Project 4

In addition to answering the bolded questions on Coursera, also attach your notebook, both as `.ipynb` and `.html`.

This project should be answered using the `Weekly` data set (attached). This data contains 1,089 weekly stock market percentage returns for 21 years, from the beginning of 1990 to the end of 2010.

Details about the columns in the data are summarized below:

- `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

In this assignment, we will be using PennGrader, a Python package built by a former TA for autograding Python notebooks. PennGrader was developed to provide students with instant feedback on their answer. You can submit your answer and know whether it's right or wrong instantly. We then record your most recent answer in our backend database. You will have 100 attempts per test case, which should be more than sufficient.

NOTE：Please remember to remove the

```
raise notImplementedError
```

after your implementation, otherwise the cell will not compile.

Getting Setup

Please run the below cells to get setup with the autograder. If you need to install packages, please copy these lines into the Terminal!

In [70]:

```
# !pip install pandas==1.0.5 --user
# pip install penngrader --user
```

In [72]:

```
# pip install seaborn --user
# pip install scikit-learn --user
# pip install statsmodels --user
```

Let's try PennGrader out! Fill in the cell below with your PennID and then run the following cell to initialize the grader.

Warning: Please make sure you only have one copy of the student notebook in your directory in Codio upon submission. The autograder looks for the variable `STUDENT_ID` across all notebooks, so if there is a duplicate notebook, it will fail.

In [73]:

```
#PLEASE ENSURE YOUR STUDENT_ID IS ENTERED AS AN INT (NOT A STRING). IF
 NOT, THE AUTOGRADER WON'T KNOW WHO
#TO ASSIGN POINTS TO YOU IN OUR BACKEND

STUDENT_ID = 49731093                   # YOUR 8-DIGIT PENNID GOES HERE
STUDENT_NAME = "Newman Ilgenfritz"      # YOUR FULL NAME GOES HERE
```

In [74]:

```
import penngrader.grader

grader = penngrader.grader.PennGrader(homework_id = 'ESE542_Online_Spri
ng_2021_HW4', student_id = STUDENT_ID)
```

In [75]:

```
# Let's import the relevant Python packages here
# Feel free to import any other packages for this project

# Data Wrangling
import pandas as pd
import numpy as np

# Statistics
import statsmodels.formula.api as smf
import statsmodels.api as sm

# Plotting
import matplotlib.pyplot as plt

%matplotlib inline
```

We're also going to run a quick (0-point) check that the pandas version set up here is correct. If you fail this, please open a Terminal window and run `pip install pandas==1.0.5 --user` otherwise none of the tests will work!

In [76]:

```
grader.grade(test_case_id = 'A0_pandas_test', answer = str(pd.__version
__))
```

```
Correct! You earned 0/0 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Part A

We are first interested in trying to predict the direction of the returns.

To start, load `Weekly.csv` into your notebook.

In [77]:

```
weeklyFile = pd.read_csv('Weekly.csv').copy()
weekly= pd.DataFrame(weeklyFile)
```

In [78]:

```
grader.grade(test_case_id = 'A0_weekly_test', answer = weekly)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

A1.

First, transform our `Direction` variable into a numerical feature that is equal to 1 if `Direction = Up`. Then, pass the dataframe into the test case to make sure it's working properly!

In [79]:

```
#raise NotImplementedError
encode = lambda x: 1 if x == 'Up' else 0
weekly['Direction'] = weekly['Direction'].map(encode)
print('weekly after encoding "direction": ')
print(weekly.head(), '\n')
```

```
weekly after encoding "direction": 
   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today  Direction
0  1990  0.816  1.572 -3.936 -0.229 -3.484  0.154976 -0.270          0
1  1990 -0.270  0.816  1.572 -3.936 -0.229  0.148574 -2.576          0
2  1990 -2.576 -0.270  0.816  1.572 -3.936  0.159837  3.514          1
3  1990  3.514 -2.576 -0.270  0.816  1.572  0.161630  0.712          1
4  1990  0.712  3.514 -2.576 -0.270  0.816  0.153728  1.178          1
```

