Shea-FA692-HW4

April 13, 2023

1 FA692 Homework 4

2 Due: Wednesday, April 12 @ 11:59PM

Name: Ryan Shea

Date: 2023-04-12

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Set seed of random number generator
CWID = 10445281 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID % 10000
np.random.seed(personal)
```

2.1 Question 1 (15pt)

2.1.1 Question 1.1

Use the `yfinance` package (or other method of your choice) to obtain the daily adjusted close prices for the S&P500 (`SPY`) from January 1, 2023 to March 15, 2023. You should inspect the dates for your data to make sure you are including everything appropriately. Create a data frame (or array) of the daily log returns of this stock; you may concatenate this to your price data. Use the `print` command to display your data.

[ ]:
```
import yfinance as yf

data = yf.download('SPY', start='2023-01-01', end='2023-03-15')

ret = data['Adj Close'].apply(np.log).diff().dropna()#[-2:]

print(ret.tail())
```

```
[*********************100%***********************]  1 of 1 completed
Date
2023-03-08    0.001631
2023-03-09   -0.018622
2023-03-10   -0.014535
2023-03-13   -0.001426
2023-03-14    0.016395
Name: Adj Close, dtype: float64
```

2.1.2 Question 1.2

Scrape data from the Bloomberg `@business` Twitter account from January 1, 2023 to March 15, 2023. Save this data to a Data Frame with time stamps. Additionally, save all the collected data to a text file with time stamps. You will need to submit the text file along with your work (-5 points if not submitted).

Note: Bloomberg tweets sometimes include the pipe "|". I recomment using tilde "~" as a delimiter instead.

Hint: Because saving the tweets can take a long time, you can comment that code out before exporting to pdf.

[ ]:
```python
# import snscrape.modules.twitter as tw

# f = open('business3.txt', 'w', encoding='utf-8')

# for tweet in tw.TwitterSearchScraper(query="(from:business) since:2023-01-01␣
↪until:2023-03-15").get_items():
#     date_str = tweet.date.strftime("%Y-%m-%d %H:%M:%S%z")
#     date_str = date_str[:-2] + ":" + date_str[-2:]
#     #f.write(date_str + "|" + tweet.content + "\n")
#     f.write(date_str + "~" + tweet.rawContent + "\n")
# f.close()

from datetime import datetime as dt
import pytz

business = []
dates = []
f = open('business3.txt', 'r', encoding='utf-8')

for l in f:
    line = l.split('~')
    date_str = line[0]
    try:
        date_time = dt.fromisoformat(date_str)
        date_time = date_time.astimezone(pytz.timezone("US/Eastern"))
        line[0] = date_time
        line[1] = line[1][:-1]
```

```python
        business.append(line)
        dates.append(date_time.date())
    except:
        business[-1][1] += " "+l[:-1]
f.close()

business = pd.DataFrame(business, columns=['Time', 'Tweet'])
business['Date'] = dates

business
```

```
[ ]:                            Time  \
0     2023-03-14 19:40:29-04:00
1     2023-03-14 19:40:29-04:00
2     2023-03-14 19:35:41-04:00
3     2023-03-14 19:31:07-04:00
4     2023-03-14 19:25:09-04:00
…                           …
26809 2022-12-31 19:00:09-05:00
26810 2022-12-31 19:00:09-05:00
26811 2022-12-31 19:00:09-05:00
26812 2022-12-31 19:00:09-05:00
26813 2022-12-31 19:00:08-05:00

                                                    Tweet        Date
0      One Japanese fintech firm is making it compuls…  2023-03-14
1      An unlikely startup guru has emerged in Japan,…  2023-03-14
2      Some US cities are late in making financial di…  2023-03-14
3      The shipping industry is looking to rethink ev…  2023-03-14
4      A biotech wants to cut fashion waste by using …  2023-03-14
…                                                   …           …
26809  Toymakers have found a new group of customers:…  2022-12-31
26810  Belarusian hackers and dissidents determined t…  2022-12-31
26811  Landlords are taking out millions in loans to …  2022-12-31
26812  It took a pandemic to make a dent in US inequa…  2022-12-31
26813  A planned train line in Mexico is billions ove…  2022-12-31

[26814 rows x 3 columns]
```

2.1.3 Question 1.3

Using your favorite sentiment analyzer (e.g., `vaderSentiment`), compute the normalized positive/neutral/negative sentiment for each tweet. Find the average of all 3 polarities of sentiment for the headlines on each date that data was collected. Concatenate these 3 scores score to your data frame of log returns. Use the `print` command to display your data.

