PROBLEM SET 2 ANSWER KEY

Chapter 4, Question 5

a). With a linear Bayes decision boundary, the LDA performs better on the test set and the QDA performs better on the train set. This is because LDA is much less flexible and has lower variance. It performs better than QDA if the number of observations are fewer.

b). QDA performs better on both the training and test sets.

c). As the sample size, n, increases, the test prediction accuracy of QDA relative to LDA improves because LDA performs better when the number of observations are less.

d). False. Using QDA can result in overfitting because its a more flexible model and fits well on the training data. Even though the training error might be lower, the test error can still be higher.

Chapter 4, Question 6

a)

$$P(\hat{Y}) = \frac{e^{-6+0.05*40+1*3.5}}{e^{-6+0.05*40+1*3.5}+1}$$

$$P(\hat{Y}) = \frac{e^{-0.5}}{e^{-0.5}+1}$$

$$P(\hat{Y}) = 0.3775$$

b) $P(\hat{Y}) = \frac{e^{-6+0.05*x+1*3.5}}{e^{-6+0.05*x+1*3.5}+1}$

$$e^{0.05x-2.5} = 0.5+0.5\left(e^{0.05x-2.5}\right)$$

$$e^{0.05x-2.5} = 1$$

$$ln\, e^{0.05x-2.5} = ln\, 1$$

$$0.05x - 2.5 = 0$$

$$x = 50$$

Chapter 4, Question 7

X = last years % profit

Y = issue stock (1,0)

$$E(X \vee Y=1) = 10$$

$$E(X \vee Y=0) = 0$$

$Var(X)=36$

$P(Y=1)=0.8\,; P(Y=0)=0.2$

Find $P(Y=1 \vee X=4)$

Using Bayes:

$$P(Y=k \vee X=x)=\frac{\pi_k \cdot f_k(x)}{\Sigma_{l=1}^{k} \pi_l \cdot f_l(x)}$$

Y = 1

$$f(X \vee Y=1)=\frac{1}{\sqrt{2\pi\sigma^2}} * e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$f_{yes}(4)=\frac{1}{\sqrt{(2\pi)(36)}} * e^{\frac{-(x-10)^2}{2(36)}}$$

$$f_{yes}(4)=\frac{e^{-\frac{1}{2}}}{72\pi}$$

Y = 0

$$f(X \vee Y=1)=\frac{1}{\sqrt{2\pi\sigma^2}} * e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$f_{yes}(4)=\frac{1}{\sqrt{(2\pi)(36)}} * e^{\frac{-(x-0)^2}{2(36)}}$$

$$f_{yes}(4)=\frac{e^{-\frac{2}{9}}}{72\pi}$$

$$P(Y=1 \vee X=4)=\frac{\pi_{yes} * f_{yes}(4)}{\pi_{no} * f_{no}(4) + \pi_{yes} * f_{yes}(4)}$$

$$P(Y=1 \vee X=4)=\frac{\frac{0.8*1}{\sqrt{72\pi}} * e^{-\frac{1}{2}}}{0.2*\left(\frac{1}{\sqrt{72\pi}} * e^{-\frac{2}{9}}\right) + 0.8*\left(\frac{1}{\sqrt{72\pi}} * e^{-\frac{1}{2}}\right)}$$

$$P(Y=1 \vee X=4)=0.752$$

Chapter 4, Question 14

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import 
QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

```
# Read the dataset
auto = pd.read_csv('Data-Auto.csv', index_col = "Unnamed: 0")
auto.head()
```

