Suggested Solutions to Selected Problems

CHAPTER 4

Problem 6

Part (A)

The ID3 decision tree induction algorithm selects the descriptive feature with the highest information gain at the root node of the decision tree. The first step in calculating information gain is to calculate the entropy for the entire dataset $\mathcal{D}$ with respect to the target feature, CANCER RISK:

$$\begin{aligned}
&H(\text{CANCERRISK}, \mathcal{D}) \\
&= - \sum_{l \in \left\{ \substack{high, \\ low} \right\}} P(\text{CANCERRISK} = l) \times log_2 \left(P(\text{CANCERRISK} = l) \right) \\
&= - \left(\left(\frac{3}{6} \times log_2 \left(\frac{3}{6} \right) \right) + \left(\frac{3}{6} \times log_2 \left(\frac{3}{6} \right) \right) \right) \\
&= 1.00 \text{ } bits
\end{aligned}$$

The table below shows the calculation of the information gain for each of the descriptive features in the dataset:

Split by Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
OBESE	*true*	DS_1	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.9183	0.9183	0.0817
	false	DS_2	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.9183		
SMOKER	*true*	DS_3	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.8113	0.5409	0.4591
	false	DS_4	$\mathbf{d}_1, \mathbf{d}_3$	0		
ALCOHOL	*true*	DS_5	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6$	0.9709	0.8091	0.1909
	false	DS_6	$\mathbf{d}_5$	0		

From this table, we can see that the feature SMOKER has the highest information gain, and consequently the ID3 algorithm will choose this feature as the one to be tested at the root node of the tree.

Part (B)

Being physically active and having a healthy diet—for example, eating vegetables and fruit—have been linked with reducing the risk of cancer. So the dataset could be extended to include features that capture the activity and diet of patients.

Problem 7

Part (A)

To answer this question we need to calculate the information gain of the STUDENT feature at the root node and show that it is less than the information gain for AGE.

To calculate information gain for the STUDENT feature we first calculate the entropy of the entire dataset $\mathcal{D}$ with respect to the target feature, BUYS

$$\begin{aligned} & H(\textsc{Buys}, \mathcal{D}) \\ &= -\sum_{l \in \left\{ \substack{yes, \\ no} \right\}} P(\textsc{Buys} = l) \times log_2 \left(P(\textsc{Buys} = l)\right) \\ &= -\left(\left(\frac{9}{14} \times log_2\left(\frac{9}{14}\right)\right) + \left(\frac{5}{14} \times log_2\left(\frac{5}{14}\right)\right)\right) \\ &= 0.9403\ bits \end{aligned}$$

Next we calculate the remainder after the dataset is split based on the values of the STUDENT feature.

Split by Feature	Feature Value	Partition	Examples	Partition Entropy	Remainder
STUDENT	*yes*	DS_1	5,6,7,9,10,11,13	0.5917	0.7885
	no	DS_2	1,2,3,4,8,12,14	0.9852	

Finally, we calculate the information gain as the difference between the original entropy and the remainder:

$$IG = 0.9403 - 0.7885 = 0.1518\ bits$$

The information gain for STUDENT is 0.1518 which is less then the 0.247 for AGE, so STUDENT is not a better feature to use at the root node.

Part (B)

There is no need to calcualte the information gain for the ID feature as the result will work out to be 0.9403—the total information required to make a prediction (and a value calculated in the answer to previous part of the question). We can know this without performing the calculation, however, because each instance has a unique value for the ID feature.

Because of this we know that ID would not be a good feature at the root node of the tree (or in fact anywhere in the tree) because it actually contains no information, and the resulting decision tree would be massively overfitted to the training data. Information measures such as the gain ratio are designed to address this limitation in information gain.

Problem 8

The target feature in this question (SCORE) is continuous. When a decision tree is predicting a continuous target, we choose as the descriptive feature to use at each node in the tree the one that results in the minimum weighted variance after the dataset has been split based on that feature. The table below shows the calculation of the weighted variance for each of the descriptive features in this domain.

Split by Feature	Level	Part.	Instances	$P(d=l)$	$var(t,\mathcal{D})$	Weighted Variance
STUDIED	*yes*	$\mathcal{D}_1$	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6$	$\frac{4}{7}$	$141\frac{2}{3}$	127.3810
	no	$\mathcal{D}_2$	$\mathbf{d}_2, \mathbf{d}_5, \mathbf{d}_7$	$\frac{3}{7}$	$108\frac{1}{3}$	
ENERGY	*alert*	$\mathcal{D}_5$	$\mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_6$	$\frac{3}{7}$	1525	829.7619
	tired	$\mathcal{D}_6$	$\mathbf{d}_1, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_7$	$\frac{4}{7}$	$308\frac{1}{3}$	

From these calculations we can see that splitting the dataset using the STUDIED feature results in the lowest weighted variance. Consequently, we should use the STUDIED feature at the root node of the tree.

Problem 9

Part (A)

Because majority voting is being used, the ensemble model will make an incorrect prediction for a query when a majority, in this case 6 or more, of the individual models make an incorrect prediction. So to calculate the overall probability of the model making an incorrect prediction, we need to sum the probabilities of the events where 6, 7, 8, 9, 10, and 11 of the models make the prediction.