In [80]:

```
grader.grade(test_case_id = 'A1_direction_test', answer = weekly)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Produce some numerical and graphical summaries of the `Weekly` data. Do there appear to be any patterns?

In [81]:

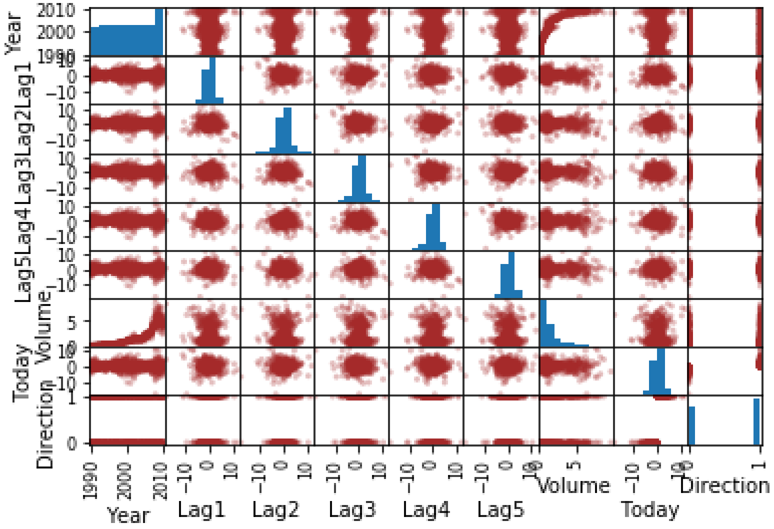
```
#raise NotImplementedError
```

```
print('features correlations: ')
print(weekly.corr(), '\n')
print('\nMatrix of pairwise scatter plots:')
axes = pd.plotting.scatter_matrix(weekly, alpha=0.30, color="brown")
#pd.plotting.scatter_matrix(weekly, alpha=0.30, color="brown")
#g = pd.plotting.scatter_matrix(weekly)
#plt.show()
#plt.close()
#sns.pairplot(weekly)
plt.show()
plt.close()
print()
```

```
features correlations: 
               Year      Lag1      Lag2      Lag3      Lag4      Lag5  \
Year       1.000000 -0.032289 -0.033390 -0.030006 -0.031128 -0.030519   
Lag1      -0.032289  1.000000 -0.074853  0.058636 -0.071274 -0.008183   
Lag2      -0.033390 -0.074853  1.000000 -0.075721  0.058382 -0.072499   
Lag3      -0.030006  0.058636 -0.075721  1.000000 -0.075396  0.060657   
Lag4      -0.031128 -0.071274  0.058382 -0.075396  1.000000 -0.075675   
Lag5      -0.030519 -0.008183 -0.072499  0.060657 -0.075675  1.000000   
Volume     0.841942 -0.064951 -0.085513 -0.069288 -0.061075 -0.058517   
Today     -0.032460 -0.075032  0.059167 -0.071244 -0.007826  0.011013   
Direction -0.022200 -0.050004  0.072696 -0.022913 -0.020549 -0.018168   

             Volume     Today  Direction  
Year       0.841942 -0.032460  -0.022200  
Lag1      -0.064951 -0.075032  -0.050004  
Lag2      -0.085513  0.059167   0.072696  
Lag3      -0.069288 -0.071244  -0.022913  
Lag4      -0.061075 -0.007826  -0.020549  
Lag5      -0.058517  0.011013  -0.018168  
Volume     1.000000 -0.033078  -0.017995  
Today     -0.033078  1.000000   0.720025  
Direction -0.017995  0.720025   1.000000  


Matrix of pairwise scatter plots:
```



Include a brief description of what relationshipis and correlations you find.

In [82]:

```
relationships = '''
    the correlations between the lag variables and today’s returns are close to zero.
    The only significant positive correlations are between Year and Volume (0.841942),
        and between Direction and Today (0.720025).


'''

#raise NotImplementedError
```

In [83]:

```
grader.grade(test_case_id = 'A1_relationships_test', answer = relationships)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

A2.

Use the full data set to perform a logistic regression with `Direction` as the response and the five lag variables as predictors.