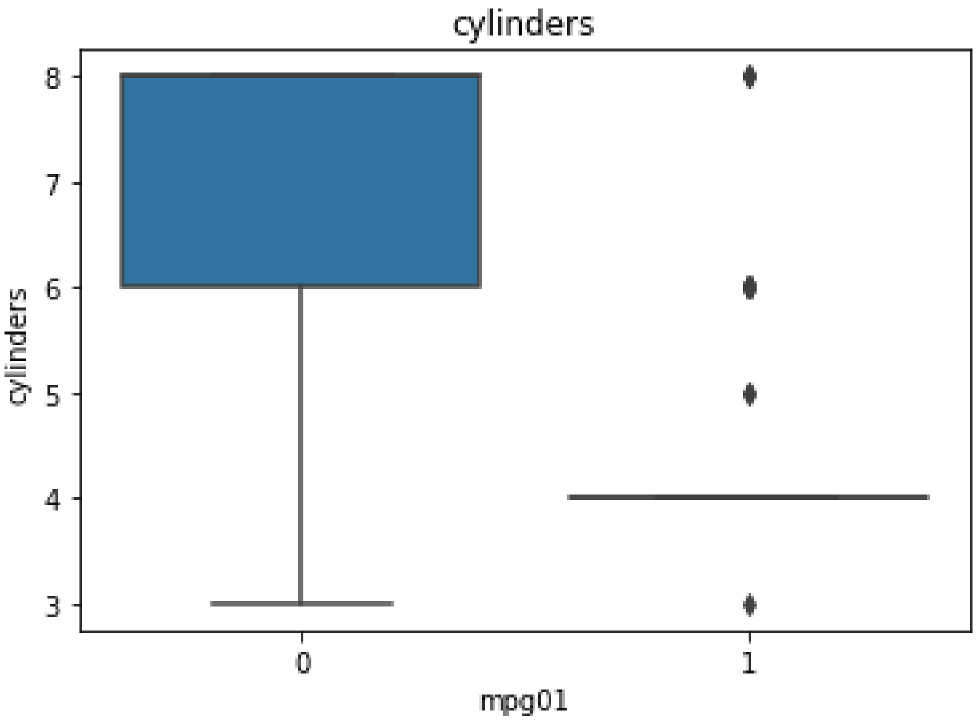
```
    mpg  cylinders  displacement  horsepower  weight  acceleration  
year  \
1  18.0          8         307.0         130    3504          12.0  
70   
2  15.0          8         350.0         165    3693          11.5  
70   
3  18.0          8         318.0         150    3436          11.0  
70   
4  16.0          8         304.0         150    3433          12.0  
70   
5  17.0          8         302.0         140    3449          10.5  
70   

   origin                       name  
1       1  chevrolet chevelle malibu  
2       1          buick skylark 320  
3       1         plymouth satellite  
4       1              amc rebel sst  
5       1                ford torino  
```

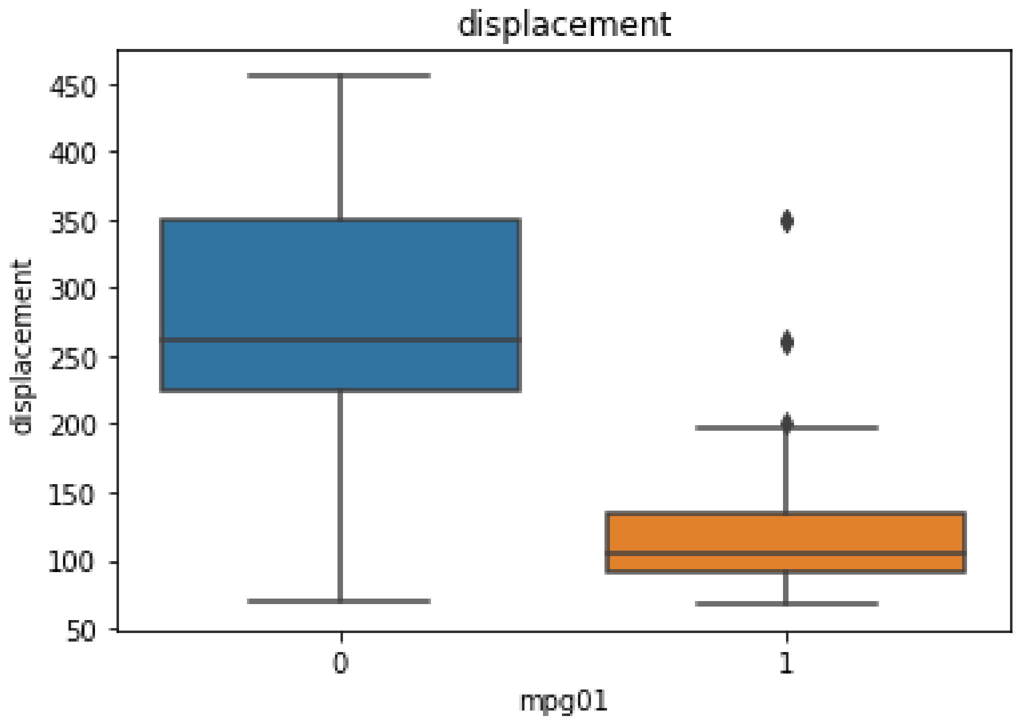
(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median,and a 0 if mpg contains a value below its median. You can compute the median using the median()function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.

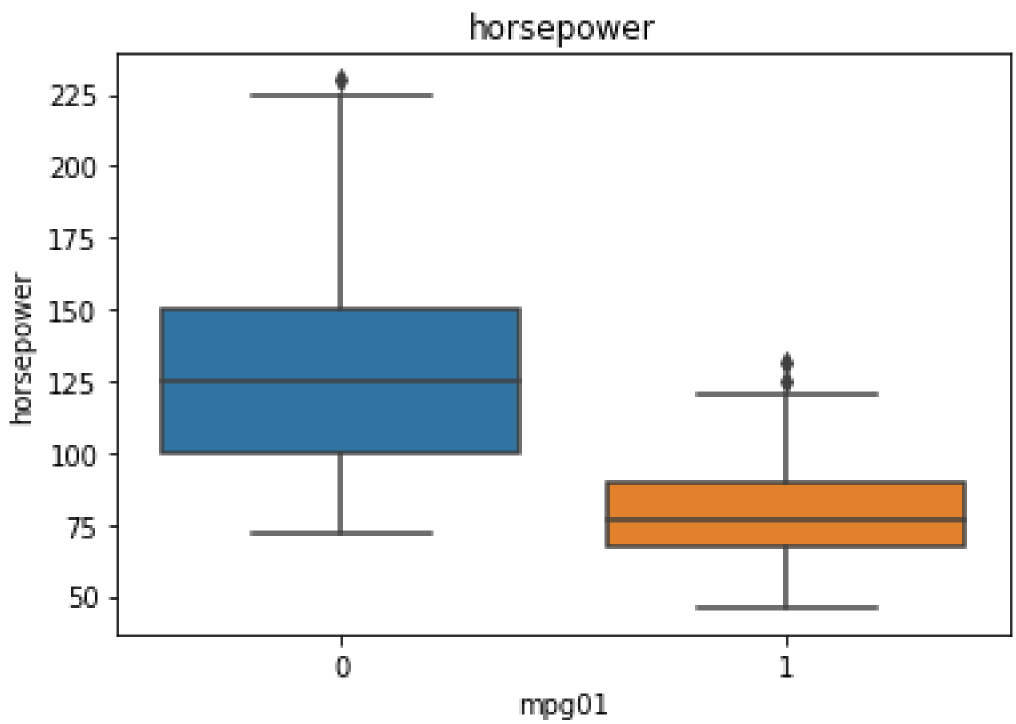
```
# SOLUTION
auto['mpg01'] = np.where(auto['mpg'] > auto['mpg'].median(), 1, 0)
```

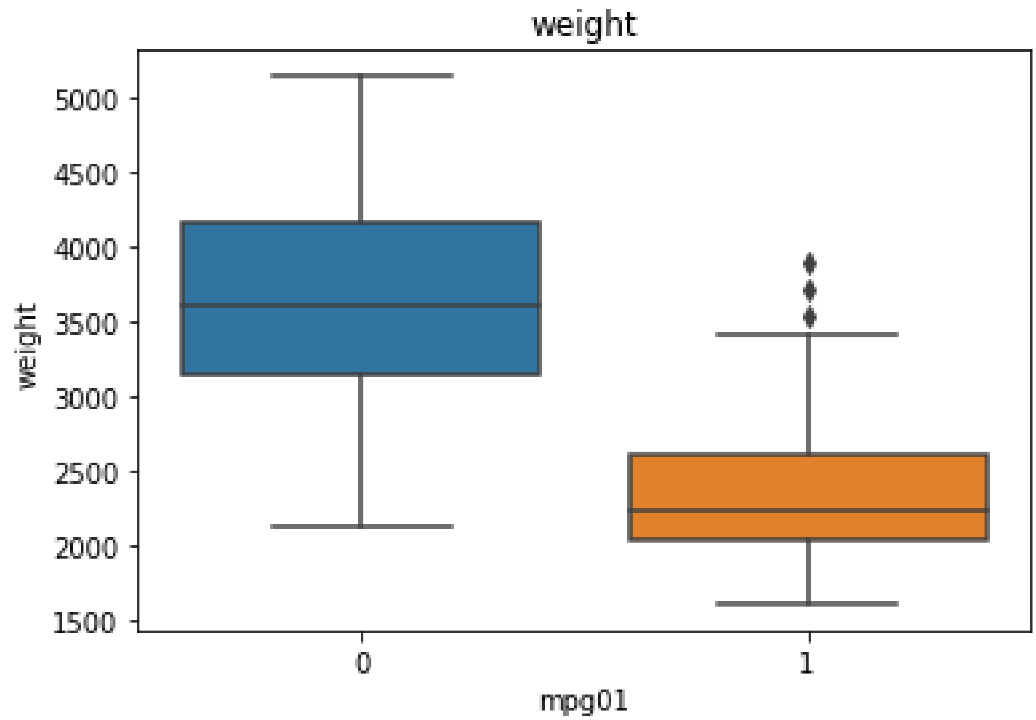
(b). Explore the data graphically in order to investigate the association between mpg01 and theother features. Which of the other features seem most likely to be useful in predicting mpg01?Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

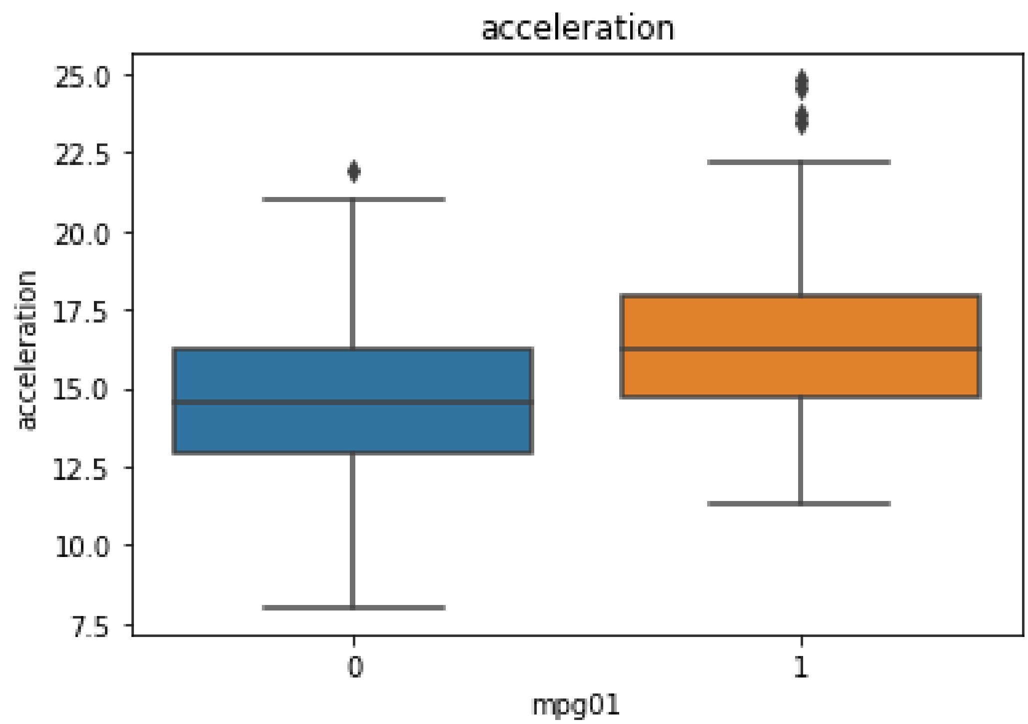
```python
# SOLUTION
features = ['cylinders', 'displacement', 'horsepower',
            'weight', 'acceleration', 'year', 'origin']

for feature in features:
    plt.figure(figsize = (6, 4))
    sns.boxplot(x = auto['mpg01'],
                y = auto[feature])
    plt.title(feature)
    plt.show()
```