Because each model in the ensemble is independent of the others, the predictions returned by the individual models 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

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

For example, we can calculate the probability of the event where exactly 6 of the models makes an incorrect prediction using the binomial distribution as follows:

$$\begin{aligned}\binom{11}{6} \times 0.2^6 \times (1-0.2)^{11-6} &= \\ \frac{11!}{6! \times (11-6)!} \times 0.000064 \times 0.32768 &= \\ 462 \times 0.000064 \times 0.32768 &= \\ &0.0097\end{aligned}$$

So, the probability of the model returning an incorrect prediction is

$$\sum_{i=6}^{11} \binom{11}{i} \times 0.2^i \times (1-0.2)^{11-i} = 0.0117$$

Part (B)

$$\sum_{i=6}^{11} \binom{11}{i} \times 0.49^i \times (1-0.49)^{11-i} = 0.4729$$

Part (C)

$$\sum_{i=11}^{21} \binom{21}{i} \times 0.49^i \times (1-0.49)^{21-i} = 0.4630$$

CHAPTER 5

Problem 4

Part (A)

In a domain where there are hundreds of thousands of items, co-absences aren't that meaningful. For example, you may be in a domain where there are so many items that most people haven't seen, listened to, bought, or visited that the majority of features will be co-absences. The technical term to describe a dataset where most of the features have zero values is **sparse data**. In these situations, you should use a metric that ignores co-absences. For a scenario such as this one, where the features are binary, the **Jaccard similarity index** is ideal as it ignores co-absences.

Part (B)

Using a similarity metric, the higher the value returned by the metric, the more similar the two items are.

Assuming you chose the **Jaccard similarity index**, then the query customer is more similar to customer $\mathbf{d}_1$ than to customer $\mathbf{d}_2$:

- $Jaccard(\mathbf{q}, \mathbf{d}_1) = \frac{2}{2+1} = 0.6667$
- $Jaccard(\mathbf{q}, \mathbf{d}_2) = \frac{1}{4} = 0.25$

There is only 1 item that customer $\mathbf{d}_1$ has bought that the query customer has not bought, item 498. As a result, the system will recommend item 498 to the query customer.

It turns out that in this instance, no matter which of the three similarity metrics we use, customer $\mathbf{d}_1$ is more similar to the query customer than customer $\mathbf{d}_2$. The supporting calculations for Russell-Rao and Sokal-Michener are

- Russell-Rao$(\mathbf{q}, \mathbf{d}_1) = \frac{2}{5} = 0.4$
- Russell-Rao$(\mathbf{q}, \mathbf{d}_2) = \frac{1}{5} = 0.2$
- Sokal-Michener$(\mathbf{q}, \mathbf{d}_1) = \frac{4}{5} = 0.8$
- Sokal-Michener$(\mathbf{q}, \mathbf{d}_2) = \frac{2}{5} = 0.4$

So, the system will recommend item 498 regardless of which similarity metric is used.

Problem 5

Part (A)

We can calculate the overlap metric between the query instance and each instance in the dataset by counting the number of feature values that are different.

ID	CLASS	Overlap Metric
1	mammal	6
2	amphibian	1
3	mammal	6
4	bird	2

Part (B)

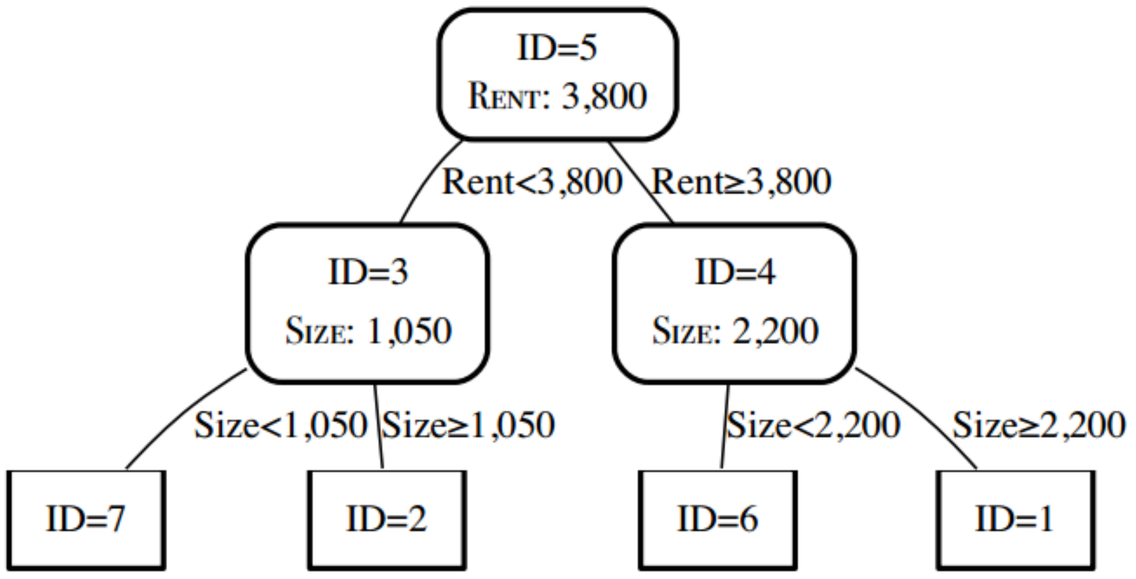
The nearest neighbor to the mystery animal is $\mathbf{d}_2$. So the mystery animal would be classified as an *amphibian*.

Part (C)

If you applied a 4-NN model to this dataset, the neighborhood defined around the query would include all the instances in the dataset irrespective of their distance from the query. As a result, any query would simply be assigned the majority class in the dataset, in this case *mammal*. So, for this dataset, a 4-NN model would massively underfit the dataset.

