Students Solutions Manual
Fundamentals of Machine Learning for Predictive Data Analytics

Students Solutions Manual
Fundamentals of Machine Learning for Predictive Data Analytics

Algorithms, Worked Examples, and Case Studies

Second Edition

John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy

The MIT Press
Cambridge, Massachusetts
London, England

Contents

Notation vii

I INTRODUCTION TO MACHINE LEARNING AND DATA ANALYTICS

1 Machine Learning for Predictive Data Analytics (Exercise Solutions) 3

2 Data to Insights to Decisions (Exercise Solutions) 7

3 Data Exploration (Exercise Solutions) 13

II PREDICTIVE DATA ANALYTICS

4 Information-Based Learning (Exercise Solutions) 35

5 Similarity-Based Learning (Exercise Solutions) 51

6 Probability-Based Learning (Exercise Solutions) 61

7 Error-Based Learning (Exercise Solutions) 71

8 Deep Learning (Exercise Solutions) 83

9 Evaluation (Exercise Solutions) 133

III BEYOND PREDICTION

10 Beyond Prediction: Unsupervised Learning (Exercise Solutions) 149

11 Beyond Prediction: Reinforcement Learning (Exercise Solutions) 159

Bibliography 167

Notation

In this section we provide a short overview of the technical notation used throughout this book.

Notational Conventions

Throughout this book we discuss the use of machine learning algorithms to train prediction models based on datasets. The following list explains the notation used to refer to different elements in a dataset. Figure 0.1[vii] illustrates the key notation using a simple sample dataset.

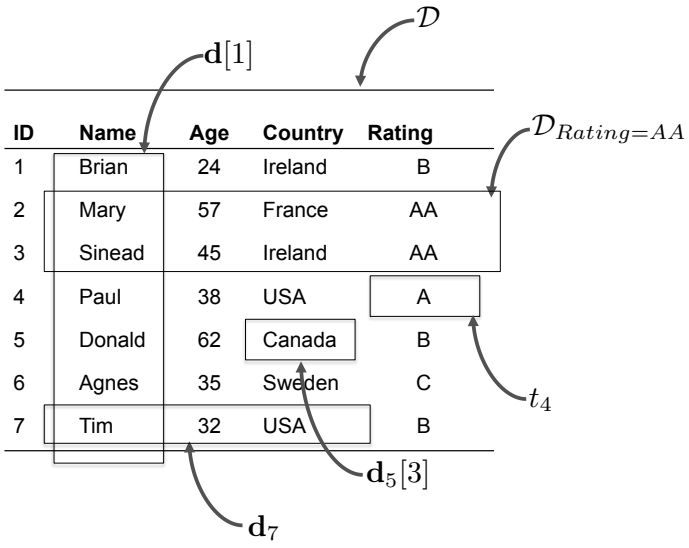


Figure 0.1
How the notation used in the book relates to the elements of a dataset.

Datasets

- $\mathcal{D}$ denotes a dataset.
- A dataset is composed of n instances, $(\mathbf{d}_1, t_1)$ to $(\mathbf{d}_n, t_n)$, where $\mathbf{d}$ is a set of m descriptive features, and t is a target feature.
- A subset of a dataset is denoted by $\mathcal{D}$ with a subscript to indicate the definition of the subset. For example, $\mathcal{D}_{f=l}$ represents the subset of instances from the dataset $\mathcal{D}$ where the feature f has the value l.

Vectors of Features

- Lowercase boldface letters refer to a vector of features. For example, $\mathbf{d}$ denotes a vector of descriptive features for an instance in a dataset, and $\mathbf{q}$ denotes a vector of descriptive features in a query.

Instances

- Subscripts are used to index into a list of instances.
- $\mathbf{x}_i$ refers to the i^{th} instance in a dataset.
- $\mathbf{d}_i$ refers to the descriptive features of the i^{th} instance in a dataset.

Individual Features

- Lowercase letters represent a single feature (e.g., f, a, b, c ...).
- Square brackets [] are used to index into a vector of features (e.g., $\mathbf{d}\,[j]$ denotes the value of the j^{th} feature in the vector $\mathbf{d}$).
- t represents the target feature.

Individual Features in a Particular Instance

- $\mathbf{d}_i\,[j]$ denotes the value of the j^{th} descriptive feature of the i^{th} instance in a dataset.
- a_i refers to the value for feature a of the i^{th} instance in a dataset.
- t_i refers to the value of the target feature of the i^{th} instance in a dataset

Indexes

- Typically, i is used to index instances in a dataset, and j is used to index features in a vector.

Models

- We use $\mathbb{M}$ to refer to a model.
- $\mathbb{M}_{\mathbf{w}}$ refers to a model $\mathbb{M}$ parameterized by a parameter vector $\mathbf{w}$.
- $\mathbb{M}_{\mathbf{w}}(\mathbf{d})$ refers to the output of a model $\mathbb{M}$ parameterized by parameters $\mathbf{w}$ for descriptive features $\mathbf{d}$.

Set Size

- Vertical bars $|\ |$ refer to counts of occurrences (e.g., $|a = l|$ represents the number of times that $a = l$ occurs in a dataset).

Feature Names and Feature Values

- We use a specific typography when referring to a feature by name in the text (e.g., POSITION, CREDITRATING, and CLAIM AMOUNT).
- For categorical features, we use a specific typography to indicate the levels in the domain of the feature when referring to a feature by name in the text (e.g., *center*, *aa*, and *soft tissue*).

Notational Conventions for Probabilities

For clarity there are some extra notational conventions used in Chapter 6[243] on probability.

Generic Events

- Uppercase letters denote generic events where an unspecified feature (or set of features) is assigned a value (or set of values). Typically, we use letters from the end of the alphabet—e.g., X, Y, Z—for this purpose.
- We use subscripts on uppercase letters to iterate over events. So, $\sum_i P(X_i)$ should be interpreted as summing over the set of events that are a complete assignment to the features in X (i.e., all the possible combinations of value assignments to the features in X).

Named Features

- Features explicitly named in the text are denoted by the uppercase initial letters of their names. For example, a feature named MENINGITIS is denoted by M.

Events Involving Binary Features

- Where a named feature is binary, we use the lowercase initial letter of the name of the feature to denote the event where the feature is true and the lowercase initial letter preceded by the $\neg$ symbol to denote the event where it is false. So, m will represent the event MENINGITIS $=$ *true*, and $\neg m$ will denote MENINGITIS $=$ *false*.

Events Involving Non-Binary Features

- We use lowercase letters with subscripts to iterate across values in the domain of a feature.
- So $\sum_i P(m_i) = P(m) + P(\neg m)$.
- In situations where a letter, for example, X, denotes a joint event, then $\sum_i P(X_i)$ should be interpreted as summing over all the possible combinations of value assignments to the features in X.

Probability of an Event

- The probability that the feature f is equal to the value v is written $P(f = v)$.

Probability Distributions

- We use bold notation $\mathbf{P}()$ to distinguish a probability distribution from a probability mass function $P()$.
- We use the convention that the first element in a probability distribution vector is the probability for a true value. For example, the probability distribution for a binary feature, A, with a probability of 0.4 of being true would be written $\mathbf{P}(A) =< 0.4, 0.6 >$.

Notational Conventions for Deep Learning

For clarity, some additional notational conventions are used in Chapter 8[381] on deep learning.

Activations

- The activation (or output) of single neuron i is denoted by a_i
- The vector of activations for a layer of neurons is denoted by $\mathbf{a}^{(k)}$ where k identifies the layer.
- A matrix of activations for a layer of neurons processing a batch of examples is denoted by $\mathbf{A}^{(k)}$ where k identifies the layer.

Activation Functions

- We use the symbol φ to generically represent activation functions. In some cases we use a subscript to indicate the use of a particular activation function. For example, φ_{SM} indicates the use of a softmax function activation function, whereas φ_{ReLU} indicates the use of a rectified linear activation function.

Categorical Targets In the context of handling categorical target features (see Section 8.4.3[463]) using a softmax function, we use the following symbols:

- We use the $\star$ symbol to indicate the index of the true category in the distribution.
- We use $\mathbf{P}$ to write the true probability distribution over the categories of the target; $\hat{\mathbf{P}}$ to write the distribution over the target categories that the model has predicted; and $\hat{\mathbf{P}}_\star$ to indicate the predicted probability for the true category.
- We write $\mathbf{t}$ to indicate the one-hot encoding vector of a categorical target.
- In some of the literature on neural networks, the term **logit** is used to refer to the result of a weighted sum calculation in a neuron (i.e., the value we normally denote z). In particular, this terminology is often used in the explanation of softmax functions; therefore, in the section on handling categorical features, we switch from our normal z notation, and instead follow this logit nomenclature, using the notation $\mathbf{l}$ to denote a vector of logits for a layer of neurons, and $\mathbf{l}_i$ to indicate the logit for the i^{th} neuron in the layer.

Elementwise Product

- We use $\odot$ to denote an elementwise product. This operation is sometimes called the **Hadamard product**.

Error Gradients (Deltas) δ

- We use the symbol δ to indicate the rate of change of the error of the network with respect to changes in the weighted sum calculated in a neuron. These δ values are the error gradients that are backpropagated during the backward pass of the backpropagation algorithm. We use a subscript to identify the particular neuron that the δ is associated with; for example, δ_i is the δ for neuron i and is equivalent to the term $\frac{\partial \mathcal{E}}{\partial z_i}$. In some cases we wish to refer to the vector of δs for the neurons in a layer l; in these cases we write $\delta^{(l)}$

Network Error

- We use the symbol $\mathcal{E}$ to denote the error of the network at the output layer.

Weights

- We use a lowercase w to indicate a single weight on a connection between two neurons. We use a double subscript to indicate the neurons that are connected, with the convention that the first subscript is the neuron the connection goes to, and the second subscript is the neuron the connection is from. For example, $w_{i,k}$ is the weight on the connection from neuron k to neuron i.
- We use $\Delta w_{i,k}$ to write the sum of error gradients calculated for the weight $w_{i,k}$. We sum errors in this way during batch gradient descent with which we sum over the examples in the batch; see Equation (8.30)[416] and also in cases in which the weight is shared by a number of neurons, whether in a convolutional neural network or during backpropagation through time.
- We use a bold capital $\mathbf{W}$ to indicate a weight matrix, and we use a superscript in brackets to indicate the layer of the network the matrix is associated with. For example, $\mathbf{W}^{(k)}$ is the weight matrix for the neurons in layer k. In an LSTM network we treat the neurons in the sigmoid and tanh layers within each gate as a separate layer of neurons, and so we write $\mathbf{W}^{(f)}$ for the weight matrix for the neurons in the forget gate, and so on for the weight matrices of the other neuron layers in the other gates. However, in a simple recurrent network we distinguish the weight matrices on the basis of whether the matrix is on the connections between the input and the hidden layer, the hidden layer and the output, or the activation memory buffer and the hidden layer. Consequently, for these matrices it is important to highlight the end of the connections the weights are applied to; we use a double subscript (similar to the subscript for a single weight), writing $\mathbf{W}_{hx}$ for the weight matrix on the connections between the input ($\mathbf{x}$) and the hidden layer ($\mathbf{h}$).

Weighted Sums and Logits

- We use a lowercase z to represent the result of the weighted sum of the inputs in a neuron. We indicate the identity of the neuron in which the calculation occurred using a subscript. For example, z_i is the result of the weighted sum calculation carried out in neuron i. Note, however, that in the section on Handling Categorical Target features, we switch to the term *logit* to refer to the output of the weight sum in a neuron and update the notation to reflect this switch; see the previous notation section on Categorical Targets for more details.
- The vector of weighted sums for a layer of neurons is denoted by $\mathbf{z}^{(k)}$ where k identifies the layer.
- A matrix of weighted sums calculations for a layer of neurons processing a batch of examples is denoted by $\mathbf{Z}^{(k)}$ where k identifies the layer.

Notational Conventions for Reinforcement Learning

For clarity there are some extra notational conventions used in Chapter 11[637] on reinforcement learning (this chapter also heavily uses the notation from the probability chapter).

Agents, States, and Actions

- In reinforcement learning we often describe an agent at time t taking an action, a_t, to move from its current state, s_t, to the next state, s_{t+1}.
- An agent's current state is often modeled as a random variable, S_t. We therefore often describe the probability that an agent is in a specific state, s, at time t as $P(S_t = s)$.
- Often states and actions are explicitly named, in which case we use the following formatting: STATE and *action*.

Transition Probabilities

- We use the $\rightarrow$ notation to represent an agent transitioning from one state to another. Therefore, the probability of an agent moving from state s_1 to state s_2 can be written

$$P(s_1 \rightarrow s_2) = P(S_{t+1} = s_2 \mid S_t = s_1)$$

- Often we condition the probability of an agent transitioning from one state, s_1, to another, s_2, on the agent taking a specific action, a. We write this

$$P(s_1 \xrightarrow{a} s_2) = P(S_{t+1} = s_2 \mid S_t = s_1, A_t = a)$$

- The dynamics of an environment in which an agent transitions between states, a Markov process, can be captured in a transition matrix

$$\mathcal{P} = \begin{bmatrix} P(s_1 \to s_1) & P(s_1 \to s_2) & \dots & P(s_1 \to s_n) \\ P(s_2 \to s_1) & P(s_2 \to s_2) & \dots & P(s_2 \to s_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_n \to s_1) & P(s_n \to s_2) & \dots & P(s_n \to s_n) \end{bmatrix}$$

- When agent decisions are allowed, leading to a Markov decision process (MDP), then the dynamics of an environment can be captured in a set of transition matrices, one for each action. For example

$$\mathcal{P}^a = \begin{bmatrix} P(s_1 \xrightarrow{a} s_1) & P(s_1 \xrightarrow{a} s_2) & \dots & P(s_1 \xrightarrow{a} s_n) \\ P(s_2 \xrightarrow{a} s_1) & P(s_2 \xrightarrow{a} s_2) & \dots & P(s_2 \xrightarrow{a} s_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_n \xrightarrow{a} s_1) & P(s_n \xrightarrow{a} s_2) & \dots & P(s_n \xrightarrow{a} s_n) \end{bmatrix}$$

I INTRODUCTION TO MACHINE LEARNING AND DATA ANALYTICS

1 Machine Learning for Predictive Data Analytics (Exercise Solutions)

1. What is **predictive data analytics**?

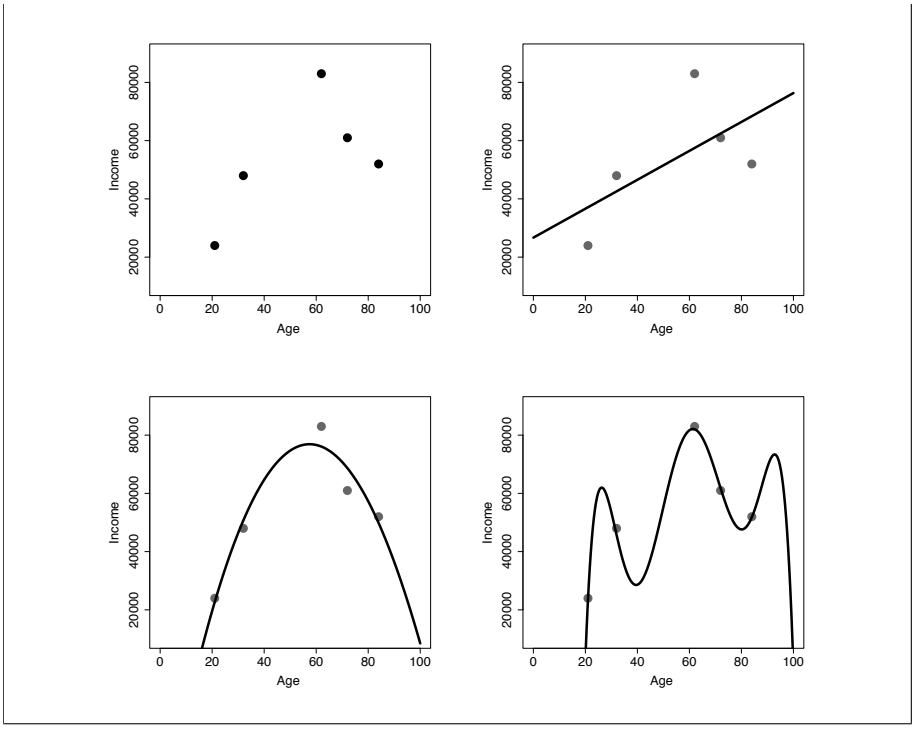
   Predictive data analytics is a subfield of data analytics that focuses on building **models** that can make predictions based on insights extracted from historical data. To build these models, we use machine learning algorithms to extract patterns from datasets.

2. What is **supervised machine learning**?

   Supervised machine learning techniques automatically learn the relationship between a set of **descriptive features** and a **target feature** from a set of historical **instances**. Supervised machine learning is a subfield of machine learning. Machine learning is defined as an automated process that extracts patterns from data. In predictive data analytics applications, we use **supervised machine learning** to build models that can make predictions based on patterns extracted from historical data.

3. Machine learning is often referred to as an **ill-posed problem**. What does this mean?

   Machine learning algorithms essentially search through all the possible patterns that exist between a set of descriptive features and a target feature to find the best model that is consistent with the training data used. It is possible to find multiple models that are consistent with a given training set (i.e., that agree with all training instances). For this reason, inductive machine learning is referred to as an ill-posed problem, as there is typically not enough information in the training data to choose a single best model. Inductive machine learning algorithms must somehow choose one of the available models as the *best*. The images below show an example of this. All the models are somewhat consistent with the training data, but which one is best?



4. The following table lists a dataset from the credit scoring domain that we discussed in the chapter. Underneath the table we list two prediction models consistent with this dataset, **Model 1** and **Model 2**.

ID	OCCUPATION	AGE	LOAN-SALARY RATIO	OUTCOME
1	industrial	39	3.40	default
2	industrial	22	4.02	default
3	professional	30	2.7 0	repay
4	professional	27	3.32	default
5	professional	40	2.04	repay
6	professional	50	6.95	default
7	industrial	27	3.00	repay
8	industrial	33	2.60	repay
9	industrial	30	4.5 0	default
10	professional	45	2.78	repay

Model 1

if LOAN-SALARY RATIO $>$ 3.00 **then**
    OUTCOME = *default*
else
    OUTCOME = *repay*
end if

Model 2

if AGE= 50 **then**
    OUTCOME = *default*
else if AGE= 39 **then**
    OUTCOME = *default*
else if AGE= 30 **and** OCCUPATION = *industrial* **then**
    OUTCOME = *default*
else if AGE= 27 **and** OCCUPATION = *professional* **then**
    OUTCOME = *default*
else
    OUTCOME = *repay*
end if

(a) Which of these two models do you think will generalize better to instances not contained in the dataset?

> **Model 1** is more likely to generalize beyond the training dataset because it is simpler and appears to be capturing a real pattern in the data.

(b) Propose an inductive bias that would enable a machine learning algorithm to make the same preference choice that you made in Part (a).

> If you are choosing between a number of models that perform equally well then prefer the simpler model over the more complex models.

(c) Do you think that the model that you rejected in Part (a) of this question is overfitting or underfitting the data?

> **Model 2** is overfitting the data. All of the decision rules in this model that predict OUTCOME = *default* are specific to single instances in the dataset. Basing predictions on single instances is indicative of a model that is overfitting.

2 Data to Insights to Decisions (Exercise Solutions)

1. An online movie streaming company has a business problem of growing **customer churn**—subscription customers canceling their subscriptions to join a competitor. Create a list of ways in which predictive data analytics could be used to help address this business problem. For each proposed approach, describe the predictive model that will be built, how the model will be used by the business, and how using the model will help address the original business problem.

   - **[Churn prediction]** A model could be built that predicts the **propensity**, or likelihood, that a customer will cancel their subscription in the next three months. This model could be run every month to identify the customers to whom the business should offer some kind of bonus to entice them to stay. The analytics problem in this case is to build a model that accurately predicts the likelihood of customers to **churn**.
   - **[Churn explanation]** By building a model that predicts the propensity of customers to cancel their subscriptions, the analytics practitioner could identify the factors that correlate strongly with customers choosing to leave the service. The business could then use this information to change its offerings so as to retain more customers. The analytics problem in this case would be to identify a small set of features that describe the company's offerings that are important in building an accurate model that predicts the likelihood of individual customers to churn.
   - **[Next-best-offer prediction]** The analytics practitioner could build a **next-best-offer** model that predicts the likely effectiveness of different bonuses that could be offered to customers to entice them to stay with the service. The company could then run this model whenever contacting a customer believed likely to leave the service and identify the least expensive bonus that is likely to entice the customer to remain a subscriber to the service. The analytics

problem in this case would be to build the most accurate next-best-offer model possible.

- **[Enjoyment prediction]** Presumably, if the company offered a better service to its customers, fewer customers would churn. The analytics practitioner could build a model that predicted the likelihood that a customer would enjoy a particular movie. The company could then put in place a service that personalized recommendations of new releases for its customers and thus reduce churn by enticing customers to stay with the service by offering them a better product. The analytics problem in this case would be to build a model that predicted, as accurately as possible, how much a customer would enjoy a given movie.

2. A national revenue commission performs audits on public companies to find and fine tax defaulters. To perform an audit, a tax inspector visits a company and spends a number of days scrutinizing the company's accounts. Because it takes so long and relies on experienced, expert tax inspectors, performing an audit is an expensive exercise. The revenue commission currently selects companies for audit at random. When an audit reveals that a company is complying with all tax requirements, there is a sense that the time spent performing the audit was wasted, and more important, that another business who is not tax compliant has been spared an investigation. The revenue commissioner would like to solve this problem by targeting audits at companies who are likely to be in breach of tax regulations, rather than selecting companies for audit at random. In this way the revenue commission hopes to maximize the yield from the audits that it performs.

   To help with **situational fluency** for this scenario, here is a brief outline of how companies interact with the revenue commission. When a company is formed, it registers with the company registrations office. Information provided at registration includes the type of industry the company is involved in, details of the directors of the company, and where the company is located. Once a company has been registered, it must provide a tax return at the end of every financial year. This includes all financial details of the company's operations during the year and is the basis of calculating the tax liability of a company. Public companies also must file public documents every year that outline how they have been performing, details of any changes in directorship, and so on.

   (a) Propose two ways in which predictive data analytics could be used to help address this business problem.[1] For each proposed approach, describe the predictive

1. Revenue commissioners around the world use predictive data analytics techniques to keep their processes as efficient as possible. Cleary and Tax (2011) is a good example.

model that will be built, how the model will be used by the business, and how using the model will help address the original business problem.

One way in which we could help to address this business problem using predictive data analytics would be to build a model that would predict the likely return from auditing a business—that is, how much unpaid tax an audit would be likely to recoup. The commission could use this model to periodically a make a prediction about every company on its register. These predictions could then be ordered from highest to lowest, and the companies with the highest predicted returns could be selected for audit. By targeting audits this way, rather than through random selection, the revenue commissioners should be able to avoid wasting time on audits that lead to no return.

Another, related way in which we could help to address this business problem using predictive data analytics would be to build a model that would predict the likelihood that a company is engaged in some kind of tax fraud. The revenue commission could use this model to periodically a make a prediction about every company on its register. These predictions could then be ordered from highest to lowest predicted likelihood, and the companies with the highest predicted propensity could be selected for audit. By targeting audits at companies likely to be engaged in fraud, rather than through random selection, the revenue commissioners should be able to avoid wasting time on audits that lead to no return.

(b) For each analytics solution you have proposed for the revenue commission, outline the type of data that would be required.

To build a model that predicts the likely yield from performing an audit, the following data resources would be required:

- Basic company details such as industry, age, and location
- Historical details of tax returns filed by each company
- Historical details of public statements issued by each company
- Details of all previous audits carried out, including the outcomes

To build a model that predicts the propensity of a company to commit fraud, the following data resources would be required:

- Basic company details such as industry, age, and location
- Historical details of tax returns filed by each company
- Historical details of public statements issued by each company

- Details of all previous audits carried out
- Details of every company the commission has found to be fraudulent

(c) For each analytics solution you have proposed, outline the capacity that the revenue commission would need in order to utilize the analytics-based insight that your solution would provide.

Utilizing the predictions of expected audit yield made by a model would be quite easy. The revenue commission already have a process in place through which they randomly select companies for audit. This process would simply be replaced by the new analytics-driven process. Because of this, the commission would require little extra capacity in order to take advantage of this system.

Similarly, utilizing the predictions of fraud likelihood made by a model would be quite easy. The revenue commission already have a process in place through which they randomly select companies for audit. This process would simply be replaced by the new analytics-driven process. Because of this, the commission would require little extra capacity in order to take advantage of this system.

3. The table below shows a sample of a larger dataset containing details of policyholders at an insurance company. The descriptive features included in the table describe each policy holders' ID, occupation, gender, age, the value of their car, the type of insurance policy they hold, and their preferred contact channel.

ID	Occupation	Gender	Age	Motor Value	Policy Type	Pref Channel
1	lab tech	female	43	42,632	planC	sms
2	farmhand	female	57	22,096	planA	phone
3	biophysicist	male	21	27,221	planA	phone
4	sheriff	female	47	21,460	planB	phone
5	painter	male	55	13,976	planC	phone
6	manager	male	19	4,866	planA	email
7	geologist	male	51	12,759	planC	phone
8	messenger	male	49	15,672	planB	phone
9	nurse	female	18	16,399	planC	sms
10	fire inspector	male	47	14,767	planC	email

(a) State whether each descriptive feature contains numeric, interval, ordinal, categorical, binary, or textual data.

ID	Ordinal	MOTORVALUE	Numeric
OCCUPATION	Textual	POLICYTYPE	Ordinal
GENDER	Categorical	AGE	Numeric
PREFCHANNEL	Categorical		

(b) How many levels does each categorical, binary, or ordinal feature have?

ID	10 are present in the sample, but there is likely to be 1 per customer
GENDER	2 (*male*, *female*)
POLICYTYPE	3 (*planA*, *planB*, *planC*)
PREFCHANNEL	3 (*sms*, *phone*, *email*)

4. Select one of the predictive analytics models that you proposed in your answer to Question 2 about the revenue commission for exploration of the design of its **analytics base table** (**ABT**).

For the answers below, the audit yield prediction model is used.

(a) What is the prediction subject for the model that will be trained using this ABT?

For the audit yield prediction model, the prediction subject is a company. We are assessing the likelihood that an audit performed on a company will yield a return, so it is the company that we are interested in assessing.

(b) Describe the domain concepts for this ABT.

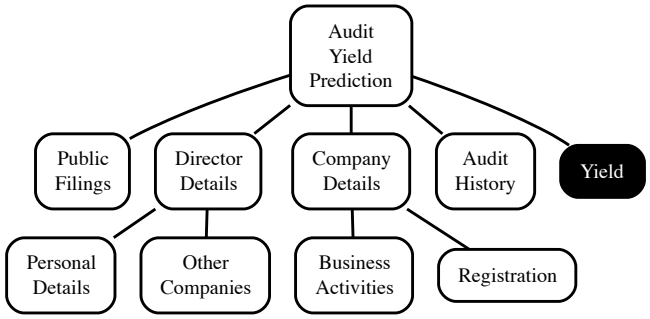
The key domain concepts for this ABT are

- *Company Details*: The details of the company itself. These could be broken down into details about the activities of the company, such as the locations it serves and the type of business activities it performs, *Business Activities*, and information provided at the time of registration, *Registration*.
- *Public Filings*: There is a wealth of information in the public documents that companies must file, and this should be included in the data used to train this model.

- *Director Details*: Company activities are heavily tied to the activities of their directors. This domain concept might be further split into details about the directors themselves, *Personal Details*, and details about other companies that the directors have links to, *Other Companies*.
- *Audit History*: Companies are often audited multiple times, and it is likely that details from previous audits would be useful in predicting the likely outcome of future audits.
- *Yield*: It is important not to forget the target feature. This would come from some measure of the yield of a previous audit.

(c) Draw a domain concept diagram for the ABT.

The following is an example domain concept diagram for the the audit yield prediction model.



(d) Are there likely to be any legal issues associated with the domain concepts you have included?

The legal issues associated with a set of domain concepts depend primarily on the data protection law within the jurisdiction within which we are working. Revenue commissions are usually given special status within data protection law and allowed access to data that other agencies would not be given. For the domain concepts given above, the one most likely to cause trouble is the *Director Details* concept. It is likely that there would be issues associated with using personal details of a company director to make decisions about a company.

3 Data Exploration (Exercise Solutions)

1. The table below shows the age of each employee at a cardboard box factory.

ID	1	2	3	4	5	6	7	8	9	10
AGE	51	39	34	27	23	43	41	55	24	25

ID	11	12	13	14	15	16	17	18	19	20
AGE	38	17	21	37	35	38	31	24	35	33

Based on this data, calculate the following **summary statistics** for the AGE feature:

(a) Minimum, maximum, and range

By simply reading through the values we can tell that the minimum value for the AGE feature is: 17.

By simply reading through the values we can tell that the maximum value for the AGE feature is: 55.

The range is simply the difference between the highest and lowest value:

$$\begin{aligned} range(\text{AGE}) &= (55 - 17) \\ &= 38 \end{aligned}$$

(b) Mean and median

We can calculate the mean of the AGE feature as follows:

$$\begin{aligned}\overline{\text{AGE}} &= \frac{1}{20} \times (51 + 39 + 34 + 27 + 23 + 43 + 41 + 55 + 24 + 25 \\ &\quad + 38 + 17 + 21 + 37 + 35 + 38 + 31 + 24 + 35 + 33) \\ &= \frac{671}{20} = 33.55\end{aligned}$$

To calculate the median of the AGE feature we first have to arrange the AGE values in ascending order:

17, 21, 23, 24, 24, 25, 27, 31, 33, **34**, **35**, 35, 37, 38, 38, 39, 41, 43, 51, 55

Because there are an even number of instances in this small dataset we take the mid-point of the middle two values as the median. These are 34 and 35 and so the median is calculated as:

$$\begin{aligned}\textit{median}(\text{AGE}) &= (34 + 35)/2 \\ &= 34.5\end{aligned}$$

(c) Variance and standard deviation

To calculate the variance we first sum the squared differences between each value for AGE and the mean of AGE. This table illustrates this:

ID	AGE	$\left(\text{AGE} - \overline{\text{AGE}}\right)$	$\left(\text{AGE} - \overline{\text{AGE}}\right)^2$
1	51	17.45	304.50
2	39	5.45	29.70
3	34	0.45	0.20
4	27	-6.55	42.90
5	23	-10.55	111.30
6	43	9.45	89.30
7	41	7.45	55.50
8	55	21.45	460.10
9	24	-9.55	91.20
10	25	-8.55	73.10
11	38	4.45	19.80
12	17	-16.55	273.90
13	21	-12.55	157.50
14	37	3.45	11.90
15	35	1.45	2.10
16	38	4.45	19.80
17	31	-2.55	6.50
18	24	-9.55	91.20
19	35	1.45	2.10
20	33	-0.55	0.30
		Sum	**1,842.95**

Based on the sum of squared differences value of $1,842.95$ we can calculate the variance as:

$$\begin{aligned} var(\text{AGE}) &= \frac{1{,}842.95}{20-1} \\ &= 96.9974 \end{aligned}$$

The standard deviation is calculated as the square root of the variance, so:

$$\begin{aligned} sd(\text{AGE}) &= \sqrt{var\,(\text{AGE})} \\ &= 9.8487 \end{aligned}$$

(d) 1^{st} quartile (25^{th} percentile) and 3^{rd} quartile (75^{th} percentile)

To calculate any percentile of the AGE feature we first have to arrange the AGE values in ascending order:

17, 21, 23, 24, 24, 25, 27, 31, 33, 34, 35, 35, 37, 38, 38, 39, 41, 43, 51, 55

We then calculate the index for the percentile value as:

$$index = n \times \frac{i}{100}$$

where n is the number of instances in the dataset and i is the percentile we would like to calculate. For the 25^{th} percentile:

$$\begin{aligned} index &= 20 \times \frac{25}{100} \\ &= 5 \end{aligned}$$

Because this is a whole number we can use this directly and so the 25^{th} percentile is at index 5 in the ordered dataset and is 24.

For the 75^{th} percentile:

$$\begin{aligned} index &= 20 \times \frac{75}{100} \\ &= 15 \end{aligned}$$

Because this is a whole number we can use this directly and so the 75^{th} percentile is at index 15 in the ordered dataset and is 38.

(e) Inter-quartile range

To calculate the inter-quartile range we subtract the lower quartile value from the upper quartile value:

$$\begin{aligned} IQR(\text{AGE}) &= (38 - 24) \\ &= 14 \end{aligned}$$

(f) 12^{th} percentile

We can use the ordered list of values above once more. For the 12^{th} percentile:

$$\begin{aligned} index &= 20 \times \frac{12}{100} \\ &= 2.4 \end{aligned}$$

Because index is not a whole number we have to calculate the percentile as follows:

$$i^{th}\text{percentile} = (1 - index\_f) \times a_{index\_w} + index\_f \times a_{index\_w+1}$$

Because $index = 2.4$, $index\_w = 2$ and $index\_f = 0.4$. Using $index\_w = 2$ we can look up AGE_2 to be 21 AGE_{2+1} to be 23. Using this we can calculate the 12^{th} percentile as:

$$\begin{aligned} 12^{th}\text{percentile of}\text{AGE} &= (1 - 0.4) \times \text{AGE}_2 + 0.4 \times \text{AGE}_{2+1} \\ &= 0.6 \times 21 + 0.4 \times 23 \\ &= 21.8 \end{aligned}$$

2. The table below shows the policy type held by customers at a life insurance company.

ID	POLICY	ID	POLICY	ID	POLICY
1	Silver	8	Silver	15	Platinum
2	Platinum	9	Platinum	16	Silver
3	Gold	10	Platinum	17	Platinum
4	Gold	11	Silver	18	Platinum
5	Silver	12	Gold	19	Gold
6	Silver	13	Platinum	20	Silver
7	Bronze	14	Silver		

(a) Based on this data, calculate the following **summary statistics** for the POLICY feature:

i. Mode and 2^{nd} mode

To calculate summary statistics for a categorical feature like this we start by counting the frequencies of each level for the feature. These are shown in this table:

Level	Frequency	Proportion
Bronze	1	0.05
Silver	8	0.40
Gold	4	0.20
Platinum	7	0.35

The proportions are calculated as the frequency of each level divided by the sum of all frequencies.

The mode is the most frequently occurring level and so in this case is *Silver*.

The 2^{nd} mode is the second most frequently occurring level and so in this case is *Platinum*.

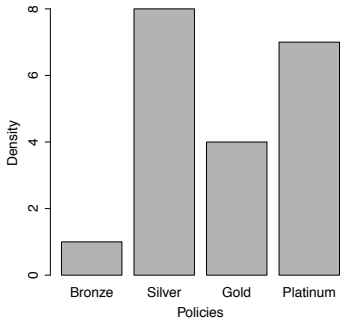
ii. Mode % and 2^{nd} mode %

The mode % is the proportion of occurrence of the mode and in this case the proportion of occurrence of *Silver* is 40%.

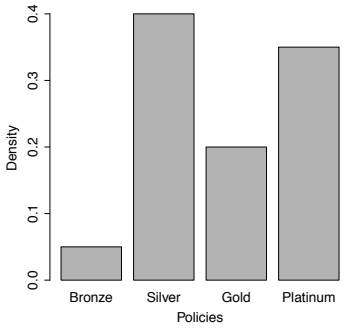
The mode % of the 2^{nd} mode, *Platinum*, is 35%.

(b) Draw a **bar plot** for the POLICY feature.

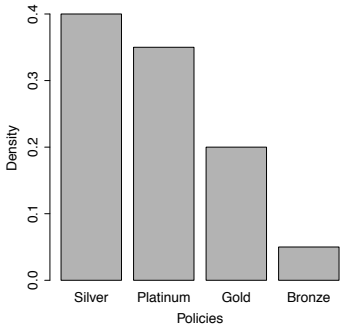
A bar plot can be drawn from the frequency table given above:



We can use proportions rather than frequencies in the plot:



In order to highlight the mode and 2^{nd} mode we could order the bars in the plot by height:



This, however, is not an especially good idea in this case as the data, although categorical, has a natural ordering and changing this in a visualisation could cause confusion.

3. An analytics consultant at an insurance company has built an **ABT** that will be used to train a model to predict the best communications channel to use to contact a potential customer with an offer of a new insurance product.[1] The following table contains an extract from this ABT—the full ABT contains 5,200 instances.