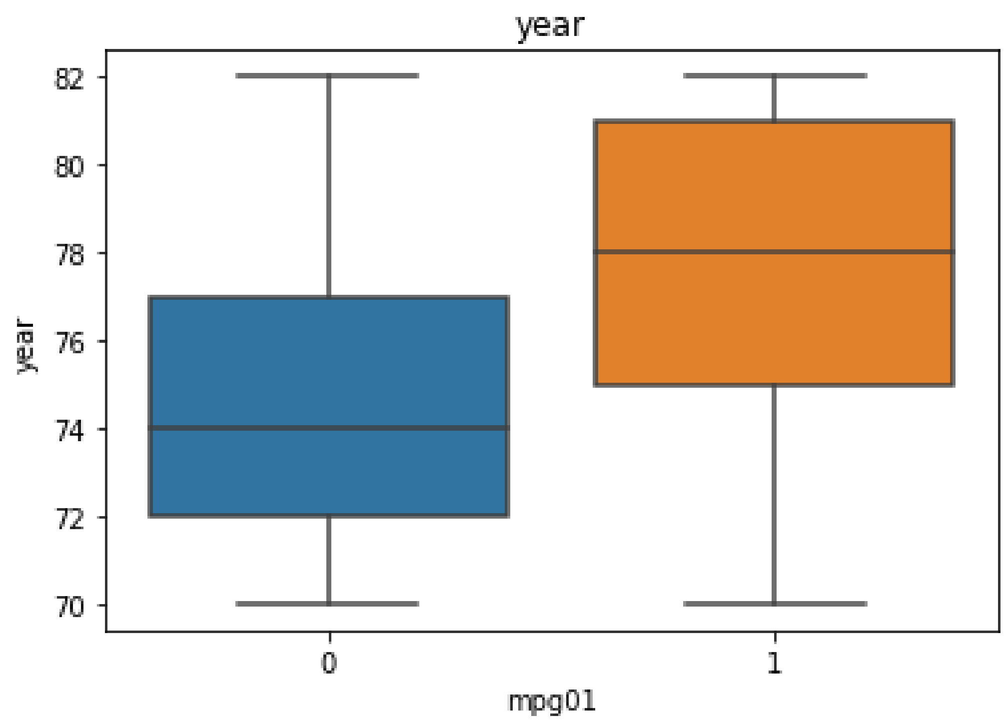


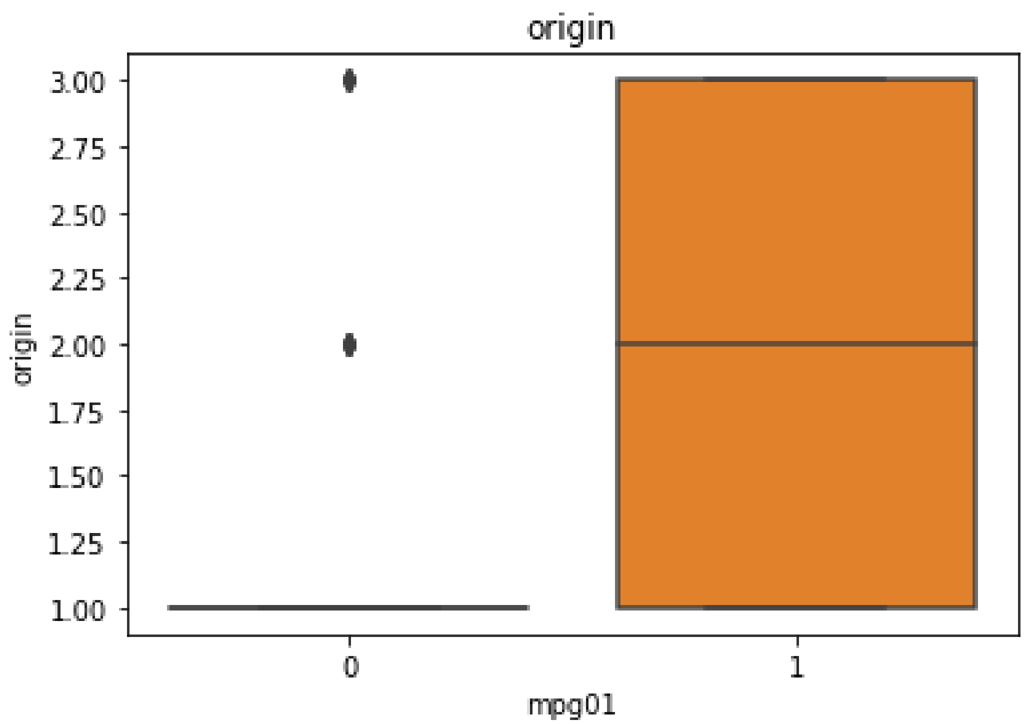

displacement
450
400
350
300
250
200
150
100
50
displacement
0
1
mpg01


horsepower
225
200
175
150
125
100
75
50
horsepower
0
1
mpg01


weight
5000
4500
4000
3500
3000
2500
2000
1500
weight
0
1
mpg01


acceleration
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
acceleration
0
1
mpg01





There is a significant difference in dsitribution between cars with mpg01 = 1 andd mpg01 = 0 as can be seen from the boxplots above. Acceleration is the only variable that does not seem to have a very strong difference so we can exclude it from our models (I have not excluded it from mine in this example).