Problem 6

Part (A)



Part (B)

The initial step in retrieving the nearest neighbor is to descend the tree to a leaf node. For this query, this descent will terminate at the node that stores instance $\mathbf{d}_7$. At this point, the *current best* variable is set to the instance stored at this node $\mathbf{d}_7$, and the *current best-distance* variable is set to the Euclidean distance between the query and $\mathbf{d}_7$:

$$\textit{current best} = \mathbf{d}_7$$

$$\textit{current best-distance} = \textit{Euclidean}(\mathbf{q}, \mathbf{d}_7) = 1{,}400.5713$$

The retrieval algorithm then ascends the tree. The first node the algorithm will encounter is the node that stores instance $\mathbf{d}_3$. The Euclidean distance between the query and $\mathbf{d}_3$ is less than *current best-distance*. Consequently, *current best* and *current best-distance* are updated to reflect this:

$$\textit{current best} = \mathbf{d}_3$$

$$\textit{current best-distance} = \textit{Euclidean}(\mathbf{q}, \mathbf{d}_3) = 951.3149$$

Because the difference between the splitting feature value at this node, SIZE = 1,050, and the query, SIZE = 1,000, is less than the *current best-distance*, the algorithm descends the other branch of the tree from this node. This descent will terminate at the node $\mathbf{d}_2$. The Euclidean distance between the query and $\mathbf{d}_2$ is less than the *current best-distance*. Consequently, *current best* and *current best-distance* are updated to reflect this:

$$\textit{current best} = \mathbf{d}_2$$

$$\textit{current best-distance} = \textit{Euclidean}(\mathbf{q}, \mathbf{d}_2) = 509.1414$$

The algorithm will then ascend the tree; because it has already visited all the nodes on the path back to the root, it does not need to check for nodes closer than the *current best* until it gets back to the root. In this instance, the Euclidean distance between the query and the instance stored at the root node, $\mathbf{d}_3$, is greater than *current best-distance*, so neither *current best* nor *current best-distance* are updated when we reach the root node. Furthermore, because the difference between the splitting feature at the root, RENT = 3,800, and the query feature value, RENT = 2,200 is larger than the *current best-distance*, the algorithm can prune the other branch from the *k-d* tree and return $\mathbf{d}_2$ as the nearest neighbor, which would indicate that the property is worth approximately \$820,000.

CHAPTER 6

Problem 5
Part (A)

There are 3 bins and 9 instances. So using equal-frequency binning we know that there will be 3 instances in each bin. In the table below we have ordered the instances in ascending order by AGE

ID	OCCUPATION	GENDER	AGE	POLICY TYPE	PREF CHANNEL
9	nurse	female	18	planC	phone
6	manager	male	19	planA	email
3	biophysicist	male	21	planA	email
1	lab tech	female	43	planC	email
4	sheriff	female	47	planB	phone
7	geologist	male	49	planC	phone
8	messenger	male	51	planB	email
5	painter	male	55	planC	phone
2	farmhand	female	57	planA	phone

If we put the first 3 instances into the *young* bin, the next 3 instances into the *middle-aged* bin, etc. we end up with the dataset in the table below.

ID	OCCUPATION	GENDER	AGE	POLICY TYPE	PREF CHANNEL
9	nurse	female	young	planC	phone
6	manager	male	young	planA	email
3	biophysicist	male	young	planA	email
1	lab tech	female	middle-aged	planC	email
4	sheriff	female	middle-aged	planB	phone
7	geologist	male	middle-aged	planC	phone
8	messenger	male	mature	planB	email
5	painter	male	mature	planC	phone
2	farmhand	female	mature	planA	phone

The thresholds for the different bins are calculated as the mid-point between the AGE values of the two instances on either side of the boundary. So, the threshold between the *young* and *middle-aged* bins would be the mid point between $\mathbf{d}_3 with\text{AGE} = 21$ and $\mathbf{d}_1 with\text{AGE} = 43$:

$$\frac{21+43}{2} = 32$$

Likewise, the threshold between the *middleaged* and *mature* bins would be the mid point between $\mathbf{d}_7 with\text{AGE} = 49$ and $\mathbf{d}_8 with\text{AGE} = 51$:

$$\frac{49+51}{2} = 50$$

Part (B)

As is always the case the ID feature should not be used as a descriptive feature during training. However, in this example there is another feature that should be removed from the dataset prior to training. The OCCUPATION feature has different and unique levels for each instance in the dataset. In other words, the OCCUPTAITON feature is equivalent to an id for each instance. Consequently, it should also be removed from the dataset prior to training a model.

Part (C)

To train a naive Bayes model using this data, we need to compute the prior probabilities of the target feature taking each level in its domain, and the conditional probability of each feature taking each level in its domain conditioned for each level that the target feature can take. The table below lists the probabilities required by a naive Bayes model to represent this domain.

$P(phone)$	$=$	0.56	$P(email)$	$=$	0.44
$P(\text{GENDER} = female \mid phone)$	$=$	0.6	$P(\text{GENDER} = female \mid email)$	$=$	0.25
$P(\text{GENDER} = male \mid phone)$	$=$	0.4	$P(\text{GENDER} = male \mid email)$	$=$	0.75
$P(\text{AGE} = young \mid phone)$	$=$	0.2	$P(\text{AGE} = young \mid email)$	$=$	0.5
$P(\text{AGE} = middle\text{-}aged \mid phone)$	$=$	0.4	$P(\text{AGE} = middle\text{-}aged \mid email)$	$=$	0.25
$P(\text{AGE} = mature \mid phone)$	$=$	0.4	$P(\text{AGE} = mature \mid email)$	$=$	0.25
$P(\text{POLICY} = planA \mid phone)$	$=$	0.2	$P(\text{POLICY} = planA \mid email)$	$=$	0.5
$P(\text{POLICY} = planB \mid phone)$	$=$	0.2	$P(\text{POLICY} = planB \mid email)$	$=$	0.25
$P(\text{POLICY} = planC \mid phone)$	$=$	0.6	$P(\text{POLICY} = planC \mid email)$	$=$	0.25

Part (D)

The first step in calculating this answer is to bin the AGE feature. We know from part (a) of this question that the threshold between the *young* and *middle-aged* bins is 32. The value for the AGE feature in the query is less then 32 so it is mapped to the *young* bin. This results in the query being defined as

$$\text{GENDER} = \textit{female},\ \text{AGE} = \textit{young},\ \text{POLICY} = \textit{planA}$$

The calculation for $P(\text{CHANNEL} = \textit{phone} \mid \mathbf{q})$ is

$$\begin{aligned}
P(\textit{phone}) &= 0.56 \\
P(\text{GENDER} = \textit{female} \mid \textit{phone}) &= 0.6 \\
P(\text{AGE} = \textit{young} \mid \textit{phone}) &= 0.2 \\
P(\text{POLICY} = \textit{planA} \mid \textit{phone}) &= 0.2
\end{aligned}$$

$$\left(\prod_{k=1}^{m} P(\mathbf{q}[k] \mid \textit{phone})\right) \times P(\textit{phone}) = 0.01344$$

The calculation for $P(\text{CHANNEL} = \textit{email} \mid \mathbf{q})$ is

$$\begin{aligned}
P(\textit{email}) &= 0.44 \\
P(\text{GENDER} = \textit{female} \mid \textit{email}) &= 0.25 \\
P(\text{AGE} = \textit{young} \mid \textit{email}) &= 0.5 \\
P(\text{POLICY} = \textit{planA} \mid \textit{email}) &= 0.5
\end{aligned}$$

$$\left(\prod_{k=1}^{m} P(\mathbf{q}[k] \mid \textit{email})\right) \times P(\textit{email}) = 0.0275$$

The target level with the highest ranking is CHANNEL = *email*, and this is the prediction returned by the model.

Problem 6

Part (A)

A naive Bayes model will label the query with the target level that has the highest probability under the assumption of conditional independence between the evidence features. So to answer this question, we need to calculate the probability of each target level given the evidence and assuming conditional independence across the evidence.

To carry out these calculations, we need to convert the raw document counts into conditional probabilities by dividing each count by the total number of documents occurring in each topic:

w_k	Count	$P(w_k \mid entertainment)$
fun	415	$\frac{415}{700} = .593$
is	695	$\frac{695}{700} = .99$
learning	35	$\frac{35}{700} = .05$
machine	70	$\frac{70}{700} = .10$

w_k	Count	$P(w_k \mid education)$
fun	200	$\frac{200}{300} = .667$
is	295	$\frac{295}{300} = .983$
learning	120	$\frac{120}{300} = .40$
machine	105	$\frac{105}{300} = .35$

We can now compute the probabilities of each target level:

$$\begin{aligned}
P(\mathit{entertainment} \mid \mathbf{q}) &= P(\mathit{entertainment}) \times P(\mathit{machine} \mid \mathit{entertainment}) \\
&\quad \times P(\mathit{learning} \mid \mathit{entertainment}) \times P(\mathit{is} \mid \mathit{entertainment}) \\
&\quad \times p(\mathit{fun} \mid \mathit{entertainment}). \\
&= 0.7 \times 0.593 \times 0.99 \times 0.5 \times 0.1 \\
&= 0.00205
\end{aligned}$$

$$\begin{aligned}
P(\mathit{education} \mid \mathbf{q}) &= P(\mathit{education}) \times P(\mathit{machine} \mid \mathit{education}) \\
&\quad \times P(\mathit{learning} \mid \mathit{education}) \times P(\mathit{is} \mid \mathit{education}) \\
&\quad \times p(\mathit{fun} \mid \mathit{education}). \\
&= 0.3 \times 0.667 \times 0.983 \times 0.4 \times 0.35 \\
&= 0.00275
\end{aligned}$$

As $P(\mathit{education} \mid \mathbf{q}) > P(\mathit{entertainment} \mid \mathbf{q})$, the naive Bayes model will predict the target level of *education*.

Part (B)

Because the word *christmas* does not appear in any document of either topic, both conditional probabilities for this word will be equal to 0: $P(christmas \mid entertainment) = 0$ and $P(christmas \mid education) = 0$. Consequently, the probability for both target levels will be 0, and the model will not be able to return a prediction.

Part (C)

The table below illustrates the smoothing of the posterior probabilities for $P(word \mid entertainment)$.