In [84]:
```
#raise NotImplementedError
fit1 = smf.glm('Direction~Lag1+Lag2+Lag3+Lag4+Lag5', data = weekly, fam
ily=sm.families.Binomial()).fit()
print(fit1.summary(), '\n')
print('Predicted probability of going Up:',fit1.predict(), '\n') #Predi
ct
```

```
                 Generalized Linear Model Regression Results
=======================================================================
=======
Dep. Variable:              Direction   No. Observations:
1089
Model:                            GLM   Df Residuals:
1083
Model Family:                Binomial   Df Model:
5
Link Function:                  logit   Scale:
1.0000
Method:                          IRLS   Log-Likelihood:
-743.37
Date:                Mon, 15 Feb 2021   Deviance:
1486.7
Time:                        17:17:10   Pearson chi2:
1.09e+03
No. Iterations:                     4
Covariance Type:            nonrobust
=======================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------
-------
```

```
Intercept       0.2303       0.062       3.712       0.000       0.109
0.352
Lag1           -0.0401       0.026      -1.522       0.128      -0.092
0.012
Lag2            0.0602       0.027       2.249       0.025       0.008
0.113
Lag3           -0.0151       0.027      -0.566       0.571      -0.067
0.037
Lag4           -0.0268       0.026      -1.013       0.311      -0.079
0.025
Lag5           -0.0135       0.026      -0.512       0.609      -0.065
0.038
============================================================================
=======

Predicted probability of going Up: [0.59979417 0.5926544  0.57835811
... 0.59836445 0.56799844 0.52963133]
```

Pass in the regression equation to `logit_equation` below. Hint: You do not need the coefficients of the equation yet, just which variables you want to include in the model. Your answer should look something like `Response~Var1+Var2` which is the input for `statsmodels.formula.api`.

In [85]:
```
logit_equation = 'Direction~Lag1+Lag2+Lag3+Lag4+Lag5'
```

In [86]:
```
grader.grade(test_case_id = 'A2_logit_test', answer = logit_equation)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

A3.

Use the `summary()` function to print the results. Do any of the predictors appear to be statistically significant? Which predictors appear to be statistically significant?

In [87]:

```
#raise NotImplementedError
print(fit1.summary(), '\n')
print('Predicted probability of going Up:',fit1.predict(), '\n') #Predi
ct
```

```
                 Generalized Linear Model Regression Results
==========================================================================
=======
Dep. Variable:              Direction   No. Observations:
1089
Model:                            GLM   Df Residuals:
1083
Model Family:                Binomial   Df Model:
5
Link Function:                  logit   Scale:
1.0000
Method:                          IRLS   Log-Likelihood:
-743.37
Date:            Mon, 15 Feb 2021   Deviance:
1486.7
Time:                    17:17:15   Pearson chi2:
1.09e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==========================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
-------
Intercept      0.2303      0.062      3.712      0.000       0.109
0.352
Lag1          -0.0401      0.026     -1.522      0.128      -0.092
0.012
Lag2           0.0602      0.027      2.249      0.025       0.008
0.113
```

```
Lag3           -0.0151       0.027      -0.566       0.571      -0.067
0.037
Lag4           -0.0268       0.026      -1.013       0.311      -0.079
0.025
Lag5           -0.0135       0.026      -0.512       0.609      -0.065
0.038
========================================================================
=======

Predicted probability of going Up: [0.59979417 0.5926544  0.57835811
... 0.59836445 0.56799844 0.52963133]
```

Type the number of apparently significant variables into `num_significant` and the names of the variables into the list `var_significant` -- the test case will only give points if both variables are correct!

In [88]:

```
# 0.05 > Lag2 therefore reject null for Lag2 only
#print(fit1.pvalues) # same as p values in summary, but carried out to
 more sig digits

num_significant = 1
var_significant = ['Lag2'] # This should be a list!
#raise NotImplementedError
```

In [89]:

```
grader.grade(test_case_id = 'A3_significant_test', answer = (num_signif
icant, var_significant))
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

A4.

Compute the overall fraction of correct predictions. Name this variable `fraction_correct_all`. What is the overall fraction of correct predictions?