ID	Occ	Gender	Age	Loc	Motor Ins	Motor Value	Health Ins	Health Type	Health Deps Adults	Health Deps Kids	Pref Channel
1	Student	female	43	urban	yes	42,632	yes	PlanC	1	2	sms
2		female	57	rural	yes	22,096	yes	PlanA	1	2	phone
3	Doctor	male	21	rural	yes	27,221	no				phone
4	Sheriff	female	47	rural	yes	21,460	yes	PlanB	1	3	phone
5	Painter	male	55	rural	yes	13,976	no				phone
⋮		⋮			⋮			⋮			
14		male	19	rural	yes	48,66	no				email
15	Manager	male	51	rural	yes	12,759	no				phone
16	Farmer	male	49	rural	no		no				phone
17		female	18	urban	yes	16,399	no				sms
18	Analyst	male	47	rural	yes	14,767	no				email
⋮		⋮			⋮			⋮			
2747		female	48	rural	yes	35,974	yes	PlanB	1	2	phone
2748	Editor	male	50	urban	yes	40,087	no				phone
2749		female	64	rural	yes	156,126	yes	PlanC	0	0	phone
2750	Reporter	female	48	urban	yes	27,912	yes	PlanB	1	2	email
⋮		⋮			⋮			⋮			
4780	Nurse	male	49	rural	no		yes	PlanB	2	2	email
4781		female	46	rural	yes	18,562	no				phone
4782	Courier	male	63	urban	no		yes	PlanA	2	0	email
4783	Sales	male	21	urban	no		no				sms
4784	Surveyor	female	45	rural	yes	17,840	no				sms
⋮		⋮			⋮			⋮			
5199	Clerk	male	48	rural	yes	19,448	yes	PlanB	1	3	email
5200	Cook	47	female	rural	yes	16,393	yes	PlanB	1	2	sms

The descriptive features in this dataset are defined as follows:

- Age: The customer's age
- Gender: The customer's gender (*male* or *female*)
- Loc: The customer's location (*rural* or *urban*)
- Occ: The customer's occupation

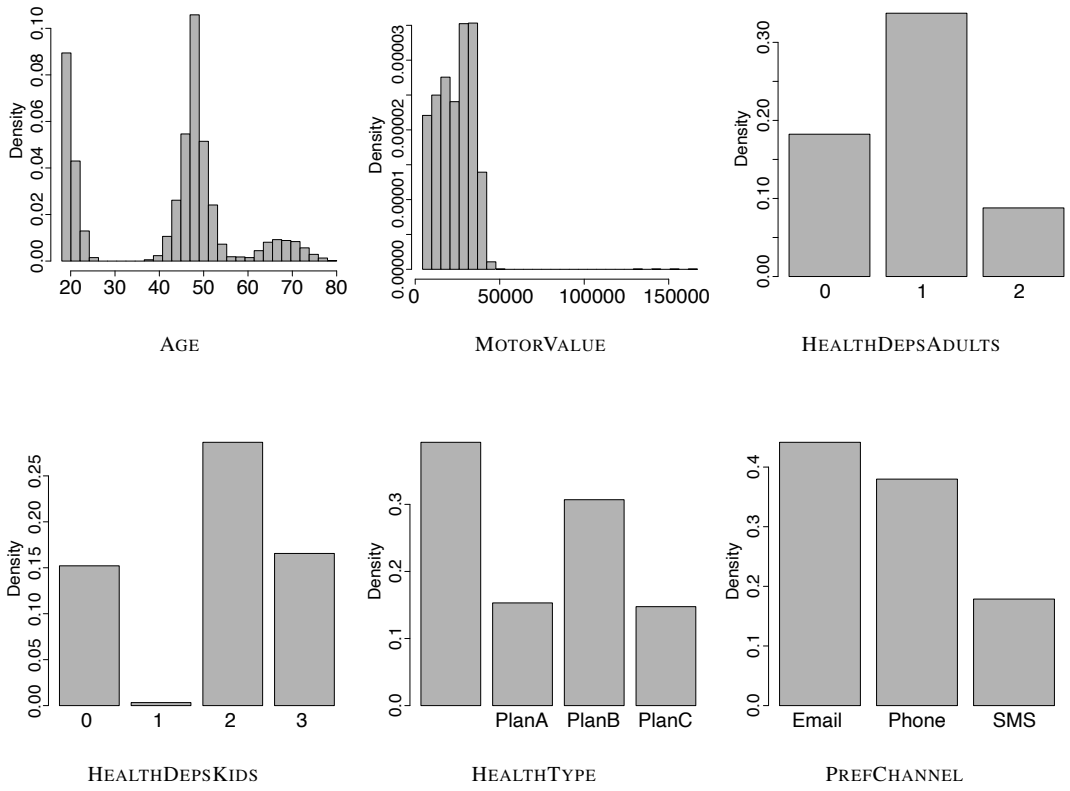
1. The data used in this question have been artificially generated for this book. Channel propensity modeling is used widely in industry; for example, see Hirschowitz (2001).

- MOTORINS: Whether the customer holds a motor insurance policy with the company (*yes* or *no*)
- MOTORVALUE: The value of the car on the motor policy
- HEALTHINS: Whether the customer holds a health insurance policy with the company (*yes* or *no*)
- HEALTHTYPE: The type of the health insurance policy (*PlanA*, *PlanB*, or *PlanC*)
- HEALTHDEPSADULTS: How many dependent adults are included on the health insurance policy
- HEALTHDEPSKIDS: How many dependent children are included on the health insurance policy
- PREFCHANNEL: The customer's preferred contact channel (*email*, *phone*, or *sms*)

The consultant generated the following **data quality report** from the ABT (visualizations of binary features have been omitted for space saving).

Feature	Count	% Miss.	Card.	Min.	1st Qrt.	Mean	Median	3rd Qrt.	Max.	Std. Dev.
AGE	5,200	0	51	18	22	41.59	47	50	80	15.66
MOTORVALUE	5,200	17.25	3,934	4,352	15,089.5	23,479	24,853	32,078	166,993	11,121
HEALTHDEPSADULTS	5,200	39.25	4	0	0	0.84	1	1	2	0.65
HEALTHDEPSKIDS	5,200	39.25	5	0	0	1.77	2	3	3	1.11

Feature	Count	% Miss.	Card.	Mode	Mode Freq.	Mode %	2nd Mode	2nd Mode Freq.	2nd Mode %
GENDER	5,200	0	2	female	2,626	50.5	male	2,574	49.5
LOC	5,200	0	2	urban	2,948	56.69	rural	2,252	43.30
OCC	5,200	37.71	1,828	Nurse	11	0.34	Sales	9	0.28
MOTORINS	5,200	0	2	yes	4,303	82.75	no	897	17.25
HEALTHINS	5,200	0	2	yes	3,159	60.75	no	2,041	39.25
HEALTHTYPE	5,200	39.25	4	PlanB	1,596	50.52	PlanA	796	25.20
PREFCHANNEL	5,200	0	3	email	2,296	44.15	phone	1,975	37.98



Discuss this data quality report in terms of the following:

(a) Missing values

> Looking at the data quality report, we can see continuous and categorical features that have significant numbers of missing: MotorValue (17.25%), HealthDepsAdults (39.25%), HealthDepsKids (39.25%), Occ (37.71%), and HealthType (39.25%).
>
> The missing values in the Occ feature look typical of this type of data. A little over a third of the customers in the dataset appear to have simply not provided this piece of information. We will discuss this feature more under cardinality, but given the large percentage of missing values and the high cardinality of this feature, imputation is probably not a good strategy. Rather this feature might be a good candidate to form the basis of a derived flag feature that simply indicates whether an occupation was provided or not.

Inspecting rows 14 to 18 of the data sample given above, we can easily see the reason for the missing values in the HEALTHDEPSADULTS, HEALTHDEPSKIDS, and HEALTHTYPE. These features always have missing values when the HEALTHINS feature has a value of *no*. From a business point of view, this makes sense—if a customer does not hold a health insurance policy, then the details of a health insurance policy will not be populated. This also explains why the missing value percentages are the same for each of these features. The explanation for the missing values for the MOTORVALUE feature is, in fact, the same. Looking at rows 4780, 4782, and 4783, we can see that whenever the MOTORINS feature has a value of *no*, then the MOTORVALUE feature has a missing value. Again, this makes sense—if a customer does not have a motor insurance policy, then none of the details of a policy will be present.

(b) Irregular cardinality

In terms of cardinality, a few things stand out. First, the AGE feature has a relatively low cardinality (given that there are 5,200 instances in the dataset). This, however, is not especially surprising as ages are given in full years, and there is naturally only a small range possible—in this data, 18– 80.

The HEALTHDEPSADULTS and HEALTHDEPSKIDS features are interesting in terms of cardinality. Both have very low values, 4 and 5 respectively. It is worth noting that a missing value counts in the cardinality calculation. For example, the only values present in the data for HEALTHDEPSADULTS are 0, 1, and 2, so it is the presence of missing values that brings cardinality to 4. We might consider changing these features to categorical features given the small number of distinct values. This, however, would lose some of the meaning captured in these features, so it should only be done after careful experimentation.

The OCC feature is interesting from a cardinality point of view. For a categorical feature to have 1,830 levels will make it pretty useless for building models. There are so many distinct values that it will be almost impossible to extract any patterns. This is further highlighted by the fact that the mode percentage for this feature is just 0.34%. This is also why no bar plot is provided for the OCC feature—there are just too many levels. Because of such high cardinality, we might just decide to remove this feature from the ABT. Another option would be to attempt to derive a feature that works at a higher level, for instance, industry, from the OCC feature. So, for example, occupations of *Occupational health nurse*, *Nurse*, *Osteopathic doctor*, and

Optometry doctor would all be transformed to *Medical*. Creating this new derived feature, however, would be a non-trivial task and would rely on the existence of an ontology or similar data resource that mapped job titles to industries.

(c) Outliers

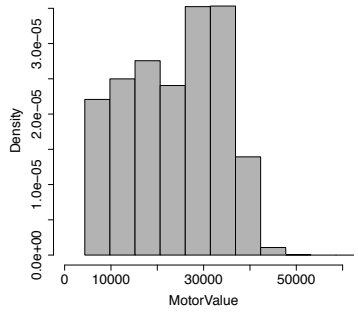
Only the MOTORVALUE feature really has an issue with outliers. We can see this in a couple of ways. First, the difference between the median and the 3^{rd} quartile and the difference between the 3^{rd} quartile and the maximum values are quite different. This suggests the presence of outliers. Second, the histogram of the MOTORVALUE feature shows huge skew to the right hand side. Finally, inspecting the data sample, we can see an instance of a very large value, 156,126 on row 2749. These outliers should be investigated with the business to determine whether they are valid or invalid, and based on this, a strategy should be developed to handle them. If valid, a clamp transformation is probably a good idea.

(d) Feature distributions

To understand the distributions of the different features, the visualizations are the most useful part of the data quality report. We'll look at the continuous features first. The AGE feature has a slightly odd distribution. We might expect age in a large population to follow a normal distribution, but this histogram shows very clear evidence of a multimodal distribution. There are three very distinct groups evident: One group of customers in their early twenties, another large group with a mean age of about 48, and a small group of older customers with a mean age of about 68. For customers of an insurance company, this is not entirely unusual, however. Insurance products tend to be targeted at specific age groups—for example, tailored motor insurance, health insurance, and life insurance policies—so it would not be unusual for a company to have specific cohorts of customers related to those products. Data with this type of distribution can also arise through merger and acquisition processes at companies. Perhaps this insurance company recently acquired another company that specialized in the senior travel insurance market? From a modeling point of view, we could hope that these three groups might be good predictors of the target feature, PREFCHANNEL.

It is hard to see much in the distribution of the MOTORVALUE feature because of the presence of the large outliers, which bunch the majority of the

data in a small portion of the graph. If we limit the histogram to a range that excludes these outliers (up to about 60,000), we can more easily see the distribution of the remaining instances.



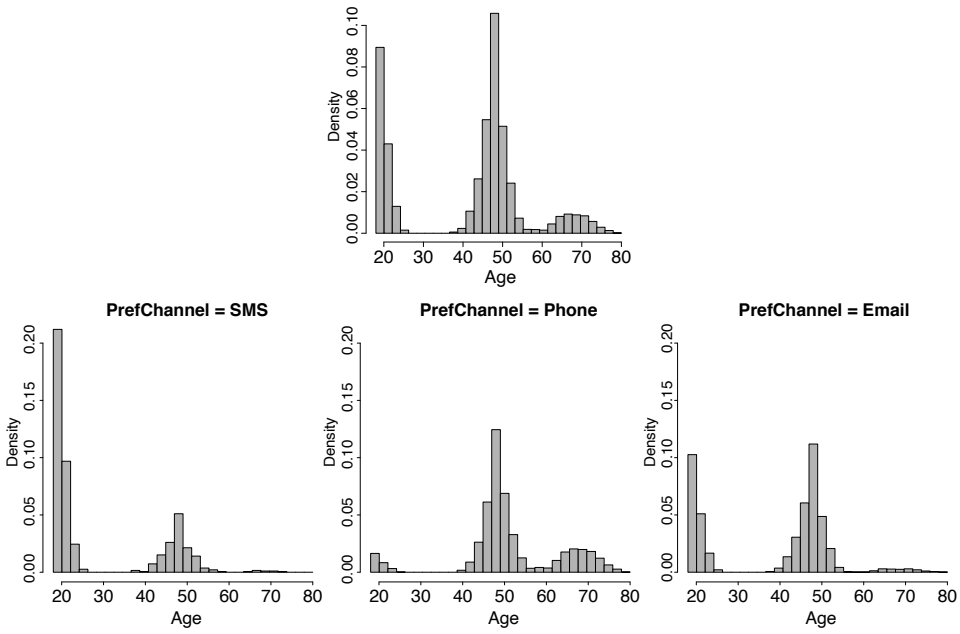
This distribution is not too far away from a unimodal distribution with left skew.

There is nothing especially remarkable about the distributions for HEALTHDEPSADULTS and HEALTHDEPSKIDS. Remembering that these features are populated only for customers with health insurance, it might be interesting for the business as a whole to learn that most customers with dependent children have more than one dependent child.

For the categorical features, the most interesting thing to learn from the distributions is that the target feature is slightly imbalanced. Many more customers prefer email contacts rather than phone or sms. Imbalanced target features can cause difficulty during the modeling process, so this should be marked for later investigation.

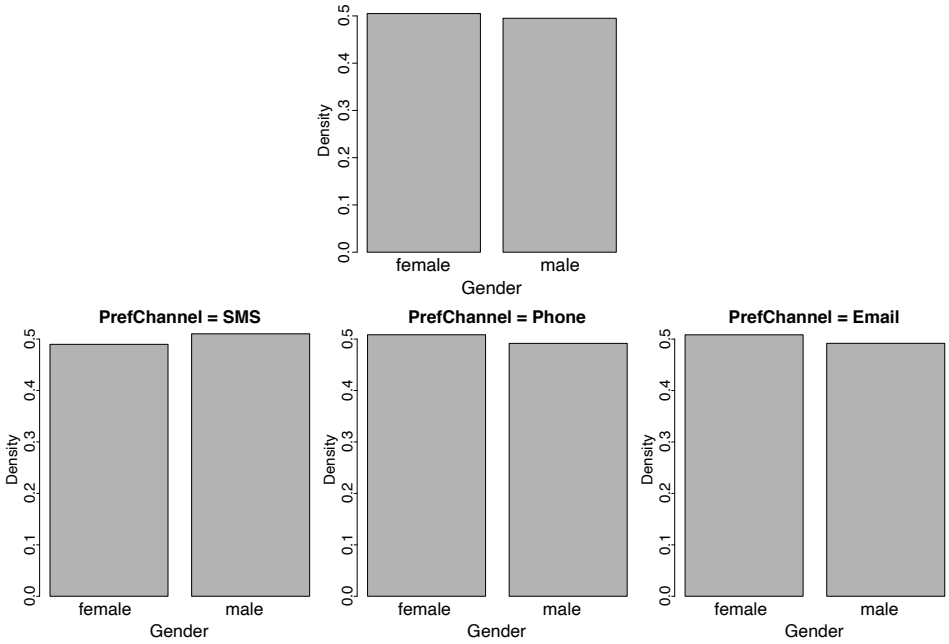
4. The following **data visualizations** are based on the channel prediction dataset given in Question 3. Each visualization illustrates the relationship between a descriptive feature and the target feature, PREFCHANNEL. Each visualization is composed of four plots: one plot of the distribution of the descriptive feature values in the entire dataset, and three plots illustrating the distribution of the descriptive feature values for each level of the target. Discuss the strength of the relationships shown in each visualization.

   (a) The visualization below illustrates the relationship between the continuous feature AGE and the target feature, PREFCHANNEL.



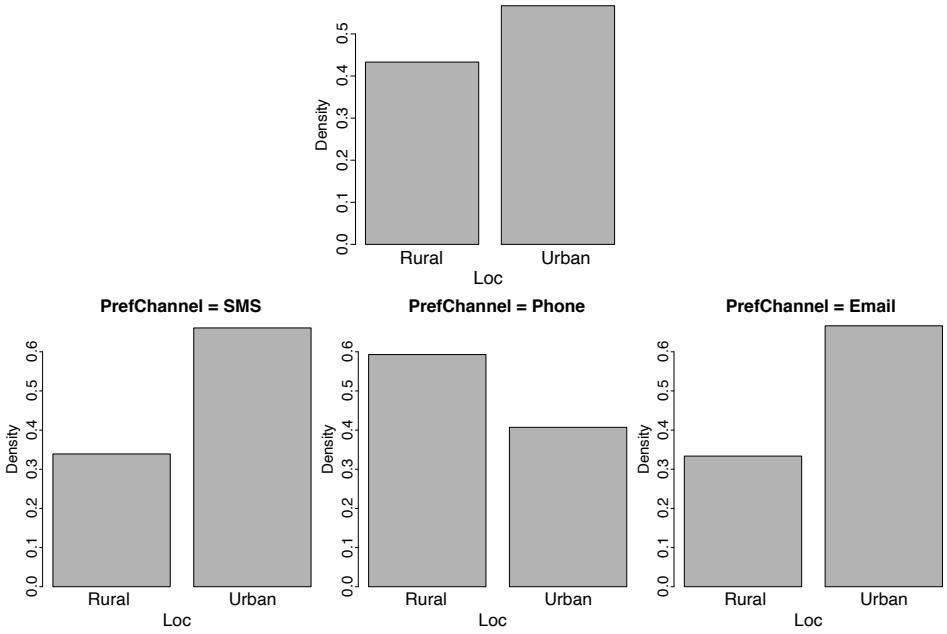
This visualization suggests a strong relationship between the AGE descriptive feature and the target feature, PREFCHANNEL. Overall we can see that the individual histograms for each data partition created by the different target levels are different from the overall histogram. Looking more deeply, we can see that the customers whose preferred channel is *sms* are predominantly younger. This is evident from the high bars in the 18–25 region of the histogram and the fact that there are very few instances in the age range above 60. We can see the opposite pattern for those customers whose preferred channel is *phone*. There are very few customers below 40 in this group. The histogram for the *email* group most closely matches the overall histogram.

(b) The visualization below illustrates the relationship between the categorical feature GENDER and the target feature PREFCHANNEL.



Each individual bar plot of GENDER created when we divide the data by the target feature is almost identical to the overall bar plot, which indicates that there is no relationship between the GENDER feature and the PREFCHANNEL feature.

(c) The visualization below illustrates the relationship between the categorical feature LOC and the target feature, PREFCHANNEL.



> The fact that the individual bar plots for each data partition are different from the overall bar plot suggests a relationship between these two features. In particular, for those customer's whose preferred channel is *phone*, the overall ratio between *rural* and *urban* locations is reversed—quite a few more rural customers prefer this channel. In the other two channel preference groups, *sms* and *email*, there are quite a few more urban dwellers. Together, this set of visualizations suggests that the LOC is reasonably predictive of the PREFCHANNEL feature.

5. The table below shows the scores achieved by a group of students on an exam.

ID	1	2	3	4	5	6	7	8	9	10
SCORE	42	47	59	27	84	49	72	43	73	59

ID	11	12	13	14	15	16	17	18	19	20
SCORE	58	82	50	79	89	75	70	59	67	35

Using this data, perform the following tasks on the SCORE feature:

(a) A **range normalization** that generates data in the range $(0, 1)$

To perform a range normalization, we need the minimum and maximum of the dataset and the high and low for the target range. From the data we can see that the minimum is 27 and the maximum is 89. In the question we are told that the low value of the target range is 0 and that the high value is 1.

Using these values, we normalize an individual value using the following equation:

$$a_i^{'} = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low$$

So, the first score in the dataset, 42, would be normalized as follows:

$$\begin{aligned} a_i^{'} &= \frac{42 - 27}{89 - 27} \times (1 - 0) + 0 \\ &= \frac{15}{62} \\ &= 0.2419 \end{aligned}$$

This is repeated for each instance in the dataset to give the full normalized data set as

ID	1	2	3	4	5	6	7	8	9	10
SCORE	0.24	0.32	0.52	0.00	0.92	0.35	0.73	0.26	0.74	0.52

ID	11	12	13	14	15	16	17	18	19	20
SCORE	0.50	0.89	0.37	0.84	1.00	0.77	0.69	0.52	0.65	0.13

(b) A **range normalization** that generates data in the range $(-1, 1)$

This normalization differs from the previous range normalization only in that the high and low values are different—in this case, -1 and 1. So the first score in the dataset, 42, would be normalized as follows:

$$\begin{aligned} a_i^{'} &= \frac{42 - 27}{89 - 27} \times (1 - (-1)) + (-1) \\ &= \frac{15}{62} \times 2 - 1 \\ &= -0.5161 \end{aligned}$$

Applying this to each instance in the dataset gives the full normalized dataset as

ID	1	2	3	4	5	6	7	8	9	10
SCORE	-0.52	-0.35	0.03	-1.00	0.84	-0.29	0.45	-0.48	0.48	0.03

ID	11	12	13	14	15	16	17	18	19	20
SCORE	0.00	0.77	-0.26	0.68	1.00	0.55	0.39	0.03	0.29	-0.74

(c) A **standardization** of the data

To perform a standardization, we use the following formula for each instance in the dataset:

$$a_i^{'} = \frac{a_i - \overline{a}}{sd(a)}$$

So we need the mean, $\overline{a}$, and standard deviation, $sd(a)$, for the feature to be standardized. In this case, the mean is calculated from the original dataset as 60.95, and the standard deviation is 17.2519. So the standardized value for the first instance in the dataset can be calculated as

$$\begin{aligned} a_i^{'} &= \frac{42 - 60.95}{17.2519} \\ &= -1.0984 \end{aligned}$$

Standardizing in the same way for the rest of the dataset gives us the following:

ID	1	2	3	4	5	6	7	8	9	10
SCORE	-1.10	-0.81	-0.11	-1.97	1.34	-0.69	0.64	-1.04	0.70	-0.11

ID	11	12	13	14	15	16	17	18	19	20
SCORE	-0.17	1.22	-0.63	1.05	1.63	0.81	0.52	-0.11	0.35	-1.50

6. The following table shows the IQs for a group of people who applied to take part in a television general-knowledge quiz.

ID	1	2	3	4	5	6	7	8	9	10
IQ	92	107	83	101	107	92	99	119	93	106

ID	11	12	13	14	15	16	17	18	19	20
IQ	105	88	106	90	97	118	120	72	100	104

Using this dataset, generate the following **binned** versions of the IQ feature:

(a) An **equal-width binning** using 5 bins.

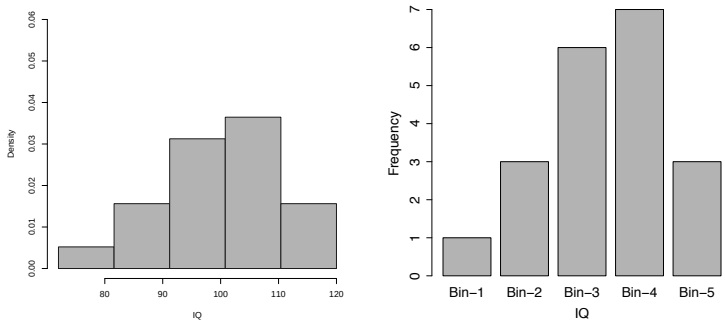
To perform an equal-width binning, we first calculate the bin size as $\frac{range}{b}$ where b is the number of bins. In this case, this is calculated as $\frac{120-72}{5} = 9.6$, where 72 and 120 are the minimum and maximum values. Using the bin, size we can calculate the following bin ranges.

Bin	Low	High
1	72.0	81.6
2	81.6	91.2
3	91.2	100.8
4	100.8	110.4
5	110.4	120.0

Once we have calculated the boundaries, we can use these to determine the bin to which each result belongs.

ID	IQ	IQ (BIN)	ID	IQ	IQ (BIN)
1	92	Bin-3	11	105	Bin-4
2	107	Bin-4	12	88	Bin-2
3	83	Bin-2	13	106	Bin-4
4	101	Bin-4	14	90	Bin-2
5	107	Bin-4	15	97	Bin-3
6	92	Bin-3	16	118	Bin-5
7	99	Bin-3	17	120	Bin-5
8	119	Bin-5	18	72	Bin-1
9	93	Bin-3	19	100	Bin-3
10	106	Bin-4	20	104	Bin-4

It is interesting to graph a histogram of the values in the dataset according to the bin boundaries as well as a bar plot showing each of the bins created.



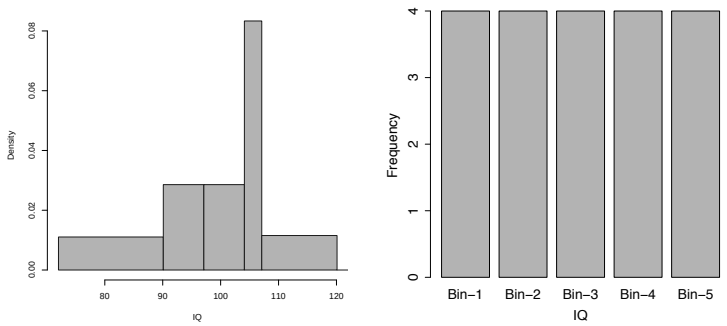
We can see from these graphs that a lot of instances belong to the middle bins and very few in the high and low bins.

(b) An **equal-frequency binning** using 5 bins

To perform an equal-frequency binning, we first determine the number of instances that will belong to each bin. This is simply the number of instances in the dataset divided by the number of bins, in this case $\frac{20}{5} = 4$. Next we sort the data by the binning feature and assign instances in order to the first bin until it is full, before moving to the second and so on. The table below shows how the instances in this dataset, sorted by RESULT, would be added to the five bins.

ID	IQ	IQ (BIN)	ID	IQ	IQ (BIN)
18	72	Bin-1	4	101	Bin-3
3	83	Bin-1	20	104	Bin-3
12	88	Bin-1	11	105	Bin-4
14	90	Bin-1	10	106	Bin-4
1	92	Bin-2	13	106	Bin-4
6	92	Bin-2	2	107	Bin-4
9	93	Bin-2	5	107	Bin-5
15	97	Bin-2	16	118	Bin-5
7	99	Bin-3	8	119	Bin-5
19	100	Bin-3	17	120	Bin-5

It is interesting to graph a histogram of the values in the dataset according to the bin boundaries as well as a bar plot showing each of the bins created.



Remember that in the histogram, because the bins are not of equal width, the bars are different heights. The area of a bar represents the density of the instances in the range represented by the bar. The key things to note here are that each bin is equally populated, but that the bins in the middle are much narrower than those at either end of the range.

II PREDICTIVE DATA ANALYTICS

4 Information-Based Learning (Exercise Solutions)

1. The image below shows a set of eight Scrabble pieces.

   (a) What is the **entropy** in bits of the letters in this set?

   We can calculate the probability of randomly selecting a letter of each type from this set: $P(O) = \frac{3}{8}$, $P(X) = \frac{1}{8}$, $P(Y) = \frac{1}{8}$, $P(M) = \frac{1}{8}$, $P(R) = \frac{1}{8}$, $P(N) = \frac{1}{8}$.

   Using these probabilities, we can calculate the entropy of the set:

$$-\left(\frac{3}{8} \times log_2\left(\frac{3}{8}\right) + \left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right) \times 5\right) = 2.4056\ bits$$

   Note that the contribution to the entropy for any letter that appears only once is the same and so has been included 5 times—once for each of *X*, *Y*, *M*, *R*, and *N*.

   (b) What would be the reduction in entropy (i.e., the **information gain**) in bits if we split these letters into two sets, one containing the vowels and the other containing the consonants?

   Information gain is the reduction in entropy that occurs after we split the original set. We know that the entropy of the initial set is 2.4056 *bits*. We calculate the remaining entropy after we split the original set using a weighted

summation of the entropies for the new sets. The two new sets are vowels {O,O,O} and consonants {X,Y,M,R,N}.

The entropy of the vowel set is

$$-\left(\frac{3}{3} \times log_2\left(\frac{3}{3}\right)\right) = 0 \; bits$$

The entropy of the consonant set is

$$-\left(\left(\frac{1}{5} \times log_2\left(\frac{1}{5}\right)\right) \times 5\right) = 2.3219 \; bits$$

The weightings used in the summation of the set entropies are just the relative size of each set. So, the weighting of the vowel set entropy is $\frac{3}{8}$, and the weighting of the consonant set entropy is $\frac{5}{8}$.

This gives the entropy remaining after we split the set of letters into vowels and consonants as

$$rem = \frac{3}{8} \times 0 + \frac{5}{8} \times 2.3219 = 1.4512 \; bits$$

The information gain is the difference between the initial entropy and the remainder:

$$IG = 2.4056 - 1.4512 = 0.9544 \; bits$$

(c) What is the maximum possible entropy in bits for a set of eight Scrabble pieces?

The maximum entropy occurs when there are eight different letters in the set. The entropy for a set with this distribution of letters is

$$\left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right) \times 8 = 3 \; bits$$

(d) In general, which is preferable when you are playing Scrabble: a set of letters with high entropy or a set of letters with low entropy?

In general, sets of letters with high entropy are preferable to lower entropy sets because the more diverse the letters in the set, the more words you are likely to be able to make from the set.

2. A convicted criminal who reoffends after release is known as a *recidivist*. The following table lists a dataset that describes prisoners released on parole and whether they reoffended within two years of release.[1]

ID	GOOD BEHAVIOR	AGE < 30	DRUG DEPENDENT	RECIDIVIST
1	false	true	false	true
2	false	false	false	false
3	false	true	false	true
4	true	false	false	false
5	true	false	true	true
6	true	false	false	false

This dataset lists six instances in which prisoners were granted parole. Each of these instances is described in terms of three binary descriptive features (GOOD BEHAVIOR, AGE < 30, DRUG DEPENDENT) and a binary target feature (RECIDIVIST). The GOOD BEHAVIOR feature has a value of *true* if the prisoner had not committed any infringements during incarceration, the AGE < 30 has a value of *true* if the prisoner was under 30 years of age when granted parole, and the DRUG DEPENDENT feature is *true* if the prisoner had a drug addiction at the time of parole. The target feature, RECIDIVIST, has a *true* value if the prisoner was arrested within two years of being released; otherwise it has a value of *false*.

(a) Using this dataset, construct the decision tree that would be generated by the **ID3** algorithm, using entropy-based information gain.

The first step in building the decision tree is to figure out which of the three descriptive features is the best one on which to split the dataset at the root node (i.e., which descriptive feature has the highest information gain). The total entropy for this dataset is computed as follows:

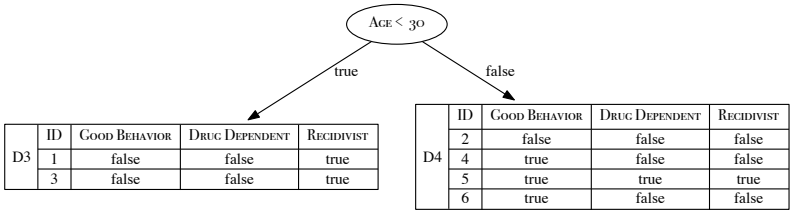
$$
\begin{aligned}
& H(\text{RECIDIVIST}, \mathcal{D}) \\
& = - \sum_{l \in \left\{ \substack{true, \\ false} \right\}} P(\text{RECIDIVIST} = l) \times log_2 \left(P(\text{RECIDIVIST} = l) \right) \\
& = - \left(\left({}^3\!/_6 \times log_2({}^3\!/_6) \right) + \left({}^3\!/_6 \times log_2({}^3\!/_6) \right) \right) = 1.00 \; bit
\end{aligned}
$$

1. This example of predicting recidivism is based on a real application of machine learning: parole boards do rely on machine learning prediction models to help them when they are making their decisions. See Berk and Bleich (2013) for a recent comparison of different machine learning models used for this task. Datasets dealing with prisoner recidivism are available online, for example, catalog.data.gov/dataset/prisoner-recidivism/. The dataset presented here is not based on real data.

The table below illustrates the computation of the information gain for each of the descriptive features:

Split by Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
GOOD BEHAVIOR	*true*	$\mathcal{D}_1$	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.9183	0.9183	0.0817
	false	$\mathcal{D}_2$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.9183		
AGE $<$ 30	*true*	$\mathcal{D}_3$	$\mathbf{d}_1, \mathbf{d}_3$	0	0.5409	0.4591
	false	$\mathcal{D}_4$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.8113		
DRUG DEPENDENT	*true*	$\mathcal{D}_5$	$\mathbf{d}_5$	0	0.8091	0.1909
	false	$\mathcal{D}_6$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6$	0.9709		

AGE $<$ 30 has the largest information gain of the three features. Consequently, this feature will be used at the root node of the tree. The figure below illustrates the state of the tree after we have created the root node and split the data based on AGE $<$ 30.



D3

ID	GOOD BEHAVIOR	DRUG DEPENDENT	RECIDIVIST
1	false	false	true
3	false	false	true

D4

ID	GOOD BEHAVIOR	DRUG DEPENDENT	RECIDIVIST
2	false	false	false
4	true	false	false
5	true	true	true
6	true	false	false

In this image we have shown how the data moves down the tree based on the split on the AGE $<$ 30 feature. Note that this feature no longer appears in these datasets because we cannot split on it again.

The dataset on the left branch contains only instances where RECIDIVIST is *true* and so does not need to be split any further.

The dataset on the right branch of the tree ($\mathcal{D}_4$) is not homogenous, so we need to grow this branch of the tree. The entropy for this dataset, $\mathcal{D}_4$, is calculated as follows:

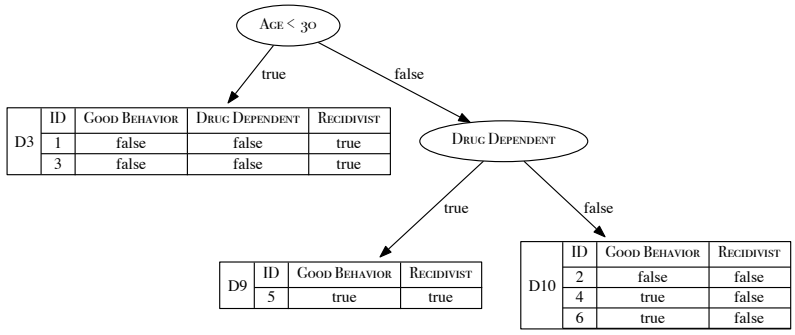
$$
\begin{aligned}
& H(\text{RECIDIVIST}, \mathcal{D}_4) \\
& = - \sum_{l \in \{true, false\}} P(\text{RECIDIVIST} = l) \times log_2 \left(P(\text{RECIDIVIST} = l) \right) \\
& = - \left(\left({}^1/_4 \times log_2({}^1/_4) \right) + \left({}^3/_4 \times log_2({}^3/_4) \right) \right) = 0.8113 \; bits
\end{aligned}
$$

The table below shows the computation of the information gain for the GOOD BEHAVIOR and DRUG DEPENDENT features in the context of the $\mathcal{D}_4$ dataset:

Split by Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
GOOD BEHAVIOR	*true*	$\mathcal{D}_7$	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.918295834	0.4591	0.3522
	false	$\mathcal{D}_8$	$\mathbf{d}_2$	0		
DRUG DEPENDENT	*true*	$\mathcal{D}_9$	$\mathbf{d}_5$	0	0	0.8113
	false	$\mathcal{D}_{10}$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_6$	0		

These calculations show that the DRUG DEPENDENT feature has a higher information gain than GOOD BEHAVIOR: 0.8113 versus 0.3522 and so should be chosen for the next split.