(c) Split the data into a training set and a test set.

```
# SOLUTION
X = auto[features]
y = auto['mpg01']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

(d) Perform LDA on the training data in order to predict mpg01 using the variables that seemedmost associated with mpg01 in (b). What is the test error of the model obtained

```
# SOLUTION
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
y_pred_lda = lda.predict(X_test)
test_error_lda = 1 - accuracy_score(y_test, y_pred_lda)
print(f"Test error rate for LDA is: {round(test_error_lda, 4)}")
```

```
Test error rate for LDA is: 0.1266
```

(e) Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
# SOLUTION
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
y_pred_qda = qda.predict(X_test)
test_error_qda = 1 - (accuracy_score(y_test, y_pred_qda))
print(f"Test error rate for QDA is: {round(test_error_qda, 4)}")
```

```
Test error rate for QDA is: 0.1139
```

(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
# SOLUTION
logreg = LogisticRegression(max_iter=200)
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
test_error_logreg = 1 - accuracy_score(y_test, y_pred_logreg)
print(f"Test error rate for Logistic Regression is:
{round(test_error_logreg, 4)}")
```

```
Test error rate for Logistic Regression is: 0.1266
```

(g) Perform naive Bayes on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
# SOLUTION
gnb = GaussianNB()
```

```
gnb.fit(X_train, y_train)
y_pred_nb = gnb.predict(X_test)
test_error_nb = 1 - accuracy_score(y_test, y_pred_nb)
print(f"Test error rate for Naive Bayes is: {round(test_error_nb,4)}")
```

```
Test error rate for Naive Bayes is: 0.1266
```

Chapter 5, Question 5

In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis

(a). Fit a logistic regression model that uses income and balance to predict default.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = pd.read_csv('Data-Default.csv')
data.head()
```

	default	student	balance	income
0	No	No	729.526495	44361.625074
1	No	Yes	817.180407	12106.134700
2	No	No	1073.549164	31767.138947
3	No	No	529.250605	35704.493935
4	No	No	785.655883	38463.495879

```
# map Yes to 1 and No to 0
data['default'] = np.where(data['default'] == 'Yes', 1, 0)
X = data[['income', 'balance']]
y = data['default']

# Fitting our regression
logreg = LogisticRegression()
logreg.fit(X, y)

# If you want to see the coefficients
print(logreg.coef_)
print(logreg.intercept_)

# Predictions
y_pred = logreg.predict_proba(X)[:, 1]
print(f"List of predictions: {y_pred}")
```

```
[[2.08091984e-05 5.64710797e-03]]
[-11.54047811]
List of predictions: [1.50473385e-03 1.26192633e-03 8.02623287e-03 ...
3.88719029e-03
 1.28189801e-01 4.29709047e-05]
```

b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

ii. Fit a multiple logistic regression model using only the training observations.

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
# i.
# Splitting our data
X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.30, random_state=42)

# ii
#New logistic regression, using only the training dataset
logreg2 = LogisticRegression()
logreg2.fit(X_train, y_train)

LogisticRegression()

# iii
pred_probs = logreg2.predict_proba(X_test)[:, 1]
pred_probs = np.where(pred_probs > 0.5, 1, 0)
pred_probs

array([0, 0, 0, ..., 0, 0, 0])

# iv
err2 = 1 - accuracy_score(y_test, pred_probs)
err2

0.031666666666666662
```

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

```
# Define a function to repeat the process
def conduct_logreg(seed, x_cols):

    X = data[x_cols]
```

```
    y = data['default']

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.3, random_state=seed)
    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)
    pred_probs = logreg.predict_proba(X_test)
    pred_probs = np.where(pred_probs > 0.5, 1, 0)
    error = 1 - logreg.score(X_test, y_test)

    print(f"Error with seed {seed} is: {error}")
    print('')
    return error

seeds = [2, 6, 9]

err_lst = []
for s in seeds:
    e = conduct_logreg(s, ['income', 'balance'])
    err_lst.append(e)

print(f'Average error from three models is:
{sum(err_lst)/len(err_lst)}')
```

```
Error with seed 2 is: 0.023666666666666614

Error with seed 6 is: 0.024666666666666615

Error with seed 9 is: 0.03066666666666662

Average error from three models is: 0.026333333333333328
```

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```
# Create our new variable
data['student'] = np.where(data['student'] == 'Yes', 1, 0)

# Calculate the error
error = conduct_logreg(42, ['student', 'balance', 'income'])
error
```

```
Error with seed 42 is: 0.03166666666666662


0.03166666666666662
```

Adding a dummy variable for student did not decrease the test error rate of the does not seem to be too important to the model