In [90]:
```
print('\nnumber of obs: ', len(weekly))
encode = lambda x: 1 if x > 0.5 else 0
predictions = pd.DataFrame({'Direction': [encode(x) for x in fit1.predi
ct()]})
print('predictions.head(): ')
print(predictions.head(), '\n')
from sklearn.metrics import confusion_matrix
x =confusion_matrix(weekly['Direction'], predictions['Direction'])
print('confusion_matrix: ')
print(x, '\n')
from sklearn.metrics import accuracy_score
# print('The accuracy score is', 100 * accuracy_score(data['Directio
n'], predictions['Direction']), '%')
aScore = accuracy_score(weekly['Direction'], predictions['Direction'])
print('aScore: ', aScore)
fraction_correct_all = aScore
```

```
number of obs:  1089
predictions.head():
   Direction
0          1
1          1
2          1
3          0
4          1

confusion_matrix:
[[ 49 435]
 [ 41 564]]

aScore:  0.5629017447199265
```

In [91]:
```
print(f'Overall fraction of correct predictions is {fraction_correct_al
l}')
```

```
Overall fraction of correct predictions is 0.5629017447199265
```

```
In [92]: grader.grade(test_case_id = 'A4_fraction_test', answer = fraction_correct_all)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

A5.

Now fit the logistic regression model using a training data period from 1990 to 2007, with `Lag2` as the only predictor.

Compute the overall fraction of correct predictions for the held out data (that is, the data from 2008, 2009 and 2010) and assign it to a variable called `fraction_correct_test`. What is the overall fraction of correct predictions?

```
In [93]: from sklearn.metrics import confusion_matrix
#Train and test split
train = weekly[weekly["Year"] < 2008]
print('train.head: ')
print(train.head(), '\n')
test = weekly[weekly["Year"] >= 2008]
print('test.head: ')
print(test.head(), '\n')
print('test obs: ', len(test), '\n')

# Model
logit_equation = 'Direction~Lag2'
fit2 = smf.glm(logit_equation, data = train, family=sm.families.Binomial()).fit()
print(fit2.summary(), '\n')
#print('Predicted probability of going Up:',fit1.predict(), '\n') #Predict

Xtest = test[weekly.columns[1:-2]]
print('Xtest.head: ')
print(Xtest.head(), '\n')
ytest = test[weekly.columns[-1]]
print('ytest.head: ')
print(ytest.head(), '\n')
#yprobs = fit2.predict(Xtest)
yprobs = fit2.predict(Xtest)

ypred =  [0 if y < 0.5 else 1 for y in yprobs]
x = confusion_matrix(ytest, ypred)
print('cm: ')
print(x, '\n')

from sklearn.metrics import accuracy_score
aScore = accuracy_score(ytest, ypred)

print('aScore: ', aScore)
#print('The accuracy score is', 100 * accuracy_score(data['Direction'],
predictions['Direction']), '%')
fraction_correct_test = aScore


#raise NotImplementedError
```

```
train.head:
   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today  Direction
0  1990  0.816  1.572 -3.936 -0.229 -3.484  0.154976 -0.270          0
1  1990 -0.270  0.816  1.572 -3.936 -0.229  0.148574 -2.576          0
2  1990 -2.576 -0.270  0.816  1.572 -3.936  0.159837  3.514          1
3  1990  3.514 -2.576 -0.270  0.816  1.572  0.161630  0.712          1
4  1990  0.712  3.514 -2.576 -0.270  0.816  0.153728  1.178          1

test.head:
     Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today  Directio
n
933  2008 -4.522 -0.402  1.125 -2.440  1.588  3.372257 -0.752
0
934  2008 -0.752 -4.522 -0.402  1.125 -2.440  4.788802 -5.412
0
935  2008 -5.412 -0.752 -4.522 -0.402  1.125  5.006464  0.409
1
```

```
936  2008  0.409 -5.412 -0.752 -4.522 -0.402  5.100980  4.871
1
937  2008  4.871  0.409 -5.412 -0.752 -4.522  4.539542 -4.596
0

test obs:  156

                 Generalized Linear Model Regression Results
=======================================================================
=======
Dep. Variable:              Direction   No. Observations:
933
Model:                            GLM   Df Residuals:
931
Model Family:                Binomial   Df Model:
1
Link Function:                  logit   Scale:
1.0000
Method:                          IRLS   Log-Likelihood:
-639.25
Date:                Mon, 15 Feb 2021   Deviance:
1278.5
Time:                        17:17:25   Pearson chi2:
933.
No. Iterations:                     4
Covariance Type:            nonrobust
=======================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------
-------
Intercept      0.2266      0.066      3.422      0.001       0.097
0.356
Lag2           0.0472      0.032      1.460      0.144      -0.016
0.110
=======================================================================
=======