The image below shows the state of the decision tree after the $\mathcal{D}_4$ partition has been split based on the feature DRUG DEPENDENT.



D3

ID	GOOD BEHAVIOR	DRUG DEPENDENT	RECIDIVIST
1	false	false	true
3	false	false	true

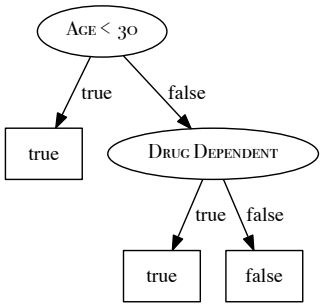
D9

ID	GOOD BEHAVIOR	RECIDIVIST
5	true	true

D10

ID	GOOD BEHAVIOR	RECIDIVIST
2	false	false
4	true	false
6	true	false

All the datasets at the leaf nodes are now pure, so the algorithm will stop growing the tree. The image below shows the tree that will be returned by the ID3 algorithm:



(b) What prediction will the decision tree generated in Part (a) of this question return for the following query?

GOOD BEHAVIOR = *false*,AGE $<$ 30 = *false*,
DRUG DEPENDENT = *true*

RECIDIVIST = *true*

(c) What prediction will the decision tree generated in Part (a) of this question return for the following query?

GOOD BEHAVIOR = *true*,AGE $<$ 30 = *true*,
DRUG DEPENDENT = *false*

RECIDIVIST = *true*

3. The following table lists a sample of data from a census.[2]

ID	AGE	EDUCATION	MARITAL STATUS	OCCUPATION	ANNUAL INCOME
1	39	bachelors	never married	transport	25K–50K
2	50	bachelors	married	professional	25K–50K
3	18	high school	never married	agriculture	$<$25K
4	28	bachelors	married	professional	25K–50K
5	37	high school	married	agriculture	25K–50K
6	24	high school	never married	armed forces	$<$25K
7	52	high school	divorced	transport	25K–50K
8	40	doctorate	married	professional	$>$50K

There are four descriptive features and one target feature in this dataset, as follows:

- AGE, a continuous feature listing the age of the individual;
- EDUCATION, a categorical feature listing the highest education award achieved by the individual (*high school*, *bachelors*, *doctorate*);
- MARITAL STATUS (*never married*, *married*, *divorced*);
- OCCUPATION (*transport* = works in the transportation industry; *professional* = doctor, lawyer, or similar; *agriculture* = works in the agricultural industry; *armed forces* = is a member of the armed forces); and
- ANNUAL INCOME, the target feature with 3 levels ($<$25K, 25K–50K, $>$50K).

(a) Calculate the **entropy** for this dataset.

2. This census dataset is based on the Census Income Dataset (Kohavi, 1996), which is available from the UCI Machine Learning Repository (Bache and Lichman, 2013) at archive.ics.uci.edu/ml/datasets/Census+Income/.

$$
\begin{aligned}
&H\left(\text{ANNUAL INCOME}, \mathcal{D}\right) \\
&= - \sum_{l \in \left\{ \begin{smallmatrix} <25K, \\ 25K–50K, \\ >50K \end{smallmatrix} \right\}} P(\text{AN. INC.} = l) \times log_2\left(P(\text{AN. INC.} = l)\right) \\
&= -\left(\left(\frac{2}{8} \times log_2\left(\frac{2}{8}\right)\right) + \left(\frac{5}{8} \times log_2\left(\frac{5}{8}\right)\right) + \left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right)\right) \\
&= 1.2988 \; bits
\end{aligned}
$$

(b) Calculate the **Gini index** for this dataset.

$$
\begin{aligned}
&Gini\left(\text{ANNUAL INCOME}, \mathcal{D}\right) \\
&= 1 - \sum_{l \in \left\{ \begin{smallmatrix} <25K, \\ 25K–50K, \\ >50K \end{smallmatrix} \right\}} P(\text{AN. INC.} = l)^2 \\
&= 1 - \left(\left(\frac{2}{8}\right)^2 + \left(\frac{5}{8}\right)^2 + \left(\frac{1}{8}\right)^2\right) = 0.5313
\end{aligned}
$$

(c) In building a decision tree, the easiest way to handle a continuous feature is to define a threshold around which splits will be made. What would be the optimal threshold to split the continuous AGE feature (use information gain based on entropy as the feature selection measure)?

First sort the instances in the dataset according to the AGE feature, as shown in the following table.

ID	AGE	ANNUAL INCOME
3	18	$<25K$
6	24	$<25K$
4	28	$25K–50K$
5	37	$25K–50K$
1	39	$25K–50K$
8	40	$>50K$
2	50	$25K–50K$
7	52	$25K–50K$

Based on this ordering, the mid-points in the AGE values of instances that are adjacent in the new ordering but that have different target levels define the possible threshold points. These points are 26, 39.5, and 45.

We calculate the information gain for each of these possible threshold points using the entropy value we calculated in part (a) of this question (1.2988 bits) as follows:

Split by Feature	Partition	Instances	Partition Entropy	Rem.	Info. Gain
>26	$\mathcal{D}_1$	$\mathbf{d}_3, \mathbf{d}_6$	0	0.4875	0.8113
	$\mathcal{D}_2$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_7, \mathbf{d}_8$	0.6500		
>39.5	$\mathcal{D}_3$	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.9710	0.9456	0.3532
	$\mathcal{D}_4$	$\mathbf{d}_2, \mathbf{d}_7, \mathbf{d}_8$	0.9033		
>45	$\mathcal{D}_5$	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_8$	1.4591	1.0944	0.2044
	$\mathcal{D}_6$	$\mathbf{d}_2, \mathbf{d}_7$	0		

The threshold AGE $>$ 26 has the highest information gain, and consequently, it is the best threshold to use if we are splitting the dataset using the AGE feature.

(d) Calculate **information gain** (based on entropy) for the EDUCATION, MARITAL STATUS, and OCCUPATION features.

We have already calculated the entropy for the full dataset in part (a) of this question as 1.2988 bits. The table below lists the rest of the calculations for the information gain for the EDUCATION, MARITAL STATUS, and OCCUPATION features.

Split by Feature	Level	Instances	Partition Gini Index	Rem.	Info. Gain
EDUCATION	*high school*	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_7$	1.0	0.5	0.7988
	bachelors	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0		
	doctorate	$\mathbf{d}_8$	0		
MARITAL STATUS	*never married*	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_6$	0.9183	0.75	0.5488
	married	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_8$	0.8113		
	divorced	$\mathbf{d}_7$	0		
OCCUPATION	*transport*	$\mathbf{d}_1, \mathbf{d}_7$	0	0.5944	0.7044
	professional	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_8$	0.9183		
	agriculture	$\mathbf{d}_3, \mathbf{d}_5$	1.0		
	armed forces	$\mathbf{d}_6$	0		

(e) Calculate the **information gain ratio** (based on entropy) for EDUCATION, MARITAL STATUS, and OCCUPATION features.

In order to calculate the information gain ratio of a feature, we divide the information gain of the feature by the entropy of the feature itself. We have

already calculated the information gain of these features in the preceding part of this question:

- IG(EDUCATION, $\mathcal{D}$) = 0.7988
- IG(MARITAL STATUS, $\mathcal{D}$) = 0.5488
- IG(OCCUPATION, $\mathcal{D}$) = 0.7044

We calculate the entropy of each feature as follows:

$$
\begin{aligned}
&H\left(\text{EDUCATION}, \mathcal{D}\right) \\
&= - \sum_{l \in \left\{ \substack{\textit{high school}, \\ \textit{bachelors}, \\ \textit{doctorate}} \right\}} P(\text{ED.} = l) \times log_2\left(P(\text{ED.} = l)\right) \\
&= -\left(\left(\frac{4}{8} \times log_2\left(\frac{4}{8}\right)\right) + \left(\frac{3}{8} \times log_2\left(\frac{3}{8}\right)\right) + \left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right)\right) \\
&= 1.4056 \; bits
\end{aligned}
$$

$$
\begin{aligned}
&H\left(\text{MARITAL STATUS}, \mathcal{D}\right) \\
&= - \sum_{l \in \left\{ \substack{\textit{never married}, \\ \textit{married}, \\ \textit{divorced}} \right\}} P(\text{MAR. STAT.} = l) \times log_2\left(P(\text{MAR. STAT.} = l)\right) \\
&= -\left(\left(\frac{3}{8} \times log_2\left(\frac{3}{8}\right)\right) + \left(\frac{4}{8} \times log_2\left(\frac{4}{8}\right)\right) + \left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right)\right) \\
&= 1.4056 \; bits
\end{aligned}
$$

$$
\begin{aligned}
&H\left(\text{OCCUPATION}, \mathcal{D}\right) \\
&= - \sum_{l \in \left\{ \substack{\textit{transport}, \\ \textit{professional}, \\ \textit{agriculture}, \\ \textit{armed forces}} \right\}} P(\text{OCC.} = l) \times log_2\left(P(\text{OCC.} = l)\right) \\
&= -(\left(\frac{2}{8} \times log_2\left(\frac{2}{8}\right)\right) + \left(\frac{3}{8} \times log_2\left(\frac{3}{8}\right)\right) \\
&+ \left(\frac{2}{8} \times log_2\left(\frac{2}{8}\right)\right) + \left(\frac{1}{8} \times log_2\left(\frac{1}{8}\right)\right)) \\
&= 1.9056 \; bits
\end{aligned}
$$

We can now calculate the information gain ratio for each feature as:

- GR(EDUCATION, $\mathcal{D}$) = $\dfrac{0.7988}{1.4056} = 0.5683$

- GR(MARITAL STATUS, $\mathcal{D}$) $= \dfrac{0.5488}{1.4056} = 0.3904$
- GR(OCCUPATION, $\mathcal{D}$) $= \dfrac{0.7044}{1.9056} = 0.3696$

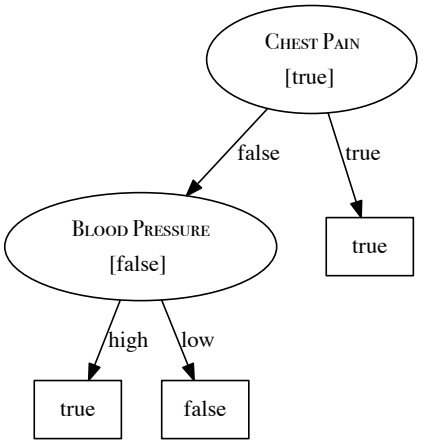
(f) Calculate **information gain** using the **Gini index** for the EDUCATION, MARITAL STATUS, and OCCUPATION features.

We have already calculated the Gini index for the full dataset in part (b) of this question as 0.5313. The table below lists the rest of the calculations of information gain for the EDUCATION, MARITAL STATUS, and OCCUPATION features.

Split by Feature	Level	Instances	Partition Gini Index	Rem.	Info. Gain
EDUCATION	*high school*	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_7$	0.5	0.25	0.2813
	bachelors	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0		
	doctorate	$\mathbf{d}_8$	0		
MARITAL STATUS	*never married*	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_6$	0.4444	0.3542	0.1771
	married	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_8$	0.375		
	divorced	$\mathbf{d}_7$	0		
OCCUPATION	*transport*	$\mathbf{d}_1, \mathbf{d}_7$	0	0.2917	0.2396
	professional	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_8$	0.4444		
	agriculture	$\mathbf{d}_3, \mathbf{d}_5$	0.5		
	armed forces	$\mathbf{d}_6$	0		

4. The following diagram shows a decision tree for the task of predicting heart disease.[3] The descriptive features in this domain describe whether the patient suffers from chest pain (CHEST PAIN) and the blood pressure of the patient (BLOOD PRESSURE). The binary target feature is HEART DISEASE. The table beside the diagram lists a pruning set from this domain.

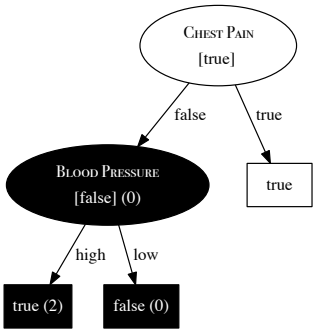
3. This example is inspired by the research reported in Palaniappan and Awang (2008).



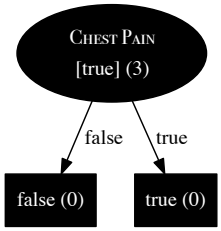
ID	Chest Pain	Blood Pressure	Heart Disease
1	false	high	false
2	true	low	true
3	false	low	false
4	true	high	true
5	false	high	false

Using the pruning set, apply **reduced error pruning** to the decision tree. Assume that the algorithm is applied in a bottom-up, left-to-right fashion. For each iteration of the algorithm, indicate the subtrees considered as pruning candidates, explain why the algorithm chooses to prune or leave these subtrees in the tree, and illustrate the tree that results from each iteration.

The first subtree that will be considered for pruning by the algorithm is the subtree under the blood pressure node. The nodes colored in black in the figure below illustrate the extent of this subtree. For each node, the value given in square brackets is the majority target level returned at that node, and the number in round brackets is the number of errors in the pruning set made as a result of predictions returned from this node.



The root node of this subtree returns *false*, and this results in 0 errors in the pruning set. The sum of the errors for the two leaf nodes of this subtree is 2. The algorithm will prune this subtree because the number of errors resulting from the leaf nodes is higher than the number of errors resulting from the root node.
The figure below illustrates the structure of the tree after the subtree under the BLOOD PRESSURE node is pruned. This figure also highlights the extent of the subtree that is considered for pruning in the second iteration of the algorithm (the entire tree in this case).



The root node of this tree returns *true* as a prediction, and consequently, it results in 3 errors on the pruning set. By contrast the number of errors made at each of the leaf nodes of this tree is 0. Because the number of errors at the leaf nodes is less than the number of errors at the root node, this tree will not be pruned. At this point all the non-leaf nodes in the tree have been tested, so the pruning algorithm will stop, and this decision tree is the one that is returned by the pruning algorithm.

5. The following table[4] lists a dataset containing the details of five participants in a heart disease study, and a target feature RISK, which describes their risk of heart disease. Each patient is described in terms of four binary descriptive features

   - EXERCISE, how regularly do they exercise
   - SMOKER, do they smoke
   - OBESE, are they overweight
   - FAMILY, did any of their parents or siblings suffer from heart disease

4. The data in this table has been artificially generated for this question, but is inspired by the results from the Framingham Heart Study: www.framinghamheartstudy.org.

ID	EXERCISE	SMOKER	OBESE	FAMILY	RISK
1	daily	false	false	yes	low
2	weekly	true	false	yes	high
3	daily	false	false	no	low
4	rarely	true	true	yes	high
5	rarely	true	true	no	high

(a) As part of the study, researchers have decided to create a predictive model to screen participants based on their risk of heart disease. You have been asked to implement this screening model using a **random forest**. The three tables below list three bootstrap samples that have been generated from the above dataset. Using these bootstrap samples, create the decision trees that will be in the random forest model (use entropy-based information gain as the feature selection criterion).

ID	EXERCISE	FAMILY	RISK
1	daily	yes	low
2	weekly	yes	high
2	weekly	yes	high
5	rarely	no	high
5	rarely	no	high

Bootstrap Sample A

ID	SMOKER	OBESE	RISK
1	false	false	low
2	true	false	high
2	true	false	high
4	true	true	high
5	true	true	high

Bootstrap Sample B

ID	OBESE	FAMILY	RISK
1	false	yes	low
1	false	yes	low
2	false	yes	high
4	true	yes	high
5	true	no	high

Bootstrap Sample C

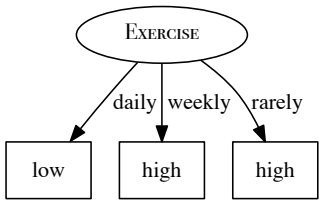
The entropy calculation for Bootstrap Sample A is:

$$
\begin{aligned}
&H\left(\text{RISK}, BoostrapSampleA\right) \\
&= - \sum_{l \in \left\{ \substack{low,} \\ high \right\}} P(\text{RISK} = l) \times log_2\left(P(\text{RISK} = l)\right) \\
&= - \left(\left(\frac{1}{5} \times log_2 \left(\frac{1}{5} \right) \right) + \left(\frac{4}{5} \times log_2 \left(\frac{4}{5} \right) \right) \right) \\
&= 0.7219 \; bits
\end{aligned}
$$

The information gain for each of the features in Bootstrap Sample A is as follows:

Split by Feature	Level	Instances	Partition Entropy	Rem.	Info. Gain
EXERCISE	*daily*	$\mathbf{d}_1$	0	0	0.7219
	weekly	$\mathbf{d}_2, \mathbf{d}_2$	0		
	rarely	$\mathbf{d}_5, \mathbf{d}_5,$	0		
FAMILY	*yes*	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_2$	0.9183	0.5510	0.1709
	no	$\mathbf{d}_5, \mathbf{d}_5$	0		

These calculations show that the EXERCISE feature has the highest information gain of the descriptive features in Bootstrap Sample A and should be added as the root node of the decision tree generated from Bootstrap Sample A. What is more, splitting on EXERCISE generates pure sets. So, the decision tree does not need to be expanded beyond this initial test and the final tree generated for Bootstrap Sample A will be as shown below.



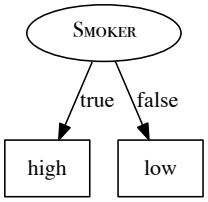
By chance, Bootstrap Sample B has the same distribution of target feature values as Bootstrap Sample A, so the entropy calculation for Bootstrap Sample B is the same as the calculation for Bootstrap Sample A:

$$
\begin{aligned}
& H\left(\text{RISK}, BoostrapSampleB\right) \\
& = - \sum_{l \in \left\{ \substack{low, \\ high} \right\}} P(\text{RISK} = l) \times log_2\left(P(\text{RISK} = l)\right) \\
& = - \left(\left(\frac{1}{5} \times log_2\left(\frac{1}{5}\right) \right) + \left(\frac{4}{5} \times log_2\left(\frac{4}{5}\right) \right) \right) \\
& = 0.7219 \; bits
\end{aligned}
$$

The information gain for each of the features in Bootstrap Sample B is as follows:

Split by Feature	Level	Instances	Partition Entropy	Rem.	Info. Gain
SMOKER	*true*	$\mathbf{d}_2, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5$	0	0	0.7219
	false	$\mathbf{d}_1$	0		
OBESE	*true*	$\mathbf{d}_4, \mathbf{d}_5$	0	0.5510	0.1709
	false	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_2$	0.9183		

These calculations show that the SMOKER feature has the highest information gain of the descriptive features in Bootstrap Sample B and should be added as the root node of the decision tree generated from Bootstrap Sample B. What is more, splitting on SMOKER generates pure sets, So the decision tree does not need to be expanded beyond this initial test. The final tree generated for Bootstrap Sample B is shown below.



The entropy calculation for Bootstrap Sample C is:

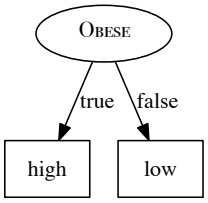
$$
\begin{aligned}
& H\left(\text{RISK}, BoostrapSampleC\right) \\
& = -\sum_{l \in \left\{ \substack{low, \\ high} \right\}} P(\text{RISK} = l) \times log_2\left(P(\text{RISK} = l)\right) \\
& = -\left(\left(\frac{2}{5} \times log_2\left(\frac{2}{5}\right)\right) + \left(\frac{3}{5} \times log_2\left(\frac{3}{5}\right)\right)\right) \\
& = 0.9710\ bits
\end{aligned}
$$

The information gain for each of the features in Bootstrap Sample C is as follows:

Split by Feature	Level	Instances	Partition Entropy	Rem.	Info. Gain
OBESE	*true*	$\mathbf{d}_4, \mathbf{d}_5$	0	0.5510	0.4200
	false	$\mathbf{d}_1, \mathbf{d}_1, \mathbf{d}_2$	0.9183		
FAMILY	*yes*	$\mathbf{d}_1, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4$	1.0	0.8	0.1709
	no	$\mathbf{d}_5$	0		

These calculations show that the OBESE feature has the highest information gain of the descriptive features in Bootstrap Sample C and should be added as the root node of the decision tree generated from Bootstrap Sample C. Splitting Bootstrap Sample C creates one pure partition for OBESE=*true* ($\mathbf{d}_4, \mathbf{d}_5$) where all the instances have RISK=*high*, and an impure partition for OBESE=*false* where two instances ($\mathbf{d}_1, \mathbf{d}_1$) have RISK=*low* and for one instance ($\mathbf{d}_2$) RISK=*high*.

Normally this would mean that we would continue to split the impure partition to create pure sets. However, in this instance there is only one feature that we can still use to split this partition, the FAMILY feature, and all the instances in this partition have the same level for this feature FAMILY=*yes*. Consequently, instead of splitting this partition further we simply create a leaf node with the majority target level within the partition: RISK=*low*. So, the final tree generated for Bootstrap Sample C will be as shown below.



(b) Assuming the random forest model you have created uses majority voting, what prediction will it return for the following query:

EXERCISE=*rarely*, SMOKER=*false*, OBESE=*true*, FAMILY=*yes*

Each of the trees in the ensemble will vote as follows:

- Tree 1: EXERCISE=*rarely* → RISK=*high*
- Tree 2: SMOKER=*false* → RISK=*low*
- Tree 3: OBESE=*true* → RISK=*high*

So, the majority vote is for RISK=*high*, and this is the prediction the model will return for this query.

5 Similarity-Based Learning (Exercise Solutions)

1. The table below lists a dataset that was used to create a nearest neighbor model that predicts whether it will be a good day to go surfing.

ID	Wave Size (ft)	Wave Period (secs)	Wind Speed (mph)	Good Surf
1	6	15	5	yes
2	1	6	9	no
3	7	10	4	yes
4	7	12	3	yes
5	2	2	10	no
6	10	2	20	no

Assuming that the model uses Euclidean distance to find the nearest neighbor, what prediction will the model return for each of the following query instances?

ID	Wave Size (ft)	Wave Period (secs)	Wind Speed (mph)	Good Surf
Q1	8	15	2	?
Q2	8	2	18	?
Q3	6	11	4	?

The table below lists the training instances along with the distances between each training instance and each query. The distance between each query instance and its nearest training instance is highlighted in bold.

ID	Wave Size (ft)	Wave Period (secs)	Wind Speed (mph)	Good Surf	Euc. Dist. to Q1	Euc. Dist. to Q2	Euc. Dist. to Q3
1	6	15	5	yes	3.61	18.49	4.12
2	1	6	9	no	13.38	12.08	8.66
3	7	10	4	yes	5.48	16.16	**1.41**
4	7	12	3	yes	**3.32**	18.06	1.73
5	2	2	10	no	16.40	10.00	11.53
6	10	2	20	no	22.29	**2.83**	18.79

From this table we can see that

- The nearest neighbor to Q1 is training instance $\mathbf{d}_4$ which is 3.32 units away from Q1. This training instance has a target level of GOOD SURF=*yes*. So the model will predict GOOD SURF=*yes* for Q1.
- The nearest neighbor to Q2 is training instance $\mathbf{d}_6$ which is 2.83 units away from Q2. This training instance has a target level of GOOD SURF=*no*. So the model will predict GOOD SURF=*no* for Q2.
- The nearest neighbor to Q3 is training instance $\mathbf{d}_3$ which is 1.41 units away from Q3. This training instance has a target level of GOOD SURF=*yes*. So the model will predict GOOD SURF=*yes* for Q3.

2. Email spam filtering models often use a **bag-of-words** representation for emails. In a bag-of-words representation, the descriptive features that describe a document (in our case, an email) each represent how many times a particular word occurs in the document. One descriptive feature is included for each word in a predefined dictionary. The dictionary is typically defined as the complete set of words that occur in the training dataset. The table below lists the bag-of-words representation for the following five emails and a target feature, SPAM, whether they are spam emails or genuine emails:

   - "*money, money, money*"
   - "*free money for free gambling fun*"
   - "*gambling for fun*"
   - "*machine learning for fun, fun, fun*"
   - "*free machine learning*"

	Bag-of-Words							
ID	MONEY	FREE	FOR	GAMBLING	FUN	MACHINE	LEARNING	SPAM
1	3	0	0	0	0	0	0	true
2	1	2	1	1	1	0	0	true
3	0	0	1	1	1	0	0	true
4	0	0	1	0	3	1	1	false
5	0	1	0	0	0	1	1	false

   (a) What target level would a nearest neighbor model using **Euclidean distance** return for the following email: "*machine learning for free*"?

   The bag-of-words representation for this query is as follows:

ID	Bag-of-Words							SPAM
	MONEY	FREE	FOR	GAMBLING	FUN	MACHINE	LEARNING	
Query	0	1	1	0	0	1	1	?

The table below shows the calculation of the Euclidean distance between the query instance and each of the instances in the training dataset:

ID	$(\mathbf{q}[i[-\mathbf{d}_j[i[)^2$							Euclidean Distance
	MONEY	FREE	FOR	GAMBLING	FUN	MACHINE	LEARNING	
1	9	1	1	0	0	1	1	3.6056
2	1	1	0	1	1	1	1	2.4495
3	0	1	0	1	1	1	1	2.2361
4	0	1	0	0	9	0	0	3.1623
5	0	0	1	0	0	0	0	1

Based on these distance calculations, the nearest neighbor to the query is instance $\mathbf{d}_5$, for which SPAM = *false*. Consequently, the model will return a prediction of SPAM = *false* for this query.

(b) What target level would a k-NN model with $k = 3$ and using **Euclidean distance** return for the same query?

Based on the distance calculations in part (a) of this question, the three nearest neighbors to the query are instances $\mathbf{d}_5$, $\mathbf{d}_3$, and $\mathbf{d}_2$. The majority of these three neighbors have a target value of SPAM = *true*. Consequently, the 3-NN model will return a prediction of SPAM = *true*.

(c) What target level would a **weighted *k*-NN** model with $k = 5$ and using a weighting scheme of the reciprocal of the squared Euclidean distance between the neighbor and the query, return for the query?

The weights for each of the instances in the dataset are

ID	Weights	
1	$\frac{1}{3.6056^2}$	$= 0.0769$
2	$\frac{1}{2.4495^2}$	$= 0.1667$
3	$\frac{1}{2.2361^2}$	$= 0.2$
4	$\frac{1}{3.1623^2}$	$= 0.1$
5	$\frac{1}{1^2}$	$= 1$

The total weight for the SPAM = *true* target level is $0.0769 + 0.1667 + 0.2 = 0.4436$. The total weight for the SPAM = *false* target level is $0.1 + 1 = 1.1$. Consequently, the SPAM = *false* has the maximum weight, and this is the prediction returned by the model.

(d) What target level would a k-NN model with $k = 3$ and using **Manhattan distance** return for the same query?

The table below shows the calculation of the Manhattan distance between the query bag-of-words vector and each instance in the dataset:

	$abs(\mathbf{q}[i[- \mathbf{d}_j[i[)$							Manhattan
ID	MONEY	FREE	FOR	GAMBLING	FUN	MACHINE	LEARNING	Distance
1	3	1	1	0	0	1	1	7
2	1	1	0	1	1	1	1	6
3	0	1	0	1	1	1	1	5
4	0	1	0	0	3	0	0	4
5	0	0	1	0	0	0	0	1

Based on these Manhattan distance calculations, the three nearest neighbors to the query are instances $\mathbf{d}_5$, $\mathbf{d}_4$, and $\mathbf{d}_3$. The majority of these three neighbors have a target value of SPAM = *false*. Consequently, the 3-NN model using Manhattan distance will return a prediction of SPAM = *false*.

(e) There are a lot of zero entries in the spam bag-of-words dataset. This is indicative of **sparse data** and is typical for text analytics. **Cosine similarity** is often a good choice when dealing with sparse non-binary data. What target level would a 3-NN model using cosine similarity return for the query?

In order to calculate the cosine similarity between the query and each instance in the dataset, we first need to calculate the vector length of each instance and the query. The table below illustrates the calculation of these vector lengths.

									Vector
ID	$\mathbf{d}[i[^2$							Sum	Length
1	9	0	0	0	0	0	0	9	3
2	1	4	1	1	1	0	0	8	2.8284
3	0	0	1	1	1	0	0	3	1.7321
4	0	0	1	0	9	1	1	12	3.4641
5	0	1	0	0	0	1	1	3	1.7321
Query	0	1	1	0	0	1	1	4	2

The second component we need to calculate is the dot product between the query and each instance. The table below illustrates the calculation of these dot products.

Pair	$(\mathbf{q}[i[\times \mathbf{d}_j[i[)$							Dot Product
$(\mathbf{q}, \mathbf{d}_1)$	0	0	0	0	0	0	0	0
$(\mathbf{q}, \mathbf{d}_2)$	0	2	1	0	0	0	0	3
$(\mathbf{q}, \mathbf{d}_3)$	0	0	1	0	0	0	0	1
$(\mathbf{q}, \mathbf{d}_4)$	0	0	1	0	0	1	1	3
$(\mathbf{q}, \mathbf{d}_5)$	0	1	0	0	0	1	1	3

We can now calculate the cosine similarity for each query-instance pair by dividing the relevant dot product by the product of the respective vector lengths. These calculations are shown below.

Pair	Cosine Similarity	
$(\mathbf{q}, \mathbf{d}_1)$	$\frac{0}{3 \times 2}$	$= 0$
$(\mathbf{q}, \mathbf{d}_2)$	$\frac{3}{2.8285 \times 2}$	$= 0.5303$
$(\mathbf{q}, \mathbf{d}_3)$	$\frac{1}{1.7321 \times 2}$	$= 0.2887$
$(\mathbf{q}, \mathbf{d}_4)$	$\frac{3}{3.4641 \times 2}$	$= 0.4330$
$(\mathbf{q}, \mathbf{d}_5)$	$\frac{3}{1.7321 \times 2}$	$= 0.8660$

When we use a similarity index, such as cosine similarity, the higher the number, the more similar the instances. Given this, the three most similar instances in the dataset to the query are instances $\mathbf{d}_5$, $\mathbf{d}_2$, and $\mathbf{d}_4$. The majority of these three neighbors have a target value of SPAM = *false*. Consequently, the 3-NN model will return a prediction of SPAM = *false*.

3. The predictive task in this question is to predict the level of corruption in a country based on a range of macroeconomic and social features. The table below lists some countries described by the following descriptive features:

   - LIFE EXP., the mean life expectancy at birth
   - TOP-10 INCOME, the percentage of the annual income of the country that goes to the top 10% of earners
   - INFANT MORT., the number of infant deaths per 1,000 births
   - MIL. SPEND, the percentage of GDP spent on the military
   - SCHOOL YEARS, the mean number years spent in school by adult females

The target feature is the **Corruption Perception Index** (CPI). The CPI measures the perceived levels of corruption in the public sector of countries and ranges from 0 (highly corrupt) to 100 (very clean).[1]

COUNTRY ID	LIFE EXP.	TOP-10 INCOME	INFANT MORT.	MIL. SPEND	SCHOOL YEARS	CPI
Afghanistan	59.61	23.21	74.30	4.44	0.40	1.5171
Haiti	45.00	47.67	73.10	0.09	3.40	1.7999
Nigeria	51.30	38.23	82.60	1.07	4.10	2.4493
Egypt	70.48	26.58	19.60	1.86	5.30	2.8622
Argentina	75.77	32.30	13.30	0.76	10.10	2.9961
China	74.87	29.98	13.70	1.95	6.40	3.6356
Brazil	73.12	42.93	14.50	1.43	7.20	3.7741
Israel	81.30	28.80	3.60	6.77	12.50	5.8069
USA	78.51	29.85	6.30	4.72	13.70	7.1357
Ireland	80.15	27.23	3.50	0.60	11.50	7.5360
UK	80.09	28.49	4.40	2.59	13.00	7.7751
Germany	80.24	22.07	3.50	1.31	12.00	8.0461
Canada	80.99	24.79	4.90	1.42	14.20	8.6725
Australia	82.09	25.40	4.20	1.86	11.50	8.8442
Sweden	81.43	22.18	2.40	1.27	12.80	9.2985
New Zealand	80.67	27.81	4.90	1.13	12.30	9.4627

We will use Russia as our query country for this question. The table below lists the descriptive features for Russia.

COUNTRY ID	LIFE EXP.	TOP-10 INCOME	INFANT MORT.	MIL. SPEND	SCHOOL YEARS	CPI
Russia	67.62	31.68	10.00	3.87	12.90	?

(a) What value would a 3-nearest neighbor prediction model using Euclidean distance return for the CPI of Russia?

> The table below lists the countries in the dataset and their CPI values by increasing Euclidean distance from Russia (column 2).

1. The data listed in this table is real and is for 2010/11 (or the most recent year prior to 2010/11 when the data was available). The data for the descriptive features in this table was amalgamated from a number of surveys retrieved from **Gapminder** (www.gapminder.org). The Corruption Perception Index is generated annually by **Transparency International** (www.transparency.org).

ID	$Euclidean(\mathbf{q}, \mathbf{d_i})$	CPI
Argentina	9.7805	2.9961
China	10.7898	3.6356
U.S.A	12.6033	7.1357
Egypt	13.7217	2.8622
Brazil	14.7394	3.7741
U.K.	15.0621	7.7751
Israel	16.0014	5.8069
Ireland	16.0490	7.5360
New Zealand	16.3806	9.4627
Canada	17.2765	8.6725
Australia	18.1472	8.8442
Germany	18.2352	8.0461
Sweden	19.8056	9.2985
Afghanistan	66.5419	1.5171
Haiti	69.6705	1.7999
Nigeria	75.2712	2.4493

The nearest three neighbors to Russia are Argentina, China, and U.S.A. The CPI value that will be returned by the model is the average CPI score for these three neighbors, which is

$$\frac{2.9961 + 3.6356 + 7.1357}{3} = 4.5891$$

(b) What value would a **weighted *k*-NN** prediction model return for the CPI of Russia? Use $k = 16$ (i.e., the full dataset) and a weighting scheme of the reciprocal of the squared Euclidean distance between the neighbor and the query.

The table below shows the calculations required to answer this question.

ID	$Euclidean(\mathbf{q}, \mathbf{d_i})$	CPI	Weight	Weight × CPI
Argentina	9.7805	2.9961	0.0105	0.0313
China	10.7898	3.6356	0.0086	0.0312
U.S.A	12.6033	7.1357	0.0063	0.0449
Egypt	13.7217	2.8622	0.0053	0.0152
Brazil	14.7394	3.7741	0.0046	0.0174
U.K.	15.0621	7.7751	0.0044	0.0343
Israel	16.0014	5.8069	0.0039	0.0227
Ireland	16.0490	7.5360	0.0039	0.0293
New Zealand	16.3806	9.4627	0.0037	0.0353
Canada	17.2765	8.6725	0.0034	0.0291
Australia	18.1472	8.8442	0.0030	0.0269
Germany	18.2352	8.0461	0.0030	0.0242
Sweden	19.8056	9.2985	0.0025	0.0237
Afghanistan	66.5419	1.5171	0.0002	0.0003
Haiti	69.6705	1.7999	0.0002	0.0004
Nigeria	75.2712	2.4493	0.0002	0.0004
Sum Weight:			0.0637	
Sum Weight × CPI:				0.3665

The value returned by the model is the sum of the instance weights multiplied by the instance target value divided by the sum of the instance weights:

$$\frac{0.3665}{0.0637} = 5.7507$$

(c) The descriptive features in this dataset are of different types. For example, some are percentages, others are measured in years, and others are measured in counts per 1,000. We should always consider normalizing our data, but it is particularly important to do this when the descriptive features are measured in different units. What value would a 3-nearest neighbor prediction model using Euclidean distance return for the CPI of Russia when the descriptive features have been normalized using range normalization?

The table below lists the range-normalized descriptive features and the un-normalized CPI.