Raw Probabilities	$P(christmas \mid entertainment)$	$=$	0
	$P(family \mid entertainment)$	$=$	0.5714
	$P(fun \mid entertainment)$	$=$	0.5929
Smoothing Parameters	k	$=$	10
	$count(entertainment)$	$=$	700
	$count(christmas \mid entertainment)$	$=$	0
	$count(family \mid entertainment)$	$=$	400
	$count(fun \mid entertainment)$	$=$	415
	$\lvert Domain(vocabulary)\rvert$	$=$	6
Smoothed Probabilities	$P(christmas \mid entertainment) = \frac{0+10}{700+(10\times 6)}$	$=$	0.0132
	$P(family \mid entertainment) = \frac{400+10}{700+(10\times 6)}$	$=$	0.5395
	$P(fun \mid entertainment) = \frac{415+10}{700+(10\times 6)}$	$=$	0.5592

The smoothing of the posterior probabilities for $P(word \mid education)$ is carried out in the same way:

Raw Probabilities	$P(christmas \mid education)$	$=$	0
	$P(family \mid education)$	$=$	0.5714
	$P(fun \mid education)$	$=$	0.5929
Smoothing Parameters	k	$=$	10
	$count(education)$	$=$	300
	$count(christmas \mid education)$	$=$	0
	$count(family \mid education)$	$=$	10
	$count(fun \mid education)$	$=$	200
	$\lvert Domain(vocabulary) \rvert$	$=$	6
Smoothed Probabilities	$P(christmas \mid entertainment) = \frac{0+10}{300+(10\times 6)}$	$=$	0.0278
	$P(family \mid entertainment) = \frac{10+10}{300+(10\times 6)}$	$=$	0.0556
	$P(fun \mid entertainment) = \frac{200+10}{300+(10\times 6)}$	$=$	0.5833

We can now compute the probabilities of each target level:

$$\begin{aligned} P(entertainment \mid \mathbf{q}) &= P(entertainment) \times P(christmas \mid entertainment) \\ &\quad \times P(family \mid entertainment) \times P(fun \mid entertainment) \\ &= 0.7 \times 0.0132 \times 0.5395 \times 0.5592 \\ &= 0.0028 \end{aligned}$$

$$\begin{aligned} P(education \mid \mathbf{q}) &= P(education) \times P(christmas \mid education) \\ &\quad \times P(family \mid education) \times P(fun \mid education) \\ &= 0.3 \times 0.0278 \times 0.0556 \times 0.5833 \\ &= 0.0003 \end{aligned}$$

As $P(entertainment \mid \mathbf{q}) > P(education \mid \mathbf{q})$, the model will predict a label of *entertainment* for this query.

CHAPTER 8

Problem 5

A single accuracy figure can hide the real performance of a model. This is particularly evident when we are dealing with imbalanced test datasets. Consider a test dataset that contains 1,000 instances overall, 900 of which belong to the positive target level, and 100 of which belong to the negative target level.

Assuming that the model always predicts the positive level, then its accuracy on the given test dataset would be 90%, which is not at all an accurate reflection of the performance of the model. Generating a simple confusion matrix for this scenario shows how poorly the model is really performing.

		Prediction	
		pos	*neg*
Target	*pos*	900	0
	neg	100	0

While measures such as average class accuracy or the F_1 measure attempt to address the issues associated with a simple accuracy measure, no single measure is perfect. All single measures compress the actual information in a set of results from a test dataset in some way, so it is always a good idea to use multiple performance measures as part of an evaluation, even though ultimately we will use a single figure to choose between alternative models.

Problem 6

Part (A)

The first step to answering this is to apply the threshold to determine whether predictions are correct or not. For Model 1, the predictions and outcomes are as follows

ID	Target	Model 1 Score	Model 1 Prediction	Model 1 Outcome
1	false	0.1026	false	TN
2	false	0.2937	false	TN
3	true	0.5120	true	TP
4	true	0.8645	true	TP
5	false	0.1987	false	TN
6	true	0.7600	true	TP
7	true	0.7519	true	TP
8	true	0.2994	false	FN
9	false	0.0552	false	TN
10	false	0.9231	true	FP
11	true	0.7563	true	TP
12	true	0.5664	true	TP
13	true	0.2872	false	FN
14	true	0.9326	true	TP
15	false	0.0651	false	TN
16	true	0.7165	true	TP
17	true	0.7677	true	TP
18	false	0.4468	false	TN
19	false	0.2176	false	TN
20	false	0.9800	true	FP
21	true	0.6562	true	TP
22	true	0.9693	true	TP
23	false	0.0275	false	TN
24	true	0.7047	true	TP
25	false	0.3711	false	TN
26	false	0.4440	false	TN
27	true	0.5440	true	TP
28	true	0.5713	true	TP
29	false	0.3757	false	TN
30	true	0.8224	true	TP

Using this, we can build the following confusion matrix for Model 1

		Prediction	
		true	*false*
Target	*true*	15	2
	false	2	11

We then repeat the same process for Model 2.

ID	Target	Model 2 Score	Model 2 Prediction	Model 2 Outcome
1	false	0.2089	false	TN
2	false	0.0080	false	TN
3	true	0.8378	true	TP
4	true	0.7160	true	TP
5	false	0.1891	false	TN
6	true	0.9398	true	TP
7	true	0.9800	true	TP
8	true	0.8578	true	TP
9	false	0.1560	false	TN
10	false	0.5600	true	FP
11	true	0.9062	true	TP
12	true	0.7301	true	TP
13	true	0.8764	true	TP
14	true	0.9274	true	TP
15	false	0.2992	false	TN
16	true	0.4569	false	FN
17	true	0.8086	true	TP
18	false	0.1458	false	TN
19	false	0.5809	true	FP
20	false	0.5783	true	FP
21	true	0.7843	true	TP
22	true	0.9521	true	TP
23	false	0.0377	false	TN
24	true	0.4708	false	FN
25	false	0.2846	false	TN
26	false	0.1100	false	TN
27	true	0.3562	false	FN
28	true	0.9200	true	TP
29	false	0.0895	false	TN
30	true	0.8614	true	TP

Using this we can build the following confusion matrix for Model 2.

		Prediction	
		true	*false*
Target	*true*	14	3
	false	3	10

Part (B)

We can calculate average class accuracy simply from the confusion matrices. For Model 1 we calculate the recall for the *true* target level as

$$recall_{true} = \frac{14}{(14+3)} = 0.8235$$

For the *false* target level, recall is

$$recall_{false} = \frac{12}{(12+1)} = 0.9231$$

We can than calculate average class accuracy as

$$average\ class\ accuracy_{AM} = \frac{(0.8235+0.9231)}{2} = 0.8733$$

For Model 2 we calculate the recalls for the two target levels as

$$recall_{true} = \frac{14}{(14+3)} = 0.8235$$

and

$$recall_{false} = \frac{10}{(10+3)} = 0.7692$$

We can than calculate average class accuracy as

$$average\ class\ accuracy_{AM} = \frac{(0.8235+0.7692)}{2} = 0.7964$$

Part (C)

Based on average class accuracy, Model 1 appears to be doing a better job than Model 2. The main difference is in the better ability of Model 1 to identify instances of the *false* target level.

Part (D)

To generate a cumulative gain chart, we first have to reorder the predictions in descending order of scores and divide the test instances into deciles. The table below shows this for Model 1 (because there are 30 instances in the test set, there are 3 instances per decile).

Decile	ID	Target	Model 1 Prediction	Model 1 Score	Model 1 Outcome
1st	20	*false*	0.9800	*true*	FP
	22	*true*	0.9693	*true*	TP
	14	*true*	0.9326	*true*	TP
2nd	10	*false*	0.9231	*true*	FP
	4	*true*	0.8645	*true*	TP
	30	*true*	0.8224	*true*	TP
3rd	17	*true*	0.7677	*true*	TP
	6	*true*	0.7600	*true*	TP
	11	*true*	0.7563	*true*	TP
4th	7	*true*	0.7519	*true*	TP
	16	*true*	0.7165	*true*	TP
	24	*true*	0.7047	*true*	TP
5th	21	*true*	0.6562	*true*	TP
	28	*true*	0.5713	*true*	TP
	12	*true*	0.5664	*true*	TP
6th	27	*true*	0.5440	*true*	TP
	3	*true*	0.5120	*true*	TP
	18	*false*	0.4468	*false*	TN
7th	26	*false*	0.4440	*false*	TN
	29	*false*	0.3757	*false*	TN
	25	*false*	0.3711	*false*	TN
8th	8	*true*	0.2994	*false*	FN
	2	*false*	0.2937	*false*	TN
	13	*true*	0.2872	*false*	FN
9th	19	*false*	0.2176	*false*	TN
	5	*false*	0.1987	*false*	TN
	1	*false*	0.1026	*false*	TN
10th	15	*false*	0.0651	*false*	TN
	9	*false*	0.0552	*false*	TN
	23	*false*	0.0275	*false*	TN

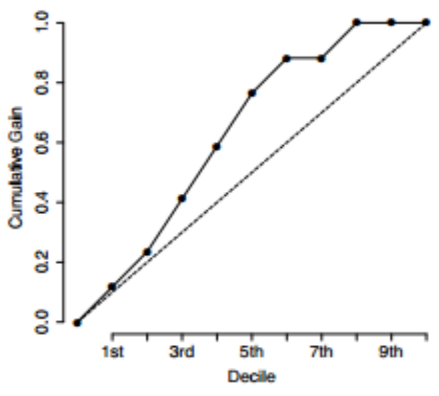
Based on this table, we can calculate the gain and lift (lift is not required to answer the question but is included for completeness) for each decile as well as the cumulative gain and cumulative lift for each decile. Recall that the definitions for gain, cumulative gain, lift, and cumulative lift always refer to the actual target levels rather than the predictions and are defined as follows:

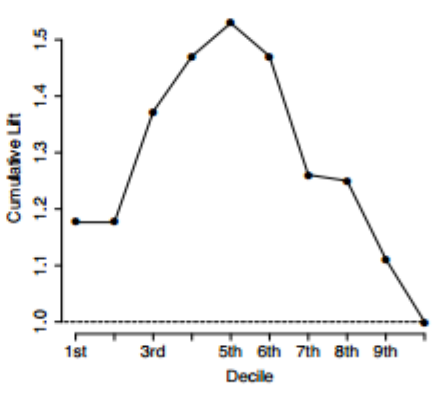
$$
\begin{aligned}
gain(dec) &= \frac{\text{num positive test instances in decile } dec}{\text{num positive test instances}}\\
cumulative\ gain(dec) &= \frac{\text{num positive test instances in all deciles up to } dec}{\text{num positive test instances}}\\
lift(dec) &= \frac{\text{\% of positive test instances in decile } dec}{\text{\% of positive test instances}}\\
cumulative\ lift(dec) &= \frac{\text{\% of positive instances in all deciles up to } dec}{\text{\% of positive test instances}}
\end{aligned}
$$

The measures for each decile for Model 1 are shown in the table below.

Decile	Positive (*true*) Count	Negative (*false*) Count	Gain	Cum. Gain	Lift	Cum. Lift
1st	2	1	0.1176	0.1176	1.1765	1.1765
2nd	2	1	0.1176	0.2353	1.1765	1.1765
3rd	3	0	0.1765	0.4118	1.7647	1.3725
4th	3	0	0.1765	0.5882	1.7647	1.4706
5th	3	0	0.1765	0.7647	1.7647	1.5294
6th	0	3	0.0000	0.7647	0.0000	1.2745
7th	1	2	0.0588	0.8235	0.5882	1.1765
8th	1	2	0.0588	0.8824	0.5882	1.1029
9th	1	2	0.0588	0.9412	0.5882	1.0458
10th	0	3	0.0000	0.9412	0.0000	0.9412

Using this table, we can now draw the required cumulative gain chart as follows (the cumulative lift chart is also shown).





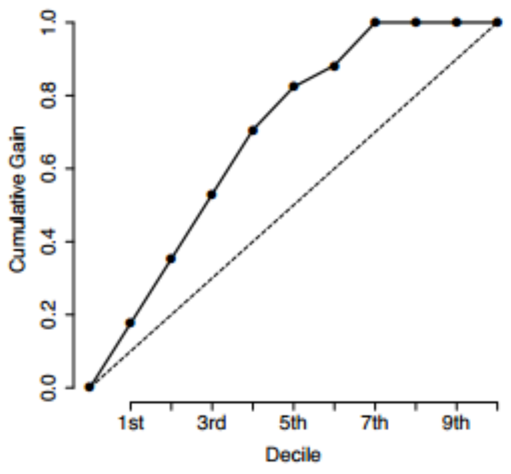
We then repeat for Model 2.

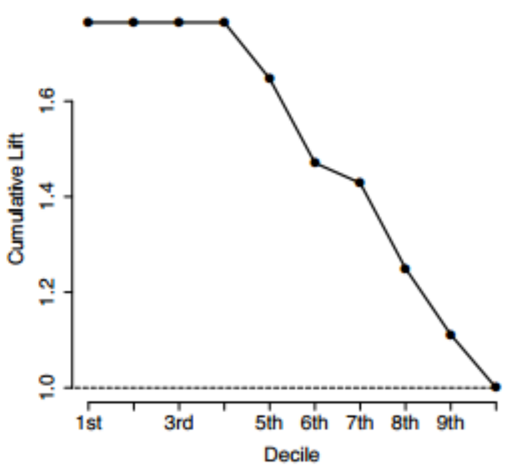
Decile	ID	Target	Model 1 Prediction	Model 1 Score	Model 1 Outcome
1st	7	*true*	0.9800	*true*	TP
	22	*true*	0.9521	*true*	TP
	6	*true*	0.9398	*true*	TP
2nd	14	*true*	0.9274	*true*	TP
	28	*true*	0.9200	*true*	TP
	11	*true*	0.9062	*true*	TP
3rd	13	*true*	0.8764	*true*	TP
	30	*true*	0.8614	*true*	TP
	8	*true*	0.8578	*true*	TP
4th	3	*true*	0.8378	*true*	TP
	17	*true*	0.8086	*true*	TP
	21	*true*	0.7843	*true*	TP
5th	12	*true*	0.7301	*true*	TP
	4	*true*	0.7160	*true*	TP
	19	*false*	0.5809	*true*	FP
6th	20	*false*	0.5783	*true*	FP
	10	*false*	0.5600	*true*	FP
	24	*true*	0.4708	*false*	FN
7th	16	*true*	0.4569	*false*	FN
	27	*true*	0.3562	*false*	FN
	15	*false*	0.2992	*false*	TN
8th	25	*false*	0.2846	*false*	TN
	1	*false*	0.2089	*false*	TN
	5	*false*	0.1891	*false*	TN
9th	9	*false*	0.1560	*false*	TN
	18	*false*	0.1458	*false*	TN
	26	*false*	0.1100	*false*	TN
10th	29	*false*	0.0895	*false*	TN
	23	*false*	0.0377	*false*	TN
	2	*false*	0.0080	*false*	TN

We can now calculate gain, cumulative gain, lift, and cumulative lift for each decile for Model 2.

Decile	Positive (*true*) Count	Negative (*false*) Count	Gain	Cum. Gain	Lift	Cum. Lift
1st	3	0	0.1765	0.1765	1.7647	1.7647
2nd	2	1	0.1176	0.2941	1.1765	1.4706
3rd	3	0	0.1765	0.4706	1.7647	1.5686
4th	3	0	0.1765	0.6471	1.7647	1.6176
5th	3	0	0.1765	0.8235	1.7647	1.6471
6th	0	3	0.0000	0.8235	0.0000	1.3725
7th	1	2	0.0588	0.8824	0.5882	1.2605
8th	1	2	0.0588	0.9412	0.5882	1.1765
9th	1	2	0.0588	1.0000	0.5882	1.1111
10th	0	3	0.0000	1.0000	0.0000	1.0000

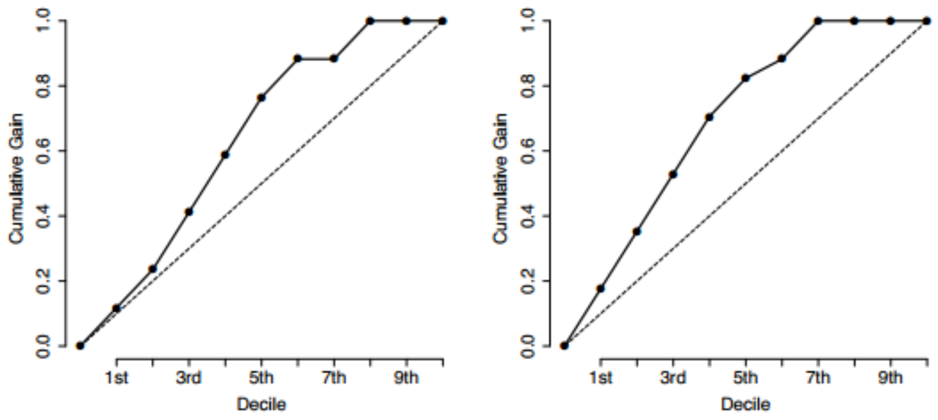
Using this table, we can now draw the required cumulative gain chart as follows (the cumulative lift chart is also shown).