Xtest.head:
      Lag1   Lag2   Lag3   Lag4   Lag5    Volume
933 -4.522 -0.402  1.125 -2.440  1.588  3.372257
934 -0.752 -4.522 -0.402  1.125 -2.440  4.788802
935 -5.412 -0.752 -4.522 -0.402  1.125  5.006464
936  0.409 -5.412 -0.752 -4.522 -0.402  5.100980
937  4.871  0.409 -5.412 -0.752 -4.522  4.539542

ytest.head:
933    0
934    0
935    1
936    1
937    0
Name: Direction, dtype: int64

cm:
[[ 7 65]
 [ 5 79]]

aScore:  0.5512820512820513
```

In [94]:

```
fraction_correct_test = fraction_correct_test
# NOTE: at a p value of 0.144, the result is not stat sig, since 0.05 < pValue
#raise NotImplementedError
```

In [95]:

```
print(f'Overall fraction of correct predictions is {fraction_correct_test}')
```

```
Overall fraction of correct predictions is 0.5512820512820513
```

Pass in the train and test datasets to make sure that they're working (feel free to rename the variables in the test case), and then run the test for `fraction_correct_test`!

In [96]:

```
grader.grade(test_case_id = 'A5_df_test', answer = (train, test))
```

```
Correct! You earned 1/1 points. You are a star!
```

```
Your submission has been successfully recorded in the gradebook.
```

In [97]:

```
grader.grade(test_case_id = 'A5_fraction_correct_test', answer = fracti
on_correct_test)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Part B

Now, we want to develop an investment strategy in which we buy if the returns are greater than 0.5% and sell otherwise.

B1.

Create a response variable, `y_i` such that

$$y_i = \begin{cases} 1 \text{ if Today } > 0.5 \\ 0 \text{ otherwise} \end{cases}$$

In [98]:

```
encode = lambda x: 1 if x > 0.50 else 0
weekly['Response'] = weekly['Today'].map(encode)
weekly.head()
```

Out[98]:

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction	Response
0	1990	0.816	1.572	-3.936	-0.229	-3.484	0.154976	-0.270	0	0
1	1990	-0.270	0.816	1.572	-3.936	-0.229	0.148574	-2.576	0	0
2	1990	-2.576	-0.270	0.816	1.572	-3.936	0.159837	3.514	1	1
3	1990	3.514	-2.576	-0.270	0.816	1.572	0.161630	0.712	1	1
4	1990	0.712	3.514	-2.576	-0.270	0.816	0.153728	1.178	1	1

In [99]:
```
grader.grade(test_case_id = 'B1_response_test', answer = weekly)
```

```
Correct! You earned 1/1 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

B2.

Fit a logistic regression model using a training data period from 1990 to 2008, with the five lag variables and volume as predictors.

In [100]:
```
train = (weekly[weekly["Year"] < 2009])
#fit3 = smf.glm('y_1~Lag1+Lag2+Lag3+Lag4+Lag5+Volume', data = weekly2,
 family=sm.families.Binomial()).fit()
fit3 = smf.glm('Response~Lag1+Lag2+Lag3+Lag4+Lag5+Volume', data = weekl
y, family=sm.families.Binomial()).fit()
```

Pass in the regression equation to `logit_equation_B` below

In [101]:
```
logit_equation_B = 'Response~Lag1+Lag2+Lag3+Lag4+Lag5+Volume'
```

In [102]:
```
grader.grade(test_case_id = 'B2_logit_test', answer = logit_equation_B)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

B3.

Use the `summary()` function to print the results. Do any of the predictors appear to be statistically significant? Which predictors appear to be statistically significant?

