BUILD A LOGISTIC REGRESSION CLASSIFIER AND ESTIMATING ERROR USING LOOCV METHOD

Question 7. In Sections 5.3.2 and 5.3.3, we saw that the cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

(a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P (Direction = "Up"| Lag1, Lag2) > 0.5. Was this observation correctly classified?

(d) Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:

```
i. Fit a logistic regression model using all but the ith observation to predict
 Direction using Lag1 and Lag2.
ii. Compute the posterior probability of the market moving up for the ith observ
ation.
iii. Use the posterior probability for the ith observation in order to predict w
hether or not the market moves up.
iv. Determine whether or not an error was made in predicting the direction for t
he ith observation. If an error was made, then indicate this as a 1, and otherwi
se indicate it as a 0.
```

(e) Take the average of the n numbers obtained in (d) iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

Understanding the Dataset: A data frame with 1089 observations on the following 9 variables.

Year

```
The year that the observation was recorded
```

Lag1

```
Percentage return for previous week
```

Lag2

```
Percentage return for 2 weeks previous
```

Lag3

```
Percentage return for 3 weeks previous
```

Lag4

```
Percentage return for 4 weeks previous
```

Lag5

```
Percentage return for 5 weeks previous
```

Volume

```
Volume of shares traded (average number of daily shares traded in billions)
```

Today

```
Percentage return for this week
```

Direction

```
A factor with levels Down and Up indicating whether the market had a positive or
negative return on a given week
```

1. Importing Libraries

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
import pylab as pl
```

2. Loading the Weekly Dataset

In [2]:

```
weekly_data = pd.read_csv("Weekly.csv")
weekly_data['Direction'] = weekly_data['Direction'].map({'Up' : 1, 'Down':0})
weekly_data = weekly_data.drop(['Year','Lag3','Lag4','Lag5','Volume','Today'],axis=1)
```

- **Replacing Up with 1 and Down with 0 in Direction column.**
- **We also dropped columns such as 'Year', 'Lag3', 'Lag4', 'Lag5', and 'Volume' as we won't be needing this columns for our model as this question requires only Direction, Lag1 and Lag2.**

3. First 5 values of weekly_data

In [3]:

```
weekly_data.head()
```

Out[3]:

	Lag1	Lag2	Direction
0	0.816	1.572	0
1	-0.270	0.816	0
2	-2.576	-0.270	1
3	3.514	-2.576	1
4	0.712	3.514	1

5. Columns in Weekly Dataset

- These helps us in understanding the columns we are dealing with.

In [4]:

```
weekly_data.columns
```

Out[4]:

```
Index(['Lag1', 'Lag2', 'Direction'], dtype='object')
```

6. Checking for missing Values

In [5]:

```
weekly_data.isnull().sum()
```

Out[5]:

```
Lag1         0
Lag2         0
Direction    0
dtype: int64
```

There are no missing values in the Dataset

7. Shape of Weekly Data

In [6]:

```
weekly_data.shape
```

Out[6]:

```
(1089, 3)
```

8. Descriptive Statistics of Weekly Data

8.1 - Data type of all the variables in the Dataset

In [7]:

```
weekly_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1089 entries, 0 to 1088
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Lag1       1089 non-null   float64
 1   Lag2       1089 non-null   float64
 2   Direction  1089 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 25.6 KB
```

We have all the Quantitative values in the Dataset

8.2 Describe()

- This funtion allows us to observe the Count, Mean, Standard Deviation, Minimum value, Maximum Value, Quantile values

In [8]:

```
weekly_data.describe()
```

Out[8]:

	Lag1	Lag2	Direction
count	1089.000000	1089.000000	1089.000000
mean	0.150585	0.151079	0.555556
std	2.357013	2.357254	0.497132
min	-18.195000	-18.195000	0.000000
25%	-1.154000	-1.154000	0.000000
50%	0.241000	0.241000	1.000000
75%	1.405000	1.409000	1.000000
max	12.026000	12.026000	1.000000

- **The mean of Lag1 is 0.150585 and mean of Lag2 is 0.151079.**
- **Standard Deviation of Lag1 and Lag2 is 2.357013 and 2.357254 respectively.**
- **Minimum value of Lag1 is -18.195 and Minimum Lag2 is -18.195**
- **Maximum Lag1 is 12.02600 and Maximum Lag2 is 12.02600.**

9. Correlation in the Dataset

In [9]:

```
weekly_data.corr()
```

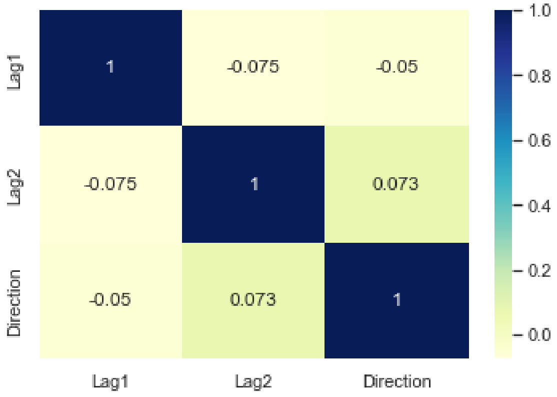
Out[9]:

	Lag1	Lag2	Direction
Lag1	1.000000	-0.074853	-0.050004
Lag2	-0.074853	1.000000	0.072696
Direction	-0.050004	0.072696	1.000000

9.1 Heatmap for Correlation Values

In [10]:

```
sns.set(rc={'figure.figsize' : (6,4)})
sns.heatmap(weekly_data.corr(), cmap="YlGnBu", annot=True)
plt.show();
```

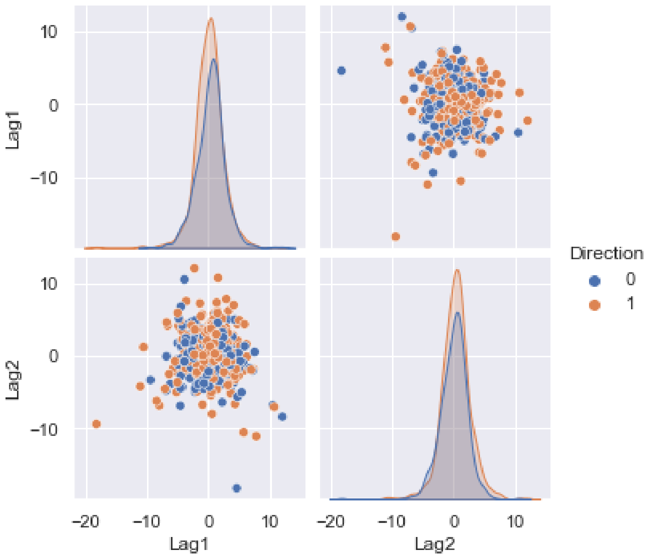


From the values we can observe that there isn't much correlation between the Dataset

9.2 Pairplot to observe Correlation

In [11]:

```
sns.set(rc={'figure.figsize' : (6,4)})
sns.pairplot(weekly_data, hue='Direction')
plt.show();
```

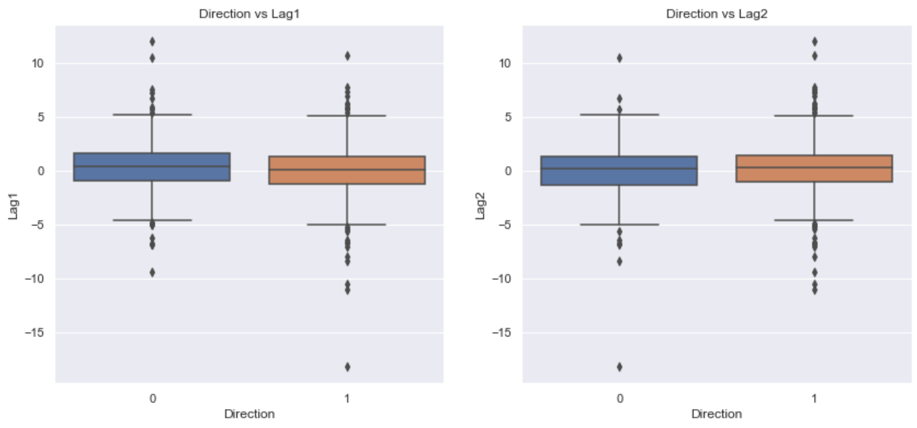


In this Pair Plot if we look closely we can see that the data points are evenly distributed in both Lag1 and Lag2 which means there isn't clear evidence about in which conditions moves Up or Down.

9.3 Box Plot of Lag 1 vs Lag2 and Direction

In [12]:

```
sns.set(rc={'figure.figsize': (14,6)})
fig, axs = plt.subplots(1, 2)
plt.subplot(1,2,1)
sns.boxplot(data=weekly_data,x="Direction",y='Lag1')
plt.title("Direction vs Lag1")
plt.subplot(1,2,2)
sns.boxplot(data=weekly_data,x="Direction",y='Lag2')
plt.title("Direction vs Lag2")
plt.show();
```

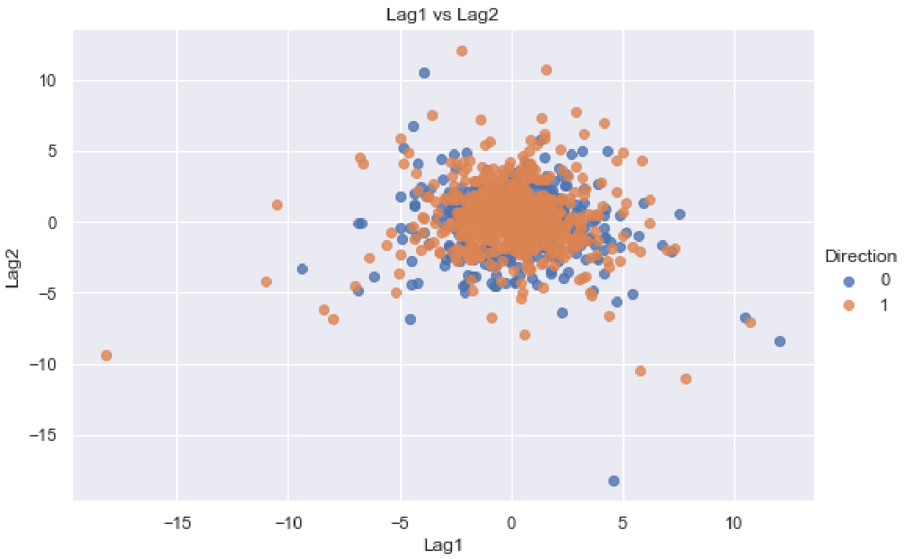


- **There are outliers present in both Direction vs Lag1 and Direction vs Lag2 plot.**
- **As observed from the pairplot, box plot indicates the same observation that Both of this features are evenly spreaded over the direction variable. This can help us train our model to the accurate values and see how it performs while classifying.**

9.4 Scatter plot of Lag1 vs Lag2

In [13]:

```
sns.lmplot(x='Lag1', y='Lag2', hue='Direction',data=weekly_data, aspect=1.5, fit_reg=Fals
plt.title("Lag1 vs Lag2")
plt.show();
```



In this scatter plot we can observe that the spread of both Lag1 and Lag22 is almost same. It is crucial to know what leads the stock market to move Up or Down based on the percentage return for previous week(Lag1) and previous 2 week(Lag2).

9. There are two types of direction in the weekly dataset

- Direction is our Response Variable.
- It contains two values ['Down', 'Up']
- Up : 1 and Down : 0

In [14]:

```
weekly_data['Direction'].unique()
```

Out[14]:

```
array([0, 1], dtype=int64)
```

10. Observe the Number of times the `Direction` went Up or Down

In [15]:

```
weekly_data.groupby('Direction').size()
```

Out[15]:

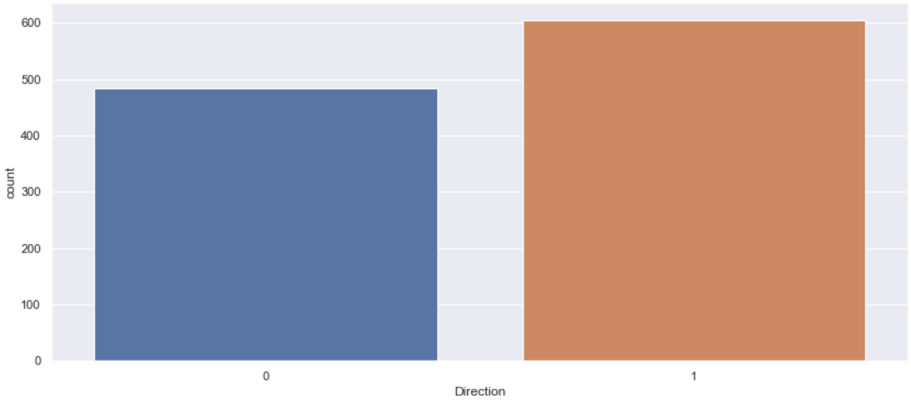
```
Direction
0    484
1    605
dtype: int64
```

We can observe that Direction has 484 rows with value `Down` and 605 rows with value `Up`

10.1 Countplot to observe the data

In [16]:

```
sns.countplot(x='Direction', data = weekly_data ,label='Count')
plt.show();
```



- **Number of times the Direction went Up = 605**
- **Number of times the Direction went Down = 484**

11. Logistic Regresion Model

In [17]:

```
weekly_data.columns
```

Out[17]:

```
Index(['Lag1', 'Lag2', 'Direction'], dtype='object')
```

11.1 Fitting the Logistic Regression Model

In [18]:

```
from statsmodels.formula.api import logit
```

In [19]:

```
weekly_model = logit("Direction ~ Lag1 + Lag2",weekly_data).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.683297
         Iterations 4
```

In [20]:

```python
print(weekly_model.summary())
```

```
                           Logit Regression Results                           
==============================================================================
====
Dep. Variable:              Direction   No. Observations:                     
1089
Model:                          Logit   Df Residuals:                         
1086
Method:                           MLE   Df Model:                             
2
Date:                Sun, 30 Oct 2022   Pseudo R-squ.:                    0.00
5335
Time:                        16:55:42   Log-Likelihood:                   -74
4.11
converged:                       True   LL-Null:                          -74
8.10
Covariance Type:            nonrobust   LLR p-value:                      0.0
1848
==============================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
------------------------------------------------------------------------------
----
Intercept      0.2212      0.061      3.599      0.000       0.101       
0.342
Lag1          -0.0387      0.026     -1.477      0.140      -0.090       
0.013
Lag2           0.0602      0.027      2.270      0.023       0.008       
0.112
==============================================================================
====
```

Lag 2 is significant associated with Direction. We can say that by considering the p-Value for Lag2 column. Let's try to observe our model using a Confusion Matrix. Confusion Matrix will help us understand our model better.

11.2 Predicting Direction column based on our Model

In [21]:

```
weekly_data['Direction_pred'] = weekly_model.predict(weekly_data[['Lag1','Lag2']])
weekly_data.head()
```

Out[21]:

	Lag1	Lag2	Direction	Direction_pred
0	0.816	1.572	0	0.570609
1	-0.270	0.816	0	0.569753
2	-2.576	-0.270	1	0.575592
3	3.514	-2.576	1	0.482496
4	0.712	3.514	1	0.599976

11.3 We will set our Posterior probability cuttoff = 0.5 and optimize the Direction_pred column

In [22]:

```
weekly_data['Direction_pred'] = weekly_data['Direction_pred'].apply(lambda x: 1 if x>0.5
y_pred = weekly_data['Direction_pred']
```

11.4 Confusion Matrix : Confusion matrix tells more about the accuracy of our model and has 4 very important metric count.

- True negatives in the upper-left position.
- False negatives in the lower-left position.
- False positives in the upper-right position.
- True positives in the lower-right position.

In [23]:

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(weekly_data['Direction'],y_pred)
print(conf_matrix)
print("True Negative :",conf_matrix[0,0])
print("True Positive :",conf_matrix[1,1])
print("False Positive :",conf_matrix[0,1])
print("False Negative :",conf_matrix[1,0])
```

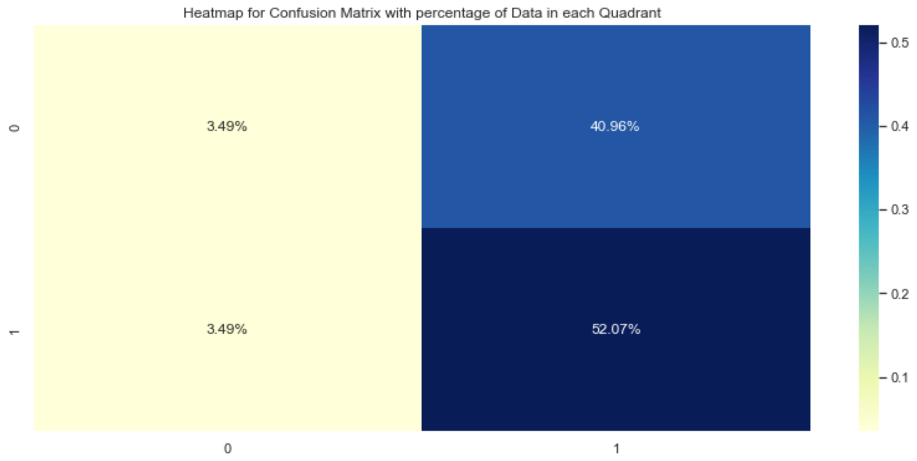
```
[[ 38 446]
 [ 38 567]]
True Negative : 38
True Positive : 567
False Positive : 446
False Negative : 38
```

- True Negative value of 38 suggests that around 38 times real value said the **Percentage Return would move Down** and value predicted by model also said **Percentage Return would move Down**.
- True Positive value of 567 says that around 567 times real value said **Percentage Return would move UP** and value predicted by model also said **Percentage Return would move UP**.
- False positive value of 446 means when model predicted the **Percentage Return would move DOWN** but in real the **Percentage Return moved UP**.
- False negative value of 38 means when model predicted **Percentage Return would move UP** but in real **Percentage Return moved Down**.

11.4.1 Heatmap of Percentage of Data in each quadrant

In [24]:

```
sns.heatmap(conf_matrix/np.sum(conf_matrix), annot=True, fmt='.2%',cmap='YlGnBu')
plt.title("Heatmap for Confusion Matrix with percentage of Data in each Quadrant")
plt.show();
```



12. Misclassification Rate

- It tells you what fraction of predictions were incorrect. It's also called Classification Error.
- Misclassification Rate(MR) is given by :

$$MR = \frac{(FalsePositive + FalseNegative)}{(TruePositive + Truenegative + FalsePositive + FalseNegative)}$$

In [25]:

```
misclassification_Rate = (conf_matrix[0,1] + conf_matrix[1,0])/(conf_matrix[0,0] + conf_r
print("Misclassification Rate using Validation Test Approach is : {:.2f} %".format(miscla
```

```
Misclassification Rate using Validation Test Approach is : 44.44 %
```

We can see that our model has a Misclassification Rate of 44.44%. Around 40.96% where the model predicted that percentage return would move Down but in real the percentage return went UP. Our model was unable to classify around 40% of the data points where the percentage return actually went UP.

13. Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

In [26]:

```
weekData = weekly_data.copy()
weekData = weekData.drop('Direction_pred', axis=1)
```

13.1 Leaving out the First Observation for fitting the Logistic Regression Model

In [27]:

```
X = weekData.iloc[1:len(weekData),:]
y = weekData.iloc[0:1,:]
```

In [28]:

```
X.head()
```

Out[28]:

	Lag1	Lag2	Direction
1	-0.270	0.816	0
2	-2.576	-0.270	1
3	3.514	-2.576	1
4	0.712	3.514	1
5	1.178	0.712	0

In [29]:

```
y
```

Out[29]:

	Lag1	Lag2	Direction
0	0.816	1.572	0

13.2 Fitting the Logistic Regression Model

In [30]:

```
weekModel = logit("Direction ~ Lag1 + Lag2",X).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.683147
         Iterations 4
```

In [31]:

```
print(weekModel.summary())
```

```
                           Logit Regression Results                           
==========================================================================
====
Dep. Variable:              Direction   No. Observations:                  
1088
Model:                          Logit   Df Residuals:                      
1085
Method:                           MLE   Df Model:                          
2
Date:                Sun, 30 Oct 2022   Pseudo R-squ.:                0.00
5387
Time:                        16:55:42   Log-Likelihood:                -74
3.26
converged:                       True   LL-Null:                       -74
7.29
Covariance Type:            nonrobust   LLR p-value:                   0.0
1785
==========================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
--------------------------------------------------------------------------
----
Intercept      0.2232      0.061      3.630      0.000       0.103       
0.344
Lag1          -0.0384      0.026     -1.466      0.143      -0.090       
0.013
Lag2           0.0608      0.027      2.291      0.022       0.009       
0.113
==========================================================================
====
```

Now that we have trained our model based on all of the observations except the first one. Let's try to predict the Direction of our first observation using the above model and see if we can correctly classify the first observation.

13.3 Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P (Direction = "Up"| Lag1, Lag2) > 0.5. Was this observation correctly classified?

In [32]:

```
y['Direction_pred'] = weekModel.predict(y[['Lag1','Lag2']])
```

In [33]:

```
y['Direction_pred'] = y['Direction_pred'].apply(lambda x: 1 if x > 0.5 else 0)
print(y)
```

```
    Lag1   Lag2  Direction  Direction_pred
0  0.816  1.572          0               1
```

The Direction is Not Correctly Classified as the original Direction trend is going DOWN but the trend predicted by the model says the trend is going UP.

14. Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:

(i.) Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.

(ii.) Compute the posterior probability of the market moving up for the ith observation.

(iii.) Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

(iv.) Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

14.1 For loop

In [56]:

```
error=[]
for i in range(1,len(weekData)):
    X = weekData.copy()
    X = X.drop(X.index[i])
    y = weekData.iloc[[i]]

    model = logit("Direction ~ Lag1 + Lag2",X).fit()
    y['Direction_pred'] = model.predict(y[['Lag1','Lag2']])
    y['Direction_pred'] = y['Direction_pred'].apply(lambda x: 1 if x > 0.5 else 0)
    error.append(y['Direction']==y['Direction_pred'])
    print(i)
```

```
         Current function value: 0.683403
         Iterations 4
28
Optimization terminated successfully.
         Current function value: 0.683179
         Iterations 4
29
Optimization terminated successfully.
         Current function value: 0.683070
         Iterations 4
30
Optimization terminated successfully.
         Current function value: 0.683130
         Iterations 4
31
Optimization terminated successfully.
         Current function value: 0.683207
         Iterations 4
32
Optimization terminated successfully.
```

14.2 Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

In [57]:

```
error
```

Out[57]:

```
[1    False
 dtype: bool,
 2    True
 dtype: bool,
 3    False
 dtype: bool,
 4    True
 dtype: bool,
 5    False
 dtype: bool,
 6    True
 dtype: bool,
 7    True
 dtype: bool,
 8    True
 dtype: bool,
 9    False
 dtype: bool,
```

14.3 For loop to estimate Number of times the prediction was wrong and error was made

In [73]:

```
count = 0
for i in range(len(error)):
    if error[i].bool() == False:
        count += 1
```

15. Take the average of the n numbers obtained in (d) (14) iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

In [75]:

```
print("The LOOCV estimate for the Error is {}".format(count/len(error)))
```

```
The LOOCV estimate for the Error is 0.4494485294117647
```

16. Conclusion

In conclusion we can say that Model does a bad job in predicting the market trend. Model is able to classify Upward Trend and Downward trend only on 50% -55 % of the data points which suggests that this model can be risky in

terms of predicting Upward Trend. Also the association between predictor variable and target variable is also quite low. For now, this model has a misclassification rate of around 45%.

In [ ]:

```

```