Part (E)

The cumulative gain charts are repeated here.



The key thing to notice here is that at the 2^{nd} decile, Model 2 has identified almost 40% of the positive instances in the dataset, whereas Model 1 has identified only just over 20%. In a scenario like the one in this question, the typical approach would be to present every contact in the contact list to a model and then rank the contacts by the output of the model. Mailshots would be sent to the top 20% of this list, or whatever percentage the people sending the mailshot could afford. Because of this, a model that ranks well should be preferred over one that simply makes correct predictions. So in this case, even though Model 2 performs worse than Model 1 based on average class accuracy, Model 2 should be preferred for this task. The issue for Model 1 arises because it gives very high prediction scores to test instances 10 and 20.

Problem 7

Part (A)

For this system, classification accuracy will obviously be important. Beyond simple accuracy, however, there is also the issue of a balance between the types of errors that the system will make, that is, false positives and false negatives. For this scenario, if we assume that *good* is the positive target level, then we should avoid false positives as much as possible.

Beyond performance of the algorithm, other considerations are important for this scenario. The first of these is prediction speed. The model will make predictions as part of a production line, which it cannot in any way slow down. Depending on the throughput speed of the production line, the model will likely need to make predictions very quickly.

The next issue that applies in this scenario is the ability to retrain the model. This ability should be taken into account in evaluating the performance of models for this task. There are two ways of thinking about this. The first is the ability to easily modify a model as new data becomes available. The second is the speed at which a model can be completely rebuilt if it cannot be easily modified.

Finally, explanation capability is another important factor to consider when evaluating models. Once the model gives a prediction, how easy is it for someone to understand this prediction? This is key in this scenario so that production line operators can understand why mistakes are being made.

Part (B)

It is not possible to say anything about the likely performance in terms of prediction accuracy of the three model types for this problem without knowing much more about the data involved, and probably actually performing experiments. We can, however, comment on the model types in terms of the other characteristics: prediction speed, capacity for retraining, training speed, and explanation capability.

For *k*-NN models, prediction speed is always an issue. Although techniques such as *k-d* trees can help in reducing the time that it takes to find the nearest neighbors for a model, *k*-NN models will take more time than the other approaches to make a prediction. This can make them unsuitable for high-throughput scenarios like the production line. In terms of retraining, *k*-NN models excel. To allow the model to take new data into account, we simply add the new instances to the data used. Similarly, full retraining is almost instant—the training dataset used by the model is simply replaced with a new one—and *k*-NN models provide some ability to explain the predictions that they make—the neighbors on which a prediction is based can be provided for explanation. Taking these characteristics together, a *k*-NN model is a reasonably good fit for this task. The main advantage is the ease with which a *k*-NN model can be retrained. The main disadvantage of using a *k*-NN model for this task is that prediction time may be too slow for integration into the production line.

A decision tree model would be a good candidate for this scenario. A decision tree can make a predictions very quickly. Decision trees also have excellent explanation capacity—by tracing the path through the decision tree, it is very easy to understand how a prediction has been made. The main disadvantage of using a decision tree for this problem is its lack of capacity for retraining. There are techniques that can be used to modify a decision tree based on some newly available data, but this remains an open research problem. On the positive side, though, decision tree training is quite fast, so if a model needs to be completely rebuilt, this does not take as long as it can for other model types—for example, regression models.

A naive Bayes model can make predictions very quickly—this simply involves evaluating the naive Bayes equation—so prediction speed is not likely to be an issue. Like decision trees, there are approaches to adapting naive Bayes models to take into account new data, but there is not a standard, widely used approach. Training time, however, is not excessive, so retraining a naive Bayes model is not a significant problem. The explanation capacity of naive Bayes models is only modest. While the conditional and prior probabilities used to make a prediction can be presented, interpreting these is not something that people find intuitively easy. For this scenario, having probabilities returned would, however, be an advantage, as it makes it very easy to tune a model to favor false precision over recall.

Finally, a logistic regression model would have no problems with prediction speed for this scenario—evaluating the regression equation is not a significant computational task. Retraining, however, is a problem. It is not easy to adapt an existing logistic regression model to take into account new data, and logistic regression models can take a very long time to train—the longest time of the four modeling approaches considered here. Logistic regression models have some explanation capability. By considering model weights together with descriptive feature values, it is possible to get a good understanding of what has contributed to a particular prediction. In some industries **score cards** are used to aid explanation (Siddiqi (2005) provides a good overview of the use of score cards for financial credit risk scoring).

Taking the four approaches together, decision trees for the explanation capability and *k*-NN models for their capacity for retraining are probably the most attractive for this problem. Given the importance of prediction speed in this scenario, decision trees would probably be slightly more suitable.

It is important to reiterate, however, that this discussion has not taken into account the ability of these different model types to cope with the actual data in this problem.