In [103]:
```
#raise NotImplementedError
print(fit3.summary(), '\n')
print('Predicted probability of going Up:',fit3.predict(), '\n') #Predi
ct
```

```
                 Generalized Linear Model Regression Results                  
==============================================================================
=======
Dep. Variable:               Response   No. Observations:                     
1089
Model:                            GLM   Df Residuals:                         
1082
Model Family:                Binomial   Df Model:                             
6
Link Function:                  logit   Scale:                                
1.0000
Method:                          IRLS   Log-Likelihood:                       
-746.35
Date:                Mon, 15 Feb 2021   Deviance:                             
1492.7
Time:                        17:17:42   Pearson chi2:                         
1.09e+03
No. Iterations:                     4                                         
Covariance Type:            nonrobust                                         
==============================================================================
=======
                 coef    std err          z      P>|z|      [0.025      
0.975]
------------------------------------------------------------------------------
-------
Intercept     -0.1616      0.086     -1.889      0.059      -0.329      
0.006
Lag1          -0.0455      0.026     -1.723      0.085      -0.097      
0.006
Lag2           0.0369      0.027      1.389      0.165      -0.015      
0.089
Lag3          -0.0128      0.026     -0.485      0.628      -0.064      
0.039
Lag4          -0.0156      0.026     -0.592      0.554      -0.067      
0.036
Lag5          -0.0142      0.026     -0.539      0.590      -0.066      
0.037
```

```
Volume          -0.0186       0.037      -0.504       0.614      -0.091
0.054
========================================================================
=======

Predicted probability of going Up: [0.48991811 0.48063181 0.49083875
... 0.46030945 0.44146026 0.42579458]
```

Type the number of apparently significant variables into `num_significant_B` and the names of the variables into the list `var_significant_B` -- the test case will only give points if both variables are correct!

In [104]:

```
print(fit3.pvalues)

num_significant_B = 1
var_significant_B = ['Lag1'] # This has to be a list!
#raise NotImplementedError
```

```
Intercept    0.058859
Lag1         0.084944
Lag2         0.164976
Lag3         0.627794
Lag4         0.553562
Lag5         0.590073
Volume       0.614221
dtype: float64
```

In [105]:

```
grader.grade(test_case_id = 'B3_significant_test', answer = (num_signif
icant_B, var_significant_B))
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

B4.

Compute the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010). Assign this value to the variable `fraction_correct` . What is the overall fraction of correct predictions?

In [106]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

test = (weekly[weekly["Year"] >= 2009])
Xtest = test[weekly.columns[1:-1]]
print(Xtest.head())
ytest = test[weekly.columns[-1]]
print(ytest.head())
print('len ytest: ', len(ytest), '\n')
yprobs = fit3.predict(Xtest)
print(yprobs)
ypred =  [0 if y < 0.5 else 1 for y in yprobs]

x =confusion_matrix(ytest, ypred)
print('cm: ')
print(x, '\n')

aScore =  accuracy_score(ytest, ypred)

print('aScore: ', aScore)
#print('The accuracy score is', 100 * accuracy_score(data['Direction'],
predictions['Direction']), '%')
fraction_correct_test = aScore

fraction_correct = 52/104
print("fraction_correct ", fraction_correct)
```

```
      Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today  Direction
985  6.760 -1.698  0.926  0.418 -2.251  3.793110 -4.448          0
986 -4.448  6.760 -1.698  0.926  0.418  5.043904 -4.518          0
987 -4.518 -4.448  6.760 -1.698  0.926  5.948758 -2.137          0
988 -2.137 -4.518 -4.448  6.760 -1.698  6.129763 -0.730          0
989 -0.730 -2.137 -4.518 -4.448  6.760  5.602004  5.173          1
985    0
986    0
```

```
987    0
988    0
989    1
Name: Response, dtype: int64
len ytest:  104

985     0.356914
986     0.549282
987     0.424508
988     0.408577
989     0.430381
          ...
1084    0.447923
1085    0.395309
1086    0.460309
1087    0.441460
1088    0.425795
Length: 104, dtype: float64
cm:
[[48  5]
 [49  2]]

aScore:  0.4807692307692308
fraction_correct  0.5
```

```
In [107]: print(f'Overall fraction of correct predictions is {fraction_correct}')
```

```
Overall fraction of correct predictions is 0.5
```

```
In [108]: grader.grade(test_case_id = 'B4_fraction_test', answer = fraction_corre
          ct)
```

```
Correct! You earned 2/2 points. You are a star!

Your submission has been successfully recorded in the gradebook.
```

Submit

You're done! Please make sure you've run all the PennGrader cells and count up your score to be sure (there are 20 points in total) and then make sure to submit this on Codio.

In [ ]: