Spring B565 HW 5
Shardul Dabhane
Friday, April 8th, 2022

1. Textbook problems. [50 points]

(a) 3.11 Exercises, Q3, Q5, Q12 (30 points).

Q3. Consider the training examples shown in Table 3.2 for a binary classification problem.

Table 3.6. Data set for Exercise 3.

Instance	a_1	a_2	a_3	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	−

(a) What is the entropy of this collection of training examples with respect to the positive class?

Answer:

The probability of positive examples is:

P(+) = 4/9

The probability of negative examples is:

P(−)=5/9.

The entropy of the training examples is:

$-4/9*\log_2(4/9)-5/9*\log_2(5/9) = 0.9911.$

(b) What are the information gains of a1 and a2 relative to these training examples?

Answer:

The probabilities for a1 are:

a1	+	-
T	3	1
F	1	4

The entropy for a1 is :

$4/9[(-3/4)*\log_2(3/4)+(-1/4)*\log_2(1/4)]+5/9[(-1/5)*\log_2(1/5)+(-4/5)*\log_2(4/5)]=0.7616$

The information gain for a1 is 0.9911 - 0.7616 = 0.2294.

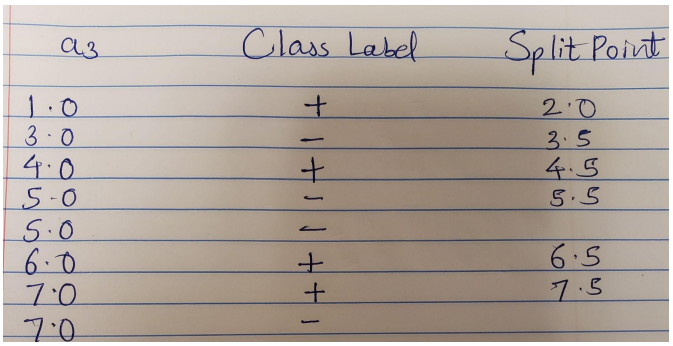
The probabilities for a2 are:

a2	+	-
T	2	3
F	2	2

The entropy for a2 is
$5/9[(-2/5)*\log_2(2/5)+(-3/5)*\log_2(3/5)]+4/9[(-2/4)*\log_2(2/4)+(-2/4)*\log_2(3/5)]= 0.9839.$

The information gain for a2 is 0.9911 − 0.9839 = 0.0072.

(c) For a3, which is a continuous attribute, compute the information gain for every possible split.
Answer:
First, we will calculate the split point for every value of a3.
The table here would be:



a3	Class Label	Split Point
1.0	+	2.0
3.0	−	3.5
4.0	+	4.5
5.0	−	5.5
5.0	−	
6.0	+	6.5
7.0	+	7.5
7.0	−	

Then we will calculate the Information Gain for every split point.
Calculations:

1) Take Split Point = 2.0

Overall Entropy = 0.9911 = Entropy(a_3)

$a_3 < 2.0 = [1+, 0-]$

Entropy($a_3 < 2.0$) = 0.0

$a_3 > 2.0 = [3+, 5-]$

$$= -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8}$$

$$= 0.954434002924$$

$$\text{Gain}(a_3 = 2.0) = \text{Entropy}(a_3) - \frac{|a_3 < 2.0|}{|a_3|}\text{Entropy}(a_3 < 2.0) - \frac{|a_3 > 2.0|}{|a_3|}\text{Entropy}(a_3 > 2.0)$$

$$= 0.9911 - \frac{8}{9} * 0.954434002924$$

$$= 0.84838$$

$$\approx 0.8484$$

We will calculate Information Gain for other split points.

2) Take Split Point = ~~3.0~~ 3.5

Overall Entropy = 0.9911 = Entropy(a_3)

$a_3 < 3.5 = [1+, 1-]$

$$\text{Entropy}(a_3 < 3.5) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}$$

$$= 1$$

$$\text{Entropy}(a_3 > 3.5) = [3+, 4-]$$

$$= -\frac{3}{7}\log\frac{3}{7} - \frac{4}{7}\log\frac{4}{7}$$

$$= 0.98522813621$$

$$\text{Gain}(a_3 = 3.5) = \text{Entropy}(a_3) - \frac{|a_3 < 3.5|}{|a_3|}\text{Entropy}(a_3 < 3.5) - \frac{|a_3 > 3.5|}{|a_3|}\text{Entropy}(a_3 > 3.5)$$

$$= 0.9911 - \frac{2}{9} * 1 - \frac{7}{9} * 0.98522813621$$

$$= 0.002589 \approx 0.26$$

3) Take Split Point = 4.5

Entropy $(a_3) = 0.9911$

$a_3 < 4.5 = [2+, 1-]$

$$\text{Entropy}(a_3 < 4.5) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}$$

$$= 0.9182958340 6$$

$a_3 > 4.5 = [2+, 4-]$

$$\text{Entropy}(a_3 > 4.5) = -\frac{2}{6}\log\frac{2}{6} - \frac{4}{6}\log\frac{4}{6}$$

$$= 0.9182958340 6$$

$$\text{Gain}(a_3 = 4.5) = \text{Entropy}(a_3) - \frac{|a_3 < 4.5|}{|a_3|}\text{Entropy}(a_3 < 4.5) - \frac{|a_3 > 4.5|}{|a_3|}\text{Entropy}(a_3 > 4.5)$$

$$= 0.9911 - \frac{3}{9} * 0.9182958340 6 - \frac{6}{9} * 0.9182958340 6$$

$$= 0.0728$$

4) Take Split Point = 5.5

Entropy $(a_3) = 0.9911$

$a_3 < 5.5 = [2+, 3-]$

$$\text{Entropy}(a_3 < 5.5) = -\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}$$

$$= 0.97095059445$$

$a_3 > 5.5 = [2+, 2-]$

$$\text{Entropy}(a_3 > 5.5) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}$$

$$= 1$$

$$\text{Gain}(a_3 = 5.5) = 0.9911 - \frac{5}{9} * 0.97095059445$$

$$- \frac{4}{9} * 1$$

$$= 0.00723$$

$$\approx 0.0072$$

5) Split Point = 6.5

Entropy $(a_3) = 0.9911$

$a_3 < 6.5 = [3+, 3-]$

Entropy $(a_3 < 6.5) = 1 = -1/2 \log 1/2 - 1/2 \log 1/2$

Entropy $(a_3 > 6.5) = [1+, 2-]$

$$= -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}$$

$$= 0.918295883406$$

$$\text{Gain}(a_3 = 6.5) = 0.9911 - \frac{6}{9} * 1$$

$$- \frac{3}{9} * 0.918295883406$$

$$= 0.0183$$

6) Split Point = 7.5

Entropy $(a_3) = 0.9911$

$a_3 < 7.5 = [+4, -4]$

$$\text{Entropy}(a_3 < 7.5) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}$$

$$= 1$$

~~Entro~~ $a_3 > 7.5 = [0+, 1-]$

Entropy $(a_3 > 7.5) = 0$

$$\text{Gain}(a_3 = 7.5) = \text{Entropy}(a_3) - \frac{|a_3 < 7.5|}{|a_3|}\text{Entropy}(a_3 < 7.5) - \frac{|a_3 > 7.5|}{|a_3|}(a_3 > 7.5)$$

$$= 0.9911 - \frac{8}{9} * 1 - \frac{1}{9} * 0$$

$$= 0.1022$$

We will create a new table with the values of entropy and information gain for every split point.

a3	Class label	Split point	Entropy	Information Gain
1.0	+	2.0	0.8484	0.1427
3.0	-	3.5	0.9885	0.0026
4.0	+	4.5	0.9183	0.0728
5.0 5.0	- -	5.5	0.9839	0.0072
6.0	+	6.5	0.9728	0.0183
7.0 7.0	+ -	7.5	0.8889	0.1022

The best information gain for a3 is at the Split point 2.0.

(d) What is the best split (among a1, a2, and a3) according to the information gain?
Answer:
a1 has the highest information gain, so a1 produces the best split with value of 0.2294.

(e) What is the best split (between a1 and a2) according to the classification error rate?
Answer:
For a1, error rate = 4/9*(1 − 3/4) + 5/9*(1 − 4/5) = **2/9**
For a2, error rate = 5/9*(1 − 3/5) + 4/9*(1 − 2/4) = **4/9**

The error rate of a2 is greater than error rate of a1, according to the error rate, a1 produces the best split.

(f) What is the best split (between a1 and a2) according to the Gini index?
Answer:
The gini index of a1 is,
$4/9[1-(3/4)^2-(1/4)^2]+5/9[1-(1/5)^2-(4/5)^2]$=**0.3444**

The gini index of a2 is,
$5/9[1-(2/5)^2-(3/5)^2]+4/9[1-(2/4)^2-(2/4)^2]$=**0.4889**

The gini index for a1 is smaller, hence it produces a better split.

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

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

(a) Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose

The contingency tables after splitting on attributes A and B are:

A=T	A=F
4	0
3	3

B=T	B=F
3	1
1	5

The overall entropy before splitting is:

$E_{original} = -0.4*\log_2 0.4 - 0.6*\log_2 0.6=0.971$

The information gain after splitting on A is:

$E_{A=T} = -4/7*\log_2 4/7 - 3/7*\log_2 3/7 = 0.9852$

$E_{A=F} = -3/3*\log_2 3/3 - 0/3*\log_2 0/3 = 0$

Information Gain=$E_{original} -7/10E_{A=T} - 3/10E_{A=F}$ = **0.2813**

The information gain after splitting on B is:
$E_{B=T} = -3/4*\log_2 3/4 - 1/4*\log 1/4 = 0.8113$
$E_{B=F} = -1/6*\log 1/6 -5/6*\log 5/6 = 0.6500$
Information Gain=$E_{original} - 4/10E_{B=T} - 6/10E_{B=F}$ = **0.2565**
Since Information Gain for A is greater than B, attribute A will be chosen to split the node.

(b) Calculate the gain in the Gini index when splitting on A and B. Which attribute would the decision tree induction algorithm choose?
Answer:
The overall gini before splitting is:
$G_{original} = 1 - 0.4^2 - 0.6^2 = 0.48$

The gain in gini after splitting on A is:
$G_{A=T} = 1 -(4/7)^2-(3/7)^2= 0.4898$
$G_{A=F} = 1 -(3/3)^2-(0/3)^2= 0$
Information Gain=$G_{original} - 7/10G_{A=T} - 3/10G_{A=F}$ = **0.1371**

The gain in gini after splitting on B is:
$G_{B=T} = 1 -(1/4)^2-(3/4)^2= 0.3750$
$G_{B=F} = 1 -(1/6)^2-(5/6)^2= 0.2778$
Information Gain=$G_{original} - 4/10G_{B=T} - 6/10G_{B=F}$ = **0.1633**

Since Information Gain for B is greater than A, attribute B will be chosen to split the node.

(c) Figure shows that entropy and the Gini index are both monotonously increasing in the range [0, 0.5] and they are both monotonously decreasing in the range [0.5, 1]. Is it possible that information gain and the gain in the Gini index favor different attributes? Explain.
Answer:
According to the results in (a) and (b), the information gains calculated are different. Information gain does not have the same behaviour when scaled with differences in measures.
Therefore, yes, it is possible that information gain and the gain in the Gini index favor different attributes.

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, T_{10} with 10 leaf nodes and T_{100} with 100 leaf nodes. The accuracies of the two decision trees on data sets A and B are shown in Table.

	Accuracy	
Data Set	T_{10}	T_{100}
A	0.86	0.97
B	0.84	0.77

(a) Based on the accuracies shown in Table 3.3, which classification model would you expect to have better performance on unseen instances?
Answer:
The training accuracy of T_{100} on dataset A is very high, but its test accuracy on dataset B which was not used for training is low. This gap between training and test accuracies implies that T_{100} suffers from overfitting. Even if the training accuracy of T_{100} is very high in dataset A, it is not representative of generalization performance on unseen instances in dataset B.
On the other hand, tree T_{10} the training accuracy on dataset A and the test accuracy of T_{10} on dataset B are not very different. **This implies that T_{10} does not suffer from overfitting or underfitting and will perform better on unseen instances.**

(b) Now, you tested $T_{10}$and T_{100} on the entire data set (A + B) and found that the classification accuracy of T_{10} on the data set (A + B) is 0.85. In contrast, the classification accuracy of T_{100} on the data set (A + B) is 0.87. Based on this new information and your observations from Table 3.3, which classification model would you finally choose for classification?
Answer:
The high accuracy of T100 on the dataset (A + B) is due to the high training accuracy of T100 on dataset A. As seen in (a), it leads to overfitting. The performance of T100 on the dataset (A+B) includes the overfitting of T100 on dataset A. T10 will give great accuracy for unseen instances in the dataset(A+B) due to its great performance for dataset B and there's not much difference in training and testing accuracy. **We would choose T10 over T100 for classification.**

(b) 4.14 Exercises, Q16, Q18 (20 points).

Q16. You are asked to evaluate the performance of two classification models, M1 and M2. The test set you have chosen contains 26 binary attributes, labeled as A through Z.

Table 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,

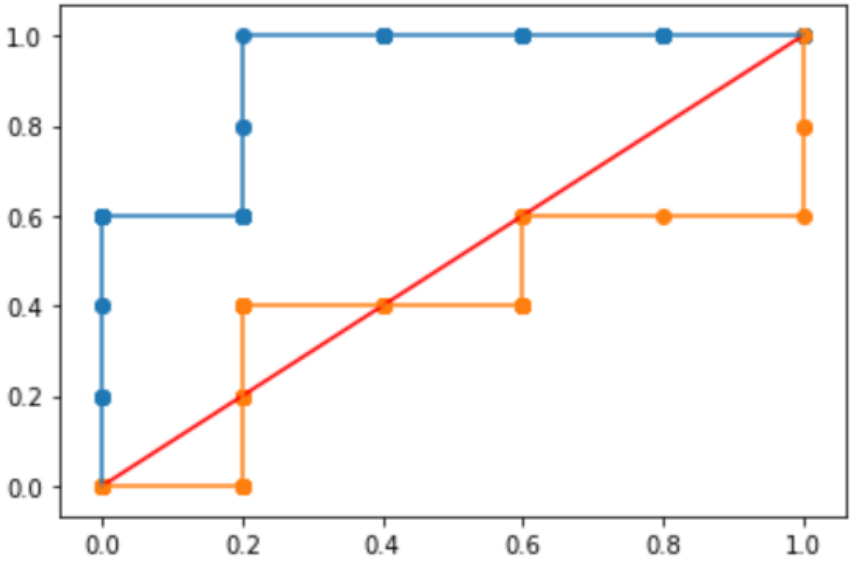
P(−)=1 − P(+) and P(−|A, . . . , Z)=1−P(+|A, . . . , Z). Assume that we are mostly interested in detecting instances from the positive class

Instance	True Class	$P(+\|A,\ldots,Z,M_1)$	$P(+\|A,\ldots,Z,M_2)$
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.

Answer:

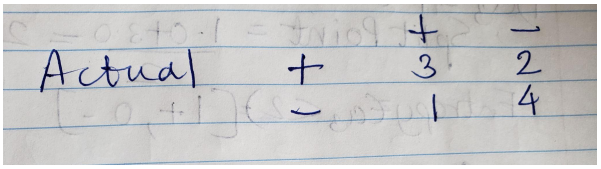
The ROC curve here would be:



M1 is better since its area under the ROC curve is larger than the area under the ROC curve for M2.

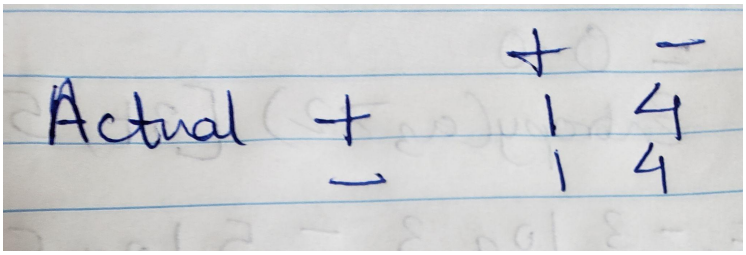
(b) For model M1, suppose you choose the cutoff threshold to be t = 0.5.

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.
Answer:
When t = 0.5, the confusion matrix for M1 is.



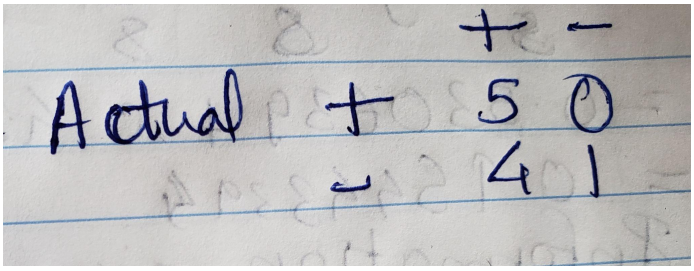
Precision = 3/4 = 75%.
Recall = 3/5 = 60%.
F-measure = (2 × 0.75 × .6)/(0.75 + 0.6) = 0.667.

(c) Repeat the analysis for part (c) 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?
Answer:
When t = 0.5, the confusion matrix for M2 is



Precision = 1/2 = 50%.
Recall = 1/5 = 20%.
F-measure = (2 × 0.5 × .2)/(0.5 + 0.2) = 0.2857.
Based on F-measure, M1 is still better than M2. This result is consistent with the ROC curve.

(d) Repeat part (c) for model M1 using the threshold t = 0.1. Which threshold do you prefer, t = 0.5 or t = 0.1? Are the results consistent with what you expect from the ROC curve?
When t = 0.1, the confusion matrix for M1 is



Precision = 5/9 = 55.6%.
Recall = 5/5 = 100%.
F-measure = (2 × 0.556 × 1)/(0.556 + 1) = 0.715.

We can see which threshold is better by computing the area under the ROC curve.

For t = 0.5, area = 0.6 × (1 - 0.2) = 0.6 × 0.8=0.48.
For t = 0.1, area = 1 × (1 - 0.8) = 1 × 0.2=0.2.
Since the area for t = 0.5 is larger than the area for t = 0.1, we prefer t = 0.5. This result is not consistent with the results using F-measure.

Q 18. 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 records 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 records in the data and the decision tree classifier predicts every test record to be positive. What is the expected error rate of the classifier on the test data?
Answer:
Expected error rate=Number of wrong predictions / Total number of predictions
Here, all records are predicted to be positive and half of the records are positive**.**
Hence, the expected error rate is 50%

(b) Repeat the previous analysis assuming that the classifier predicts each test record to be the positive class with a probability of 0.8 and the negative class with a probability of 0.2.
Answer:
The table for this problem would be

	Actual +	Actual -
Predicted +	0.5*0.8=0.4	0.25

Predicted -	0.25	0.5*0.2=0.1

The error rate would be:
=number of wrong predictions/total predictions=0.25+-.25=0.5
The expected error rate is 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 record to be positive?
Answer:
Here, all records are predicted to be positive and two-thirds of the records are positive.
Hence, the expected error rate is 33%.

(d) Repeat the previous analysis assuming that the classifier predicts each test record to be the positive class with a probability of 2/3 and the negative class with a probability of 1/3.
Answer:
The table for this case is:

	Actual +	Actual -
Predicted +	2/3*2/3=4/9	2/9
Predicted -	2/9	1/3*1/3=1/9

The expected error rate here would be:

=number of wrong predictions/total predictions=2/9+2/9=4/9=0.444.
The expected error rate is 44.4%.

(10 points) 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)?

In [1]:
```
#import the required modules

import pandas as pd
import numpy as np

from sklearn import preprocessing
from sklearn import metrics

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import train_test_split
```

In [2]:
```
#Read the csv file and dataset

data=pd.read_csv("auto-mpg.csv",na_values="?")
data.describe()
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

In [3]:
```
#change the horsepower field to numeric and fill missing values.
#We changed the horsepower field to numeric for calculations later

data['horsepower'] = pd.to_numeric(data['horsepower'])
data = data.fillna(0)
data.head()
```

Out[3]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

In [4]:
```
#Drop car name as we don't need it for predictions

data1=data.drop('car name',1)


#Drop mpg as it is the value we need to predict

data1=data1.drop('mpg',1)

X=data1[list(data1.columns)]

y=data['mpg']

#Perform feature scaling (using L2 norm)
X_normalized=preprocessing.normalize(X, norm='l2')

#Split of training and testing data according to problem statement. 1/3rd data for testing
X_train, X_test, y_train, y_test =train_test_split(X_normalized, y, test_size=0.33,random_state=123)
```

In [5]:
```
#Define the Linear Regression model

lr = LinearRegression()
model = lr.fit(X_train, y_train)

