B565 (Spring 2022) - DATA MINING
HOMEWORK 5

Name: Ameya Dalvi **Email: abdalvi@iu.edu**

1. Textbook problems.

Q3. Consider the training examples shown in Table 3.6 for a binary classification problem.

Instance	a1	a2	a3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
8	T	F	7.0	+
9	F	T	5.0	-

Q. 3)

a) The class attribute for the given collection of training examples has two values '+' and '−'.

+	4
−	5

$$\text{ENTROPY}(E) = -\sum_{i=1}^{N} P_i \log P_i$$

where, P_i is the probability of randomly selecting an example in class i and N is number of classes

$\therefore P(+) = \frac{4}{9}$, $P(-) = \frac{5}{9}$

$$\therefore E = -\frac{4}{9}\log_2\left(\frac{4}{9}\right) - \frac{5}{9}\log_2\left(\frac{5}{9}\right)$$

$= 0.5199 + 0.4711$

$= 0.991$

b) We know that Entropy before split = 0.991. a_1 is split into 'T' and 'F' according to Target Class as:-

	T	F
+	3	1
−	1	4

$$\text{Weighted Entropy} = \sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i)$$

where, k → number of partitions

n_i → number of records in partition i

$$= \frac{4}{9}\left[-\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right] + \frac{5}{9}\left[-\frac{1}{5}\log_2\left(\frac{1}{5}\right) - \frac{4}{5}\log_2\left(\frac{4}{5}\right)\right]$$

$= \frac{4}{9} \times (0.811) + \frac{5}{9} \times (0.721)$

= Entropy after split = 0.76

Information Gain = E before split − E after split

$= 0.991 - 0.76$

$= 0.231$

Similarly, a_2 is split as,

	T	F
+	2	2
−	3	2

$$\text{Entropy after split} = \frac{5}{9}\left[-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right] + \frac{4}{9}\left[-\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log\left(\frac{2}{4}\right)\right]$$

$= 0.5388 + 0.445$

$= 0.9838$

Information Gain for a_2 $= 0.991 - 0.98$

$= 0.0072$.

c) For a_3, it is a continuous attribute, hence we first sort the values and then choose the split position according to mid point between two successive values.

Class:	+	−	+	−	−	+	−	+	−
a_3	1.0	3.0	4.0	5.0	5.0	6.0	7.0	7.0	8.0
	2.0		3.5		4.5		5.5	6.5	7.5
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
+	1	3	1	3	2	2	2	2	3	1	4	0
−	0	5	1	4	1	4	3	2	3	2	4	1

$$E_{2.0} = \frac{1}{9}\left[0 - 1\log_2(1)\right] + \frac{8}{9}\left[-\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right)\right]$$

$= \frac{8}{9} \times 0.95$

$= 0.845$

Information Gain for $E_{2.0}$ = E before split $- E_{2.0} = 0.991 - 0.845$

$= 0.146$

$$E_{3.5} = \frac{2}{9}\left[-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right] + \frac{7}{9}\left[-\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right]$$

$= 0.223 + 0.766$

$= 0.989$

Information Gain for $E_{3.5}$ = E Before split $- E_{3.5} = 0.991 - 0.989$

$= 0.002$

$$E_{4.5} = \frac{3}{9}\left[-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right] + \frac{6}{9}\left[-\frac{2}{6}\log_2\left(\frac{2}{6}\right) - \frac{4}{6}\log_2\left(\frac{4}{6}\right)\right]$$

$= 0.306 + 0.612$

$= 0.918$

Information Gain for $E_{4.5}$ = E before split $- E_{4.5} = 0.991 - 0.918$

$= 0.0073$

$$E_{5.5} = \frac{5}{9}\left[-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right] + \frac{4}{9}\left[-\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right]$$

$= 0.539 + 0.445$

$= 0.984$

Information Gain for $E_{5.5}$ = E before split $- E_{5.5} = 0.991 - 0.984$

$= 0.007$

$$E_{6.5} = \frac{6}{9}\left[-\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right)\right] + \frac{3}{9}\left[-\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right]$$

$= 0.667 + 0.306$

$= 0.973$

Information Gain for $E_{6.5}$ = E before split $- E_{6.5} = 0.991 - 0.973$

$= 0.018$

$$E_{7.5} = \frac{8}{9}\left[-\frac{4}{8}\log_2\left(\frac{4}{8}\right) - \frac{4}{8}\log_2\left(\frac{4}{8}\right)\right] + \frac{1}{9}\left[-0 - 1\log_2(1)\right]$$

$= 0.889$

Information Gain for $E_{7.5}$ = E before split $- E_{7.5} = 0.991 - 0.889$

$= 0.102$

d) Information gain for a_1 is the highest, hence the best split can happen at a_1.

e) Misclassification error rate is calculated as:

$$\text{Error}(E) = 1 - \max(P_1, P_2 \cdots P_n)$$

a_1 is split into 'T' and 'F' according to Target class

	T	F
+	3	1
−	1	4

$$\therefore \text{Weighted Error} = \frac{4}{9}\left[1 - \max\left(\frac{1}{4}, \frac{3}{4}\right)\right] + \frac{5}{9}\left[1 - \max\left(\frac{1}{5}, \frac{4}{5}\right)\right]$$

$$= \frac{2}{9} = 0.222$$

a_2 is split into 'T' and 'F' according to Target

	T	F
+	2	2
−	3	2

$$\therefore \text{Weighted Error} = \frac{5}{9}\left[1 - \max\left(\frac{2}{5}, \frac{3}{5}\right)\right] + \frac{4}{9}\left[1 - \max\left(\frac{2}{4}, \frac{2}{4}\right)\right]$$

$$= \frac{5}{9}\left[1 - \frac{3}{5}\right] + \frac{4}{9}\left[1 - \frac{2}{4}\right]$$

$$= \frac{4}{9} = 0.445$$

Therefore a_1 is a better split due to the misclassification error.

f) Gini Index is calculated as:

$$GI = 1 - \sum_{i=1}^{N}(P_i)^2$$

a_1 is split as:

	T	F
+	3	1
−	1	4

$$\text{Weighted G.I} = \frac{4}{9}\left[1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2\right] + \frac{5}{9}\left[1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2\right]$$

$$= \frac{4}{9}\left[1 - \frac{9}{16} - \frac{1}{16}\right] + \frac{5}{9}\left[1 - \frac{1}{25} - \frac{16}{25}\right]$$

$$= \frac{6}{36} + \frac{8}{45}$$

$$= 0.345$$

a_2 is split as:

	T	F
+	2	2
−	3	2

$$\text{Weighted G.I} = \frac{5}{9}\left[1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2\right] + \frac{4}{9}\left[1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right]$$

$$= \frac{5}{9}\left[1 - \frac{4}{25} - \frac{9}{25}\right] + \frac{4}{9}\left[1 - \frac{4}{16} - \frac{4}{16}\right]$$

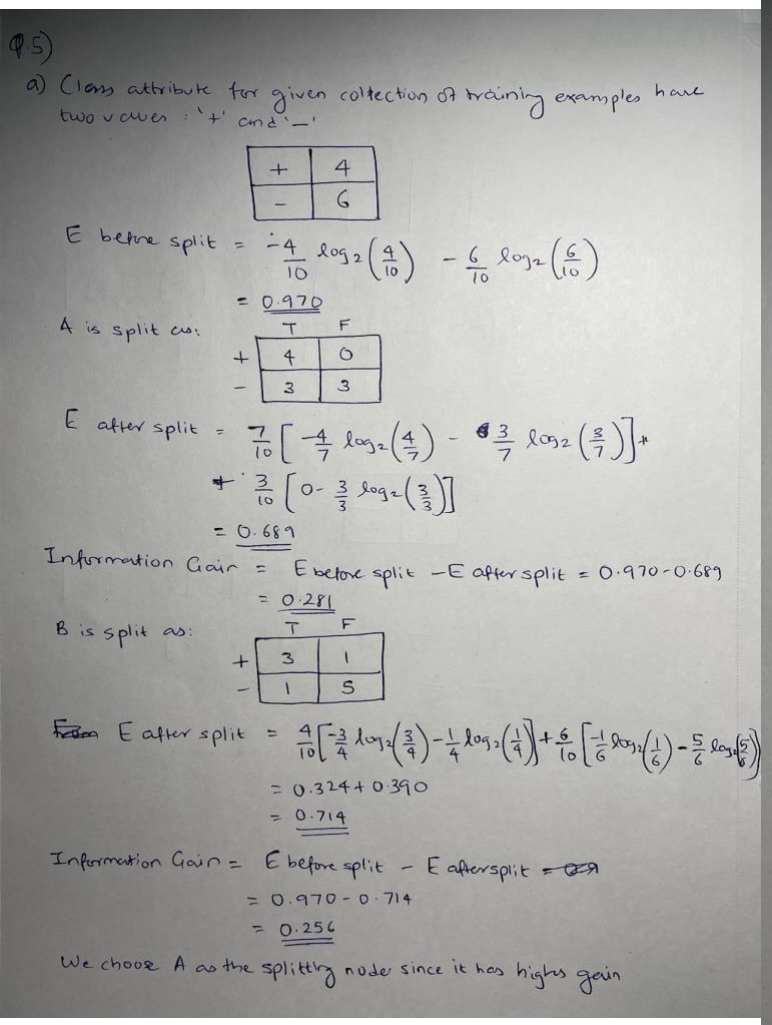
$$= \frac{12}{45} + \frac{8}{36}$$

$$= 0.489$$

We choose a_1 as the best split since it has lower gini index.

Q5. Consider the following data set for a binary class problem.

Instance	a2	Target Class
T	F	+
T	T	+
T	T	+
T	F	-
T	T	+
F	F	-
F	F	-
F	F	-
T	T	-
T	F	-



Q5)

a) Class attribute for given collection of training examples have two values : '+' and '−'

+	4
−	6

E before split $= -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right)$

$= 0.970$

A is split as:

	T	F
+	4	0
−	3	3

E after split $= \frac{7}{10}\left[-\frac{4}{7}\log_2\left(\frac{4}{7}\right) - \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] + \frac{3}{10}\left[0 - \frac{3}{3}\log_2\left(\frac{3}{3}\right)\right]$

$= 0.689$

Information Gain = E before split − E after split = 0.970 − 0.689

$= 0.281$

B is split as:

	T	F
+	3	1
−	1	5

E after split $= \frac{4}{10}\left[-\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right] + \frac{6}{10}\left[-\frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{5}{6}\log_2\left(\frac{5}{6}\right)\right]$

$= 0.324 + 0.390$

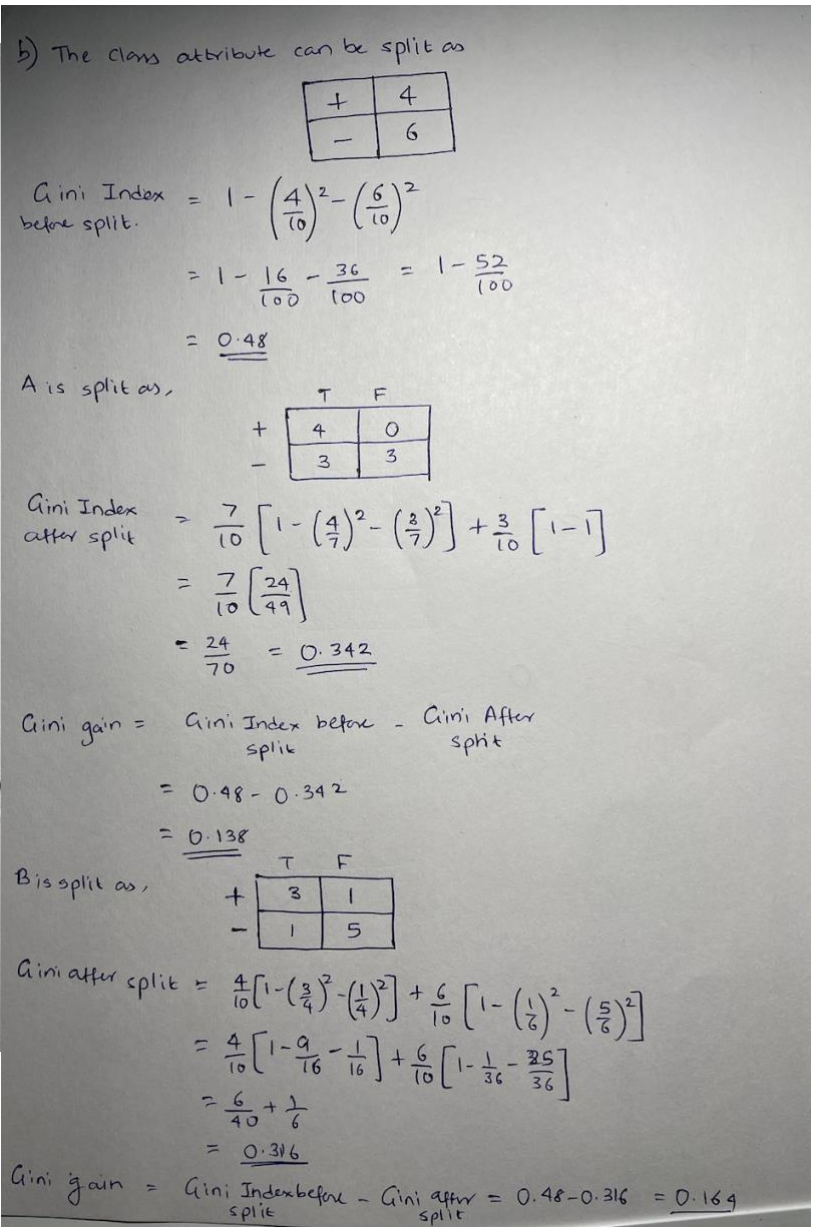
$= 0.714$

Information Gain = E before split − E after split

$= 0.970 - 0.714$

$= 0.256$

We choose A as the splitting node since it has higher gain



b) The class attribute can be split as

+	4
−	6

Gini Index before split $= 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2$

$= 1 - \frac{16}{100} - \frac{36}{100} = 1 - \frac{52}{100}$

$= 0.48$

A is split as,

	T	F
+	4	0
−	3	3

Gini Index after split $= \frac{7}{10}\left[1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2\right] + \frac{3}{10}\left[1 - 1\right]$

$= \frac{7}{10}\left[\frac{24}{49}\right]$

$= \frac{24}{70} = 0.342$

Gini gain = Gini Index before split − Gini After split

$= 0.48 - 0.342$

$= 0.138$

B is split as,

	T	F
+	3	1
−	1	5

Gini after split $= \frac{4}{10}\left[1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2\right] + \frac{6}{10}\left[1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2\right]$

$= \frac{4}{10}\left[1 - \frac{9}{16} - \frac{1}{16}\right] + \frac{6}{10}\left[1 - \frac{1}{36} - \frac{25}{36}\right]$

$= \frac{6}{40} + \frac{1}{6}$

$= 0.316$

Gini gain = Gini Index before split − Gini after split = 0.48 − 0.316 = 0.164

c) Figure 3.11 shows that entropy and the Gini index are both monotonically increasing on the range [0, 0.5] and they are both monotonically decreasing on the range [0.5, 1]. Is it possible that information gain and the gain in the Gini index favor different attributes? Explain.

Yes, it's feasible that the Gini Index's information gain and gain favor different attributes for splitting. Even if the entropy and gini index both increase and subsequently decrease steadily over time, the gain might vary depending on the entropy and gini index. When the gains are computed with these impurity metrics in mind before the split, the scaling changes, and the behavior changes as a result. The same may be seen in the preceding problem as A is selected on the basis of information gain, whereas B is selected on the basis of gain in gini index.

Q12.Consider a labeled data set containing 100 data instances, which is randomly partitioned into two sets A and B, each containing 50 instances. We use A as the training set to learn two decision trees, T10 with 10 leaf nodes and T100 with 100 leaf nodes. The accuracies of the two decision trees on data sets A and B are shown in Table

	Accuracy	Accuracy
Data Set	**T10**	**T100**
A	**0.86**	**0.97**
B	**0.84**	**0.77**

a) **Based on the accuracies shown in Table 3.7 , which classification model would you expect to have better performance on unseen instances?**

In this case, A is the training set and B is the testing set, and we have two models T10 and T100 to examine. We can observe from the table that, although having a higher training accuracy (on A), T100 performs poorly in testing, with a testing accuracy of 77 percent compared to 0.97 for training. T100, is thus, overfitting to our data. T10, on the other hand, performs consistently on both the training and test sets; its training and testing accuracy are almost identical and equally good. As a result, I believe T10 would perform better in unseen instances.

b) **Now, you tested T10 and T100 on the entire data set (A+B) and found that the classification accuracy of T10 on data set (A+B) is 0.85, whereas the classification accuracy of T100 on the data set (A+B)is 0.87. Based on this new information and your observations from Table 3.7 , which classification model would you finally choose for classification?**

T10 and T100 seem to perform almost equally well on the combined dataset (A+B). T10 has an accuracy of 0.85 and T100 is 0.87. Hence, we can see, although T100 is performing slightly better here, it's not a significant difference. Also, T100 is computationally more expensive compared to T10 as we increase the number of splits in our model. Hence T10 can be used finally.

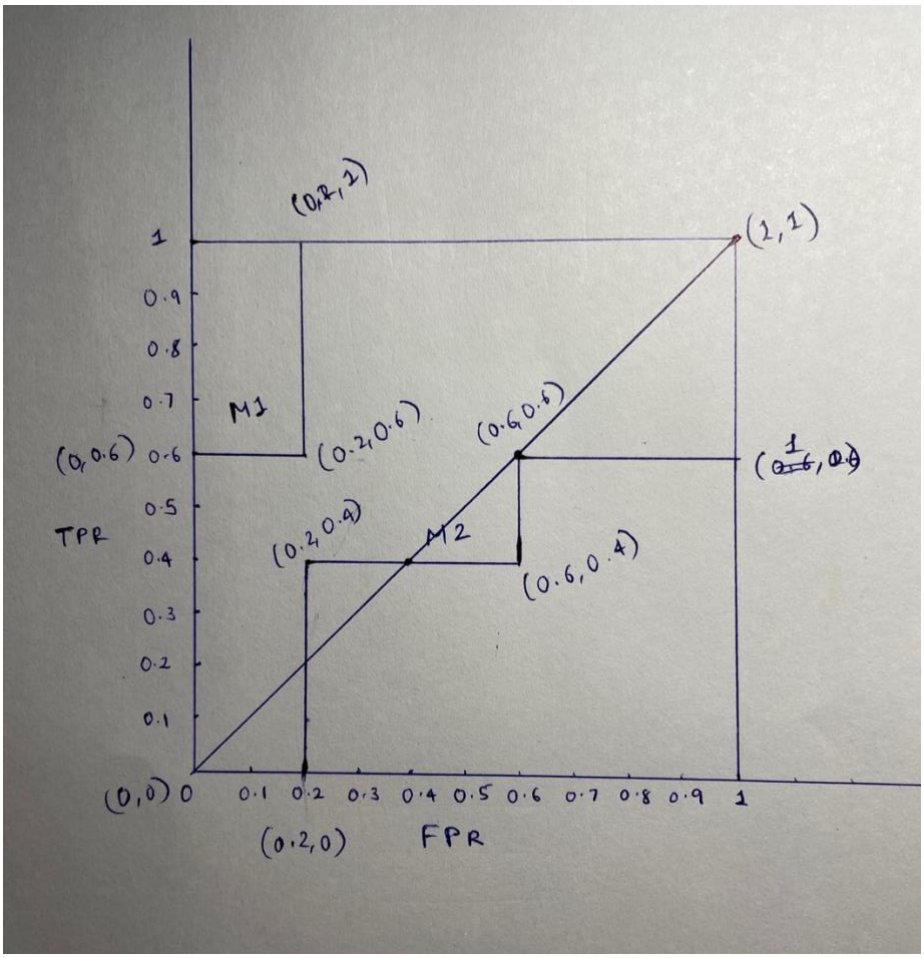
Q16. You are asked to evaluate the performance of two classification models, and . The test set you have chosen contains 26 binary attributes, labeled as A through Z. Table 4.13 shows the posterior probabilities obtained by applying the models to the test set. (Only the posterior probabilities for the positive class are shown). As this is a two-class problem, and . Assume that we are mostly interested in detecting instances from the positive class.

Instance	True Class	P(+|A,…,Z,M1)	P(+|A,…,Z,M2)
1	+	0.73	0.61
2	+	0.69	0.03
3	-	0.44	0.68
4	-	0.55	0.31
5	+	0.67	0.45
6	+	0.47	0.09
7	-	0.08	0.38
8	-	0.15	0.05
9	+	0.45	0.01
10	-	0.35	0.04

a. Plot the ROC curve for both M1 and M2. (You should plot them on the same graph.) Which model do you think is better? Explain your reasons.

M1 : Threshold values : (0.1,0.5,0.6,0.8)
0.1: (TPR,FPR) = (1,0.8)
0.5: (TPR,FPR) = (0.6,0.2)
0.6: (TPR,FPR) = (0.6,0)
0.8: (TPR,FPR) = (0,0)

M2 : Threshold values are (0.1,0.5,0.6,0.8)
0.1: (TPR,FPR) = (0.4,0.6)
0.5: (TPR,FPR) = (0.2,0.2)
0.6: (TPR,FPR) = (0.2,0.2)
0.8: (TPR,FPR) = (0,0)



b. For model M1, suppose you choose the cutoff threshold to be t = 0.5. In other words, any test instances whose posterior probability is greater than t will be classified as a positive example. Compute the precision, recall, and F-measure for the model at this threshold value.

Let t = 0.5, then confusion matrix of M1 will be:-

		+	-
Actual	**+**	**3**	**2**
	-	**1**	**4**

Precision = $\frac{TP}{TP + FP}$ **= 3/4**

= 0.75

= 75%

Recall = $\frac{TP}{TP + FN}$ **= 3/5**

= 0.60

= 60%

F-measure = $\frac{2 * Precision * Recall}{Precision + Recall}$

= (2 x 0.75 x 0.60) / (0.75 + 0.60) = 0.667

c. Repeat the analysis for part (b) using the same cutoff threshold on model M2. Compare the F-measure results for both models. Which model is better? Are the results consistent with what you expect from the ROC curve?

Let t = 0.5, then confusion matrix of M1 will be:-

		+	-
Actual	**+**	**1**	**4**
	-	**1**	**4**

Precision = $\frac{TP}{TP + FP}$ **= 1/2**

= 0.50

= 50%

Recall = $\frac{TP}{TP + FN}$ **= 1/5**

= 0.20

= 20%

F-measure = $\frac{2 * Precision * Recall}{Precision + Recall}$

= (2 x 0.50 x 0.20) / (0.50 + 0.20) = 0.2857

Based on, F-measure of M1 and the current calculation of M2 we can conclude that M1 is better than M2 and same can be seen in ROC plot also.

d. Repeat part (b) for model M1 using the threshold t=0.1. Which threshold do you prefect=0.5 or t=0.1? Are the results consistent with what you expect from the ROC curve?

Let t = 0.5, then confusion matrix of M1 will be:-

		+	-
Actual	**+**	**5**	**0**
	-	**4**	**1**

Precision = $\frac{TP}{TP + FP}$ **= 5/9**

= 0.55

= 55.5%

Recall = $\frac{TP}{TP + FN}$ **= 5/5**

= 100%

F-measure = $\frac{2 * Precision * Recall}{Precision + Recall}$

= (2 x 0.55 x 1) / (0.55 + 1) = 0.714

Based on the F-measure, t=0.1 is better than t=0.5

For t=0.1, FPR = 0.8 and TPR = 1,
For t=0.5, FPR = 0.2 and TPR = 0.6.

Since point (0.2,0.6) is closer to the point (0,1) t=0.5 will be selected and this finding is inconsistent with the results using f-measure.

Q18. Consider the task of building a classifier from random data, where the attribute values are generated randomly irrespective of the class labels. Assume the data set contains instances from two classes, " + " and " -" Half of the data set is used for training while the remaining half is used for testing.

a. **Suppose there are an equal number of positive and negative instances in the data and the decision tree classifier predicts every test instance to bepositive. What is the expected error rate of the classifier on the test data?**

   Suppose we have 50 positive and 50 negative instances in the data, after splitting the data into train and test we can expect to have 25 positive and 25 negative instances in train and test each. Now if our model predicts every test instance to be positive, then 25 of the negative values in test would be predicted incorrectly as positive, hence the error rate would thus be 25/50 = 0.5 = 50%.

b. **Repeat the previous analysis assuming that the classifier predicts each test instance to be positive class with probability 0.8 and negative class with probability 0.2.**

   Suppose we have 20 instances, of which 10 are positive and 10 negative, Hence, after the split, we would have 5 positive and 5 negative instances in train and test respectively. Now, probability of our model predicting test instance as positive is 0.8, hence ⅘ positive instances would be predicted correctly, similarly, probability of our model predicting test instance as negative is 0.2, hence ⅕ positive instances would be predicted correctly. So out of the total 10 instances, 5 instances would be predicted correctly as either positive or negative, hence the error rate thus would be 50%.

c. **Suppose two-thirds of the data belong to the positive class and the remaining one-third belong to the negative class. What is the expected error of a classifier that predicts every test instance to be positive?**

   Suppose we have 200 instances, of which 134 are positive and 66 negative, Hence, after the split, we would have 67 positive and 33 negative instances in train and test respectively. Now if our model predicts every test instance to be positive, then 33 of the negative values in test would be predicted incorrectly as positive, hence the error rate would thus be 33/100 = 33%.

d. Repeat the previous analysis assuming that the classifier predicts each test instance to be positive class with probability 2/3 and negative class with probability 1/3.

Suppose we have 200 instances, of which 134 are positive and 66 negative, Hence, after the split, we would have 67 positive and 33 negative instances in train and test respectively. Now, probability of our model predicting test instance as positive is ⅔ , hence ⅔ * 67 = 44.67 ~ 45 and positive instances would be predicted correctly and 22.33 ~ 22 instances incorrectly, similarly, probability of our model predicting test instance as negative is ⅓ , hence ⅓ * 33 = 11 positive instances would be predicted correctly and 22 incorrectly. So out of the total 100 instances, 22 + 22 = 44 instances would be predicted incorrectly and hence the error rate thus would be 22 + 22.34 ~ 44.34%.

2. Linear regression:

a) Dataset: the [auto-mpg] dataset on Kaggle.

[213]:
```
data1=pd.read_csv('/Users/ameyadalvi/Downloads/auto-mpg.csv')
data1
```

[213]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

b) Perform feature scaling (using L2 norm) over the auto data set. Use two thirds of the data for training and the remaining one third for testing. Train a multivariate linear regression (sklearn.linear_model.LinearRegression) with "mpg" as the response and all other variables except "car name" as the predictors. What's the coefficient for the "year" attribute, and what does the coefficient suggest? What's the accuracy (mean squared error) of the model on the test data (one third of the mpg data set)?

0.1. Feature Scaling

[294]:
```
scaled_data = preprocessing.normalize(data, norm='l2')
```

[296]:
```
scaled_data = pd.DataFrame(scaled_data, columns=columns)
scaled_data
```

[296]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
0	0.005113	0.002272	0.087201	0.036926	0.995286	0.003409	0.019883	0.000284
1	0.004039	0.002154	0.094240	0.044427	0.994364	0.003096	0.018848	0.000269
2	0.005210	0.002316	0.092048	0.043419	0.994580	0.003184	0.020262	0.000289
3	0.004637	0.002319	0.088104	0.043472	0.994936	0.003478	0.020287	0.000290
4	0.004905	0.002308	0.087137	0.040395	0.995153	0.003030	0.020197	0.000289
...	...	...	...	...	...	...	...	...
387	0.009656	0.001431	0.050068	0.030756	0.997778	0.005579	0.029325	0.000358
388	0.020609	0.001874	0.045433	0.024356	0.997650	0.011522	0.038407	0.000937
389	0.013900	0.001737	0.058639	0.036487	0.996865	0.005039	0.035618	0.000434
390	0.010645	0.001521	0.045620	0.030033	0.997938	0.007071	0.031174	0.000380
391	0.011375	0.001468	0.043665	0.030088	0.998048	0.007118	0.030088	0.000367

392 rows × 8 columns

0.2. Train-Test split

[302]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.33, random_state=42)
```

0.3. Linear Regression

[303]:
```
lin_reg = LinearRegression().fit(X_train, y_train)
```

[304]:
```
lin_reg.score(X_train, y_train)
```

[304]:
```
0.9404605356967636
```

[305]:
```
lin_reg.coef_
```

[305]:
```
array([[ 0.84020765, -0.11601289, -0.1712066 , -1.61009463, -0.27982384,
         0.56987164,  0.84684289]])
```

[306]:
```
pd.DataFrame(lin_reg.coef_.T, X.columns, columns=['coef'])
```

[306]:

	coef
cylinders	0.840208
displacement	-0.116013
horsepower	-0.171207
weight	-1.610095
acceleration	-0.279824
model year	0.569872
origin	0.846843

[307]:
```
y_pred = lin_reg.predict(X_test)
y_pred
```

[307]:
```
array([[0.01267267],
       [0.00866908],
```

Coefficient of the year attribute is 0.5698, The coefficient suggests that every unit increase in model year results in 56.98% increase in the mpg and same for decrease.

Mean Square Error:

[308]:
```
print("The mean squared error is:",mean_squared_error(y_test,y_pred))
```

```
The mean squared error is: 1.874651041530226e-06
```

c) Try linear regression with regularization (Ridge and Lasso) as implemented in sklearn (RidgeCV and LassoCV). Use the cross-validation approach and compare the coefficients for the different attributes.

0.4. LassoCV

[309]:
```
lasso_reg = LassoCV(cv=5, random_state=0).fit(X_train, y_train)
```

[310]:
```
lasso_reg.score(X_train, y_train)
```

[310]:
```
0.9324314168665049
```

[311]:
```
lasso_reg.coef_
```

[311]:
```
array([ 0.        ,  0.00883293, -0.04832203, -0.        ,  0.        ,
        0.64119412,  0.        ])
```

[312]:
```
pd.DataFrame(lasso_reg.coef_.T, X.columns, columns=['coef'])
```

[312]:

	coef
cylinders	0.000000
displacement	0.008833
horsepower	-0.048322
weight	-0.000000
acceleration	0.000000
model year	0.641194
origin	0.000000

[313]:
```
lasso_reg.alpha_
```

[313]:
```
1.312731356606791e-07
```

[314]:
```
y_pred = lasso_reg.predict(X_test)
y_pred
```

[314]:
```
array([0.01260688, 0.00894652, 0.01923548, 0.01460651, 0.0104539 ,
       0.01397322, 0.00126352, 0.01322659, 0.00658744, 0.01677195
```

[315]:
```
print("The mean squared error is:",mean_squared_error(y_test,y_pred))
```

```
The mean squared error is: 1.8765639641889238e-06
```

0.5. RidgeCV

[316]:
```
ridge_reg = RidgeCV(cv=5).fit(X_train, y_train)
```

[317]:
```
ridge_reg.score(X_train, y_train)
```

[317]:
```
0.5023618383898814
```

[318]:
```
ridge_reg.coef_
```

[318]:
```
array([[ 0.00077346, -0.08655993, -0.00562391,  0.00290521,  0.0157326 ,
         0.06747499,  0.0030243 ]])
```

[319]:
```
pd.DataFrame(ridge_reg.coef_.T, X.columns, columns=['coef'])
```

[319]:

	coef
cylinders	0.000773
displacement	-0.086560
horsepower	-0.005624
weight	0.002905
acceleration	0.015733
model year	0.067475
origin	0.003024

[320]:
```
ridge_reg.alpha_
```

[320]:
```
0.1
```

[321]:
```
y_pred = ridge_reg.predict(X_test)
y_pred
```

[321]:
```
array([[0.01113881],
       [0.01075597],
       [0.01126822],
       [0.01114685]
```

[322]:
```
print("The mean squared error is:",mean_squared_error(y_test,y_pred))
```