COUNTRY ID	LIFE EXP.	TOP-10 INCOME	INFANT MORT.	MIL. SPEND	SCHOOL YEARS	CPI
Afghanistan	0.3940	0.0445	0.8965	0.6507	0.0000	1.5171
Haiti	0.0000	1.0000	0.8815	0.0000	0.2174	1.7999
Nigeria	0.1698	0.6313	1.0000	0.1384	0.2681	2.4493
Egypt	0.6869	0.1762	0.2145	0.2652	0.3551	2.8622
Argentina	0.8296	0.3996	0.1359	0.0963	0.7029	2.9961
China	0.8053	0.3090	0.1409	0.2786	0.4348	3.6356
Brazil	0.7582	0.8148	0.1509	0.2004	0.4928	3.7741
Israel	0.9785	0.2629	0.0150	1.0000	0.8768	5.8069
U.S.A	0.9034	0.3039	0.0486	0.6922	0.9638	7.1357
Ireland	0.9477	0.2016	0.0137	0.0757	0.8043	7.5360
U.K.	0.9459	0.2508	0.0249	0.3749	0.9130	7.7751
Germany	0.9501	0.0000	0.0137	0.1818	0.8406	8.0461
Canada	0.9702	0.1063	0.0312	0.1996	1.0000	8.6725
Australia	1.0000	0.1301	0.0224	0.2651	0.8043	8.8442
Sweden	0.9821	0.0043	0.0000	0.1760	0.8986	9.2985
New Zealand	0.9617	0.2242	0.0312	0.1547	0.8623	9.4627

We also need to normalize the query in order to generate a prediction, so we show the normalized version of the descriptive features for Russia in the table below.

COUNTRY ID	LIFE EXP.	TOP-10 INCOME	INFANT MORT.	MIL. SPEND	SCHOOL YEARS	CPI
Russia	0.6099	0.3754	0.0948	0.5658	0.9058	?

The table below lists the countries in the dataset by increasing Euclidean distance—calculated using the normalized descriptive features—between Russia and the country (column 2). Notice that this ordering is different from the ordering of the countries when we used the unnormalized descriptive features.

ID	$Euclidean(\mathbf{q}, \mathbf{d_i})$	CPI
Egypt	0.00004	2.8622
Brazil	0.00048	3.7741
China	0.00146	3.6356
Afghanistan	0.00217	1.5171
Argentina	0.00233	2.9961
United States	0.00742	7.1357
United Kingdom	0.01275	7.7751
Ireland	0.01302	7.5360
Germany	0.01339	8.0461
New Zealand	0.01531	9.4627
Canada	0.01685	8.6725
Israel	0.01847	5.8069
Sweden	0.01918	9.2985
Australia	0.02316	8.8442
Nigeria	0.03753	2.4493
Haiti	0.13837	1.7999

In this instance, the three nearest neighbors to Russia are Egypt, Brazil, and China. The CPI value that will be returned by the model is the average CPI score for these three neighbors:

$$\frac{2.8622 + 3.7741 + 3.6356}{3} = 3.4240$$

(d) What value would a **weighted *k*-NN** prediction model—with $k = 16$ (i.e., the full dataset) and using a weighting scheme of the reciprocal of the squared Euclidean distance between the neighbor and the query—return for the CPI of Russia when it is applied to the range-normalized data?

The table below shows the calculations required to answer this question.

ID	$Euclidean(\mathbf{q}, \mathbf{d_i})$	CPI	Weight	Weight×CPI
Egypt	0.00004	2.8622	809,250,011.4	2,316,224,862.0
Brazil	0.00048	3.7741	4,284,287.3	16,169,371.4
China	0.00146	3.6356	471,369.7	1,713,699.1
Afghanistan	0.00217	1.5171	211,391.8	320,701.1
Argentina	0.00233	2.9961	184,029.5	551,366.0
United States	0.00742	7.1357	18,176.9	129,704.9
United Kingdom	0.01275	7.7751	6,154.1	47,849.0
Ireland	0.01302	7.5360	5,899.5	44,459.1
Germany	0.01339	8.0461	5,575.7	44,863.0
New Zealand	0.01531	9.4627	4,263.8	40,347.0
Canada	0.01685	8.6725	3,520.9	30,535.1
Israel	0.01847	5.8069	2,932.5	17,028.7
Sweden	0.01918	9.2985	2,717.1	25,265.1
Australia	0.02316	8.8442	1,864.8	16,492.9
Nigeria	0.03753	2.4493	710.1	1,739.3
Haiti	0.13837	1.7999	52.2	94.0
Sum Weight:			814,452,958	
Sum Weight × CPI:				2,335,378,378

The value returned by the model is the sum of the instance weights multiplied by the instance target value divided by the sum of the instance weights:

$$\frac{2{,}335{,}378{,}378}{814{,}452{,}958} = 2.8674$$

(e) The actual 2011 CPI for Russia was 2.4488. Which of the predictions made was the most accurate? Why do you think this was?

The most accurate prediction made was the one based on normalized data using the weighted k-NN model, 2.8674. There are two main reasons for this. First, this example illustrates the importance of normalizing data. Because the data ranges in this dataset are so different from each other, normalization is crucial. The second main reason is the small size of the dataset. Using three nearest neighbors probably tends to underfit slightly for such a small dataset. Using weighted distances allows for this.

6 Probability-Based Learning (Exercise Solutions)

1. (a) Three people flip a fair coin. What is the probability that exactly two of them will get heads?

There are 8 possible outcomes:

Person1	Person2	Person3
Heads	Heads	Heads
Heads	Heads	Tails
Heads	Tails	Heads
Heads	Tails	Tails
Tails	Heads	Heads
Tails	Heads	Tails
Tails	Tails	Heads
Tails	Tails	Tails

In 3 of these outcomes there are 2 Heads. So the probability of exactly two people getting heads is

$$\frac{3}{8} = 0.375$$

(b) Twenty people flip a fair coin. What is the probability that exactly eight of them will get heads?

We could use the same approach as we used in part (a) to answer this question: list all possible outcomes and count the number of outcomes that match our criteria. However, this approach doesn't scale up to problems where there are a lot of possible outcomes. For example, in part (a) with 3 people flipping the coin there were $2^3 = 8$ possible outcomes. However, now with 20 people

flipping the coin there are $2^{20} = 1{,}048{,}576$ possible outcomes—clearly, too many outcomes for us to list out. So what should we do?

Because each coin flip is independent of the others, the coin flips can be viewed as a sequence of independent binary experiments. Consequently, we can calculate the probability of getting k outcomes, each with a probability of p, in a sequence of n experiments using the **binomial distribution** as

$$\binom{n}{k} \times p^k \times (1-p)^{n-k}$$

where n is the number of binary experiments, k is the number of particular results we are looking for (e.g., the number of heads we are looking for), and p is the probability of getting the result we are looking for (e.g., the probability of getting a head).

So, we can calculate the probability of the event where exactly 8 of the coin flips comes up heads using the binomial distribution as follows

$$\begin{aligned}\binom{20}{8} \times 0.5^8 \times (1-0.5)^{20-8} &= \frac{20!}{8! \times (20-8)!} \times 0.5^8 \times 0.5^{12} \\ &= 125970 \times 0.00390625 \times 0.000244141 \\ &= 0.120134354\end{aligned}$$

Note: the ! symbol represents the factorial operation, for example

$$6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$$

(c) Twenty people flip a fair coin. What is the probability that at least four of them will get heads?

The probability that at least 4 people will get heads is equal to 1 minus probability that less then 4 people will get heads.

The probability that less then 4 people will get heads is simply the sum of the following probabilities:

- the probability that exactly 3 people will get heads
- the probability that exactly 2 people will get heads
- the probability that exactly 1 person will get heads

We can calculate each of these probabilities using the binomial distribution as follows:

The probability of exactly 3 people getting heads:

$$\binom{20}{3} \times 0.5^3 \times (1-0.5)^{20-3} = \frac{20!}{3! \times (20-3)!} \times 0.5^3 \times 0.5^{17}$$
$$= 1140 \times 0.125 \times (7.62939 \times 10^{-6})$$
$$= 0.001087189$$

The probability of exactly 2 people getting heads:

$$\binom{20}{2} \times 0.5^2 \times (1-0.5)^{20-2} = \frac{20!}{2! \times (20-2)!} \times 0.5^2 \times 0.5^{18}$$
$$= 190 \times 0.25 \times (3.8147 \times 10^{-6})$$
$$= 0.000181198$$

The probability of exactly 1 person getting heads:

$$\binom{20}{1} \times 0.5^1 \times (1-0.5)^{20-1} = \frac{20!}{1! \times (20-1)!} \times 0.5^1 \times 0.5^{19}$$
$$= 20 \times 0.5 \times (1.90735 \times 10^{-6})$$
$$= 0.0000190735$$

Probability of 3 people or less getting heads is:

$$0.001087189 + 0.000181198 + 0.0000190735 = 0.0012874605$$

So the probability of at least 4 people getting heads is:

$$1 - 0.0012874605 = 0.9987125$$

2. The table below gives details of symptoms that patients presented and whether they were suffering from meningitis.

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

Using this dataset, calculate the following probabilities:

(a) $P(\text{VOMITING} = \textit{true})$

This can be calculated easily by counting:
$P(\text{VOMITING} = \textit{true}) = \frac{6}{10} = 0.6$

(b) $P(\text{HEADACHE} = \textit{false})$

This can be calculated easily by counting:
$P(\text{HEADACHE} = \textit{false}) = \frac{3}{10} = 0.3$

(c) $P(\text{HEADACHE} = \textit{true}, \text{VOMITING} = \textit{false})$

This can be calculated easily by counting:
$P(\text{HEADACHE} = \textit{true}, \text{VOMITING} = \textit{false}) = \frac{1}{10} = 0.1$
Or using the product rule:
$P(\text{HEADACHE} = \textit{true}, \text{VOMITING} = \textit{false}) = P(\text{HEADACHE} = \textit{true} \mid \text{VOMITING} = \textit{false}) \times P(\text{VOMITING} = \textit{false}) = \frac{1}{4} \times \frac{4}{10} = 0.1$

(d) $P(\text{VOMITING} = \textit{false} \mid \text{HEADACHE} = \textit{true})$

This can be calculated easily by counting:
$P(\text{VOMITING} = \textit{false} \mid \text{HEADACHE} = \textit{true}) = \frac{1}{7} = 0.1429$

(e) $P(\text{MENINGITIS} \mid \text{FEVER} = \textit{true}, \text{VOMITING} = \textit{false})$

This can be calculated easily by counting. First,
$P(\text{MENINGITIS} = \textit{true} \mid \text{FEVER} = \textit{true}, \text{VOMITING} = \textit{false}) = \frac{1}{4} = 0.25.$
Then,
$P(\text{MENINGITIS} = \textit{false} \mid \text{FEVER} = \textit{true}, \text{VOMITING} = \textit{false}) = \frac{3}{4} = 0.75$
So,
$\mathbf{P}(\text{MENINGITIS} \mid \text{FEVER} = \textit{true}, \text{VOMITING} = \textit{false}) = \langle 0.25, 0.75 \rangle$

3. Predictive data analytics models are often used as tools for process quality control and fault detection. The task in this question is to create a naive Bayes model to monitor a wastewater treatment plant.[1] The table below lists a dataset containing details of activities at a wastewater treatment plant for 14 days. Each day is described in terms

1. The dataset in this question is inspired by the Waste Water Treatment Dataset that is available from the UCI Machine Learning repository (Bache and Lichman, 2013) at archive.ics.uci.edu/ml/machine-learning-databases/water-treatment. The creators of this dataset reported their work in Bejar et al. (1991).

of six descriptive features that are generated from different sensors at the plant. SS-IN measures the solids coming into the plant per day; SED-IN measures the sediment coming into the plant per day; COND-IN measures the electrical conductivity of the water coming into the plant.[2] The features SS-OUT, SED-OUT, and COND-OUT are the corresponding measurements for the water flowing out of the plant. The target feature, STATUS, reports the current situation at the plant: *ok*, everything is working correctly; *settler*, there is a problem with the plant settler equipment; or *solids*, there is a problem with the amount of solids going through the plant.

ID	SS -IN	SED -IN	COND -IN	SS -OUT	SED -OUT	COND -OUT	STATUS
1	168	3	1,814	15	0.001	1,879	ok
2	156	3	1,358	14	0.01	1,425	ok
3	176	3.5	2,200	16	0.005	2,140	ok
4	256	3	2,070	27	0.2	2,700	ok
5	230	5	1,410	131	3.5	1,575	settler
6	116	3	1,238	104	0.06	1,221	settler
7	242	7	1,315	104	0.01	1,434	settler
8	242	4.5	1,183	78	0.02	1,374	settler
9	174	2.5	1,110	73	1.5	1,256	settler
10	1,004	35	1,218	81	1,172	33.3	solids
11	1,228	46	1,889	82.4	1,932	43.1	solids
12	964	17	2,120	20	1,030	1,966	solids
13	2,008	32	1,257	13	1,038	1,289	solids

(a) Create a naive Bayes model that uses probability density functions to model the descriptive features in this dataset (assume that all the descriptive features are normally distributed).

The prior probabilities of each of the target feature levels are

$$P(\text{STATUS} = ok) = \frac{4}{13} = 0.3077$$

$$P(\text{STATUS} = settler) = \frac{5}{13} = 0.3846$$

$$P(\text{STATUS} = ok) = \frac{4}{13} = 0.3077$$

2. The conductivity of water is affected by inorganic dissolved solids and organic compounds, such as oil. Consequently, water conductivity is a useful measure of water purity.

To create the probability density functions required by the model, we simply need to fit a normal distribution to each feature for each level of the target. To do this, we calculate the mean and standard deviation for each feature for the set of instances where the target takes a given value. The table below lists the normal probability distributions fitted to each descriptive feature and target level.

$P(\text{SS-IN} \mid ok)$	=	$N\left(x, \mu = 189, \sigma = 45.42\right)$
$P(\text{SED-IN} \mid ok)$	=	$N\left(x, \mu = 3.125, \sigma = 0.25\right)$
$P(\text{COND-IN} \mid ok)$	=	$N\left(x, \mu = 1{,}860.5, \sigma = 371.4\right)$
$P(\text{SS-OUT} \mid ok)$	=	$N\left(x, \mu = 18, \sigma = 6.06\right)$
$P(\text{SED-OUT} \mid ok)$	=	$N\left(x, \mu = 0.054, \sigma = 0.10\right)$
$P(\text{COND-OUT} \mid ok)$	=	$N\left(x, \mu = 2{,}036, \sigma = 532.19\right)$
$P(\text{SS-IN} \mid settler)$	=	$N\left(x, \mu = 200.8, \sigma = 55.13\right)$
$P(\text{SED-IN} \mid settler)$	=	$N\left(x, \mu = 4.4, \sigma = 1.78\right)$
$P(\text{COND-IN} \mid settler)$	=	$N\left(x, \mu = 1{,}251.2, \sigma = 116.24\right)$
$P(\text{SS-OUT} \mid settler)$	=	$N\left(x, \mu = 98, \sigma = 23.38\right)$
$P(\text{SED-OUT} \mid settler)$	=	$N\left(x, \mu = 1.018, \sigma = 1.53\right)$
$P(\text{COND-OUT} \mid settler)$	=	$N\left(x, \mu = 1{,}372, \sigma = 142.58\right)$
$P(\text{SS-IN} \mid solids)$	=	$N\left(x, \mu = 1{,}301, \sigma = 485.44\right)$
$P(\text{SED-IN} \mid solids)$	=	$N\left(x, \mu = 32.5, \sigma = 11.96\right)$
$P(\text{COND-IN} \mid solids)$	=	$N\left(x, \mu = 1{,}621, \sigma = 453.04\right)$
$P(\text{SS-OUT} \mid solids)$	=	$N\left(x, \mu = 49.1, \sigma = 37.76\right)$
$P(\text{SED-OUT} \mid solids)$	=	$N\left(x, \mu = 1{,}293, \sigma = 430.95\right)$
$P(\text{COND-OUT} \mid solids)$	=	$N\left(x, \mu = 832.85, \sigma = 958.31\right)$

(b) What prediction will the naive Bayes model return for the following query?

SS-IN = 222, SED-IN = 4.5, COND-IN = 1,518, SS-OUT = 74 SED-OUT = 0.25, COND-OUT = 1,642

The calculation for STATUS = *ok*:

$P(ok)$	=	0.3077		
$P(\text{SS-IN} \mid ok)$	=	$N\left(222, \mu = 189, \sigma = 45.42\right)$	=	0.0068
$P(\text{SED-IN} \mid ok)$	=	$N\left(4.5, \mu = 3.125, \sigma = 0.25\right)$	=	4.3079×10^{-7}
$P(\text{COND-IN} \mid ok)$	=	$N\left(1{,}518, \mu = 1{,}860.5, \sigma = 371.4\right)$	=	0.0007
$P(\text{SS-OUT} \mid ok)$	=	$N\left(74, \mu = 18, \sigma = 6.06\right)$	=	1.7650×10^{-20}
$P(\text{SED-OUT} \mid ok)$	=	$N\left(0.25, \mu = 0.054, \sigma = 0.10\right)$	=	0.5408
$P(\text{COND-OUT} \mid ok)$	=	$N\left(1{,}642, \mu = 2{,}036, \sigma = 532.19\right)$	=	0.0006

$$\left(\prod_{k=1}^{m} P(\mathbf{q}[k] \mid ok)\right) \times P(ok) = 3.41577 \times 10^{-36}$$

The calculation for STATUS = *settler*:

$P(settler)$	=	0.3846		
$P(\text{SS-IN} \mid settler)$	=	$N\left(222, \mu = 200.8, \sigma = 55.13\right)$	=	0.0067
$P(\text{SED-IN} \mid settler)$	=	$N\left(4.5, \mu = 4.4, \sigma = 1.78\right)$	=	0.2235
$P(\text{COND-IN} \mid settler)$	=	$N\left(1{,}518, \mu = 1{,}251.2, \sigma = 116.24\right)$	=	0.0002
$P(\text{SS-OUT} \mid settler)$	=	$N\left(74, \mu = 98, \sigma = 23.38\right)$	=	0.0101
$P(\text{SED-OUT} \mid settler)$	=	$N\left(0.25, \mu = 1.018, \sigma = 1.53\right)$	=	0.2303
$P(\text{COND-OUT} \mid settler)$	=	$N\left(1{,}642, \mu = 1{,}372, \sigma = 142.58\right)$	=	0.0005

$$\left(\prod_{k=1}^{m} P(\mathbf{q}[k] \mid settler)\right) \times P(settler) = 1.53837 \times 10^{-13}$$

The calculation for STATUS = *solids*:

$P(solids)$	=	0.3077		
$P(\text{SS-IN} \mid solids)$	=	$N\left(x, \mu = 1{,}301, \sigma = 485.44\right)$	=	6.9496×10^{-5}
$P(\text{SED-IN} \mid solids)$	=	$N\left(x, \mu = 32.5, \sigma = 11.96\right)$	=	0.0022
$P(\text{COND-IN} \mid solids)$	=	$N\left(x, \mu = 1{,}621, \sigma = 453.04\right)$	=	0.0009
$P(\text{SS-OUT} \mid solids)$	=	$N\left(x, \mu = 49.1, \sigma = 37.76\right)$	=	0.0085
$P(\text{SED-OUT} \mid solids)$	=	$N\left(x, \mu = 1{,}293, \sigma = 430.95\right)$	=	1.0291×10^{-5}
$P(\text{COND-OUT} \mid solids)$	=	$N\left(x, \mu = 832.85, \sigma = 958.31\right)$	=	0.0003

$$\left(\prod_{k=1}^{m} P(\mathbf{q}[k] \mid solids)\right) \times P(solids) = 1.00668 \times 10^{-21}$$

Recall that because we are using the heights of the PDFs rather than calculating the actual probabilities for each feature taking a value, the score of each target level is a relative ranking and should not be interpreted as a probability.

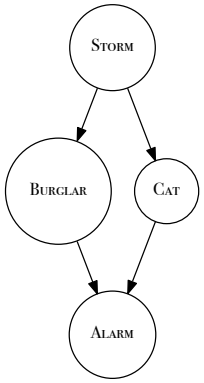
That said, the target level with the highest ranking is STATUS = *settler*. This indicates that there was a problem with the plant's settler equipment on the day of the query.

4. The following is a description of the causal relationship between storms, the behavior of burglars and cats, and house alarms:

Stormy nights are rare. Burglary is also rare, and if it is a stormy night, burglars are likely to stay at home (burglars don't like going out in storms). Cats don't like storms either, and if there is a storm, they like to go inside. The alarm on your house is designed to be triggered if a burglar breaks into your house, but sometimes it can be set off by your cat coming into the house, and sometimes it might not be triggered even if a burglar breaks in (it could be faulty or the burglar might be very good).

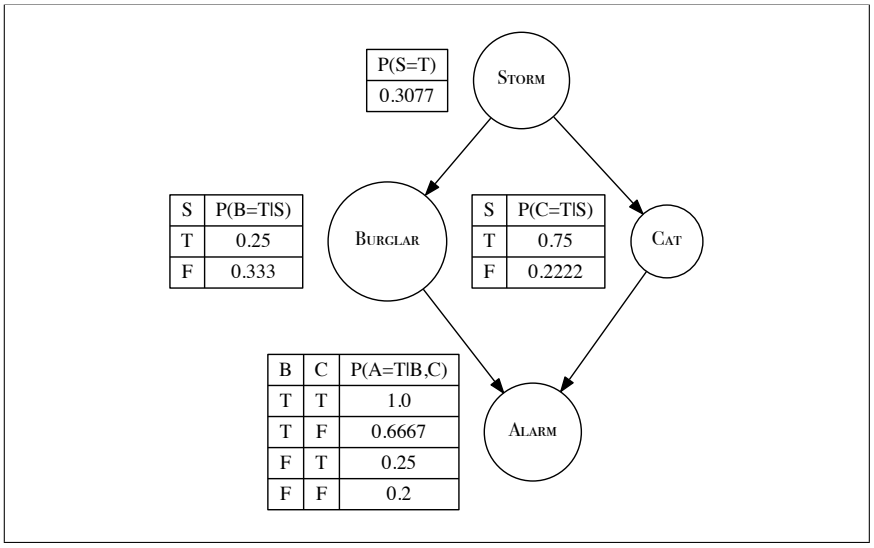
(a) Define the topology of a Bayesian network that encodes these causal relationships.

The figure below illustrates a Bayesian network that encodes the described causal relationships. Storms directly affect the behavior of burglars and cats, and this is reflected by links from the storm node to the burglar and cat nodes. The behavior of burglars and cats both affect whether the alarm goes off, and hence there are links from each of these nodes to the alarm node.



(b) The table below lists a set of instances from the house alarm domain. Using the data in this table, create the conditional probability tables (CPTs) for the network you created in Part (a) of this question.

ID	STORM	BURGLAR	CAT	ALARM
1	false	false	false	false
2	false	false	false	false
3	false	false	false	false
4	false	false	false	false
5	false	false	false	true
6	false	false	true	false
7	false	true	false	false
8	false	true	false	true
9	false	true	true	true
10	true	false	true	true
11	true	false	true	false
12	true	false	true	false
13	true	true	false	true



(c) What value will the Bayesian network predict for ALARM, given that there is both a burglar and a cat in the house but there is no storm?

Because both the parent nodes for ALARM are known, the probability distribution over ALARM is independent of the feature STORM. Consequently, we can read the relevant probability distribution over ALARM directly from the conditional probability table for the ALARM node. Examining the conditional probability table, we can see that when BURGLAR = *true*, and CAT =

true, then ALARM = *true* is the MAP prediction. In other words, the network would predict that the alarm would sound in this situation.

(d) What value will the Bayesian network predict for ALARM, given that there is a storm but we don't know if a burglar has broken in or where the cat is?

In this case, the values of the parents of the target feature are unknown. Consequently, we need to sum out both the parents for each value of the target. The network would calculate the probability of the event ALARM = *true* as follows:

$$
\begin{aligned}
P(a \mid s) = \frac{P(a,s)}{P(s)} &= \frac{\sum_{i,j} P(a, B_i, C_j, s)}{P(s)} \\
\sum_{i,j} P(a, B_i, C_j, s) &= \sum_{i,j} P(a \mid B_i, C_j) \times P(B_i \mid s) \times P(C_j \mid s) \times P(s) \\
&= (P(a \mid b, c) \times P(b \mid s) \times P(c \mid s) \times P(s)) \\
&+ (P(a \mid b, \neg c) \times P(b \mid s) \times P(\neg c \mid s) \times P(s)) \\
&+ (P(a \mid \neg b, c) \times P(\neg b \mid s) \times P(c \mid s) \times P(s)) \\
&+ (P(a \mid \neg b, \neg c) \times P(\neg b \mid s) \times P(\neg c \mid s) \times P(s)) \\
&= (1.0 \times 0.25 \times 0.75 \times 0.3077) + (0.6667 \times 0.25 \times 0.25 \times 0.3077) \\
&+ (0.25 \times 0.75 \times 0.75 \times 0.3077) + (0.2 \times 0.75 \times 0.25 \times 0.3077) \\
&= 0.125324 \\
P(a \mid s) = \frac{P(a,s)}{P(s)} &= \frac{0.125324}{0.3077} = 0.4073
\end{aligned}
$$

This implies that $P(\text{ALARM} = \textit{false}) = 0.5927$, so in this instance, ALARM = *false* is the MAP level for the target, and this is the prediction the model will return.

7 Error-Based Learning (Exercise Solutions)

1. A multivariate linear regression model has been built to predict the **heating load** in a residential building on the basis of a set of descriptive features describing the characteristics of the building. Heating load is the amount of heat energy required to keep a building at a specified temperature, usually 65° Fahrenheit during the winter regardless of outside temperature. The descriptive features used are the overall surface area of the building, the height of the building, the area of the building's roof, and the percentage of wall area in the building that is glazed. This kind of model would be useful to architects or engineers when designing a new building.[1] The trained model is

$$\begin{aligned}\text{HEATING LOAD} = &-26.030 + 0.0497 \times \text{SURFACE AREA} \\ &+ 4.942 \times \text{HEIGHT} - 0.090 \times \text{ROOF AREA} \\ &+ 20.523 \times \text{GLAZING AREA}\end{aligned}$$

Use this model to make predictions for each of the query instances shown in the following table.

ID	SURFACE AREA	HEIGHT	ROOF AREA	GLAZING AREA
1	784.0	3.5	220.5	0.25
2	710.5	3.0	210.5	0.10
3	563.5	7.0	122.5	0.40
4	637.0	6.0	147.0	0.60

1. This question is inspired by Tsanas and Xifara (2012), and although the data used is artificially generated, it is based on the Energy Efficiency Dataset available from the UCI Machine Learning Repository (Bache and Lichman, 2013) at archive.ics.uci.edu/ml/datasets/Energy+efficiency/.

Calculating the predictions made by the model simply involves inserting the descriptive features from each query instance into the prediction model.

1: $-26.030 + 0.0497 \times 784.0 + 4.942 \times 3.5 - 0.090 \times 220.5 + 20.523 \times 0.25 = 15.5$

2: $-26.030 + 0.0497 \times 710.5 + 4.942 \times 3.0 - 0.09 \times 210.5 + 20.523 \times 0.10 = 7.2$

3: $-26.03 + 0.0497 \times 563.5 + 4.942 \times 7.0 - 0.09 \times 122.5 + 20.523 \times 0.40 = 33.8$

4: $-26.03 + 0.0497 \times 637.0 + 4.942 \times 6.0 - 0.09 \times 147.0 + 20.523 \times 0.60 = 34.4$

2. You have been hired by the European Space Agency to build a model that predicts the amount of oxygen that an astronaut consumes when performing five minutes of intense physical work. The descriptive features for the model will be the age of the astronaut and their average heart rate throughout the work. The regression model is

$$\text{OXYCON} = \mathbf{w}[0] + \mathbf{w}[1] \times \text{AGE} + \mathbf{w}[2] \times \text{HEARTRATE}$$

The table that follows shows a historical dataset that has been collected for this task.

ID	OXYCON	AGE	HEART RATE	ID	OXYCON	AGE	HEART RATE
1	37.99	41	138	7	44.72	43	158
2	47.34	42	153	8	36.42	46	143
3	44.38	37	151	9	31.21	37	138
4	28.17	46	133	10	54.85	38	158
5	27.07	48	126	11	39.84	43	143
6	37.85	44	145	12	30.83	43	138

(a) Assuming that the current weights in a multivariate linear regression model are $\mathbf{w}[0] = -59.50$, $\mathbf{w}[1] = -0.15$, and $\mathbf{w}[2] = 0.60$, make a prediction for each training instance using this model.

The following table shows the predictions made using the given model weights.

ID	OXYCON	AGE	HEART RATE	Prediction
1	37.99	41	138	17.15
2	47.34	42	153	26.00
3	44.38	37	151	25.55
4	28.17	46	133	13.40
5	27.07	48	126	8.90
6	37.85	44	145	20.90
7	44.72	43	158	28.85
8	36.42	46	143	19.40
9	31.21	37	138	17.75
10	54.85	38	158	29.60
11	39.84	43	143	19.85
12	30.83	43	138	16.85

(b) Calculate the sum of squared errors for the set of predictions generated in Part (a).

The following table shows the predictions made by the model and sum of squared error calculation based on these predictions.

Initial Weights

$\mathbf{w}[0]$:	-59.50	$\mathbf{w}[1]$:	-0.15	$\mathbf{w}[2]$:	0.60

Iteration 1

ID	OXYCON	Prediction	Error	Squared Error	*errorDelta* $(\mathcal{D}, \mathbf{w}[0])$	*errorDelta* $(\mathcal{D}, \mathbf{w}[1])$	*errorDelta* $(\mathcal{D}, \mathbf{w}[2])$
1	37.99	17.15	20.84	434.41	20.84	854.54	2,876.26
2	47.34	26.00	21.34	455.41	21.34	896.29	3,265.05
3	44.38	25.55	18.83	354.60	18.83	696.74	2,843.45
4	28.17	13.40	14.77	218.27	14.77	679.60	1,964.93
5	27.07	8.90	18.17	330.09	18.17	872.08	2,289.20
6	37.85	20.90	16.95	287.35	16.95	745.86	2,457.94
7	44.72	28.85	15.87	251.91	15.87	682.48	2,507.71
8	36.42	19.40	17.02	289.72	17.02	782.98	2,434.04
9	31.21	17.75	13.46	181.26	13.46	498.14	1,857.92
10	54.85	29.60	25.25	637.57	25.25	959.50	3,989.52
11	39.84	19.85	19.99	399.47	19.99	859.44	2,858.12
12	30.83	16.85	13.98	195.52	13.98	601.25	1,929.61
			Sum	4,035.56	216.48	9,128.90	3,1273.77
	Sum of squared errors ($Sum/2$)			2,017.78			

(c) Assuming a learning rate of 0.000002, calculate the weights at the next iteration of the gradient descent algorithm.

To calculate the updated weight values we apply the weight update rule for multivariate linear regression with gradient descent for each weight as fol-

lows (using *errorDelta* values given in the answer to the previous part):

$$\begin{aligned}
\mathbf{w}[0] &\leftarrow \mathbf{w}[0] + \alpha \times errorDelta(\mathcal{D}, \mathbf{w}[0]) \\
&\leftarrow -59.50 + 0.000002 \times 216.48 \\
&\leftarrow -59.4996 \\
\mathbf{w}[1] &\leftarrow \mathbf{w}[1] + \alpha \times errorDelta(\mathcal{D}, \mathbf{w}[1]) \\
&\leftarrow -0.15 + 0.000002 \times 9128.9 \\
&\leftarrow -0.1317 \\
\mathbf{w}[2] &\leftarrow \mathbf{w}[2] + \alpha \times errorDelta(\mathcal{D}, \mathbf{w}[2]) \\
&\leftarrow 0.60 + 0.000002 \times 31273.77 \\
&\leftarrow 0.6625
\end{aligned}$$

(d) Calculate the sum of squared errors for a set of predictions generated using the new set of weights calculated in Part (c).

The new sum of squared errors calculated using these new weights is given by the following table.

ID	OXYCON	Prediction	Error	Squared Error
1	37.99	26.53	11.46	131.38
2	47.34	36.34	11.00	121.07
3	44.38	35.67	8.71	75.87
4	28.17	22.56	5.61	31.53
5	27.07	17.66	9.41	88.56
6	37.85	30.77	7.08	50.10
7	44.72	39.52	5.20	27.08
8	36.42	29.18	7.24	52.37
9	31.21	27.06	4.16	17.27
10	54.85	40.18	14.67	215.31
11	39.84	29.58	10.26	105.21
12	30.83	26.27	4.57	20.84
			Sum	936.57
	Sum of squared errors ($Sum/2$)			468.29

3. A multivariate logistic regression model has been built to predict the propensity of shoppers to perform a repeat purchase of a free gift that they are given. The descriptive features used by the model are the age of the customer, the socioeconomic band to which the customer belongs (*a*, *b*, or *c*), the average amount of money the customer spends on each visit to the shop, and the average number of visits the customer makes

to the shop per week. This model is being used by the marketing department to determine who should be given the free gift. The weights in the trained model are shown in the following table.

Feature	Weight
Intercept (**w** [0])	-3.82398
AGE	-0.02990
SOCIOECONOMIC BAND B	-0.09089
SOCIOECONOMIC BAND C	-0.19558
SHOP VALUE	0.02999
SHOP FREQUENCY	0.74572

Use this model to make predictions for each of the following query instances.

ID	AGE	SOCIOECONOMIC BAND	SHOP FREQUENCY	SHOP VALUE
1	56	b	1.60	109.32
2	21	c	4.92	11.28
3	48	b	1.21	161.19
4	37	c	0.72	170.65
5	32	a	1.08	165.39

Calculating the predictions made by the model simply involves inserting the descriptive features from each query instance into the prediction model. The only extra thing that must be considered in this case is the categorical descriptive feature SOCIOECONOMIC BAND. We can note from the regression equation that this one feature has been expanded into two: SOCIOECONOMIC BAND B and SOCIOECONOMIC BAND C. These are binary features, indicating that the original feature was set to the level *b* or *c*. It is assumed that when both of these features are set to 0, then the original feature was set to *a* (the choice of which level to leave out is arbitrary). The other pieces of information required are that the *yes* level is the positive level, and the classification threshold is 0.5.
With this information, the predictions can be made as follows:

1: $Logistic(-3.82398 + -0.0299 \times 56 + -0.09089 \times 1 + -0.19558 \times 0 + 0.74572 \times 1.6 + 0.02999 \times 109.32)$
$= Logistic(-1.12) = \frac{1}{1+e^{1.12}}$
$= 0.25 \Rightarrow no$

2: $Logistic(-3.82398 + -0.0299 \times 21 + -0.09089 \times 0 + -0.19558 \times 1 + 0.74572 \times 4.92 + 0.02999 \times 11.28)$

$= Logistic(-0.64) = \frac{1}{1+e^{0.64}}$
$= 0.35 \Rightarrow no$

3: $Logistic(-3.82398 + -0.0299 \times 48 + -0.09089 \times 1 + -0.19558 \times 0 + 0.74572 \times 1.21 + 0.02999 \times 161.19)$
$= Logistic(0.39) = \frac{1}{1+e^{-0.39}}$
$= 0.60 \Rightarrow yes$

4: $Logistic(-3.82398 + -0.0299 \times 37 + -0.09089 \times 0 + -0.19558 \times 1 + 0.74572 \times 0.72 + 0.02999 \times 170.65)$
$= Logistic(0.53) = \frac{1}{1+e^{-0.53}}$
$= 0.63 \Rightarrow yes$

5: $Logistic(-3.82398 + -0.0299 \times 32 + -0.09089 \times 0 + -0.19558 \times 0 + 0.74572 \times 1.08 + 0.02999 \times 165.39)$
$= Logistic(0.98) = \frac{1}{1+e^{-0.98}}$
$= 0.73 \Rightarrow yes$

4. The use of the **kernel trick** is key in writing efficient implementations of the **support vector machine** approach to predictive modelling. The kernel trick is based on the fact that the result of a **kernel function** applied to a support vector and a query instance is equivalent to the result of calculating the dot product between the support vector and the query instance after a specific set of basis functions have been applied to both—in other words, $kernel(\mathbf{d}, \mathbf{q}) = \boldsymbol{\phi}(\mathbf{d}) \cdot \boldsymbol{\phi}(\mathbf{q})$.

   (a) Using the support vector $\langle \mathbf{d}[1], \mathbf{d}[2] \rangle$ and the query instance $\langle \mathbf{q}[1], \mathbf{q}[2] \rangle$ as examples, show that applying a polynomial kernel with $p = 2$, $kernel(\mathbf{d}, \mathbf{q}) = (\mathbf{d} \cdot \mathbf{q} + 1)^2$, is equivalent to calculating the dot product of the support vector and query instance after applying the following set of basis functions:

$$\phi_0(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = \mathbf{d}[1]^2 \qquad \phi_1(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = \mathbf{d}[2]^2$$
$$\phi_2(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = \sqrt{2} \times \mathbf{d}[1] \times \mathbf{d}[2] \qquad \phi_3(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = \sqrt{2} \times \mathbf{d}[1]$$
$$\phi_4(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = \sqrt{2} \times \mathbf{d}[2] \qquad \phi_5(\langle \mathbf{d}[1], \mathbf{d}[2] \rangle) = 1$$

To answer this question we should first calculate the result of applying the polynomial kernel function to the support vector and query instance:

$$\begin{aligned}
kernel(\mathbf{d},\mathbf{q}) &= (\mathbf{d}\cdot\mathbf{q}+1)^2 \\
&= (\langle \mathbf{d}[1],\mathbf{d}[2]\rangle \cdot \langle \mathbf{q}[1],\mathbf{q}[2]\rangle + 1)^2 \\
&= (\mathbf{d}[1]\times\mathbf{q}[1]+\mathbf{d}[2]\times\mathbf{q}[2]+1)^2 \\
&= (\mathbf{d}[1]\times\mathbf{q}[1]+\mathbf{d}[2]\times\mathbf{q}[2]+1)\times(\mathbf{d}[1]\times\mathbf{q}[1]+\mathbf{d}[2]\times\mathbf{q}[2]+1) \\
&= (\mathbf{d}[1]\times\mathbf{q}[1])^2 + (\mathbf{d}[1]\times\mathbf{q}[1]\times\mathbf{d}[2]\times\mathbf{q}[2]) + (\mathbf{d}[1]\times\mathbf{q}[1]) \\
&\quad + (\mathbf{d}[2]\times\mathbf{q}[2]\times\mathbf{d}[1]\times\mathbf{q}[1]) + (\mathbf{d}[2]\times\mathbf{q}[2])^2 + (\mathbf{d}[2]\times\mathbf{q}[2]) \\
&\quad + (\mathbf{d}[1]\times\mathbf{q}[1]) + (\mathbf{d}[2]\times\mathbf{q}[2]) + 1 \\
&= (\mathbf{d}[1]\times\mathbf{q}[1])^2 + (\mathbf{d}[2]\times\mathbf{q}[2])^2 + 2\times(\mathbf{d}[1]\times\mathbf{q}[1]\times\mathbf{d}[2]\times\mathbf{q}[2]) \\
&\quad + 2\times(\mathbf{d}[1]\times\mathbf{q}[1]) + 2\times(\mathbf{d}[2]\times\mathbf{q}[2]) + 1
\end{aligned}$$

We then apply the set of basis functions to the support vector

$$\begin{aligned}
\boldsymbol{\phi}(\mathbf{d}) &= \langle \phi_0(\mathbf{d}),\phi_1(\mathbf{d}),\phi_2(\mathbf{d}),\phi_3(\mathbf{d}),\phi_4(\mathbf{d}),\phi_5(\mathbf{d})\rangle \\
&= \left\langle \mathbf{d}[1]^2, \mathbf{d}[2]^2, \sqrt{2}\times\mathbf{d}[1]\times\mathbf{d}[2], \sqrt{2}\times\mathbf{d}[1], \sqrt{2}\times\mathbf{d}[2], 1\right\rangle
\end{aligned}$$

and to the query instance:

$$\begin{aligned}
\boldsymbol{\phi}(\mathbf{q}) &= \langle \phi_0(\mathbf{q}),\phi_1(\mathbf{q}),\phi_2(\mathbf{q}),\phi_3(\mathbf{q}),\phi_4(\mathbf{q}),\phi_5(\mathbf{q})\rangle \\
&= \left\langle \mathbf{q}[1]^2, \mathbf{q}[2]^2, \sqrt{2}\times\mathbf{q}[1]\times\mathbf{q}[2], \sqrt{2}\times\mathbf{q}[1], \sqrt{2}\times\mathbf{q}[2], 1\right\rangle
\end{aligned}$$

we then calculate the dot product between the transformed support vector and query instance as:

$$\begin{aligned}
\boldsymbol{\phi}(\mathbf{d})\cdot\boldsymbol{\phi}(\mathbf{q}) &= \left\langle \mathbf{d}[1]^2, \mathbf{d}[2]^2, \sqrt{2}\times\mathbf{d}[1]\times\mathbf{d}[2], \sqrt{2}\times\mathbf{d}[1], \sqrt{2}\times\mathbf{d}[2], 1\right\rangle \\
&\quad \cdot \left\langle \mathbf{q}[1]^2, \mathbf{q}[2]^2, \sqrt{2}\times\mathbf{q}[1]\times\mathbf{q}[2], \sqrt{2}\times\mathbf{q}[1], \sqrt{2}\times\mathbf{q}[2], 1\right\rangle \\
&= \mathbf{d}[1]^2\times\mathbf{q}[1]^2 + \mathbf{d}[2]^2\times\mathbf{q}[2]^2 + \sqrt{2}\times\mathbf{d}[1]\times\mathbf{d}[2]\times\sqrt{2}\times\mathbf{q}[1]\times\mathbf{q}[2] \\
&\quad + \sqrt{2}\times\mathbf{d}[1]\times\sqrt{2}\times\mathbf{q}[1] + \sqrt{2}\times\mathbf{d}[2]\times\sqrt{2}\times\mathbf{q}[2] + 1\times 1 \\
&= (\mathbf{d}[1]\times\mathbf{q}[1])^2 + (\mathbf{d}[2]\times\mathbf{q}[2])^2 + 2\times(\mathbf{d}[1]\times\mathbf{q}[1]\times\mathbf{d}[2]\times\mathbf{q}[2]) \\
&\quad + 2\times(\mathbf{d}[1]\times\mathbf{q}[1]) + 2\times(\mathbf{d}[2]\times\mathbf{q}[2]) + 1
\end{aligned}$$

This is equivalent to the the result of applying the polynomial kernel function calculated above which demonstrates that these two calculations are equivalent–in other words $kernel(\mathbf{d},\mathbf{q}) = \boldsymbol{\phi}(\mathbf{d})\cdot\boldsymbol{\phi}(\mathbf{q})$.

(b) A support vector machine model has been trained to distinguish between dosages of two drugs that cause a dangerous interaction and those that interact safely. This model uses just two continuous features, DOSE1 and DOSE2, and two target levels, *dangerous* (the positive level, $+1$) and *safe* (the negative level, -1). The support vectors in the trained model are shown in the following table.

DOSE1	DOSE2	CLASS
0.2351	0.4016	+1
-0.1764	-0.1916	+1
0.3057	-0.9394	-1
0.5590	0.6353	-1
-0.6600	-0.1175	-1

In the trained model the value of w_0 is 0.3074, and the values of the $\boldsymbol{\alpha}$ parameters are $\langle 7.1655, 6.9060, 2.0033, 6.1144, 5.9538\rangle$.

i. Using the version of the support vector machine prediction model that uses basis functions (see Equation 7.46) with the basis functions given in Part (a), calculate the output of the model for a query instance with DOSE1 $= 0.90$ and DOSE2 $= -0.90$.

The first step in this calculation is to transform the support vectors using the set of basis functions

$$\boldsymbol{\phi}(\langle 0.2351, 0.4016\rangle) = \langle 0.0553, 0.1613, 0.1335, 0.3325, 0.5679, 1.0\rangle$$
$$\boldsymbol{\phi}(\langle -0.1764, -0.1916\rangle) = \langle 0.0311, 0.0367, 0.0478, -0.2495, -0.2710, 1.0\rangle$$
$$\boldsymbol{\phi}(\langle 0.3057, -0.9394\rangle) = \langle 0.0935, 0.8825, -0.4061, 0.4323, -1.3285, 1.0\rangle$$
$$\boldsymbol{\phi}(\langle 0.5590, 0.6353\rangle) = \langle 0.3125, 0.4036, 0.5022, 0.7905, 0.8984, 1.0\rangle$$
$$\boldsymbol{\phi}(\langle -0.6600, -0.1175\rangle) = \langle 0.4356, 0.0138, 0.1097, -0.9334, -0.1662, 1.0\rangle$$

The query instance then also needs to be transformed

$$\boldsymbol{\phi}(\langle 0.91, -0.93\rangle) = \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle$$

The output of the support vector machine can then be calculated as:

$$
\begin{aligned}
&\mathbb{M}_{\boldsymbol{\alpha},\boldsymbol{\phi},0.3074}(\langle 0.91, -0.93\rangle) \\
&\quad = (-1 \times 7.1655 \times (\langle 0.0553, 0.1613, 0.1335, 0.3325, 0.5679, 1.0\rangle \\
&\qquad \cdot \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 6.9060 \times (\langle 0.0311, 0.0367, 0.0478, -0.2495, -0.2710, 1.0\rangle \\
&\qquad \cdot \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 2.0033 \times (\langle 0.0935, 0.8825, -0.4061, 0.4323, -1.3285, 1.0\rangle \\
&\qquad \cdot \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 6.1144 \times (\langle 0.3125, 0.4036, 0.5022, 0.7905, 0.8984, 1.0\rangle \\
&\qquad \cdot \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 5.9538 \times (\langle 0.4356, 0.0138, 0.1097, -0.9334, -0.1662, 1.0\rangle \\
&\qquad \cdot \langle 0.8281, 0.8649, -1.1968, 1.2869, -1.3152, 1.0\rangle) + 0.3074) \\
&\quad = 5.3689 + 7.4596 - 8.9686 - 4.8438 - 1.2331 \\
&\quad = -2.2170
\end{aligned}
$$

Because the output of the model is negative the model makes a prediction of the negative level—*safe*.

ii. Using the version of the support vector machine prediction model that uses a kernel function (see Equation 7.47) with the polynomial kernel function, calculate the output of the model for a query instance with $\text{DOSE1} = 0.22$ and $\text{DOSE2} = 0.16$.

The output of the model can be calculated as

$$
\begin{aligned}
&\mathbb{M}_{\boldsymbol{\alpha},\boldsymbol{\phi},0.3074}(\langle 0.22, 0.16\rangle) \\
&\quad = \Big(-1 \times 7.1655 \times (\langle 0.2351, 0.4016\rangle \cdot \langle 0.22, 0.16\rangle + 1)^2 + 0.3074\Big) \\
&\qquad + \Big(1 \times 6.9060 \times (\langle -0.1764, -0.1916\rangle \cdot \langle 0.22, 0.16\rangle + 1)^2 + 0.3074\Big) \\
&\qquad + \Big(1 \times 2.0033 \times (\langle 0.3057, -0.9394\rangle \cdot \langle 0.22, 0.16\rangle + 1)^2 + 0.3074\Big) \\
&\qquad + \Big(1 \times 6.1144 \times (\langle 0.559, 0.6353\rangle \cdot \langle 0.22, 0.16\rangle + 1)^2 + 0.3074\Big) \\
&\qquad + \Big(1 \times 5.9538 \times (\langle -0.66, -0.1175\rangle \cdot \langle 0.22, 0.16\rangle + 1)^2 + 0.3074\Big) \\
&\quad = 9.2314 + 6.2873 - 1.3769 - 8.8624 - 3.8536 \\
&\quad = 1.4257
\end{aligned}
$$

Because the output of the model is positive, the model predicts the positive level—*dangerous*.

iii. Verify that the answers calculated in Parts (i) and (ii) of this question would have been the same if the alternative approach (basis functions or the polynomial kernel function) had been used in each case.

First we will calculate the output of the model for the query instance $\langle 0.91, -0.93\rangle$ using the polynomial kernel

$$\begin{aligned}
&\mathbb{M}_{\boldsymbol{\alpha},\boldsymbol{\phi},0.3074}(\langle 0.91, -0.93\rangle)\\
&\quad = \Big(-1 \times 7.1655 \times (\langle 0.2351, 0.4016\rangle \cdot \langle 0.91, -0.93\rangle + 1)^2 + 0.3074\Big)\\
&\qquad + \Big(1 \times 6.9060 \times (\langle -0.1764, -0.1916\rangle \cdot \langle 0.91, -0.93\rangle + 1)^2 + 0.3074\Big)\\
&\qquad + \Big(1 \times 2.0033 \times (\langle 0.3057, -0.9394\rangle \cdot \langle 0.91, -0.93\rangle + 1)^2 + 0.3074\Big)\\
&\qquad + \Big(1 \times 6.1144 \times (\langle 0.559, 0.6353\rangle \cdot \langle 0.91, -0.93\rangle + 1)^2 + 0.3074\Big)\\
&\qquad + \Big(1 \times 5.9538 \times (\langle -0.66, -0.1175\rangle \cdot \langle 0.91, -0.93\rangle + 1)^2 + 0.3074\Big)\\
&\quad = 5.3689 + 7.4596 - 8.9686 - 4.8438 - 1.2331\\
&\quad = -2.2170
\end{aligned}$$

This is the same result calculated previously, and would lead to the same prediction.

Next we will calculate the output of the model for the query instance $\langle 0.22, 0.16\rangle$ using the set of basis functions. To this, first the query instance needs to be transformed using the basis functions

$$\boldsymbol{\phi}(\langle 0.22, 0.16\rangle) = \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle$$

The output of the support vector machine can then be calculated as:

$$
\begin{aligned}
&\mathbb{M}_{\boldsymbol{\alpha},\boldsymbol{\phi},0.3074}(\langle 0.22, 0.16\rangle) \\
&\quad = (-1 \times 7.1655 \times (\langle 0.0553, 0.1613, 0.1335, 0.3325, 0.5679, 1.0\rangle \\
&\qquad \cdot \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 6.9060 \times (\langle 0.0311, 0.0367, 0.0478, -0.2495, -0.2710, 1.0\rangle \\
&\qquad \cdot \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 2.0033 \times (\langle 0.0935, 0.8825, -0.4061, 0.4323, -1.3285, 1.0\rangle \\
&\qquad \cdot \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 6.1144 \times (\langle 0.3125, 0.4036, 0.5022, 0.7905, 0.8984, 1.0\rangle \\
&\qquad \cdot \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle) + 0.3074) \\
&\qquad + (1 \times 5.9538 \times (\langle 0.4356, 0.0138, 0.1097, -0.9334, -0.1662, 1.0\rangle \\
&\qquad \cdot \langle 0.0484, 0.0256, 0.0498, 0.3111, 0.2263, 1.0\rangle) + 0.3074) \\
&\quad = 9.2314 + 6.2873 - 1.3769 - 8.8624 - 3.8536 \\
&\quad = 1.4257
\end{aligned}
$$

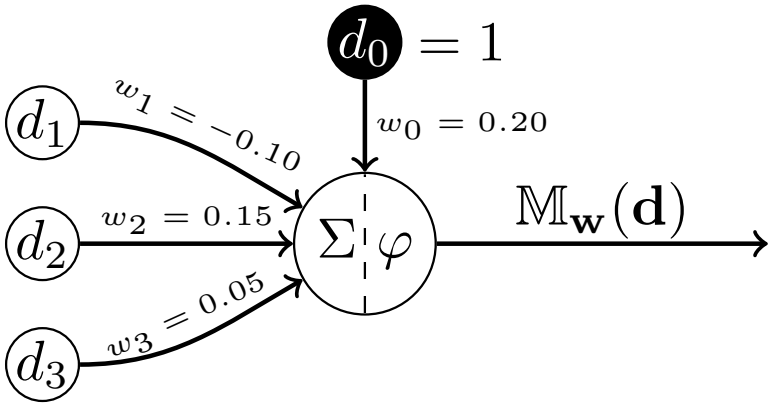
Again, this output is the same as that calculated previously using the polynomial kernel function.

iv. Compare the amount of computation required to calculate the output of the support vector machine using the polynomial kernel function with the amount required to calculate the output of the support vector machine using the basis functions.

It is clear, even in this small example, that the calculation using the polynomial kernel function is much more efficient than the calculation using the basis function transformation. Making the transformation using the basis functions and calculating the dot product in this higher dimensional space takes much more computational effort than calculating the polynomial kernel function. This is the advantage of the kernel trick.

8 Deep Learning (Exercise Solutions)

1. The following image shows an artificial neuron that takes three inputs



(a) Calculate the **weighted sum** for this neuron for the input vector

$$\mathbf{d} = [0.2, 0.5, 0.7]$$

$$\begin{aligned} z &= (1 \times w_0) + (0.2 \times w_1) + (0.5 \times w_2) + (0.7 \times w_3) \\ &= (1 \times 0.2) + (0.2 \times -0.1) + (0.5 \times 0.15) + (0.7 \times 0.05) \\ &= 0.2 + -0.02 + 0.075 + 0.035 \\ &= 0.29 \end{aligned}$$

(b) What would be the output from this neuron if the activation function φ is a threshold activation with $\theta = 1$?

Using a threshold activation function the weighted sum z is mapped to an output as follows:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } z \geqslant \theta \\ 0 & \textit{otherwise} \end{cases}$$

In this instance $z = 0.29$ and $\theta = 1$ and so $\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = 0$

(c) What would be the output from this neuron if the activation function φ is the logistic function?

The logistic function is defined as:

$$\textit{logistic}(z) = \frac{1}{1 + e^{-z}}$$

For $z = 0.29$ the result of this calculation is: 0.571996133

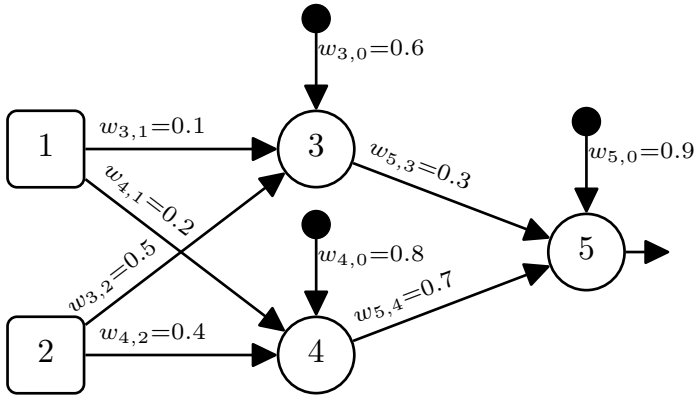
(d) What would be the output from this neuron if the activation function φ is the rectified linear function?

The rectified linear function is defined as:

$$\textit{rectifier}(z) = \textit{max}(0, z)$$

For $z = 0.29$ the output of the rectified linear function is 0.29.

2. The following image shows an artificial neural network with two sensing neurons (Neurons 1 and 2) and 3 processing neurons (Neurons 3, 4, and 5)



(a) Assuming that the processing neurons in this network use a **logistic** activation function, what would be the output of Neuron 5 if the network received the input vector: Neuron 1 = 0.7 and Neuron 2 = 0.3?

The weighted sum calculation for Neuron 3 is:

$$\begin{aligned} z_3 &= (0.6 \times 1) + (0.1 \times 0.7) + (0.5 \times 0.3) \\ &= (0.6) + (0.07) + (0.15) \\ &= 0.82 \end{aligned}$$

The weighted sum calculation for Neuron 4 is:

$$\begin{aligned} z_4 &= (0.8 \times 1) + (0.2 \times 0.7) + (0.4 \times 0.3) \\ &= (0.8) + (0.14) + (0.12) \\ &= 1.06 \end{aligned}$$

Using a logistic activation function the activation for Neuron 3 is:

$$\begin{aligned} a_3 &= \mathit{logistic}(z_3) \\ &= \frac{1}{1 + e^{-0.82}} \\ &= 0.69423634 \end{aligned}$$

Using a logistic activation function the activation for Neuron 4 is:

$$\begin{aligned} a_4 &= \mathit{logistic}(z_4) \\ &= \frac{1}{1 + e^{-1.06}} \\ &= 0.742690545 \end{aligned}$$

The weighted sum calculation for Neuron 5 can now be calculated as:

$$\begin{aligned} z_5 &= (0.9 \times 1) + (0.3 \times 0.69423634) + (0.7 \times 0.742690545) \\ &= (0.9) + (0.208270902) + (0.519883382) \\ &= 1.628154284 \end{aligned}$$

And the final output of the network can now be calculated by passing z_5 through the activation function for Neuron 5, which for this question is the

logistic function:

$$\begin{aligned} a_5 &= logistic(z_5) \\ &= \frac{1}{1+e^{-1.628154284}} \\ &= 0.835916637 \end{aligned}$$

(b) Assuming that the processing neurons in this network use a **ReLU** activation function, what would be the output of Neuron 5 if the network received the input vector: Neuron 1 = 0.7 and Neuron 2 = 0.3?

From the solutions to part one of this question the weighted sum calculations for Neurons 3 and 4 are as follows:

$$\begin{aligned} z_3 &= 0.82 \\ z_4 &= 1.06 \end{aligned}$$

Both of these values are greater than 0 and so using the rectifier function ($max(0,z)$) as an activation function the activations for each of these neurons is:

$$\begin{aligned} a_3 &= 0.82 \\ a_4 &= 1.06 \end{aligned}$$

The weighted sum calculation for Neuron 5 can now be calculated as:

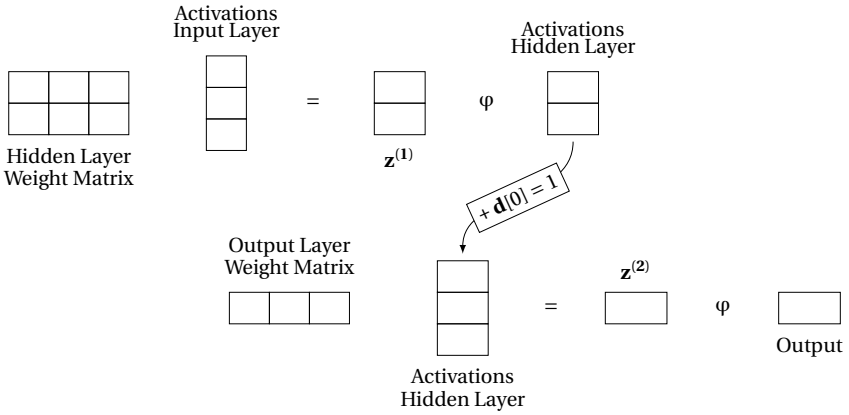
$$\begin{aligned} z_5 &= (0.9 \times 1) + (0.3 \times 0.82) + (0.7 \times 1.06) \\ &= (0.9) + (0.246) + (0.742) \\ &= 1.888 \end{aligned}$$

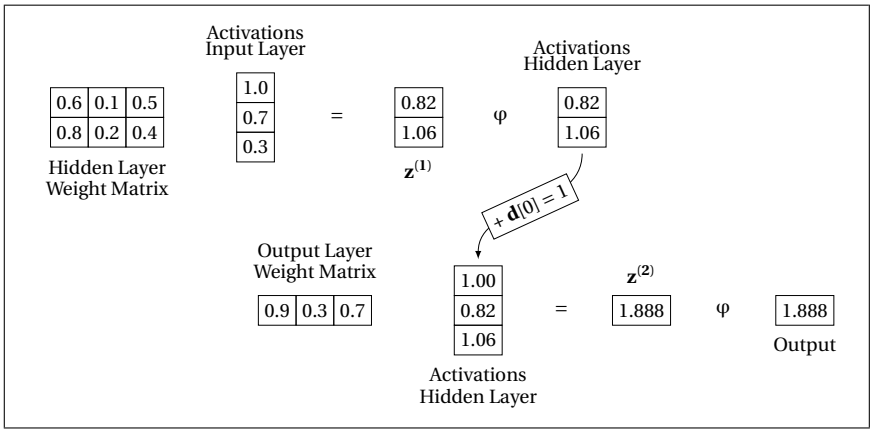
z_5 is greater than 0 and so passing this value through the recitifier activation function ($max(0,z)$) does not change it, and so the activation for Neuron 5 (and the output of the network) is:

$$a_5 = 1.888$$

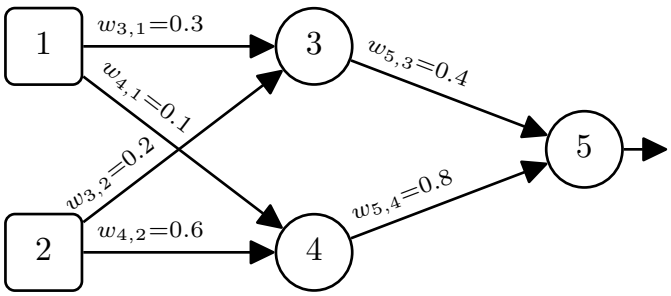
(c) The following image provides a template diagram for the sequence of matrix operations that our neural network would use to process the input vector Neuron 1 =

0.7 and Neuron 2 = 0.3. Assuming that the processing neurons in the network use a ReLU activation function, fill in the diagram with the appropriate weights, bias terms, weighted sum values, and activations.





3. The following image shows an artificial network with two layers of **linear neurons** (i.e., neurons that have no activation function and whose output is simply the result of the weighted sum of their inputs). Furthermore, these neurons have no bias terms.



(a) Calculate the output for Neuron 5 for the input vector: Neuron 1 = 0.9, Neuron 2 = 0.5.

Weighted sum for Neuron 3:

$$\begin{aligned} a_3 &= ((0.9 \times 0.3) + (0.5 \times 0.2)) \\ &= 0.37 \end{aligned}$$

Weighted sum for Neuron 4:

$$\begin{aligned} a_4 &= ((0.9 \times 0.1) + (0.5 \times 0.6)) \\ &= 0.39 \end{aligned}$$

Weighted sum, and activation, for Neuron 5:

$$\begin{aligned} a_4 &= ((0.37 \times 0.4) + (0.39 \times 0.8)) \\ &= 0.46 \end{aligned}$$

(b) Calculate the weight matrix for the single layer network that would implement the equivalent mapping that this two-layer network implements.

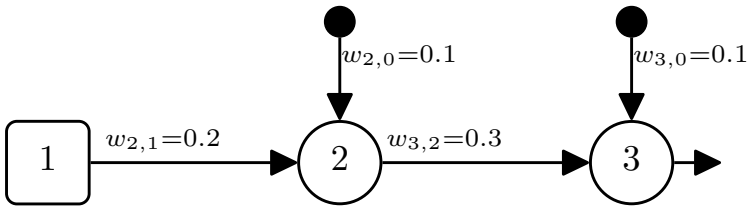
$$\begin{aligned} \mathbf{W}' &= \mathbf{W}_2 \cdot \mathbf{W}_1 \\ &= \begin{bmatrix} 0.4 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.3 & 0.2 \\ 0.1 & 0.6 \end{bmatrix} \\ &= \begin{bmatrix} 0.2 & 0.56 \end{bmatrix} \end{aligned}$$

(c) Show that the single-layer network using the weight matrix you calculated in Part 2 generates the same output as the network for the input vector: Neuron 1 = 0.9,

Neuron 2 = 0.5.

$$
\begin{aligned}
activation &= \begin{bmatrix} 0.2 & 0.56 \end{bmatrix} \cdot \begin{bmatrix} 0.9 \\ 0.5 \end{bmatrix} \\
&= ((0.2 \times 0.9) + (0.56 \times 0.5)) \\
&= 0.46
\end{aligned}
$$

4. The following image illustrates the topology of a simple feedforward neural network that has a single sensing neuron (Neuron 1), a single hidden processing neuron (Neuron 2), and a single processing output neuron (Neuron 3).



   (a) Assuming that the processing neurons use **logistic** activation functions, that the input to the network is Neuron 1 = 0.2 and that the desired output for this input is 0.7:

      i. Calculate the output generated by the network in response to this input.

$$
\begin{aligned}
z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.2)) \\
&= ((0.1 \times 1) + (0.2 \times 0.2)) \\
&= 0.14
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= logistic(0.14) \\
&= 0.534942945
\end{aligned}
$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.3 \times 0.534942945)) \\
&= 0.260482884
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= logistic(0.260482884) \\
&= 0.564754992
\end{aligned}
$$

ii. Calculate the δ values for each of the neurons in the network (i.e., δ_3, δ_2).

Calculations for δ_3:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_3} &= -(t_k - a_k) \\ &= -(0.7 - 0.564754992) \\ &= -0.135245008\end{aligned}$$

$$\begin{aligned}\frac{\partial a_3}{\partial z_3} &= logistic(z_3) \times (1 - logistic(z_3)) \\ &= 0.564754992 \times (1 - 0.564754992) \\ &= 0.245806791\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= -0.135245008 \times 0.245806791 \\ &= -0.033244142\end{aligned}$$

Calculations for δ_2:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_2} &= w_{3,2} \times \delta_3 \\ &= 0.3 \times -0.033244142 \\ &= -0.009973242\end{aligned}$$

$$\begin{aligned}\frac{\partial a_2}{\partial z_2} &= logistic(z_2) \times (1 - logistic(z_2)) \\ &= 0.534942945 \times (1 - 0.534942945) \\ &= 0.248778991\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{\partial \mathcal{E}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \\ &= -0.009973242 \times 0.248778991 \\ &= -0.002481133\end{aligned}$$

iii. Using the δ values you calculated above, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial \mathcal{E}/\partial w_{3,2}$,

$\partial\mathcal{E}/\partial w_{3,0}$, $\partial\mathcal{E}/\partial w_{2,1}$, $\partial\mathcal{E}/\partial w_{2,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{3,2}} &= \delta_3 \times a_2 \\ &= -0.033244142 \times 0.534942945 \\ &= -0.017783719\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{3,0}} &= \delta_3 \times a_0 \\ &= -0.033244142 \times 1 \\ &= -0.033244142\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{2,1}} &= \delta_2 \times a_1 \\ &= -0.002481133 \times 0.2 \\ &= -0.000496227\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{2,0}} &= \delta_2 \times a_0 \\ &= -0.002481133 \times 1 \\ &= -0.002481133\end{aligned}$$

iv. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{3,2}, w_{3,0,}, w_{2,1}, w_{2,0,}$) after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial\mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$\begin{aligned} w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\ &= 0.3 - 0.1 \times -0.017783719 \\ &= 0.301778372 \end{aligned}$$

$$\begin{aligned} w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\ &= 0.1 - 0.1 \times -0.033244142 \\ &= 0.103324414 \end{aligned}$$

$$\begin{aligned} w_{2,1}^{t+1} &= w_{2,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,1}} \\ &= 0.2 - 0.1 \times -0.000496227 \\ &= 0.200049623 \end{aligned}$$

$$\begin{aligned} w_{2,0}^{t+1} &= w_{2,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,0}} \\ &= 0.1 - 0.1 \times -0.002481133 \\ &= 0.100248113 \end{aligned}$$

v. Calculate the reduction in the error of the network for this example using the new weights, compared with using the original weights.

The output for the network using the new weights is:

$$\begin{aligned} z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.2)) \\ &= ((0.100248113 \times 1) + (0.200049623 \times 0.2)) \\ &= 0.140258038 \end{aligned}$$

$$\begin{aligned} a_2 &= \mathit{logistic}(0.140258038) \\ &= 0.535007139 \end{aligned}$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.103324414 \times 1) + (0.301778372 \times 0.535007139)) \\
&= 0.264777998
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= logistic(0.264777998) \\
&= 0.565810465
\end{aligned}
$$

The error with the new weights is:

$$
\begin{aligned}
Error' &= 0.7 - 0.565810465 \\
&0.134189535
\end{aligned}
$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$
\begin{aligned}
Error\ Reduction &= 0.135245008 - 0.134189535 \\
&= 0.001055473
\end{aligned}
$$

(b) Assuming that the processing neurons are **ReLU**s, that the input to the network is Neuron 1 = 0.2, and that the desired output for this input is 0.7:

i. Calculate the output generated by the network in response to this input.

$$
\begin{aligned}
z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.2)) \\
&= ((0.1 \times 1) + (0.2 \times 0.2)) \\
&= 0.14
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= rectifier(0.14) \\
&= max(0, 0.14) \\
&= 0.14
\end{aligned}
$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.3 \times 0.14)) \\
&= 0.142
\end{aligned}
$$

$$\begin{aligned} a_3 &= rectifier(0.142) \\ &= max(0, 0.142) \\ &= 0.142 \end{aligned}$$

ii. Calculate the δ values for each of the neurons in the network (i.e., δ_3, δ_2).

Calculations for δ_3:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial a_3} &= -(t_k - a_k) \\ &= -(0.7 - 0.142) \\ &= -0.558 \end{aligned}$$

$$\begin{aligned} \frac{\partial a_3}{\partial z_3} &= \begin{cases} 1 & \text{if } z_3 > 1 \\ 0 & otherwise \end{cases} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= -0.558 \times 1 \\ &= -0.558 \end{aligned}$$

Calculations for δ_2:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial a_2} &= w_{3,2} \times \delta_3 \\ &= 0.3 \times -0.558 \\ &= -0.1674 \end{aligned}$$

$$\begin{aligned} \frac{\partial a_2}{\partial z_2} &= \begin{cases} 1 & \text{if } z_2 > 1 \\ 0 & otherwise \end{cases} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \delta_2 &= \frac{\partial \mathcal{E}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \\ &= -0.1674 \times 1 \\ &= -0.1674 \end{aligned}$$

iii. Using the δ values you have calculated in the preceding, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial\mathcal{E}/\partial w_{3,2}$, $\partial\mathcal{E}/\partial w_{3,0}$, $\partial\mathcal{E}/\partial w_{2,1}$, $\partial\mathcal{E}/\partial w_{2,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{3,2}} &= \delta_3 \times a_2 \\ &= -0.558 \times 0.14 \\ &= -0.07812\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{3,0}} &= \delta_3 \times a_0 \\ &= -0.558 \times 1 \\ &= -0.558\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{2,1}} &= \delta_2 \times a_1 \\ &= -0.1674 \times 0.2 \\ &= -0.03348\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{2,0}} &= \delta_2 \times a_0 \\ &= -0.1674 \times 1 \\ &= -0.1674\end{aligned}$$

iv. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{3,2}, w_{3,0,}, w_{2,1}, w_{2,0}$) after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial\mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$\begin{aligned} w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\ &= 0.3 - 0.1 \times -0.07812 \\ &= 0.307812 \end{aligned}$$

$$\begin{aligned} w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\ &= 0.1 - 0.1 \times -0.558 \\ &= 0.1558 \end{aligned}$$

$$\begin{aligned} w_{2,1}^{t+1} &= w_{2,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,1}} \\ &= 0.2 - 0.1 \times -0.03348 \\ &= 0.203348 \end{aligned}$$

$$\begin{aligned} w_{2,0}^{t+1} &= w_{2,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,0}} \\ &= 0.1 - 0.1 \times -0.1674 \\ &= 0.11674 \end{aligned}$$

v. Calculate the reduction in the error for this example using the new weights for the network, compared with using the original weights.

The output for the network using the new weights is:

$$\begin{aligned} z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.2)) \\ &= ((0.11674 \times 1) + (0.203348 \times 0.2)) \\ &= 0.1574096 \end{aligned}$$

$$\begin{aligned} a_2 &= rectifier(0.1574096) \\ &= max(0, 0.1574096) \\ &= 0.1574096 \end{aligned}$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.1558 \times 1) + (0.307812 \times 0.1574096)) \\
&= 0.204252564
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= rectifier(0.204252564) \\
&= max(0, 0.204252564) \\
&= 0.204252564
\end{aligned}
$$

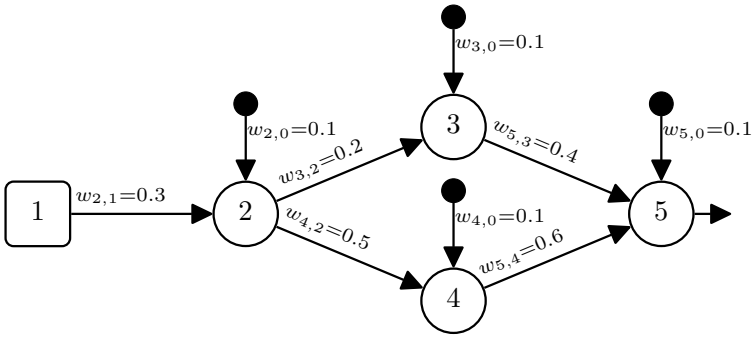
The error with the new weights is:

$$
\begin{aligned}
Error' &= 0.7 - 0.204252564 \\
&0.495747436
\end{aligned}
$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$
\begin{aligned}
Error\ Reduction &= 0.558 - 0.495747436 \\
&= 0.062252564
\end{aligned}
$$

5. The following image illustrates the topology of a simple feedforward neural network that has a single sensing neuron (Neuron 1), three hidden processing neuron (Neurons 2, 3, and 4), and a single processing output neuron (Neuron 5).