# Finding coefficients
coefficients_list = pd.DataFrame(model.coef_, X.columns,columns=['Coefficients'])
coefficients_list
```

Out[5]:

	Coefficients
cylinders	-2456.030542
displacement	8.539630
horsepower	-163.080173
weight	-61.750582
acceleration	-525.006011
model year	980.794359
origin	1208.596008

The coefficient of the year attribute is 980.794359, which means that as the year increases the mpg value also increases. It means the attributes are positively correlated.

In [6]:
```
#Prediction on the test data
y_pred = model.predict(X_test)

#Calculate accuracy(mean squared error)
print('Accuracy here (Mean Squared Error): ',round(metrics.mean_squared_error(y_test, y_pred), 3))
```

```
Accuracy here (Mean Squared Error):  11.507
```

(10 points) 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.

In [7]:
```
#Set range of values for alphas and find best parameters using RidgeCV for Ridge regression and LassoCV for Lasso Regression
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
alphas = np.logspace(-6, 6, 13)
clf_ridge = RidgeCV(cv=RepeatedKFold(n_splits=10,n_repeats=3,random_state=1),alphas=alphas).fit(X_train, y_train)
clf_ridge_alpha=clf_ridge.alpha_
alphas = np.logspace(-6, 6, 13)
clf_lasso = LassoCV(cv=RepeatedKFold(n_splits=10,n_repeats=3,random_state=1),alphas=alphas).fit(X_train, y_train)
clf_lasso_alpha=clf_lasso.alpha_
print(clf_ridge_alpha)
print(clf_lasso_alpha)
```

```
0.0001
0.001
```

In [8]:
```
#Use the best alpha value to find the coefficients for Ridge Regression

model_ridge = Ridge(alpha=clf_ridge_alpha)
model_ridge.fit(X_train,y_train)
y_pred = model_ridge.predict(X_test)
coefficients_list_ridge = pd.DataFrame(model_ridge.coef_, X.columns,columns=['Coefficients'])
print(coefficients_list_ridge)
print('Accuracy (Mean Squared Error): ',round(metrics.mean_squared_error(y_test, y_pred), 3))
```

```
              Coefficients
cylinders     -144.450046
displacement    -40.091388
horsepower     -135.194772
weight          -55.469389
acceleration   -309.568770
model year      885.361242
origin          153.798514
Accuracy (Mean Squared Error):  11.953
```

In [10]:
```
#Use the best alpha value to find the coefficients for Ridge Regression

model_lasso = Lasso(alpha=clf_lasso_alpha)
smodel_lasso.fit(X_train,y_train)
y_pred = model_lasso.predict(X_test)
coefficients_list_lasso = pd.DataFrame(model_lasso.coef_, X.columns,columns=['Coefficients'])
print(coefficients_list_lasso)
print('Accuracy (Mean Squared Error): ',round(metrics.mean_squared_error(y_test, y_pred), 3))
```

```
              Coefficients
cylinders       -0.000000
displacement    -38.064104
horsepower      -93.725406
weight            0.000000
acceleration     -0.000000
model year      813.792933
origin            0.000000
Accuracy (Mean Squared Error):  12.165
```

(10 points) 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?

In case of Ridge regression, the positive coefficient values are larger for model year and origin compared to the same coefficients in Lasso Regression.

For negative coefficient values, the value is larger for cylinders,displacement and smaller for horsepower compared to the same coefficients in Lasso Regression.

The MSE is the best for linear regression without regularization regardless of whether the regularization is Lasso or Ridge. Between Ridge and Lasso, Ridge performs better on the test dataset than Lasso.

In [ ]:

This paper is about presenting a systemic approach to achieving fairness in binary classification. The approach reduces fair classification to a sequence of cost-sensitive classification problems whose solutions give us a randomized classifier with the lowest error subject to the required constraints. For this approach, the paper introduces two reduction techniques that will work for any problem and overcome the many disadvantages faced.

Machine learning systems have recently come under scrutiny due to their inadvertent discrimination against minorities when it comes to allocating resources like loans or providing opportunities like jobs. Machine Learning researchers have now turned their attention toward the design of fair classification and regression algorithms. The paper studies the task of binary classification by subjecting it to fairness constraints with respect to a pre-defined attribute like race or sex. There has been previous work on this problem which can be classified into two groups.The first group consists of approaches that involve specific quantitative definitions of fairness and putting them in machine learning methods. This is done by relaxing the desired definitions of fairness and enforcing weaker constraints. This includes a lack of correlation and the resulting fairness usually only holds under strong distributional assumptions tied to specific classifiers like SVM. The second group of approaches consists of the elimination of restrictions on specific classifiers and treating the classification as a "black box". On top of this, a wrapper is implemented which works either by pre-processing or post-processing the predictions of the classifier.

The approach in the paper is based on the second group of approaches while eliminating its disadvantages. This approach allows any definition of fairness that can be formalized via linear inequalities on conditional moments, such as demographic parity or equalized odds. This will reduce binary classification to a bunch of cost-sensitive classification problems. This approach is algorithmic, applicable to any classifier family and the results are finite-sample guarantees.
The problem is formulated as such: a binary classification setting where the training examples consist of triples (X, A, Y), where $X \in \chi$ is a feature vector, $A \in A$ is a protected attribute, and $Y \in \{0, 1\}$ is a label. X can contain A or other features which indicate A. The problem requires to learn an accurate classifier $h : \chi \rightarrow \{0, 1\}$ from some set of classifiers H, such as linear threshold rules, decision trees, or neural nets, while satisfying some definition of fairness.
There are two well known quantitative definitions of fairness that are used in this paper based on previous work: Demographic parity and equalized odds. Each definition can be viewed as a special case of a general set of linear constraints. It's formula is:

$$M\mu(h) \leq c$$

where the Matrix M and linear vector c describe the constraints and $\mu(h)$ is is a vector of conditional movements. The paper then solves an example for Demographic parity constraints by expressing each equality constraint as a pair of inequality constraints, which allows us to control the enforcement of the constraint. The violations of the constraints would be minimal. The paper gives us an example of Equalized Odds as well.

The paper then has a section that gives us details about how to solve the fair classification problem by reduction to a sequence of cost-sensitive problems. We input a data set of training examples into the

algorithm which includes parameters denoting the losses for predicting labels 0 and 1. The algorithm outputs an equation that allows us to specify different costs for a variety of training examples essential for incorporating fairness constraints. Then, in the reduction stage, we rewrite the problem as a saddle point problem. This is done by introducing a Lagrange multiplier for each of the constraints and we form a Lagrangian. The paper then performs the error analysis on the three errors introduced in the results of the reductions performed. There are errors introduced in the process as they use empirical values instead of actual values, which is unavoidable. The other two errors are a result of placing a bound on the values of the Lagrangian and the suboptimality of the level v. To bound the statistical error, the paper proposes to use the Rademacher complexity of the classifier family , which we denote by Rn(H), where n is the number of training examples.
In some situations, where the number of constraints is very small, the paper proposes to use a deterministic classifier, even if it means low accuracy. The paper asks us to consider a grid of values λ(the Lagrangian result values), calculate the best response for each value, and then select the value with the desired tradeoff between accuracy and fairness. This is called as grid search.

The paper then consists of a section which have the results of the experiments performed by using the exponentiated-gradient reductions and comparing the results with previous approaches for demographic parity and equalized odds. They applied their reductions on 4 datasets, splitting each dataset into 75% examples for training and 25% examples for testing. They ran their reduction across a wide range of tradeoffs between the classification error and fairness constraints. Almost all the datasets had substantial reduction in disparity without much impact on classifier accuracy. The reduction performed better than all other approaches used in previous papers. The paper then concludes that the reduction techniques they proposed work for any type of classifier in practice. There is optimization of the tradeoff between accuracy and any definition of fairness. However, there is further research needed in cases where we need to achieve fairness without training-time access to protected attributes. Also, further research is needed to handle the tradeoff between accuracy and multiple definitions of fairness.

I really liked the paper as it is detailed in its approach and presents the idea really well. We need to ensure that the machine learning algorithms that we use are fair and not discriminatory, even by accident. However, the equations used in the paper are a bit complex and people need to be familiar with advanced mathematics.