```
The mean squared error is: 1.1796890606887199e-05
```

d) Finally, compare the results obtained for ordinary linear regression, Ridge, and Lasso (using the α values that gave the lowest test MSE for the latter two). Does the type of regularization used affect the importance of the attributes? How can you interpret these results?

[334]:
```
pd.DataFrame(lin_reg.coef_.T, X.columns, columns=['coef'])
```

[334]:

	coef
cylinders	0.840208
displacement	-0.116013
horsepower	-0.171207
weight	-1.610095
acceleration	-0.279824
model year	0.569872
origin	0.846843

[332]:
```
pd.DataFrame(lasso_reg.coef_.T, X.columns, columns=['coef'])
```

[332]:

	coef
cylinders	0.000000
displacement	0.008833
horsepower	-0.048322
weight	-0.000000
acceleration	0.000000
model year	0.641194
origin	0.000000

[333]:
```
pd.DataFrame(ridge_reg.coef_.T, X.columns, columns=['coef'])
```

[333]:

	coef
cylinders	0.000773
displacement	-0.086560
horsepower	-0.005624
weight	0.002905
acceleration	0.015733
model year	0.067475
origin	0.003024

Only anomaly i find is the displacement attribute as for Lasso, It is positive for Lasso and Negative for rest both, but this could be because of low value of cross validation parameter.RidgeCV has a mean squared error higher, as compared to LassoCV and Linear Regression, which is justified as LassoCV is built in such a way that it drops the less important features by converting their coefficients to 0. Alpha value that gave lowest MSE among Lasso and Ridge is 1.312. Hence the type of regularization used does affect the importance of attributes.

3. Write a summary for this paper: A Reductions Approach to Fair Classification.

Summary:

The task of binary classification under fairness constraints with regard to a pre-defined protected attributes, is investigated in this research paper. According to the authors, research in this topic may be divided into **two** categories:

Approach 1 - It includes particular quantitative definitions of fairness into existing machine learning algorithms by loosening the intended definitions of fairness and imposing only weaker restrictions, such as absence of correlation. However, the resultant fairness guarantees are often only valid under strong distributional assumptions, and the methods are restricted to certain classifier families, such as SVMs.

Approach 2- This strategy avoids the need for specialized classifier families and treats the underlying classification method as a black box, with a wrapper that either pre-processes the input or processes the classifier's predictions. Pre-processing, however, still leads to significant injustice, according to the authors, since it concentrates on certain conceptions of fairness. Furthermore, while post-processing allows for a broader choice of fairness standards, it does not guarantee that the most accurate fair classifier will be found.

Author has used following definitions::

- Demographic (or statistical) parity
- Equalized Odds
- Equality of opportunity
- Balance for negative class
- Error-rate balance
- Overall accuracy equality

Exponentiated-gradient reduction
The authors coupled the notion of a saddle point issue with a Lagrange multiplier (λ) in this technique. The author also discusses the game-theoretic viewpoint, in which the saddle point may be considered as a point of equilibrium between two payers: Q-player picking Q and λ-player choosing λ.

Grid Search
The authors of this technique discuss a situation in which the number of restrictions is relatively modest (e.g., the case for demographic parity or equalized odds with binary protected attribute). According to the authors, it would be appropriate to explore grid values of λ and compute the optimum response for each value before selecting the value with the best accuracy and fairness tradeoff.

Results:
• After reviewing all of the algorithms, it was discovered that all of them required access to the protected attribute during training. During testing, only the post-processing algorithm required access to the protected property.
• The reduction dominated or equaled the baselines on the test data after running it over a wide variety of tradeoffs between classification error and fairness restrictions.
• On occasion, the grid-search failed to obtain the lowest disparity on the training set. Its performance on the testing set, on the other hand, was quite similar to the exponentiated gradient-reduction.
• Grid-search is not possible with non-binary characteristics.

Conclusion:
To integrate fairness in a binary classifier, the author offered two reduction techniques. These reduction techniques address the shortcomings of previous approaches, such as the inability to guarantee fairness and the lack of a robust fairness definition. During training time for accessing protected characteristics, these reduction strategies optimize the accuracy and fairness definition trade-off.