(a) Assuming that the processing neurons use **logistic** activation functions, that the input to the network is Neuron 1 = 0.5 and that the desired output for this input is 0.9:

i. Calculate the output generated by the network in response to this input.

$$
\begin{aligned}
z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.5)) \\
&= ((0.1 \times 1) + (0.3 \times 0.5)) \\
&= 0.25
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= logistic(0.25) \\
&= 0.562176501
\end{aligned}
$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.2 \times 0.562176501)) \\
&= 0.2124353
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= logistic(0.2124353) \\
&= 0.552909994
\end{aligned}
$$

$$
\begin{aligned}
z_4 &= ((w_{4,0} \times 1) + (w_{4,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.5 \times 0.552909994)) \\
&= 0.38108825
\end{aligned}
$$

$$
\begin{aligned}
a_4 &= logistic(0.38108825) \\
&= 0.594135549
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\
&= ((0.1 \times 1) + (0.4 \times 0.552909994) + (0.6 \times 0.594135549)) \\
&= 0.677645327
\end{aligned}
$$

$$
\begin{aligned}
a_5 &= logistic(0.677645327) \\
&= 0.663212956
\end{aligned}
$$

ii. Calculate the δ values for each of the processing neurons in the network (i.e., δ_5, δ_4, δ_3, δ_2).

Calculations for δ_5:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_5} &= -(t_k - a_k) \\ &= -(0.9 - 0.663212956) \\ &= -0.236787044\end{aligned}$$

$$\begin{aligned}\frac{\partial a_5}{\partial z_5} &= logistic(z_5) \times (1 - logistic(z_5)) \\ &= 0.677645327 \times (1 - 0.677645327) \\ &= 0.223361531\end{aligned}$$

$$\begin{aligned}\delta_3 = \frac{\partial \mathcal{E}}{\partial a_5} \times \frac{\partial a_5}{\partial z_5} \\ &= -0.236787044 \times 0.223361531 \\ &= -0.052889117\end{aligned}$$

Calculations for δ_4:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_4} &= w_{5,4} \times \delta_5 \\ &= 0.6 \times -0.052889117 \\ &= -0.03173347\end{aligned}$$

$$\begin{aligned}\frac{\partial a_4}{\partial z_4} &= logistic(z_4) \times (1 - logistic(z_4)) \\ &= 0.594135549 \times (1 - 0.594135549) \\ &= 0.241138498\end{aligned}$$

$$\begin{aligned}\delta_4 = \frac{\partial \mathcal{E}}{\partial a_4} \times \frac{\partial a_4}{\partial z_4} \\ &= -0.03173347 \times 0.241138498 \\ &= -0.007652161\end{aligned}$$

Calculations for δ_3:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_3} &= w_{5,3} \times \delta_5 \\ &= 0.4 \times -0.052889117 \\ &= -0.021155647\end{aligned}$$

$$\begin{aligned}\frac{\partial a_3}{\partial z_3} &= logistic(z_3) \times (1 - logistic(z_3)) \\ &= 0.552909994 \times (1 - 0.552909994) \\ &= 0.247200532\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= -0.021155647 \times 0.247200532 \\ &= -0.005229687\end{aligned}$$

Calculations for δ_2:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_2} &= (w_{3,2} \times \delta_3) + (w_{4,2} \times \delta_4) \\ &= (0.2 \times -0.005229687) + (0.5 \times -0.007652161) \\ &= -0.004872018\end{aligned}$$

$$\begin{aligned}\frac{\partial a_2}{\partial z_2} &= logistic(z_2) \times (1 - logistic(z_2)) \\ &= 0.562176501 \times (1 - 0.562176501) \\ &= 0.246134083\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{\partial \mathcal{E}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \\ &= -0.004872018 \times 0.246134083 \\ &= -0.00119917\end{aligned}$$

iii. Using the δ values you have calculated, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial\mathcal{E}/\partial w_{5,4}$, $\partial\mathcal{E}/\partial w_{5,3}$, $\partial\mathcal{E}/\partial w_{5,0}$, $\partial\mathcal{E}/\partial w_{4,2}$, $\partial\mathcal{E}/\partial w_{4,0}$, $\partial\mathcal{E}/\partial w_{3,2}$, $\partial\mathcal{E}/\partial w_{3,0}$, $\partial\mathcal{E}/\partial w_{2,1}$,

$\partial\mathcal{E}/\partial w_{2,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{5,4}} &= \delta_5 \times a_4 \\ &= -0.052889117 \times 0.594135549 \\ &= -0.031423304\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{5,3}} &= \delta_5 \times a_3 \\ &= -0.052889117 \times 0.552909994 \\ &= -0.029242921\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{5,0}} &= \delta_5 \times a_0 \\ &= -0.052889117 \times 1 \\ &= -0.052889117\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{4,2}} &= \delta_4 \times a_2 \\ &= -0.007652161 \times 0.562176501 \\ &= -0.004301865\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{4,0}} &= \delta_4 \times a_0 \\ &= -0.007652161 \times 1 \\ &= -0.007652161\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{3,2}} &= \delta_3 \times a_2 \\ &= -0.005229687 \times 0.562176501 \\ &= -0.002940007\end{aligned}$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,0}} = \delta_3 \times a_0$$
$$= -0.005229687 \times 1$$
$$= -0.005229687$$

$$\frac{\partial \mathcal{E}}{\partial w_{2,1}} = \delta_2 \times a_1$$
$$= -0.00119917 \times 0.5$$
$$= -0.000599585$$

$$\frac{\partial \mathcal{E}}{\partial w_{2,0}} = \delta_2 \times a_0$$
$$= -0.00119917 \times 1$$
$$= -0.00119917$$

iv. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{5,4}$, $w_{5,3}$, $w_{5,0}$, $w_{4,2}$, $w_{4,0}$, $w_{3,2}$, $w_{3,0,}$, $w_{2,1}$, $w_{2,0,}$) after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$w_{5,4}^{t+1} = w_{5,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,4}}$$
$$= 0.6 - (0.1 \times -0.031423304)$$
$$= 0.60314233$$

$$w_{5,3}^{t+1} = w_{5,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,3}}$$
$$= 0.4 - (0.1 \times -0.029242921)$$
$$= 0.402924292$$

$$
\begin{aligned}
w_{5,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,0}} \\
&= 0.1 - (0.1 \times -0.052889117) \\
&= 0.105288912
\end{aligned}
$$

$$
\begin{aligned}
w_{4,2}^{t+1} &= w_{4,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,2}} \\
&= 0.5 - (0.1 \times -0.004301865) \\
&= 0.500430187
\end{aligned}
$$

$$
\begin{aligned}
w_{4,0}^{t+1} &= w_{4,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,0}} \\
&= 0.1 - (0.1 \times -0.007652161) \\
&= 0.100765216
\end{aligned}
$$

$$
\begin{aligned}
w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\
&= 0.2 - (0.1 \times -0.002940007) \\
&= 0.200294001
\end{aligned}
$$

$$
\begin{aligned}
w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\
&= 0.1 - (0.1 \times -0.005229687) \\
&= 0.100522969
\end{aligned}
$$

$$
\begin{aligned}
w_{2,1}^{t+1} &= w_{2,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,1}} \\
&= 0.3 - (0.1 \times -0.000599585) \\
&= 0.300059958
\end{aligned}
$$

$$
\begin{aligned}
w_{2,0}^{t+1} &= w_{2,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,0}} \\
&= 0.1 - (0.1 \times -0.00119917) \\
&= 0.100119917
\end{aligned}
$$

v. Calculate the reduction in the error of the network for this example using the new weights, compared with using the original weights.

The output for the network using the new weights is:

$$\begin{aligned} z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.5)) \\ &= ((0.100119917 \times 1) + (0.300059958 \times 0.5)) \\ &= 0.250149896 \end{aligned}$$

$$\begin{aligned} a_2 &= logistic(0.250149896) \\ &= 0.562213395 \end{aligned}$$

$$\begin{aligned} z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\ &= ((0.100522969 \times 1) + (0.200294001 \times 0.562213395)) \\ &= 0.213130939 \end{aligned}$$

$$\begin{aligned} a_3 &= logistic(0.213130939) \\ &= 0.55308195 \end{aligned}$$

$$\begin{aligned} z_4 &= ((w_{4,0} \times 1) + (w_{4,2} \times a_2)) \\ &= ((0.100765216 \times 1) + (0.500430187 \times 0.562213395)) \\ &= 0.38211377 \end{aligned}$$

$$\begin{aligned} a_4 &= logistic(0.38211377) \\ &= 0.594382817 \end{aligned}$$

$$\begin{aligned} z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\ &= ((0.105288912 \times 1) + (0.402924292 \times 0.55308195) + (0.60314233 \times 0.594382817)) \\ &= 0.686636503 \end{aligned}$$

$$\begin{aligned} a_5 &= logistic(0.686636503) \\ &= 0.665218283 \end{aligned}$$

The error with the new weights is:

$$Error' = 0.9 - 0.665218283$$
$$0.234781717$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$\begin{aligned} Error\ Reduction &= 0.236787044 - 0.234781717 \\ &= 0.002005326 \end{aligned}$$

(b) Assuming that the processing neurons are **ReLU**s, that the input to the network is Neuron 1 = 0.5 and that the desired output for this input is 0.9

i. Calculate the output generated by the network in response to this input.

$$\begin{aligned} z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.5)) \\ &= ((0.1 \times 1) + (0.3 \times 0.5)) \\ &= 0.25 \end{aligned}$$

$$\begin{aligned} a_2 &= rectifier(0.25) \\ &= max(0, 0.25) \\ &= 0.25 \end{aligned}$$

$$\begin{aligned} z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\ &= ((0.1 \times 1) + (0.2 \times 0.25)) \\ &= 0.15 \end{aligned}$$

$$\begin{aligned} a_3 &= rectifier(0.15) \\ &= max(0, 0.15) \\ &= 0.15 \end{aligned}$$

$$\begin{aligned} z_4 &= ((w_{4,0} \times 1) + (w_{4,2} \times a_2)) \\ &= ((0.1 \times 1) + (0.5 \times 0.25)) \\ &= 0.225 \end{aligned}$$

$$\begin{aligned} a_4 &= rectifier(0.225) \\ &= max(0, 0.225) \\ &= 0.225 \end{aligned}$$

$$\begin{aligned} z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\ &= ((0.1 \times 1) + (0.4 \times 0.15) + (0.6 \times 0.225)) \\ &= 0.295 \end{aligned}$$

$$\begin{aligned} a_5 &= rectifier(0.295) \\ &= max(0, 0.295) \\ &= 0.295 \end{aligned}$$

ii. Calculate the δ values for each of the processing neurons in the network (i.e., δ_5, δ_4, δ_3, δ_2).

Calculations for δ_5:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial a_5} &= -(t_k - a_k) \\ &= -(0.9 - 0.295) \\ &= -0.605 \end{aligned}$$

$$\begin{aligned} \frac{\partial a_5}{\partial z_5} &= \begin{cases} 1 & \text{if } z_5 > 1 \\ 0 & otherwise \end{cases} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \delta_5 &= \frac{\partial \mathcal{E}}{\partial a_5} \times \frac{\partial a_5}{\partial z_5} \\ &= -0.605 \times 1 \\ &= -0.605 \end{aligned}$$

Calculations for δ_4:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_4} &= w_{5,4} \times \delta_5 \\ &= 0.6 \times -0.605 \\ &= -0.363\end{aligned}$$

$$\begin{aligned}\frac{\partial a_4}{\partial z_4} &= \begin{cases} 1 & \text{if } z_4 > 1 \\ 0 & \textit{otherwise} \end{cases} \\ &= 1\end{aligned}$$

$$\begin{aligned}\delta_4 &= \frac{\partial \mathcal{E}}{\partial a_4} \times \frac{\partial a_4}{\partial z_4} \\ &= -0.363 \times 1 \\ &= -0.363\end{aligned}$$

Calculations for δ_3:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_3} &= w_{5,3} \times \delta_5 \\ &= 0.4 \times -0.605 \\ &= -0.242\end{aligned}$$

$$\begin{aligned}\frac{\partial a_3}{\partial z_3} &= \begin{cases} 1 & \text{if } z_3 > 1 \\ 0 & \textit{otherwise} \end{cases} \\ &= 1\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= -0.242 \times 1 \\ &= -0.242\end{aligned}$$

Calculations for δ_2:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_2} &= (w_{3,2} \times \delta_3) + (w_{4,2} \times \delta_4) \\ &= (0.2 \times -0.242) + (0.5 \times -0.363) \\ &= -0.2299\end{aligned}$$

$$\begin{aligned}\frac{\partial a_2}{\partial z_2} &= \begin{cases} 1 & \text{if } z_2 > 1 \\ 0 & \textit{otherwise} \end{cases} \\ &= 1\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{\partial \mathcal{E}}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \\ &= -0.2299 \times 1 \\ &= -0.2299\end{aligned}$$

iii. Using the δ values you have calculated, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial\mathcal{E}/\partial w_{5,4}$, $\partial\mathcal{E}/\partial w_{5,3}$, $\partial\mathcal{E}/\partial w_{5,0}$, $\partial\mathcal{E}/\partial w_{4,2}$, $\partial\mathcal{E}/\partial w_{4,0}$, $\partial\mathcal{E}/\partial w_{3,2}$, $\partial\mathcal{E}/\partial w_{3,0}$, $\partial\mathcal{E}/\partial w_{2,1}$, $\partial\mathcal{E}/\partial w_{2,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,4}} &= \delta_5 \times a_4 \\ &= -0.605 \times 0.225 \\ &= -0.136125\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,3}} &= \delta_5 \times a_3 \\ &= -0.605 \times 0.15 \\ &= -0.09075\end{aligned}$$

$$\frac{\partial \mathcal{E}}{\partial w_{5,0}} = \delta_5 \times a_0$$
$$= -0.605 \times 1$$
$$= -0.605$$

$$\frac{\partial \mathcal{E}}{\partial w_{4,2}} = \delta_4 \times a_2$$
$$= -0.363 \times 0.25$$
$$= -0.09075$$

$$\frac{\partial \mathcal{E}}{\partial w_{4,0}} = \delta_4 \times a_0$$
$$= -0.363 \times 1$$
$$= -0.363$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,2}} = \delta_3 \times a_2$$
$$= -0.242 \times 0.25$$
$$= -0.0605$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,0}} = \delta_3 \times a_0$$
$$= -0.242 \times 1$$
$$= -0.242$$

$$\frac{\partial \mathcal{E}}{\partial w_{2,1}} = \delta_2 \times a_1$$
$$= -0.2299 \times 0.5$$
$$= -0.11495$$

$$\frac{\partial \mathcal{E}}{\partial w_{2,0}} = \delta_2 \times a_0$$
$$= -0.2299 \times 1$$
$$= -0.2299$$

iv. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{5,4}$, $w_{5,3}$, $w_{5,0}$, $w_{4,2}$, $w_{4,0}$, $w_{3,2}$, $w_{3,0,}$, $w_{2,1}$, $w_{2,0,}$)

after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$\begin{aligned} w_{5,4}^{t+1} &= w_{5,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,4}} \\ &= 0.6 - (0.1 \times -0.136125) \\ &= 0.6136125 \end{aligned}$$

$$\begin{aligned} w_{5,3}^{t+1} &= w_{5,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,3}} \\ &= 0.4 - (0.1 \times -0.09075) \\ &= 0.409075 \end{aligned}$$

$$\begin{aligned} w_{5,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,0}} \\ &= 0.1 - (0.1 \times -0.605) \\ &= 0.1605 \end{aligned}$$

$$\begin{aligned} w_{4,2}^{t+1} &= w_{4,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,2}} \\ &= 0.5 - (0.1 \times -0.09075) \\ &= 0.509075 \end{aligned}$$

$$\begin{aligned} w_{4,0}^{t+1} &= w_{4,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,0}} \\ &= 0.1 - (0.1 \times -0.363) \\ &= 0.1363 \end{aligned}$$

$$\begin{aligned} w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\ &= 0.2 - (0.1 \times -0.0605) \\ &= 0.20605 \end{aligned}$$

$$
\begin{aligned}
w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\
&= 0.1 - (0.1 \times -0.242) \\
&= 0.1242
\end{aligned}
$$

$$
\begin{aligned}
w_{2,1}^{t+1} &= w_{2,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,1}} \\
&= 0.3 - (0.1 \times -0.11495) \\
&= 0.311495
\end{aligned}
$$

$$
\begin{aligned}
w_{2,0}^{t+1} &= w_{2,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{2,0}} \\
&= 0.1 - (0.1 \times -0.2299) \\
&= 0.12299
\end{aligned}
$$

v. Calculate the reduction in the error of the network for this example using the new weights, compared with using the original weights.

The output for the network using the new weights is:

$$
\begin{aligned}
z_2 &= ((w_{2,0} \times 1) + (w_{2,1} \times 0.5)) \\
&= ((0.12299 \times 1) + (0.311495 \times 0.5)) \\
&= 0.2787375
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= rectifier(0.2787375) \\
&= max(0, 0.2787375) \\
&= 0.2787375
\end{aligned}
$$

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,2} \times a_2)) \\
&= ((0.1242 \times 1) + (0.20605 \times 0.2787375)) \\
&= 0.181633862
\end{aligned}
$$

$$\begin{aligned} a_3 &= rectifier(0.181633862) \\ &= max(0, 0.181633862) \\ &= 0.181633862 \end{aligned}$$

$$\begin{aligned} z_4 &= ((w_{4,0} \times 1) + (w_{4,2} \times a_2)) \\ &= ((0.1363 \times 1) + (0.509075 \times 0.2787375)) \\ &= 0.278198293 \end{aligned}$$

$$\begin{aligned} a_4 &= rectifier(0.278198293) \\ &= max(0, 0.278198293) \\ &= 0.278198293 \end{aligned}$$

$$\begin{aligned} z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\ &= ((0.1605 \times 1) + (0.409075 \times 0.181633862) + (0.6136125 \times 0.278198293)) \\ &= 0.405507822 \end{aligned}$$

$$\begin{aligned} a_5 &= rectifier(0.405507822) \\ &= max(0, 0.405507822) \\ &= 0.405507822 \end{aligned}$$

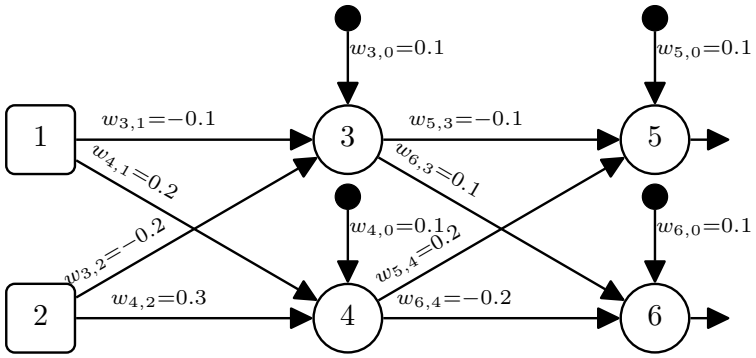
The error with the new weights is:

$$\begin{aligned} Error' &= 0.9 - 0.405507822 \\ &0.494492178 \end{aligned}$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$\begin{aligned} Error\ Reduction &= 0.605 - 0.494492178 \\ &= 0.110507822 \end{aligned}$$

6. The following image illustrates the topology of a feedforward neural network that has two sensing neurons (Neurons 1 and 2), two hidden processing neuron (Neurons 3, and 4), and two processing output neurons (Neurons 5 and 6).



(a) Assuming that the processing neurons use **logistic** activation functions, that the input to the network is Neuron 1 = 0.3 and Neuron 2 = 0.6, and that the desired output for this input is Neuron 5 = 0.7 and Neuron 6 = 0.4:

i. Calculate the output generated by the network in response to this input.

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,1} \times a_1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (-0.1 \times 0.3) + (-0.2 \times 0.6)) \\
&= -0.05
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= \mathit{logistic}(-0.05) \\
&= 0.487502604
\end{aligned}
$$

$$
\begin{aligned}
z_4 &= ((w_{4,0} \times 1) + (w_{4,1} \times a_1) + (w_{4,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.2 \times 0.3) + (0.3 \times 0.6)) \\
&= 0.34
\end{aligned}
$$

$$
\begin{aligned}
a_4 &= \mathit{logistic}(0.34) \\
&= 0.584190523
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\
&= ((0.1 \times 1) + (-0.1 \times 0.487502604) + (0.2 \times 0.584190523)) \\
&= 0.168087844
\end{aligned}
$$

$$a_5 = logistic(0.168087844)$$
$$= 0.541923301$$

$$z_6 = ((w_{6,0} \times 1) + (w_{6,3} \times a_3) + (w_{6,4} \times a_4))$$
$$= ((0.1 \times 1) + (0.1 \times 0.487502604) + (-0.2 \times 0.584190523))$$
$$= 0.031912156$$

$$a_6 = logistic(0.031912156)$$
$$= 0.507977362$$

ii. Calculate the sum of squared errors for this network for this example.

$$SSE = (t_5 - a_5)^2 + (t_6 - a_6)^2$$
$$= (0.7 - 0.541923301)^2 + (0.4 - 0.507977362)^2$$
$$= (0.158076699)^2 + (-0.107977362)^2$$
$$= 0.036647354$$

iii. Calculate the δ values for each of the processing neurons in the network (i.e., δ_6, δ_5, δ_4, δ_3).

Calculations for δ_6:

$$\frac{\partial \mathcal{E}}{\partial a_6} = -(t_6 - a_6)$$
$$= -(0.4 - 0.507977362)$$
$$= -(-0.107977362)$$
$$= 0.107977362$$

$$\frac{\partial a_6}{\partial z_6} = logistic(z_6) \times (1 - logistic(z_6))$$
$$= 0.507977362 \times (1 - 0.507977362)$$
$$= 0.249936362$$

$$\begin{aligned}\delta_6 &= \frac{\partial \mathcal{E}}{\partial a_6} \times \frac{\partial a_6}{\partial z_6} \\ &= 0.107977362 \times 0.249936362 \\ &= 0.026987469\end{aligned}$$

Calculations for δ_5:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_5} &= -(t_5 - a_5) \\ &= -(0.7 - 0.541923301) \\ &= -0.158076699\end{aligned}$$

$$\begin{aligned}\frac{\partial a_5}{\partial z_5} &= logistic(z_5) \times (1 - logistic(z_5)) \\ &= 0.541923301 \times (1 - 0.541923301) \\ &= -0.039241345\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_5} \times \frac{\partial a_5}{\partial z_5} \\ &= -0.158076699 \times 0.248242437 \\ &= -0.039241345\end{aligned}$$

Calculations for δ_4:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_4} &= (w_{5,4} \times \delta_5) + (w_{6,4} \times \delta_6) \\ &= (0.2 \times -0.039241345) + (-0.2 \times 0.026987469) \\ &= -0.013245763\end{aligned}$$

$$\begin{aligned}\frac{\partial a_4}{\partial z_4} &= logistic(z_4) \times (1 - logistic(z_4)) \\ &= 0.584190523 \times (1 - 0.584190523) \\ &= 0.242911956\end{aligned}$$

$$\begin{aligned}\delta_4 &= \frac{\partial \mathcal{E}}{\partial a_4} \times \frac{\partial a_4}{\partial z_4} \\ &= -0.013245763 \times 0.242911956 \\ &= -0.003217554\end{aligned}$$

Calculations for δ_3:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial a_3} &= (w_{5,3} \times \delta_5) + (w_{6,3} \times \delta_6) \\ &= (-0.1 \times -0.039241345) + (0.1 \times 0.026987469) \\ &= 0.006622881\end{aligned}$$

$$\begin{aligned}\frac{\partial a_3}{\partial z_3} &= \mathit{logistic}(z_3) \times (1 - \mathit{logistic}(z_3)) \\ &= 0.487502604 \times (1 - 0.487502604) \\ &= 0.249843815\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= 0.006622881 \times 0.249843815 \\ &= 0.001654686\end{aligned}$$

iv. Using the δ values you calculated above, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial\mathcal{E}/\partial w_{6,4}$, $\partial\mathcal{E}/\partial w_{6,3}$, $\partial\mathcal{E}/\partial w_{6,0}$, $\partial\mathcal{E}/\partial w_{5,4}$, $\partial\mathcal{E}/\partial w_{5,3}$, $\partial\mathcal{E}/\partial w_{5,0}$, $\partial\mathcal{E}/\partial w_{4,2}$, $\partial\mathcal{E}/\partial w_{4,1}$, $\partial\mathcal{E}/\partial w_{4,0}$, $\partial\mathcal{E}/\partial w_{3,2}$, $\partial\mathcal{E}/\partial w_{3,1}$, $\partial\mathcal{E}/\partial w_{3,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{6,4}} &= \delta_6 \times a_4 \\ &= 0.026987469 \times 0.584190523 \\ &= 0.015765824\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{6,3}} &= \delta_6 \times a_3 \\ &= 0.026987469 \times 0.487502604 \\ &= 0.013156461\end{aligned}$$

$$\frac{\partial \mathcal{E}}{\partial w_{5,0}} = \delta_6 \times a_0$$
$$= 0.026987469 \times 1$$
$$= 0.026987469$$

$$\frac{\partial \mathcal{E}}{\partial w_{5,4}} = \delta_5 \times a_4$$
$$= -0.039241345 \times 0.584190523$$
$$= -0.022924422$$

$$\frac{\partial \mathcal{E}}{\partial w_{5,3}} = \delta_5 \times a_3$$
$$= -0.039241345 \times 0.487502604$$
$$= -0.019130258$$

$$\frac{\partial \mathcal{E}}{\partial w_{5,0}} = \delta_5 \times a_0$$
$$= -0.039241345 \times 1$$
$$= -0.039241345$$

$$\frac{\partial \mathcal{E}}{\partial w_{4,2}} = \delta_4 \times a_2$$
$$= -0.003217554 \times 0.6$$
$$= -0.001930532$$

$$\frac{\partial \mathcal{E}}{\partial w_{4,1}} = \delta_4 \times a_1$$
$$= -0.003217554 \times 0.3$$
$$= -0.000965266$$

$$\frac{\partial \mathcal{E}}{\partial w_{4,0}} = \delta_4 \times a_0$$
$$= -0.003217554 \times 1$$
$$= -0.003217554$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,2}} = \delta_3 \times a_2$$
$$= 0.001654686 \times 0.6$$
$$= 0.000992812$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,1}} = \delta_3 \times a_1$$
$$= 0.001654686 \times 0.3$$
$$= 0.000496406$$

$$\frac{\partial \mathcal{E}}{\partial w_{3,0}} = \delta_3 \times a_0$$
$$= 0.001654686 \times 1$$
$$= 0.001654686$$

v. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{6,4}$,$w_{6,3}$,$w_{6,0}$,$w_{5,4}$, $w_{5,3}$, $w_{5,0}$, $w_{4,2}$,$w_{4,1}$,$w_{4,0}$, $w_{3,2}$, $w_{3,1}$, $w_{3,0,}$) after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$w_{6,4}^{t+1} = w_{6,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,4}}$$
$$= -0.2 - (0.1 \times 0.015765824)$$
$$= -0.201576582$$

$$w_{6,3}^{t+1} = w_{6,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,3}}$$
$$= 0.1 - (0.1 \times 0.013156461)$$
$$= 0.098684354$$

$$
\begin{aligned}
w_{6,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,0}} \\
&= 0.1 - (0.1 \times 0.026987469) \\
&= 0.097301253
\end{aligned}
$$

$$
\begin{aligned}
w_{5,4}^{t+1} &= w_{5,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,4}} \\
&= 0.2 - (0.1 \times -0.022924422) \\
&= 0.202292442
\end{aligned}
$$

$$
\begin{aligned}
w_{5,3}^{t+1} &= w_{5,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,3}} \\
&= -0.1 - (0.1 \times -0.019130258) \\
&= -0.098086974
\end{aligned}
$$

$$
\begin{aligned}
w_{5,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,0}} \\
&= 0.1 - (0.1 \times -0.039241345) \\
&= 0.103924135
\end{aligned}
$$

$$
\begin{aligned}
w_{4,2}^{t+1} &= w_{4,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,2}} \\
&= 0.3 - (0.1 \times -0.001930532) \\
&= 0.300193053
\end{aligned}
$$

$$
\begin{aligned}
w_{4,1}^{t+1} &= w_{4,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,1}} \\
&= 0.2 - (0.1 \times -0.000965266) \\
&= 0.200096527
\end{aligned}
$$

$$
\begin{aligned}
w_{4,0}^{t+1} &= w_{4,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,0}} \\
&= 0.1 - (0.1 \times -0.003217554) \\
&= 0.100321755
\end{aligned}
$$

$$
\begin{aligned}
w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\
&= -0.2 - (0.1 \times 0.000992812) \\
&= -0.200099281
\end{aligned}
$$

$$
\begin{aligned}
w_{3,1}^{t+1} &= w_{3,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,1}} \\
&= -0.1 - (0.1 \times 0.000496406) \\
&= -0.100049641
\end{aligned}
$$

$$
\begin{aligned}
w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\
&= 0.1 - (0.1 \times 0.001654686) \\
&= 0.099834531
\end{aligned}
$$

vi. Calculate the reduction in the sum of squared error of the network for this example using the new weights, compared with using the original weights.

The output for the network using the new weights is:

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,1} \times a_1) + (w_{3,2} \times a_2)) \\
&= ((0.099834531 \times 1) + (-0.100049641 \times 0.3) + (-0.200099281 \times 0.6)) \\
&= -0.050239929
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= logistic(-0.050239929) \\
&= 0.487442659
\end{aligned}
$$

$$
\begin{aligned}
z_4 &= ((w_{4,0} \times 1) + (w_{4,1} \times a_1) + (w_{4,2} \times a_2)) \\
&= ((0.100321755 \times 1) + (0.200096527 \times 0.3) + (0.300193053 \times 0.6)) \\
&= 0.340466545
\end{aligned}
$$

$$
\begin{aligned}
a_4 &= logistic(0.340466545) \\
&= 0.584303848
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\
&= ((0.103924135 \times 1) + (-0.098086974 \times 0.487442659) + (0.202292442 \times 0.584303848)) \\
&= 0.174312611
\end{aligned}
$$

$$
\begin{aligned}
a_5 &= logistic(0.174312611) \\
&= 0.543468144
\end{aligned}
$$

$$
\begin{aligned}
z_6 &= ((w_{6,0} \times 1) + (w_{6,3} \times a_3) + (w_{6,4} \times a_4)) \\
&= ((0.097301253 \times 1) + (0.098684354 \times 0.487442659) + (-0.201576582 \times 0.584303848)) \\
&= 0.027622244
\end{aligned}
$$

$$
\begin{aligned}
a_6 &= logistic(0.027622244) \\
&= 0.506905122
\end{aligned}
$$

If we now calculate the sum of squared errors on this example for this network using the new weights we get:

$$
\begin{aligned}
SSE &= (t_5 - a_5)^2 + (t_6 - a_6)^2 \\
&= (0.7 - 0.543468144)^2 + (0.4 - 0.506905122)^2 \\
&= (0.156531856)^2 + (-0.106905122)^2 \\
&= 0.035930927
\end{aligned}
$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$
\begin{aligned}
Error\ Reduction &= 0.036647354 - 0.035930927 \\
&= 0.000716426
\end{aligned}
$$

(b) Assuming that the processing neurons use a **rectifier** activation functions, that the input to the network is Neuron 1 = 0.3 and Neuron 2 = 0.6 and that the desired output for this input is Neuron 5 = 0.7 and Neuron 6 = 0.4:

i. Calculate the output generated by the network in response to this input.

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,1} \times a_1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (-0.1 \times 0.3) + (-0.2 \times 0.6)) \\
&= -0.05
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= rectifier(-0.05) \\
&= max(0, -0.05) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
z_4 &= ((w_{4,0} \times 1) + (w_{4,1} \times a_1) + (w_{4,2} \times a_2)) \\
&= ((0.1 \times 1) + (0.2 \times 0.3) + (0.3 \times 0.6)) \\
&= 0.34
\end{aligned}
$$

$$
\begin{aligned}
a_4 &= rectifier(0.34) \\
&= max(0, 0.34) \\
&= 0.34
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\
&= ((0.1 \times 1) + (-0.1 \times 0) + (0.2 \times 0.34)) \\
&= 0.168
\end{aligned}
$$

$$
\begin{aligned}
a_5 &= rectifier(0.168) \\
&= max(0, 0.168) \\
&= 0.168
\end{aligned}
$$

$$
\begin{aligned}
z_6 &= ((w_{6,0} \times 1) + (w_{6,3} \times a_3) + (w_{6,4} \times a_4)) \\
&= ((0.1 \times 1) + (0.1 \times 0) + (-0.2 \times 0.34)) \\
&= 0.032
\end{aligned}
$$

$$\begin{aligned} a_6 &= rectifier(0.032) \\ &= max(0, 0.032) \\ &= 0.032 \end{aligned}$$

ii. Calculate the sum of squared errors for this network on this example.

$$\begin{aligned} SSE &= (t_5 - a_5)^2 + (t_6 - a_6)^2 \\ &= (0.7 - 0.168)^2 + (0.4 - 0.032)^2 \\ &= (0.532)^2 + (0.368)^2 \\ &= 0.418448 \end{aligned}$$

iii. Calculate the δ values for each of the processing neurons in the network (i.e., δ_6, δ_5, δ_4, δ_3).

Calculations for δ_6:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial a_6} &= -(t_6 - a_6) \\ &= -(0.4 - 0.032) \\ &= -(0.368) \\ &= -0.368 \end{aligned}$$

$$\begin{aligned} \frac{\partial a_6}{\partial z_6} &= \begin{cases} 1 & \text{if } z_6 > 1 \\ 0 & otherwise \end{cases} \\ &= 1 \end{aligned}$$

$$\begin{aligned} \delta_6 &= \frac{\partial \mathcal{E}}{\partial a_6} \times \frac{\partial a_6}{\partial z_6} \\ &= -0.368 \times 1 \\ &= -0.368 \end{aligned}$$

Calculations for δ_5:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial a_5} &= -(t_5 - a_5) \\ &= -(0.7 - 0.168) \\ &= -0.532 \end{aligned}$$

$$\frac{\partial a_5}{\partial z_5} = \begin{cases} 1 & \text{if } z_5 > 1 \\ 0 & otherwise \end{cases}$$
$$= 1$$

$$\delta_5 = \frac{\partial \mathcal{E}}{\partial a_5} \times \frac{\partial a_5}{\partial z_5}$$
$$= -0.532 \times 1$$
$$= -0.532$$

Calculations for δ_4:

$$\frac{\partial \mathcal{E}}{\partial a_4} = (w_{5,4} \times \delta_5) + (w_{6,4} \times \delta_6)$$
$$= (0.2 \times -0.532) + (-0.2 \times -0.368)$$
$$= -0.0328$$

$$\frac{\partial a_4}{\partial z_4} = \begin{cases} 1 & \text{if } z_4 > 1 \\ 0 & otherwise \end{cases}$$
$$= 1$$

$$\delta_4 = \frac{\partial \mathcal{E}}{\partial a_4} \times \frac{\partial a_4}{\partial z_4}$$
$$= -0.0328 \times 1$$
$$= -0.0328$$

Calculations for δ_3:

$$\frac{\partial \mathcal{E}}{\partial a_3} = (w_{5,3} \times \delta_5) + (w_{6,3} \times \delta_6)$$
$$= (-0.1 \times -0.532) + (0.1 \times -0.368)$$
$$= 0.0164$$

$$\frac{\partial a_3}{\partial z_3} = \begin{cases} 1 & \text{if } z_3 > 1 \\ 0 & otherwise \end{cases}$$
$$= 0$$

$$\begin{aligned}\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} \times \frac{\partial a_3}{\partial z_3} \\ &= 0.0164 \times 0 \\ &= 0\end{aligned}$$

iv. Using the δ values you calculated above, calculate the sensitivity of the error of the network to changes in each of the weights of the network (i.e., $\partial\mathcal{E}/\partial w_{6,4}$, $\partial\mathcal{E}/\partial w_{6,3}$, $\partial\mathcal{E}/\partial w_{6,0}$, $\partial\mathcal{E}/\partial w_{5,4}$, $\partial\mathcal{E}/\partial w_{5,3}$, $\partial\mathcal{E}/\partial w_{5,0}$, $\partial\mathcal{E}/\partial w_{4,2}$, $\partial\mathcal{E}/\partial w_{4,1}$, $\partial\mathcal{E}/\partial w_{4,0}$, $\partial\mathcal{E}/\partial w_{3,2}$, $\partial\mathcal{E}/\partial w_{3,1}$, $\partial\mathcal{E}/\partial w_{3,0}$).

To calculate the sensitivity of the error of the network to changes in a weight we multiply the δ for the neuron that uses the weight in its weighted sum by the activation (or input) that the weight was applied to in the calculation of the weight sum. The calculations for each of the weights is shown below.

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{6,4}} &= \delta_6 \times a_4 \\ &= -0.368 \times 0.34 \\ &= -0.12512\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{6,3}} &= \delta_6 \times a_3 \\ &= -0.368 \times 0 \\ &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,0}} &= \delta_6 \times a_0 \\ &= -0.368 \times 1 \\ &= -0.368\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,4}} &= \delta_5 \times a_4 \\ &= -0.532 \times 0.34 \\ &= -0.18088\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,3}} &= \delta_5 \times a_3 \\ &= -0.532 \times 0 \\ &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{5,0}} &= \delta_5 \times a_0 \\ &= -0.532 \times 1 \\ &= -0.532\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{4,2}} &= \delta_4 \times a_2 \\ &= -0.0328 \times 0.6 \\ &= -0.01968\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{4,1}} &= \delta_4 \times a_1 \\ &= -0.0328 \times 0.3 \\ &= -0.00984\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{4,0}} &= \delta_4 \times a_0 \\ &= -0.0328 \times 1 \\ &= -0.0328\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{3,2}} &= \delta_3 \times a_2 \\ &= 0 \times 0.6 \\ &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{3,1}} &= \delta_3 \times a_1 \\ &= 0 \times 0.3 \\ &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{3,0}} &= \delta_3 \times a_0 \\ &= 0 \times 1 \\ &= 0\end{aligned}$$

v. Assuming a learning rate of $\alpha = 0.1$, calculate the updated values for each of the weights in the network ($w_{6,4}$,$w_{6,3}$,$w_{6,0}$,$w_{5,4}$, $w_{5,3}$, $w_{5,0}$, $w_{4,2}$,$w_{4,1}$,$w_{4,0}$, $w_{3,2}$, $w_{3,1}$, $w_{3,0,}$) after the processing of this single training example.

We update weights as follows:

$$w_{i,k}^{t+1} = w_{i,k}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{i,k}}$$

The calculation for each weight update is below:

$$\begin{aligned}w_{6,4}^{t+1} &= w_{6,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,4}} \\ &= -0.2 - (0.1 \times -0.12512) \\ &= -0.187488\end{aligned}$$

$$\begin{aligned}w_{6,3}^{t+1} &= w_{6,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,3}} \\ &= 0.1 - (0.1 \times 0) \\ &= 0.1\end{aligned}$$

$$\begin{aligned}w_{6,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{6,0}} \\ &= 0.1 - (0.1 \times -0.368) \\ &= 0.1368\end{aligned}$$

$$\begin{aligned}w_{5,4}^{t+1} &= w_{5,4}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,4}} \\ &= 0.2 - (0.1 \times -0.18088) \\ &= 0.218088\end{aligned}$$

$$
\begin{aligned}
w_{5,3}^{t+1} &= w_{5,3}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,3}} \\
&= -0.1 - (0.1 \times 0) \\
&= -0.1
\end{aligned}
$$

$$
\begin{aligned}
w_{5,0}^{t+1} &= w_{5,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{5,0}} \\
&= 0.1 - (0.1 \times -0.532) \\
&= 0.1532
\end{aligned}
$$

$$
\begin{aligned}
w_{4,2}^{t+1} &= w_{4,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,2}} \\
&= 0.3 - (0.1 \times -0.01968) \\
&= 0.301968
\end{aligned}
$$

$$
\begin{aligned}
w_{4,1}^{t+1} &= w_{4,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,1}} \\
&= 0.2 - (0.1 \times -0.00984) \\
&= 0.200984
\end{aligned}
$$

$$
\begin{aligned}
w_{4,0}^{t+1} &= w_{4,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{4,0}} \\
&= 0.1 - (0.1 \times -0.0328) \\
&= 0.10328
\end{aligned}
$$

$$
\begin{aligned}
w_{3,2}^{t+1} &= w_{3,2}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,2}} \\
&= -0.2 - (0.1 \times 0) \\
&= -0.2
\end{aligned}
$$

$$
\begin{aligned}
w_{3,1}^{t+1} &= w_{3,1}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,1}} \\
&= -0.1 - (0.1 \times 0) \\
&= -0.1
\end{aligned}
$$

$$
\begin{aligned}
w_{3,0}^{t+1} &= w_{3,0}^{t} - \alpha \times \frac{\partial \mathcal{E}}{\partial w_{3,0}} \\
&= 0.1 - (0.1 \times 0) \\
&= 0.1
\end{aligned}
$$

vi. Calculate the reduction in the sum of squared error of the network for this example using the new weights, compared with using the original weights.

The output for the network using the new weights is:

$$
\begin{aligned}
z_3 &= ((w_{3,0} \times 1) + (w_{3,1} \times a_1) + (w_{3,2} \times a_2)) \\
&= ((0.1 \times 1) + (-0.1 \times 0.3) + (-0.2 \times 0.6)) \\
&= -0.05
\end{aligned}
$$

$$
\begin{aligned}
a_3 &= rectifier(-0.05) \\
&= max(0, -0.05) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
z_4 &= ((w_{4,0} \times 1) + (w_{4,1} \times a_1) + (w_{4,2} \times a_2)) \\
&= ((0.10328 \times 1) + (0.200984 \times 0.3) + (0.301968 \times 0.6)) \\
&= 0.344756
\end{aligned}
$$

$$
\begin{aligned}
a_4 &= rectifier(0.344756) \\
&= max(0, 0.344756) \\
&= 0.344756
\end{aligned}
$$

$$
\begin{aligned}
z_5 &= ((w_{5,0} \times 1) + (w_{5,3} \times a_3) + (w_{5,4} \times a_4)) \\
&= ((0.1532 \times 1) + (-0.1 \times 0) + (0.218088 \times 0.344756)) \\
&= 0.228387147
\end{aligned}
$$

$$
\begin{aligned}
a_5 &= rectifier(0.228387147) \\
&= max(0, 0.228387147) \\
&= 0.228387147
\end{aligned}
$$

$$\begin{aligned} z_6 &= ((w_{6,0} \times 1) + (w_{6,3} \times a_3) + (w_{6,4} \times a_4)) \\ &= ((0.1368 \times 1) + (0.1 \times 0) + (-0.187488 \times 0.344756)) \\ &= 0.072162387 \end{aligned}$$

$$\begin{aligned} a_6 &= rectifier(0.072162387) \\ &= max(0, 0.072162387) \\ &= 0.072162387 \end{aligned}$$

If we now calculate the sum of squared errors on this example for this network using the new weights we get:

$$\begin{aligned} SSE &= (t_5 - a_5)^2 + (t_6 - a_6)^2 \\ &= (0.7 - 0.228387147)^2 + (0.4 - 0.072162387)^2 \\ &= (0.471612853)^2 + (0.327837613)^2 \\ &= 0.329896184 \end{aligned}$$

Subtracting this new error from the error with the original weights (which we calculated as part of the calculation for δ_3 above) gives us the reduction in error:

$$\begin{aligned} Error\ Reduction &= 0.418448 - 0.329896184 \\ &= 0.088551816 \end{aligned}$$

7. Assuming a fully connected feedforward network where all the neurons uses a linear activation function (i.e., $a_i = z_i$) and with the following topology:
   (a) 100 neurons in the input layer
   (b) 5 hidden layers with 2,000 neurons in each layer
   (c) 10 neurons in the output layer

   If all the inputs to the network have been standardized to have a mean value of 0 and a standard deviation of 1, and the initial weights for the network are sampled from a normal distribution with mean 0.0 and standard deviation of $\sigma = 0.01$, then:

   (a) Calculate the variance of the z values across for the neurons in the first hidden layer in the first iteration of training.

$$
\begin{aligned}
var(Z^{(1)} &= n_{in}^{(1)} \times var(W^{(1)}) \times var(\mathbf{d}^{(1)} \\
&= 100 \times 0.0001 \times 1 \\
&= 0.01
\end{aligned} \tag{8.1}
$$

(b) Calculate the variance of the z values across for the neurons in the last hidden layer in the first iteration of training.

$$
\begin{aligned}
var(Z^{(5)} &= (n_{in}^{(2,\ldots,5)} \times var(W^{(2,\ldots,5)})^4 \times var(\mathbf{d}^{(2)} \\
&= (2000 \times 0.0001)^4 \times 0.01 \\
&= 0.00016
\end{aligned} \tag{8.2}
$$

Assuming that the variance of the δs for the output layer is equal to 1:

(a) Calculate the variance of the δs across for the neurons in the last hidden layer in the first iteration of training.

$$
\begin{aligned}
var(\delta^{(5)} &= n_{out}^{(5)} \times var(W^{(5)}) \times var(\mathbf{d}^{(5)} \\
&= 10 \times 0.0001 \times 1 \\
&= 0.001
\end{aligned} \tag{8.3}
$$

(b) Calculate the variance of the δs values across for the neurons in the first hidden layer in the first iteration of training.

$$
\begin{aligned}
var(\delta^{(1)} &= (n_{out}^{(1,\ldots,4)} \times var(W^{(1,\ldots,4)})^4 \times var(\mathbf{d}^{(4)} \\
&= (2000 \times 0.0001)^4 \times 0.001 \\
&= 0.0000016
\end{aligned} \tag{8.4}
$$

(c) Is the training dynamic of this network stable, or is it suffering from vanishing or exploding gradients?

The training dynamic of this network is exhibiting vanishing gradients because the variance of the δs is reducing through each layer that they are backpropagated through.

9 Evaluation (Exercise Solutions)

1. The table below shows the predictions made for a categorical target feature by a model for a test dataset. Based on this test set, calculate the evaluation measures listed below.

ID	Target	Prediction
1	false	false
2	false	false
3	false	false
4	false	false
5	true	true
6	false	false
7	true	true
8	true	true
9	false	false
10	false	false
11	false	false
12	true	true
13	false	false
14	true	true
15	false	false
16	false	false
17	true	false
18	true	true
19	true	true
20	true	true

(a) A confusion matrix and the misclassification rate

The confusion matrix can be written as

		Prediction *true*	Prediction *false*
Target	*true*	8	1
	false	0	11

Misclassification rate can be calculated as

$$\begin{aligned} misclassification\ rate &= \frac{(FP+FN)}{(TP+TN+FP+FN)} \\ &= \frac{(0+1)}{(8+11+0+1)} \\ &= 0.05 \end{aligned}$$

(b) The average class accuracy (harmonic mean)

First, we calculate the recall for each target level:

$$recall_{true} = \frac{8}{9} = 0.889$$

$$recall_{false} = \frac{11}{11} = 1.000$$

Then we can calculate a harmonic mean as

$$\begin{aligned} average\ class\ accuracy_{HM} &= \frac{1}{\frac{1}{|levels(t)|}\sum_{l \in levels(t)} \frac{1}{recall_l}} \\ &= \frac{1}{\frac{1}{2}\left(\frac{1}{0.889} + \frac{1}{1}\right)} \\ &= 0.941 \end{aligned}$$

(c) The precision, recall, and F_1 measure

We can calculate precision and recall as follows (assuming that the *true* target level is the positive level):

$$\begin{aligned} precision &= \frac{TP}{(TP + FP)} \\ &= \frac{8}{(8 + 0)} \\ &= 1.000 \end{aligned}$$

$$\begin{aligned} recall &= \frac{TP}{(TP + FN)} \\ &= \frac{8}{(8 + 1)} \\ &= 0.889 \end{aligned}$$

Using these figures, we can calculate the F_1 measure as

$$\begin{aligned} F_1\ measure &= 2 \times \frac{(precision \times recall)}{(precision + recall)} \\ &= 2 \times \frac{(1.000 \times 0.889)}{(1.000 + 0.889)} \\ &= 0.941 \end{aligned}$$

2. The table below shows the predictions made for a continuous target feature by two different prediction models for a test dataset.

ID	Target	Model 1 Prediction	Model 2 Prediction	ID	Target	Model 1 Prediction	Model 2 Prediction
1	2,623	2,664	2,691	16	2,570	2,577	2,612
2	2,423	2,436	2,367	17	2,528	2,510	2,557
3	2,423	2,399	2,412	18	2,342	2,381	2,421
4	2,448	2,447	2,440	19	2,456	2,452	2,393
5	2,762	2,847	2,693	20	2,451	2,437	2,479
6	2,435	2,411	2,493	21	2,296	2,307	2,290
7	2,519	2,516	2,598	22	2,405	2,355	2,490
8	2,772	2,870	2,814	23	2,389	2,418	2,346
9	2,601	2,586	2,583	24	2,629	2,582	2,647
10	2,422	2,414	2,485	25	2,584	2,564	2,546
11	2,349	2,407	2,472	26	2,658	2,662	2,759
12	2,515	2,505	2,584	27	2,482	2,492	2,463
13	2,548	2,581	2,604	28	2,471	2,478	2,403
14	2,281	2,277	2,309	29	2,605	2,620	2,645
15	2,295	2,280	2,296	30	2,442	2,445	2,478

(a) Based on these predictions, calculate the evaluation measures listed below for each model.

i. The sum of squared errors

The sum of squared errors for Model 1 can be calculated as follows.

		Model 1					
ID	Target	Pred.	Error	Error^2	|Error|	SST	SST^2
1	2,623.4	2,664.3	40.9	1,674.2	40.9	173.5	30,089.5
2	2,423.0	2,435.9	12.9	167.4	12.9	-54.9	3,017.9
3	2,423.3	2,398.5	-24.8	615.0	24.8	-92.3	8,528.0
4	2,448.1	2,447.1	-1.1	1.2	1.1	-43.8	1,918.8
5	2,761.7	2,847.3	85.7	7,335.9	85.7	356.4	127,043.9
6	2,434.9	2,411.2	-23.7	560.9	23.7	-79.6	6,341.4
7	2,519.0	2,516.4	-2.6	6.7	2.6	25.5	652.8
8	2,771.6	2,870.2	98.6	9,721.7	98.6	379.4	143,913.2
9	2,601.4	2,585.9	-15.6	242.0	15.6	95.0	9,028.8
10	2,422.3	2,414.2	-8.1	65.0	8.1	-76.7	5,875.6
11	2,348.8	2,406.7	57.9	3,352.0	57.9	-84.1	7,079.6
12	2,514.7	2,505.2	-9.4	89.3	9.4	14.4	206.2
13	2,548.4	2,581.2	32.8	1,075.2	32.8	90.3	8,157.2
14	2,281.4	2,276.9	-4.5	20.4	4.5	-214.0	45,776.8
15	2,295.1	2,279.7	-15.4	238.5	15.4	-211.2	44,597.1
16	2,570.5	2,576.6	6.1	37.2	6.1	85.7	7,346.2
17	2,528.1	2,510.2	-17.9	320.8	17.9	19.4	375.1
18	2,342.2	2,380.9	38.7	1,496.9	38.7	-110.0	12,093.6
19	2,456.0	2,452.1	-3.9	15.1	3.9	-38.8	1,501.8
20	2,451.1	2,436.7	-14.4	208.5	14.4	-54.2	2,934.9
21	2,295.8	2,307.2	11.4	129.8	11.4	-183.7	33,730.7
22	2,405.0	2,354.9	-50.1	2,514.9	50.1	-136.0	18,492.1
23	2,388.9	2,418.1	29.2	853.2	29.2	-72.8	5,297.2
24	2,629.5	2,582.4	-47.1	2,215.7	47.1	91.5	8,380.0
25	2,583.8	2,563.5	-20.3	411.7	20.3	72.7	5,281.6
26	2,658.2	2,662.0	3.9	15.1	3.9	171.2	29,298.7
27	2,482.3	2,491.8	9.4	88.6	9.4	0.9	0.8
28	2,470.8	2,477.7	6.9	47.7	6.9	-13.1	172.6
29	2,604.9	2,619.8	14.9	221.7	14.9	128.9	16,624.4
30	2,441.6	2,444.9	3.3	10.9	3.3	-46.0	2,117.8
Mean	2,490.9	2,497.3	6.5	1,125.1	23.7	6.5	19,529.1
Std Dev	127.0	142.0	33.5	2,204.9	24.1	142.0	34,096.6

$$\begin{aligned} sum\ of\ squared\ errors &= \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \\ &= 16{,}876.6 \end{aligned}$$

The sum of squared errors for Model 2 can be calculated as follows.

ID	Target	Model 1 Prediction	Error	Error^2	|Error|	SST	SST^2
1	2,623.4	2,690.6	67.2	4,511.2	67.2	199.7	39,884.8
2	2,423.0	2,367.4	-55.6	3,095.8	55.6	-123.5	15,255.8
3	2,423.3	2,412.2	-11.1	123.5	11.1	-78.7	6,187.8
4	2,448.1	2,439.9	-8.3	68.7	8.3	-51.0	2,602.4
5	2,761.7	2,693.1	-68.6	4,704.4	68.6	202.2	40,882.2
6	2,434.9	2,493.0	58.1	3,374.5	58.1	2.1	4.6
7	2,519.0	2,598.1	79.1	6,253.1	79.1	107.2	11,494.6
8	2,771.6	2,813.8	42.2	1,781.3	42.2	323.0	104,307.2
9	2,601.4	2,582.9	-18.5	343.9	18.5	92.0	8,469.5
10	2,422.3	2,485.1	62.8	3,940.2	62.8	-5.8	33.8
11	2,348.8	2,471.7	122.9	15,104.0	122.9	-19.1	366.3
12	2,514.7	2,583.6	68.9	4,749.9	68.9	92.7	8,598.5
13	2,548.4	2,604.0	55.6	3,091.3	55.6	113.1	12,797.6
14	2,281.4	2,309.4	28.0	783.9	28.0	-181.4	32,919.1
15	2,295.1	2,296.0	0.9	0.8	0.9	-194.8	37,962.7
16	2,570.5	2,611.7	41.2	1,697.4	41.2	120.8	14,595.0
17	2,528.1	2,557.1	29.0	839.9	29.0	66.3	4,390.0
18	2,342.2	2,420.5	78.3	6,135.2	78.3	-70.3	4,946.7
19	2,456.0	2,392.5	-63.5	4,027.2	63.5	-98.3	9,669.3
20	2,451.1	2,478.7	27.6	760.6	27.6	-12.2	147.8
21	2,295.8	2,290.0	-5.8	34.1	5.8	-200.9	40,358.2
22	2,405.0	2,490.4	85.4	7,286.1	85.4	-0.5	0.2
23	2,388.9	2,345.5	-43.3	1,878.7	43.3	-145.3	21,122.7
24	2,629.5	2,646.7	17.2	295.6	17.2	155.8	24,275.8
25	2,583.8	2,546.3	-37.6	1,410.9	37.6	55.4	3,069.4
26	2,658.2	2,759.3	101.1	10,227.4	101.1	268.4	72,047.6
27	2,482.3	2,462.8	-19.5	381.8	19.5	-28.1	787.8
28	2,470.8	2,403.4	-67.4	4,542.6	67.4	-87.4	7,645.6
29	2,604.9	2,644.9	40.0	1,601.7	40.0	154.1	23,736.8
30	2,441.6	2,478.0	36.4	1,327.0	36.4	-12.9	166.2
Mean	2,490.9	2,512.3	21.4	3,145.8	48.0	21.4	18,290.9
Std Dev	127.0	135.8	52.7	3,382.5	29.4	135.8	23,625.8

$$\begin{aligned} sum\ of\ squared\ errors &= \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \\ &= 47{,}186.3 \end{aligned}$$

ii. The R^2 measure

The R^2 measure is calculated as

$$R^2 = 1 - \frac{sum\ of\ squared\ errors}{total\ sum\ of\ squares}$$

The sum of squared error values from the previous part can be used. So, for Model 1,

$$\begin{aligned} total\ sum\ of\ squares &= \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2 \\ &= 292{,}937.1 \end{aligned}$$

and

$$\begin{aligned} R^2 &= 1 - \frac{16{,}876.6}{292{,}937.1} \\ &= 0.942 \end{aligned}$$

For Model 2,

$$total\ sum\ of\ squares = 274{,}363.1$$

and

$$\begin{aligned} R^2 &= 1 - \frac{47{,}186.3}{274{,}363.1} \\ &= 0.828 \end{aligned}$$

(b) Based on the evaluation measures calculated, which model do you think is performing better for this dataset?

Model 1 has a higher R^2 value than Model 2, 0.942 compared to 0.828, which indicates that it is better able to capture the pattern in this dataset. An R^2 value this high suggests quite a powerful model.

3. A credit card issuer has built two different credit scoring models that predict the propensity of customers to default on their loans. The outputs of the first model for a test dataset are shown in the table below.

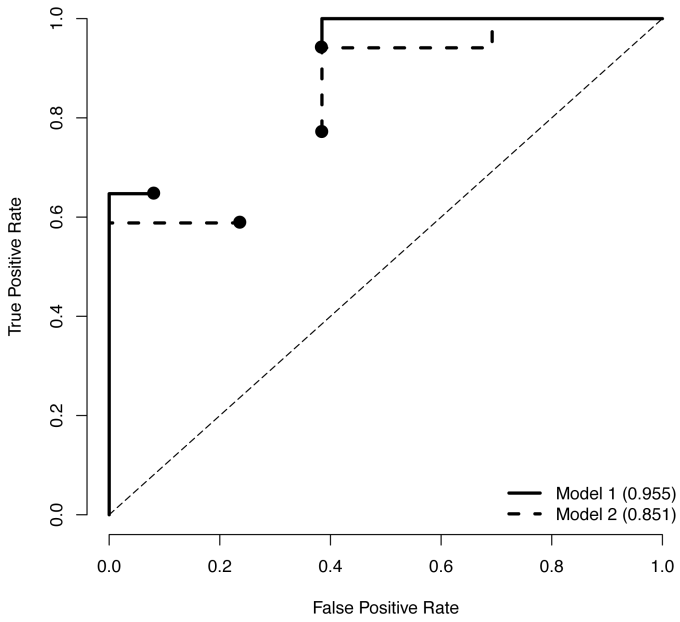
ID	Target	Score	Prediction	ID	Target	Score	Prediction
1	bad	0.634	bad	16	good	0.072	good
2	bad	0.782	bad	17	bad	0.567	bad
3	good	0.464	good	18	bad	0.738	bad
4	bad	0.593	bad	19	bad	0.325	good
5	bad	0.827	bad	20	bad	0.863	bad
6	bad	0.815	bad	21	bad	0.625	bad
7	bad	0.855	bad	22	good	0.119	good
8	good	0.500	good	23	bad	0.995	bad
9	bad	0.600	bad	24	bad	0.958	bad
10	bad	0.803	bad	25	bad	0.726	bad
11	bad	0.976	bad	26	good	0.117	good
12	good	0.504	bad	27	good	0.295	good
13	good	0.303	good	28	good	0.064	good
14	good	0.391	good	29	good	0.141	good
15	good	0.238	good	30	good	0.670	bad

The outputs of the second model for the same test dataset are shown in the table below.

ID	Target	Score	Prediction	ID	Target	Score	Prediction
1	bad	0.230	bad	16	good	0.421	bad
2	bad	0.859	good	17	bad	0.842	good
3	good	0.154	bad	18	bad	0.891	good
4	bad	0.325	bad	19	bad	0.480	bad
5	bad	0.952	good	20	bad	0.340	bad
6	bad	0.900	good	21	bad	0.962	good
7	bad	0.501	good	22	good	0.238	bad
8	good	0.650	good	23	bad	0.362	bad
9	bad	0.940	good	24	bad	0.848	good
10	bad	0.806	good	25	bad	0.915	good
11	bad	0.507	good	26	good	0.096	bad
12	good	0.251	bad	27	good	0.319	bad
13	good	0.597	good	28	good	0.740	good
14	good	0.376	bad	29	good	0.211	bad
15	good	0.285	bad	30	good	0.152	bad

Based on the predictions of these models, perform the following tasks to compare their performance.

(a) The image below shows an **ROC curve** for each model. Each curve has a point missing.



Calculate the missing point in the ROC curves for Model 1 and Model 2. To generate the point for Model 1, use a threshold value of 0.51. To generate the point for Model 2, use a threshold value of 0.43.

To plot an ROC curve, it is easiest to sort the data according to the prediction scores generated. Based on the threshold being used to find a point for the ROC plot, we can create a new set of predictions from which we will calculate the true positive rate (TPR) and the false positive rate (FPR) that are used to plot the ROC curve. The table below shows the prediction scores for Model 1 in ascending order, the new predictions based on a threshold of 0.51, as well as the outcome of each of these predictions—whether it is a true positive (TP), false positive (FP), true negative (TN), or false negative (FN).

ID	Target	Score	Prediction	Outcome
28	good	0.064	good	TN
16	good	0.072	good	TN
26	good	0.117	good	TN
22	good	0.119	good	TN
29	good	0.141	good	TN
15	good	0.238	good	TN
27	good	0.295	good	TN
13	good	0.303	good	TN
19	bad	0.325	good	FN
14	good	0.391	good	TN
3	good	0.464	good	TN
8	good	0.500	good	TN
12	good	0.504	good	FP
17	bad	0.567	bad	TP
4	bad	0.593	bad	TP
9	bad	0.600	bad	TP
21	bad	0.625	bad	TP
1	bad	0.634	bad	TP
30	good	0.670	bad	FP
25	bad	0.726	bad	TP
18	bad	0.738	bad	TP
2	bad	0.782	bad	TP
10	bad	0.803	bad	TP
6	bad	0.815	bad	TP
5	bad	0.827	bad	TP
7	bad	0.855	bad	TP
20	bad	0.863	bad	TP
24	bad	0.958	bad	TP
11	bad	0.976	bad	TP
23	bad	0.995	bad	TP

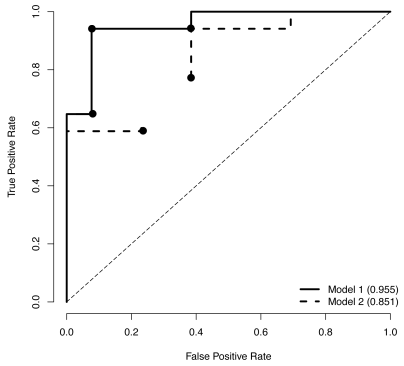
Based on these predictions, we can build a confusion matrix from which we can calculate the true positive rate and false positive rate that we use to plot a point in ROC space. The confusion matrix for Model 1 using a threshold of 0.48 is shown below.

		Prediction	
		bad	*good*
Target	*bad*	16	1
	good	1	12

So we can calculate the TPR and FPR as

$$\begin{aligned} TPR &= \frac{16}{16+1} = 0.9412 \\ FPR &= \frac{1}{12+1} = 0.0769 \end{aligned}$$

Using these figures, we can plot an extra point on the ROC curve and connect it to the existing points to complete the curve (other points for other thresholds are required to complete the curve, but they all result in the same TPR score and so a horizontal line).



The table below shows the prediction scores for Model 2 in ascending order, the new predictions based on a threshold of 0.43, as well as the outcome of each of these predictions—whether it is a true positive (TP), false positive (FP), true negative (TN), or false negative (FN).

ID	Target	Score	Prediction	Outcome
26	good	0.096	good	TN
30	good	0.152	good	TN
3	good	0.154	good	TN
29	good	0.211	good	TN
1	bad	0.230	good	FN
22	good	0.238	good	TN
12	good	0.251	good	TN
15	good	0.285	good	TN
27	good	0.319	good	TN
4	bad	0.325	good	FN
20	bad	0.340	good	FN
23	bad	0.362	good	FN
14	good	0.376	good	TN
16	good	0.421	good	TN
19	bad	0.480	bad	TP
7	bad	0.501	bad	TP
11	bad	0.507	bad	TP
13	good	0.597	bad	FP
8	good	0.650	bad	FP
28	good	0.740	bad	FP
10	bad	0.806	bad	TP
17	bad	0.842	bad	TP
24	bad	0.848	bad	TP
2	bad	0.859	bad	TP
18	bad	0.891	bad	TP
6	bad	0.900	bad	TP
25	bad	0.915	bad	TP
9	bad	0.940	bad	TP
5	bad	0.952	bad	TP
21	bad	0.962	bad	TP

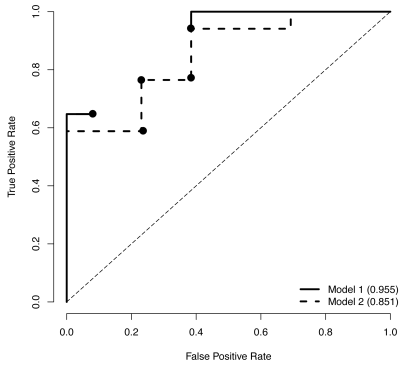
Based on these predictions, we can build a confusion matrix from which we can calculate the true positive rate and false positive rate that we use to plot a point in ROC space. The confusion matrix for Model 2 using a threshold of 0.48 is shown below.

		Prediction	
		bad	*good*
Target	*bad*	13	4
	good	3	10

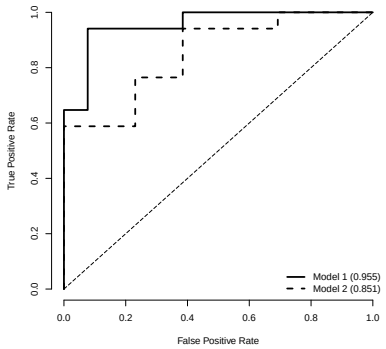
So we can calculate the TPR and FPR as

$$\begin{aligned} TPR &= \frac{13}{13+4} = 0.7647 \\ FPR &= \frac{3}{10+3} = 0.2308 \end{aligned}$$

Using these figures, we can plot an extra point on the ROC curve and connect it to the existing points to complete the curve (other points for other thresholds are required to complete the curve, but they all result in the same TPR score and so a horizontal line).



For completeness, we show both complete curves together below.



(b) The **area under the ROC curve** (AUC) for Model 1 is 0.955 and for Model 2 is 0.851. Which model is performing best?

Based on the higher AUC, we can conclude that Model 1 is performing better at this task. Furthermore, the ROC curve for Model 1 dominates the curve for Model 2 (i.e., is always higher), which means that there is no operating point (or threshold value) for which Model 2 is better.

(c) Based on the AUC values for Model 1 and Model 2, calculate the **Gini coefficient** for each model.

The Gini coefficient is calculated as

$$Gini\ coefficient = (2 \times ROC\ index) - 1$$

So for Model 1, the Gini coefficient is

$$Gini\ coefficient = (2 \times 0.955) - 1 \ = 0.91$$

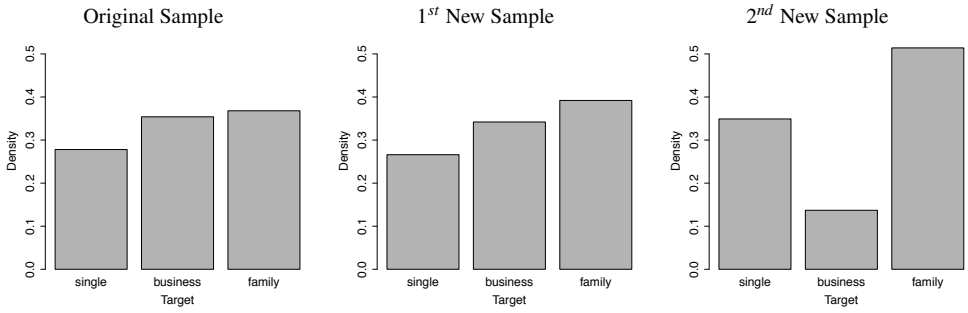
For Model 2, the Gini coefficient is

$$Gini\ coefficient = (2 \times 0.851) - 1 \ = 0.702$$

4. A retail supermarket chain has built a prediction model that recognizes the household that a customer comes from as being one of *single*, *business*, or *family*. After deployment, the analytics team at the supermarket chain uses the **stability index** to monitor the performance of this model. The table below shows the frequencies of predictions of the three different levels made by the model for the original validation dataset at the time the model was built, for the month after deployment, and for a monthlong period six months after deployment.

Target	Original Sample	1^{st} New Sample	2^{nd} New Sample
single	123	252	561
business	157	324	221
family	163	372	827

Bar plots of these three sets of prediction frequencies are shown in the following images.



Calculate the **stability index** for the two new periods and determine whether the model should be retrained at either of these points.

The stability index is calculated as

$$stability\ index = \sum_{l \in levels} \left(\left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} - \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \times log_e \left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} / \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \right)$$

where l is a target level, $|\mathcal{A}|$ refers to the size of the test set on which performance measures were originally calculated, $|\mathcal{A}_{t=l}|$ refers to the number of instances in the original test set for which the model made a prediction of level l for target t, $|\mathcal{B}|$ and $|\mathcal{B}_{t=l}|$ refer to the same measurements on the newly collected dataset. The following table shows the components of calculating this for the two new periods.

	Original		1st New Sample			2nd New Sample		
Target	Count	%	Count	%	SI_t	Count	%	SI_t
single	123	0.2777	252	0.2658	0.00052	561	0.3487	0.01617
business	157	0.3544	324	0.3418	0.00046	221	0.1374	0.20574
family	163	0.3679	372	0.3924	0.00157	827	0.5140	0.04881
Sum	443		948		0.003	1,609		0.271

For the first new sample, the stability index is 0.003, which indicates that there is practically no difference between the distribution of target levels predicted for the original validation dataset and for the data in the newt period.
For the second sample, the stability index is 0.271, which indicates a massive difference between the distribution of target levels at this point in time compared to the distribution predicted for the original validation set. This suggests that concept drift has occurred and that the model should be retrained.

III BEYOND PREDICTION

10 Beyond Prediction: Unsupervised Learning (Exercise Solutions)

1. The following table shows a small dataset in which each instance describes measurements taken using three sensors when a valve in an oil well was opened. The three descriptive features, PRESSURE, TEMPERATURE, and VOLUME measure characteristics of the oil flowing through the valve when it was opened. The ***k*-means clustering** approach is to be applied to this dataset with $k = 3$ and using **Euclidean distance**. The initial cluster centroids for the three clusters C_1, C_2 , and C_3 are $\mathbf{c}_1 = \langle -0.929, -1.040, -0.831 \rangle$, $\mathbf{c}_2 = \langle -0.329, -1.099, 0.377 \rangle$, and $\mathbf{c}_3 = \langle -0.672, -0.505, 0.110 \rangle$. The following table also shows the distance to these three cluster centers for each instance in the dataset.

				Cluster Distances Iter. 1		
ID	PRESSURE	TEMPERATURE	VOLUME	$Dist(\mathbf{d}_i, \mathbf{c}_1)$	$Dist(\mathbf{d}_i, \mathbf{c}_2)$	$Dist(\mathbf{d}_i, \mathbf{c}_3)$
1	-0.392	-1.258	-0.666	0.603	1.057	1.117
2	-0.251	-1.781	-1.495	1.204	1.994	2.093
3	-0.823	-0.042	1.254	2.314	1.460	1.243
4	0.917	-0.961	0.055	2.049	1.294	1.654
5	-0.736	-1.694	-0.686	0.697	1.284	1.432
6	1.204	-0.605	0.351	2.477	1.611	1.894
7	0.778	-0.436	-0.220	1.911	1.422	1.489
8	1.075	-1.199	-0.141	2.125	1.500	1.896
9	-0.854	-0.654	0.771	1.650	0.793	0.702
10	-1.027	-0.269	0.893	1.891	1.201	0.892
11	-0.288	-2.116	-1.165	1.296	1.848	2.090
12	-0.597	-1.577	-0.618	0.666	1.136	1.298
13	-1.113	-0.271	0.930	1.930	1.267	0.960
14	-0.849	-0.430	0.612	1.569	0.879	0.538
15	1.280	-1.188	0.053	2.384	1.644	2.069

(a) Assign each instance to its nearest cluster to generate the clustering at the first iteration of *k*-means on the basis of the initial cluster centroids.

The clusters to be assigned to each instance are shown below.

				Cluster Distances Iter. 1			Iter. 1
ID	PRESSURE	TEMP.	VOLUME	$Dist(\mathbf{d}_i, \mathbf{c}_1)$	$Dist(\mathbf{d}_i, \mathbf{c}_2)$	$Dist(\mathbf{d}_i, \mathbf{c}_3)$	Cluster
1	-0.392	-1.258	-0.666	0.603	1.057	1.117	C_1
2	-0.251	-1.781	-1.495	1.204	1.994	2.093	C_1
3	-0.823	-0.042	1.254	2.314	1.460	1.243	C_3
4	0.917	-0.961	0.055	2.049	1.294	1.654	C_2
5	-0.736	-1.694	-0.686	0.697	1.284	1.432	C_1
6	1.204	-0.605	0.351	2.477	1.611	1.894	C_2
7	0.778	-0.436	-0.220	1.911	1.422	1.489	C_2
8	1.075	-1.199	-0.141	2.125	1.500	1.896	C_2
9	-0.854	-0.654	0.771	1.650	0.793	0.702	C_3
10	-1.027	-0.269	0.893	1.891	1.201	0.892	C_3
11	-0.288	-2.116	-1.165	1.296	1.848	2.090	C_1
12	-0.597	-1.577	-0.618	0.666	1.136	1.298	C_1
13	-1.113	-0.271	0.930	1.930	1.267	0.960	C_3
14	-0.849	-0.430	0.612	1.569	0.879	0.538	C_3
15	1.280	-1.188	0.053	2.384	1.644	2.069	C_2

(b) On the basis of the clustering calculated in Part (a), calculate a set of new cluster centroids.

The instances that have been assigned to the first cluster, C_1, are

ID	PRESSURE	TEMPERATURE	VOLUME
1	-0.392	-1.258	-0.666
2	-0.251	-1.781	-1.495
5	-0.736	-1.694	-0.686
11	-0.288	-2.116	-1.165
12	-0.597	-1.577	-0.618

And, so, the new cluster centroid is the average of these values:

$\mathbf{c}_1 = \langle -0.453, -1.685, -0.926 \rangle$.

The instances that have been assigned to the second cluster, C_2, are

ID	PRESSURE	TEMPERATURE	VOLUME
4	0.917	-0.961	0.055
6	1.204	-0.605	0.351
7	0.778	-0.436	-0.220
8	1.075	-1.199	-0.141
15	1.280	-1.188	0.053

And, so, the new cluster centroid is the average of these values:

$\mathbf{c}_2 = \langle 1.051, -0.878, 0.020 \rangle$

The instances that have been assigned to the third cluster, C_3, are

ID	PRESSURE	TEMPERATURE	VOLUME
3	-0.823	-0.042	1.254
9	-0.854	-0.654	0.771
10	-1.027	-0.269	0.893
13	-1.113	-0.271	0.930
14	-0.849	-0.430	0.612

And, so, the new cluster centroid is the average of these values:

$\mathbf{c}_3 = \langle -0.933, -0.333, 0.892 \rangle$.

2. The following table shows details of two different clusterings of the dataset from Question 1—one with $k = 2$ and one with $k = 3$—and partial workings to calculate the silhouette for the clusterings.

	$k = 2$ clustering					$k = 3$ clustering				
		Nearest					Nearest			
ID	Cluster	Cluster	$a(i)$	$b(i)$	$s(i)$	ID	Cluster	Cluster	$a(i)$	$b(i)$	$s(i)$
$\mathbf{d}_1$	C_1	C_2	??	1.898	??	$\mathbf{d}_1$	C_1	C_2	0.732	1.681	0.565
$\mathbf{d}_2$	C_1	C_2	1.608	2.879	0.442	$\mathbf{d}_2$	C_1	??	??	??	??
$\mathbf{d}_3$	C_2	C_1	0.624	2.594	0.76	$\mathbf{d}_3$	C_3	C_2	0.624	2.422	0.742
$\mathbf{d}_4$	C_1	C_2	1.261	2.142	0.411	$\mathbf{d}_4$	C_2	C_1	0.482	1.884	??
$\mathbf{d}_5$	C_1	C_2	1.452	2.098	0.308	$\mathbf{d}_5$	C_1	C_3	0.619	2.098	0.705
$\mathbf{d}_6$	C_1	??	??	??	??	$\mathbf{d}_6$	C_2	C_3	0.68	2.24	0.697
$\mathbf{d}_7$	C_1	C_2	1.42	2.061	0.311	$\mathbf{d}_7$	C_2	C_1	0.777	1.935	0.598
$\mathbf{d}_8$	C_1	C_2	1.272	2.432	??	$\mathbf{d}_8$	C_2	C_1	0.558	1.842	0.697
$\mathbf{d}_9$	C_2	C_1	0.496	2.067	0.76	$\mathbf{d}_9$	C_3	C_1	0.496	2.04	0.757
$\mathbf{d}_{10}$	C_2	C_1	0.344	2.375	??	$\mathbf{d}_{10}$	C_3	??	0.344	??	??
$\mathbf{d}_{11}$	C_1	C_2	1.565	2.802	0.441	$\mathbf{d}_{11}$	C_1	C_2	0.769	2.201	0.651
$\mathbf{d}_{12}$	C_1	C_2	1.338	??	??	$\mathbf{d}_{12}$	C_1	C_2	0.592	1.935	0.694
$\mathbf{d}_{13}$	C_2	C_1	0.379	2.444	0.845	$\mathbf{d}_{13}$	C_3	C_1	0.379	2.436	0.844
$\mathbf{d}_{14}$	C_2	C_1	??	2.056	??	$\mathbf{d}_{14}$	C_3	C_1	0.459	2.038	??
$\mathbf{d}_{15}$	C_1	C_2	1.425	2.53	0.437	$\mathbf{d}_{15}$	C_2	C_1	0.579	2.101	0.725

(a) A number of values are missing from these workings (indicated by ??). Calculate the missing values. The distances between each instance in the dataset from Question 1 (using **Euclidean distance**) are shown in the following distance matrix, and will be useful for this exercise.

	$\mathbf{d}_1$	$\mathbf{d}_2$	$\mathbf{d}_3$	$\mathbf{d}_4$	$\mathbf{d}_5$	$\mathbf{d}_6$	$\mathbf{d}_7$	$\mathbf{d}_8$	$\mathbf{d}_9$	$\mathbf{d}_{10}$	$\mathbf{d}_{11}$	$\mathbf{d}_{12}$	$\mathbf{d}_{13}$	$\mathbf{d}_{14}$	$\mathbf{d}_{15}$
$\mathbf{d}_1$	0.00														
$\mathbf{d}_2$	0.99	0.00													
$\mathbf{d}_3$	2.31	3.30	0.00												
$\mathbf{d}_4$	1.52	2.11	2.30	0.00											
$\mathbf{d}_5$	0.56	0.95	2.55	1.95	0.00										
$\mathbf{d}_6$	2.00	2.63	2.29	0.55	2.46	0.00									
$\mathbf{d}_7$	1.50	2.12	2.21	0.61	2.02	0.73	0.00								
$\mathbf{d}_8$	1.56	1.98	2.62	0.35	1.96	0.78	0.82	0.00							
$\mathbf{d}_9$	1.63	2.60	0.78	1.94	1.79	2.10	1.92	2.20	0.00						
$\mathbf{d}_{10}$	1.95	2.93	0.47	2.23	2.15	2.32	2.13	2.52	0.44	0.00					
$\mathbf{d}_{11}$	1.00	0.47	3.23	2.07	0.78	2.61	2.20	1.94	2.49	2.86	0.00				
$\mathbf{d}_{12}$	0.38	0.96	2.43	1.77	0.19	2.26	1.83	1.78	1.69	2.04	0.83	0.00			
$\mathbf{d}_{13}$	2.01	2.98	0.49	2.32	2.19	2.41	2.22	2.61	0.49	0.09	2.91	2.09	0.00		
$\mathbf{d}_{14}$	1.59	2.57	0.75	1.93	1.81	2.08	1.83	2.21	0.28	0.37	2.51	1.70	0.44	0.00	
$\mathbf{d}_{15}$	1.82	2.26	2.68	0.43	2.21	0.66	0.94	0.28	2.31	2.62	2.19	2.03	2.71	2.33	0.00

To illustrate how this exercise is completed the complete workings for calculating the silhouette with for instance $\mathbf{d}_6$ will be shown. In this clustering

$\mathbf{d}_6$ is a member of C_1. The first step is calculating the silhouette value for $\mathbf{d}_6$ is to calculate $a(i)$, the average distance between $\mathbf{d}_6$ and the other members of cluster C_1. The members of C_1 are $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_{11}, \mathbf{d}_{12}, \mathbf{d}_{15}\}$. From the distance matrix provided, the distance from $\mathbf{d}_6$ to each other member of C_1 is:

$$\begin{array}{c} \\ \mathbf{d}_6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_4 & \mathbf{d}_5 & \mathbf{d}_6 & \mathbf{d}_8 & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{15} \end{array} \\ \left[\begin{array}{ccccccccc} 2.00 & 2.63 & 0.55 & 2.46 & 0.73 & 0.78 & 2.61 & 2.26 & 0.66 \end{array}\right] \end{array}$$

The average distance between $\mathbf{d}_6$ and the other members of C_1, $a(i)$, is the average of these values: 1.631.

The next step in the algorithm is to calculate the average distance from $\mathbf{d}_6$ to each member of the other clusters in the clustering. In this case there is just one, C_2. The distances from $\mathbf{d}_6$ to the members of C_2 are:

$$\begin{array}{c} \\ \mathbf{d}_6 \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{d}_3 & \mathbf{d}_9 & \mathbf{d}_{10} & \mathbf{d}_{13} & \mathbf{d}_{14} \end{array} \\ \left[\begin{array}{ccccc} 2.29 & 2.10 & 2.32 & 2.41 & 2.08 \end{array}\right] \end{array}$$

which gives an average of 2.240 which is the value for $b(i)$ as there is only one other cluster. The silhouette width for $\mathbf{d}_6$ is then calculated as

$$\frac{2.240 - 1.631}{max(2.240, 1.631)} = 0.272$$

Other values are calculated similarly. The completed table of all values is shown below.

		$k = 2$ clustering Nearest			
ID	Cluster	Cluster	$a(i)$	$b(i)$	$s(i)$
$\mathbf{d}_1$	C_1	C_2	1.259	1.898	0.337
$\mathbf{d}_2$	C_1	C_2	1.608	2.879	0.442
$\mathbf{d}_3$	C_2	C_1	0.624	2.594	0.76
$\mathbf{d}_4$	C_1	C_2	1.261	2.142	0.411
$\mathbf{d}_5$	C_1	C_2	1.452	2.098	0.308
$\mathbf{d}_6$	C_1	C_2	1.631	2.24	0.272
$\mathbf{d}_7$	C_1	C_2	1.42	2.061	0.311
$\mathbf{d}_8$	C_1	C_2	1.272	2.432	0.477
$\mathbf{d}_9$	C_2	C_1	0.496	2.067	0.76
$\mathbf{d}_{10}$	C_2	C_1	0.344	2.375	0.855
$\mathbf{d}_{11}$	C_1	C_2	1.565	2.802	0.441
$\mathbf{d}_{12}$	C_1	C_2	1.338	1.991	0.328
$\mathbf{d}_{13}$	C_2	C_1	0.379	2.444	0.845
$\mathbf{d}_{14}$	C_2	C_1	0.459	2.056	0.776
$\mathbf{d}_{15}$	C_1	C_2	1.425	2.53	0.437

		$k = 3$ clustering			
		Nearest			
ID	Cluster	Cluster	$a(i)$	$b(i)$	$s(i)$
$\mathbf{d}_1$	C_1	C_2	0.732	1.681	0.565
$\mathbf{d}_2$	C_1	C_2	0.843	2.219	0.62
$\mathbf{d}_3$	C_3	C_2	0.624	2.422	0.742
$\mathbf{d}_4$	C_2	C_1	0.482	1.884	0.744
$\mathbf{d}_5$	C_1	C_3	0.619	2.098	0.705
$\mathbf{d}_6$	C_2	C_3	0.68	2.24	0.697
$\mathbf{d}_7$	C_2	C_1	0.777	1.935	0.598
$\mathbf{d}_8$	C_2	C_1	0.558	1.842	0.697
$\mathbf{d}_9$	C_3	C_1	0.496	2.04	0.757
$\mathbf{d}_{10}$	C_3	C_2	0.344	2.363	0.855
$\mathbf{d}_{11}$	C_1	C_2	0.769	2.201	0.651
$\mathbf{d}_{12}$	C_1	C_2	0.592	1.935	0.694
$\mathbf{d}_{13}$	C_3	C_1	0.379	2.436	0.844
$\mathbf{d}_{14}$	C_3	C_1	0.459	2.038	0.775
$\mathbf{d}_{15}$	C_2	C_1	0.579	2.101	0.725

(b) On the basis of the completed table, calculate the silhouette for each clustering.

The silhouette for each clustering is simply the average of the silhouette entries. For the clustering with $k = 2$ this is 0.517, and for the clustering with $k = 3$ this is 0.711.

(c) On the basis of the silhouette, would you choose 2 or 3 for the value of k for this dataset?

As it has a higher silhouette there is more evidence that three clusters exist in this dataset than two. So $k = 3$ should be chosen.

3. A city tax service has performed a clustering of individual taxpayers using k-means clustering in order to better understand groups that might exist within their taxpayer base. The clustering has divided the taxpayers into three clusters. Four descriptive features have been used to describe each taxpayer:

   - AGE: The age of the taxpayer.
   - YEARSINCURRENTEMPLOYMENT: The number of years that the taxpayer has been in their current job.
   - TOTALINCOME: The taxpayer's total income for the current tax year.
   - EFFECTIVETAXRATE: The effective tax rate paid by the taxpayer (this is simply tax paid divided by total income).

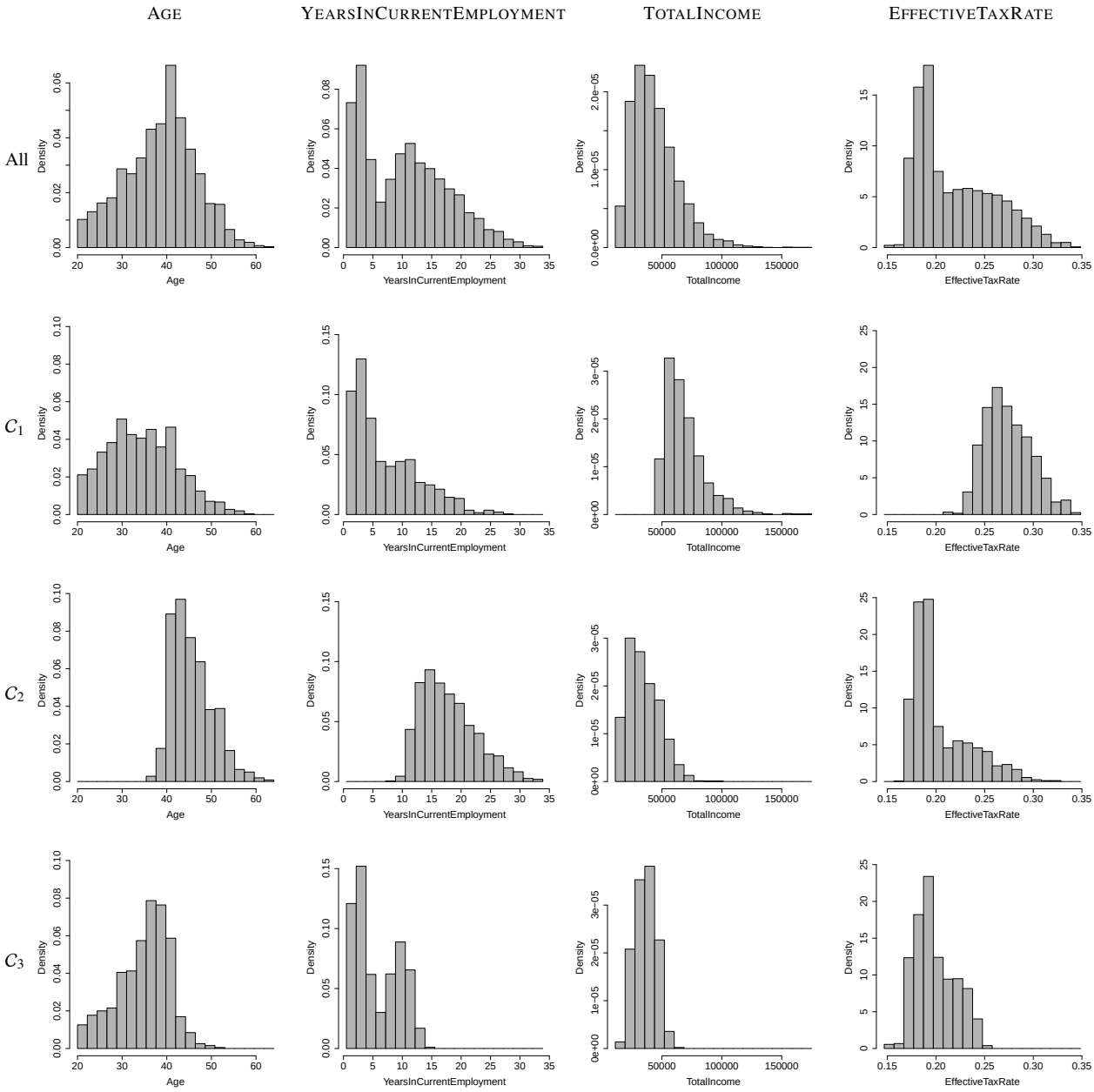
   The following table shows summary statistics of the four descriptive features for each of the three clusters found.

Feature	Cluster	Min.	1^{st} Qrt.	Mean	Median	3^{rd} Qrt.	Max	Std. Dev.
AGE	C_1	20	28	34.6	34	40	59	7.8
	C_2	36	43	45.8	45	48	64	4.5
	C_3	20	32	34.9	36	39	52	5.8
YEARSIN CURRENT EMPLOYMENT	C_1	0.50	2.74	7.18	5.11	10.76	27.40	5.56
	C_2	8.16	14.25	17.81	17.04	20.71	33.89	4.60
	C_3	0.50	2.44	5.73	4.38	9.32	14.12	3.73
TOTALINCOME	C_1	46 247.70	57 355.06	68 843.26	64 977.64	75 967.11	175 000	16 387.77
	C_2	11 182.46	24 222.04	34 711.67	32 637.42	44 102.08	93 800.98	13 412.08
	C_3	15 505.02	29 636.07	36 370.00	36 421.53	42 912.04	64 075.62	8 744.26
EFFECTIVE TAXRATE	C_1	0.210	0.256	0.274	0.271	0.291	0.349	0.024
	C_2	0.167	0.183	0.204	0.192	0.220	0.321	0.030
	C_3	0.147	0.183	0.199	0.194	0.214	0.252	0.021

   The following table shows the information gain calculated when each descriptive feature is used to predict the membership of a single cluster versus the rest of the population.

	Information Gain		
Feature	C_1	C_2	C_3
AGE	0.0599	0.4106	0.1828
YEARSINCURRENTEMPLOYMENT	0.0481	0.5432	0.3073
TOTALINCOME	0.5015	0.0694	0.1830
EFFECTIVETAXRATE	0.5012	0.0542	0.2166

   The following images show histograms of the values of the four descriptive features both for the full dataset and when divided into the three clusters found.



Using the information provided, write a description of what it means for a taxpayer to be a member of each of the clusters.

To perform this task the best place to start is to analyse the information gain values provided in the final table. These indicate which descriptive features are likely to be most useful in describing each cluster. For C_1 the descriptive features with the highest information gain values are TOTALINCOME and EFFECTIVETAXRATE; for C_2 it is YEARSINCURRENTEMPLOYMENT and AGE; and for C_3 it is YEARSINCURRENTEMPLOYMENT and EFFECTIVETAXRATE. The selection of two descriptive features for each cluster is arbitrary here but seems to be suggested by the information gain values. We will always use the other descriptive features in our descriptions too.

Using each of these pairs of descriptive features an examination of the summary statistics in the table provided and the histograms can help understand how to define each of the clusters found. For example, the first cluster, C_1, can be largely defined as containing taxpayers with relatively large incomes who pay a high effective tax rate. They can also be seen to be relatively young and not long in their current positions. It is common in this type of application of clustering to name clusters so as to easily capture their meaning. We could describe these taxpayers as *high flyers* given their youth and high earnings.

Similarly, we can describe the members of the second cluster, C_2, as being older and having stayed in their current potions for a long time. They also tend towards lower incomes and pay slightly low tax rates, although this distribution has a long tail. We might describe these as *golden years* taxpayers.

Finally, members of the the third cluster, C_3, are largely defined by not having been long in their current jobs and paying a fairly low effective tax rate. They also tend to be in their thirties and have relatively low incomes. We might describe these taxpayers as being those in the *middle*.

11 Beyond Prediction: Reinforcement Learning (Exercise Solutions)

1. An agent in an environment completes an episode and receives the following rewards:

$$\{r_0 = -33, r_1 = -11, r_2 = -12, r_3 = 27, r_4 = 87, r_5 = 156\}$$

   (a) Calculate the discounted return at time $t = 0$ on the basis of this sequence of rewards using a discounting factor of 0.72.

> We can apply Equation (11.8)[642] to this sequence of rewards to calculate the discount rate, as viewed from $t = 0$ where $\gamma = 0.72$:
>
> $$\begin{aligned} G &= \left(0.72^0 \times -33\right) + \left(0.72^1 \times -11\right) + \left(0.72^2 \times -12\right) \\ &\quad + \left(0.72^3 \times 27\right) + \left(0.72^4 \times 87\right) + \left(0.72^5 \times 156\right) \\ &= (-33) + (0.72 \times -11) + (0.5184 \times -12) \\ &\quad + (0.3732 \times 27) + (0.2687 \times 87) + (0.1935 \times 156) \\ &= 16.4985 \end{aligned}$$

   (b) Calculate the discounted return at time $t = 0$ on the basis of this sequence of rewards using a discount rate of 0.22.

> We can apply Equation (11.8)[642] to this sequence of rewards to calculate the discounted return, as viewed from $t = 0$ where $\gamma = 0.72$:
>
> $$\begin{aligned} G &= \left(0.22^0 \times -33\right) + \left(0.22^1 \times -11\right) + \left(0.22^2 \times -12\right) \\ &\quad + \left(0.22^3 \times 27\right) + \left(0.22^4 \times 87\right) + \left(0.22^5 \times 156\right) \\ &= (-33) + (0.22 \times -11) + (0.0484 \times -12) \\ &\quad + (0.0106 \times 27) + (0.0023 \times 87) + (0.0005 \times 156) \\ &= -35.4365 \end{aligned}$$

2. To try to better understand the slightly baffling behavior of her new baby, Maria—a scientifically minded new mother—monitored her baby girl over the course of a day

recording her activity at 20 minute intervals. The activity stream looked like this (with time flowing down through the columns):

SLEEPING	SLEEPING	SLEEPING	CRYING	SLEEPING	SLEEPING
CRYING	SLEEPING	HAPPY	HAPPY	CRYING	HAPPY
SLEEPING	SLEEPING	CRYING	HAPPY	SLEEPING	HAPPY
SLEEPING	CRYING	SLEEPING	HAPPY	SLEEPING	HAPPY
SLEEPING	CRYING	SLEEPING	HAPPY	SLEEPING	HAPPY
HAPPY	SLEEPING	HAPPY	HAPPY	SLEEPING	HAPPY
HAPPY	SLEEPING	HAPPY	SLEEPING	HAPPY	HAPPY
HAPPY	HAPPY	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	SLEEPING	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	HAPPY	HAPPY	SLEEPING	SLEEPING	SLEEPING
SLEEPING	HAPPY	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	CRYING	CRYING	SLEEPING	SLEEPING	SLEEPING

Maria noticed that her baby could occupy one of three states—HAPPY, CRYING, or SLEEPING—and moved quite freely between them.

(a) On the basis of this sequence of states, calculate a transition matrix that gives the probability of moving between each of the three states.

The first step to building the transition matrix is to count the frequency of each possible state transition. Working down through the list of states we can count the number of times we move from one state to the next. For example, in the first five states {SLEEPING, CRYING, SLEEPING, SLEEPING, SLEEPING} there is one transition from SLEEPING → CRYING, one transition from CRYING → SLEEPING, and two transitions from SLEEPING → SLEEPING. This would give the following partial transition frequency matrix:

	SLEEPING	CRYING	HAPPY
SLEEPING	2	1	0
CRYING	1	0	0
HAPPY	0	0	0

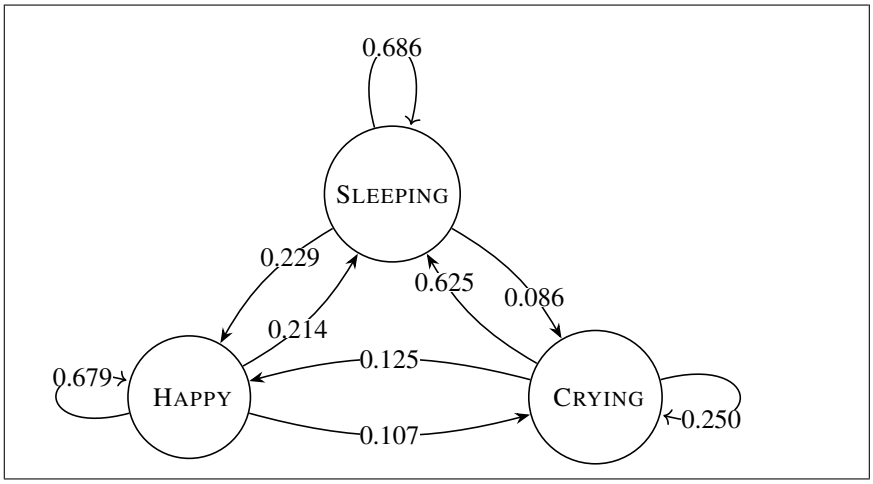
The final frequency table is shown in the table below:

	SLEEPING	CRYING	HAPPY
SLEEPING	24	3	8
CRYING	5	2	1
HAPPY	6	3	19

By normalising each row in the table (dividing each value by the sum of values in the row) we can calculate the final transition matrix:

	SLEEPING	CRYING	HAPPY
SLEEPING	0.686	0.086	0.229
CRYING	0.625	0.250	0.125
HAPPY	0.214	0.107	0.679

(b) Draw a **Markov process** diagram to capture the behavior of a small baby as described.



3. The following table shows the action-value table for a reinforcement learning agent learning to play the **TwentyTwos** game after 20 episodes of training have elapsed.

State	Action	Value	State	Action	Value	State	Action	Value
PL-DL	*Twist*	0.706	PH-DL	*Twist*	−0.038	PM-DH	*Twist*	0.533
PL-DL	*Stick*	0.284	PH-DL	*Stick*	0.164	PM-DH	*Stick*	−0.526
PM-DL	*Twist*	−0.985	PL-DH	*Twist*	0.386	PH-DH	*Twist*	0.154
PM-DL	*Stick*	0.589	PL-DH	*Stick*	−0.832	PH-DH	*Stick*	0.103
BUST	*Twist*	0.000	TIE	*Twist*	0.000	WIN	*Twist*	0.000
BUST	*Stick*	0.000	TIE	*Stick*	0.000	WIN	*Stick*	0.000
LOSE	*Twist*	0.000				TWENTYTWO	*Twist*	0.000
LOSE	*Stick*	0.000				TWENTYTWO	*Stick*	0.000

In the answers to the following questions, assume that after the initial cards have been dealt to the player and the dealer, the following cards are coming up next in the deck: 10♥, 2♣, 7♣, *K*♥, 9♦.

(a) At the beginning of the first episode the player is dealt (2♥, *K*♣), the dealer is dealt (*A*♦, 3♦), and the dealer's visible card is the *A*♦. Given these cards, what state is the TwentyTwos playing agent in?

> The value of the player's hand is 12 and the value of the dealer's visible card is 11 so the agent finds itself in the PL-DH state.

(b) Assuming that the next action that the agent will take is selected using a **greedy action selection policy**, what action will the agent choose to take (*Stick* or *Twist*)?

> To answer this question we need to examine the action-value table to find the expected discounted return for each possible action from this state:
>
> - $Q(\text{PL-DH}, \textit{Twist}) = 0.386$
> - $Q(\text{PL-DH}, \textit{Stick}) = -0.832$
>
> As the expected return for the *Twist* action is higher this is the action that the agent will select using a greedy action selection policy.

(c) Simulate taking the action that the agent selected in Part (b) and determine the state that the agent will move to following this action and the reward that they will receive. (**Note:** If cards need to be dealt to the player or dealer, use cards from the list given at the beginning of this question.)

> The agent selected the *Twist* action and the next card in the deck that will be dealt is the 10♥. So, the cards in the player's hand will now be (2♥, *K*♣, 10♥)

which have a value of 22. This means that the agent will move to the PH-DH state. As this is a non-terminal state the agent will receive a reward of 0.

(d) Assuming that **Q-learning** is being used with $\alpha = 0.2$ and $\gamma = 0.9$, update the entry in the action-value table for the action simulated in Part (c).

To update the $Q(\text{PL-DH}, \textit{Twist})$ entry in the action-value table we can use Equation 11.24[658] as follows:

$$Q(\text{PL-DH}, \textit{Twist}) \leftarrow Q(\text{PL-DH}, \textit{Twist}) + \alpha \times \Big(R(\text{PL-DH}, \textit{Twist}) + \gamma \times \max_a Q(\text{PH-DH}, a) - Q(\text{PL-DH}, \textit{Twist})\Big)$$

To fill in all value in this table we need to determine $\max_a Q(\text{PH-DH}, a)$, the action from PH-DH with the highest expected return. The action value function entries for the two possible actions are:

- $Q(\text{PH-DH}, \textit{Twist}) = 0.154$
- $Q(\text{PH-DH}, \textit{Stick}) = 0.103$

At this point in the learning process the *Twist* has the higher expected return and so $Q(\text{PH-DH}, \textit{Twist})$ is used in the action-value function entry update equation:

$$\begin{aligned} &Q(\text{PL-DH}, \textit{Twist}) \\ &\quad \leftarrow Q(\text{PL-DH}, \textit{Twist}) + \\ &\quad \alpha \times (R(\text{PL-DH}, \textit{Twist}) + \gamma \times Q(\text{PH-DH}, \textit{Twist}) - Q(\text{PL-DH}, \textit{Twist})) \\ &\quad 0.386 + 0.2 \times (0 + 0.9 \times 0.154 - 0.386) \\ &\quad 0.361 \end{aligned}$$

(e) Assuming that a **greedy action selection policy** is used again and that **Q-learning** is still being used with $\alpha = 0.2$ and $\gamma = 0.9$, select the next action that the agent will perform, simulate this action, and update the entry in the action-value table for the action. (**Note:** If cards need to be dealt to the player or dealer, continue to use cards from the list given at the beginning of this question.)

The agent finds itself in the PH-DH state. So, assuming again that a greedy action selection policy is being used it will choose the action with the highest

value in the action-value function table. The entries for *Twist* and *Stick* from this state are:

- $Q(\text{PH-DH}, \textit{Twist}) = 0.154$
- $Q(\text{PH-DH}, \textit{Stick}) = 0.103$

At this point in the learning process the *Twist* has the higher expected return and so $Q(\text{PH-DH}, \textit{Twist})$ will be selected. In this case, unfortunately, this is not a good idea. The player's hand has a total value of 22 so any new card will send them bust. In this instance the card dealt from the deck described above is the 2♣ which takes the player agent in the the BUST state with a reward of -1.

The action-value function table entry for $Q(\text{PH-DH}, \textit{Twist})$ can be updated as:

$$
\begin{aligned}
&Q(\text{PH-DH}, \textit{Twist}) \\
&\qquad \leftarrow Q(\text{PH-DH}, \textit{Twist}) + \\
&\qquad\quad \alpha \times \Big(R(\text{PH-DH}, \textit{Twist}) + \gamma \times \max_a Q(\text{BUST}, a) - Q(\text{PH-DH}, \textit{Twist}) \Big)
\end{aligned}
$$

The BUST state is a terminal state so it doesn't matter which action is selected next as all return an expected return of 0.0. So the update becomes:

$$
\begin{aligned}
&Q(\text{PH-DH}, \textit{Twist}) \\
&\qquad \leftarrow Q(\text{PH-DH}, \textit{Twist}) + \\
&\qquad\quad \alpha \times (R(\text{PH-DH}, \textit{Twist}) + \gamma \times Q(\text{BUST}, \textit{Twist}) - Q(\text{PH-DH}, \textit{Twist})) \\
&\qquad\quad 0.154 + 0.2 \times (-1 + 0.9 \times 0 - 0.154) \\
&\qquad\quad 0.039
\end{aligned}
$$

(f) On the basis of the changes made to the TwentyTwos playing agent's action-value table following the two actions taken in the previous parts of this question, how has the agent's **target policy** changed?

The first action taken didn't change the overall policy in a noticeable way. The action-value function table entry for $Q(\text{PL-DH}, \textit{Twist})$ reduced slightly, but at 0.36 it is still higher than the value for $Q(\text{PL-DH}, \textit{Stick})$ and so a greedy target policy will still select the *Twist*action in that state.

The results of the changes resulting from the second action are more impactful. The choice to *Twist* in the PH-DH state led to the player going bust

and reduced the action-value function table entry for $Q(\text{PH-DH}, \textit{Twist})$ from 0.154 to 0.039. This is less than the entry for $Q(\text{PH-DH}, \textit{Stick})$, at 0.103, and so the target policy from that state will not swap from the *Twist* action to the *Stick* action—probably a moire sensible choice.

Bibliography

Alimoglu, Fevzi, and Ethem Alpaydin. 1996. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the fifth Turkish artificial intelligence and artificial neural networks symposium (TAINN'96)*.

Anguita, Davide, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st international European symposium on artificial neural networks, computational intelligence, and machine learning (ESANN'13)*, 437–442.

Bache, K., and M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Bejar, J., U. Cortés, and M. Poch. 1991. LINNEO+: A classification methodology for ill-structured domains, Research report RT-93-10-R, Dept. Llenguatges i Sistemes Informatics, Universitat Politècnica de Catalunya.

Berk, Richard A., and Justin Bleich. 2013. Statistical procedures for forecasting criminal behavior. *Criminology & Public Policy* 12 (3): 513–544.

Bray, Freddie, Jacques Ferlay, Isabelle Soerjomataram, Rebecca L. Siegel, Lindsey A. Torre, and Ahmedin Jemal. 2018. Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians* 68 (6): 394–424. doi:10.3322/caac.21492.

Cleary, Duncan, and Revenue Irish Tax. 2011. Predictive analytics in the public sector: Using data mining to assist better target selection for audit. In *The proceedings of the 11th European conference on egovernment: Faculty of administration, University of Ljubljana, Ljubljana, Slovenia, 16–17 June 2011*, 168. Academic Conferences Limited.

Dua, Dheeru, and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Fawcett, Tom. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27 (8): 861–874.

Hirschowitz, Anton. 2001. Closing the CRM loop: The 21st century marketer's challenge: Transforming customer insight into customer value. *Journal of Targeting, Measurement and Analysis for Marketing* 10 (2): 168–178.

Kansagara, Devan, Honora Englander, Amanda Salanitro, David Kagen, Cecelia Theobald, Michele Freeman, and Sunil Kripalani. 2011. Risk prediction models for hospital readmission: A systematic review. *JAMA* 306 (15): 1688–1698.

Klubička, Filip, Giancarlo D. Salton, and John D. Kelleher. 2018. Is it worth it? Budget-related evaluation metrics for model selection. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Kohavi, Ron. 1996. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the twenty-fifth ACM SIGKDD international conference on knowledge discovery and data mining KDD*, 202–207.

Mac Namee, B., P. Cunningham, S. Byrne, and O. I. Corrigan. 2002. The problem of bias in training data in regression problems in medical decision support. *Artificial Intelligence in Medicine* 24 (1): 51–70.

Mangasarian, Olvi L., and William H. Wolberg. 1990. Cancer diagnosis via linear programming. *SIAM News* 23 (5): 1–18.

Mishne, Gilad, and Natalie S. Glance. 2006. Predicting movie sales from blogger sentiment. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, 155–158.

Osowski, Stainslaw, Linh Tran Hoai, and T. Markiewicz. 2004. Support vector machine-based expert system for reliable heartbeat recognition. *IEEE Transactions on Biomedical Engineering* 51 (4): 582–589. doi:10.1109/TBME.2004.824138.

Palaniappan, Sellappan, and Rafiah Awang. 2008. Intelligent heart disease prediction system using data mining techniques. *International Journal of Computer Science and Network Security* 8 (8): 343–350.

Rubin, Daniel J. 2015. Hospital readmission of patients with diabetes. *Current Diabetes Reports* 15 (4): 17.

Siddiqi, Naeem. 2005. *Credit risk scorecards: Developing and implementing intelligent credit scoring*. Wiley.

Tsanas, Athanasios, and Angeliki Xifara. 2012. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings* 49: 560–567.