```
[ ]: from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
pos, neg, neu = [], [], []
analyser = SentimentIntensityAnalyzer()
for tweet in business['Tweet']:
    score = analyser.polarity_scores(tweet)
    pos.append(score['pos'])
    neg.append(score['neg'])
    neu.append(score['neu'])

business['Positive'] = pos
business['Negative'] = neg
business['Neutral'] = neu

business.head()
```

```
[ ]:                         Time  \
0 2023-03-14 19:40:29-04:00
1 2023-03-14 19:40:29-04:00
2 2023-03-14 19:35:41-04:00
3 2023-03-14 19:31:07-04:00
4 2023-03-14 19:25:09-04:00

                                               Tweet        Date  Positive  \
0  One Japanese fintech firm is making it compuls…  2023-03-14     0.000
1  An unlikely startup guru has emerged in Japan,…  2023-03-14     0.000
2  Some US cities are late in making financial di…  2023-03-14     0.097
3  The shipping industry is looking to rethink ev…  2023-03-14     0.000
4  A biotech wants to cut fashion waste by using …  2023-03-14     0.000

   Negative  Neutral
0     0.000    1.000
1     0.000    1.000
2     0.079    0.824
3     0.368    0.632
4     0.290    0.710
```

2.2 Question 2 (15pt)

2.2.1 Question 2.1

Linearly regress SPY returns as a function of the lagged returns (2 lags). This should be of the form $r_t = \beta_0 + \beta_1 r_{t-1} + \beta_2 r_{t-2}$. Evaluate the performance of this model with the mean squared error of the training data.

```
[ ]: df = pd.DataFrame({"log": ret})
df['pos'] = business.pivot_table(index='Date', values='Positive',␣
 ↪aggfunc='mean')
```

```
df['neg'] = business.pivot_table(index='Date', values='Negative',␣
↪aggfunc='mean')
df['neu'] = business.pivot_table(index='Date', values='Neutral', aggfunc='mean')


for i in range(1, 3):
    df[f'log_{i}'] = df['log'].shift(i)
    df[f'pos_{i}'] = df['pos'].shift(i)
    df[f'neg_{i}'] = df['neg'].shift(i)
    df[f'neu_{i}'] = df['neu'].shift(i)

df = df.dropna() # lose the first 2 rows

print(f"{df.shape = }")

df.head()
```

```
df.shape = (46, 12)
```

[ ]:

```
                 log       pos       neg       neu     log_1     pos_1  \
Date
2023-01-06  0.022673  0.072002  0.070873  0.857135 -0.011479  0.067030
2023-01-09 -0.000567  0.075860  0.057659  0.866483  0.022673  0.072002
2023-01-10  0.006988  0.060580  0.074328  0.865099 -0.000567  0.075860
2023-01-11  0.012569  0.068954  0.068449  0.862574  0.006988  0.060580
2023-01-12  0.003634  0.072893  0.071808  0.855305  0.012569  0.068954

               neg_1     neu_1     log_2     pos_2     neg_2     neu_2
Date
2023-01-06  0.071558  0.861423  0.007691  0.080386  0.076459  0.843152
2023-01-09  0.070873  0.857135 -0.011479  0.067030  0.071558  0.861423
2023-01-10  0.057659  0.866483  0.022673  0.072002  0.070873  0.857135
2023-01-11  0.074328  0.865099 -0.000567  0.075860  0.057659  0.866483
2023-01-12  0.068449  0.862574  0.006988  0.060580  0.074328  0.865099
```

[ ]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

y = df['log']
X = df[['log_1', 'log_2']]

linreg = LinearRegression()
linreg.fit(X, y)

preds = linreg.predict(X)

print(f"Intercept : {linreg.intercept_}", end='\n\n')
```

```
for feat, coef in zip(X.columns, linreg.coef_):
    print(f"{feat:>9} : {coef}")

print()
print(f"MSE: {mean_squared_error(y, preds)}")
```

```
Intercept : 0.0007263457920817806

    log_1 : 0.05614851635772608
    log_2 : -0.12133150745716237

MSE: 0.00011083584269359552
```

2.2.2 Question 2.2

Linearly regress `SPY` returns as a function of the lagged sentiment scores (2 lags). This should be of the form $r_t = \beta_0 + \beta_1^{pos} s_{t-1}^{pos} + \beta_1^{neu} s_{t-1}^{neu} + \beta_1^{neg} s_{t-1}^{neg} + \beta_2^{pos} s_{t-2}^{pos} + \beta_2^{neu} s_{t-2}^{neu} + \beta_2^{neg} s_{t-2}^{neg}$. Evaluate the performance of this model with the mean squared error of the training data.

[ ]:
```
y = df['log']
X = df[['pos_1', 'pos_2', 'neg_1', 'neg_2', 'neu_1', 'neu_2']]

linreg = LinearRegression()
linreg.fit(X, y)

preds = linreg.predict(X)

print(f"Intercept : {linreg.intercept_}", end='\n\n')

for feat, coef in zip(X.columns, linreg.coef_):
    print(f"{feat:>9} : {coef}")

print()
print(f"MSE: {mean_squared_error(y, preds)}")
```

```
Intercept : -39.639385562760594

    pos_1 : -11.529138946763046
    pos_2 : 50.98372428142845
    neg_1 : -10.99345559023581
    neg_2 : 51.318176402586
    neu_1 : -11.428918323943893
    neu_2 : 51.02968537813942

MSE: 9.73307238249023e-05
```

2.2.3 Question 2.3

Linearly regress SPY returns as a function of the lagged returns and sentiment (2 lags each). This should be of the form $r_t = \beta_0 + \beta_{1,r} r_{t-1} + \beta_{2,r} r_{t-2} + \beta_1^{pos} s_{t-1}^{pos} + \beta_1^{neu} s_{t-1}^{neu} + \beta_1^{neg} s_{t-1}^{neg} + \beta_2^{pos} s_{t-2}^{pos} + \beta_2^{neu} s_{t-2}^{neu} + \beta_2^{neg} s_{t-2}^{neg}$. Evaluate the performance of this model with the mean squared error of the training data.

```
[ ]: y = df['log']
     X = df[['log_1', 'log_2', 'pos_1', 'pos_2', 'neg_1', 'neg_2', 'neu_1', 'neu_2']]

     linreg = LinearRegression()
     linreg.fit(X, y)

     preds = linreg.predict(X)

     print(f"Intercept : {linreg.intercept_}", end='\n\n')

     for feat, coef in zip(X.columns, linreg.coef_):
         print(f"{feat:>9} : {coef}")

     print()
     print(f"MSE: {mean_squared_error(y, preds)}")
```

```
Intercept : -43.294419357032616

    log_1 : 0.06832441586556065
    log_2 : -0.02872132228146571
    pos_1 : -9.941652601626126
    pos_2 : 53.08011846350946
    neg_1 : -9.406361871438579
    neg_2 : 53.36617138063956
    neu_1 : -9.858793198413219
    neu_2 : 53.11377869190214

MSE: 9.679835761538497e-05
```

2.2.4 Question 2.4

Compare the performance of these 3 linear regressions. Compare also to the performance from Homework 3 when only the compound sentiment score was used.

The MSEs are lower using the different sentiment scores compared to just the compound score. It does the best when it uses all of the different features as well.

2.3 Question 3 (15pt)

2.3.1 Question 3.1

Regress SPY returns with a random forest as a function of the lagged returns (2 lags). This should be of the form $r_t = f(r_{t-1}, r_{t-2})$. Evaluate the performance of this model with the mean squared error of the training data.

[ ]:
```
from sklearn.ensemble import RandomForestRegressor

y = df['log']
X = df[['log_1', 'log_2']]

rf = RandomForestRegressor()
rf.fit(X, y)

preds = rf.predict(X)

print("*" * 5, "|FEATURE IMPORTANCES|", "*" * 5)
for feat, imp in zip(X.columns, rf.feature_importances_):
    print(f"{feat:>4} : {imp}")

print()
print(f"  MSE : {mean_squared_error(y, preds)}")
```

```
***** |FEATURE IMPORTANCES| *****
log_1 : 0.5194776560977129
log_2 : 0.48052234390228704

  MSE : 2.649891959962792e-05
```

2.3.2 Question 3.2

Regress SPY returns with a random forest as a function of the lagged sentiments (2 lags). This should be of the form $r_t = f(s^{pos}_{t-1}, s^{neu}_{t-1}, s^{neg}_{t-1}, s^{pos}_{t-2}, s^{neu}_{t-2}, s^{neg}_{t-2})$. Evaluate the performance of this model with the mean squared error of the training data.

[ ]:
```
y = df['log']
X = df[['pos_1', 'pos_2', 'neg_1', 'neg_2', 'neu_1', 'neu_2']]

rf = RandomForestRegressor()
rf.fit(X, y)

preds = rf.predict(X)
print("*" * 5, "|FEATURE IMPORTANCES|", "*" * 5, sep=' ')
for feat, imp in zip(X.columns, rf.feature_importances_):
    print(f"{feat:>4} : {imp}")

print()
print(f"  MSE : {mean_squared_error(y, preds)}")
```

```
***** |FEATURE IMPORTANCES| *****
pos_1 : 0.15730213951233896
pos_2 : 0.15919690693901034
neg_1 : 0.309288748696683
```

```
neg_2 : 0.16301310084952395
neu_1 : 0.11615854369598574
neu_2 : 0.09504056030645801

  MSE : 1.6281129434211215e-05
```

2.3.3 Question 3.3

Regress SPY returns with a random forest as a function of the lagged returns and sentiment (2 lags each). This should be of the form $r_t = f(r_{t-1}, r_{t-2}, s^{pos}_{t-1}, s^{neu}_{t-1}, s^{neg}_{t-1}, s^{pos}_{t-2}, s^{neu}_{t-2}, s^{neg}_{t-2})$. Evaluate the performance of this model with the mean squared error of the training data.

[ ]:
```
y = df['log']
X = df[['log_1', 'log_2', 'pos_1', 'pos_2', 'neg_1', 'neg_2', 'neu_1', 'neu_2']]

rf = RandomForestRegressor()
rf.fit(X, y)

preds = rf.predict(X)

print("*" * 5, "|FEATURE IMPORTANCES|", "*" * 5)
for feat, imp in zip(X.columns, rf.feature_importances_):
    print(f"{feat:>4} : {imp}")

print()
print(f"  MSE : {mean_squared_error(y, preds)}")
```

```
***** |FEATURE IMPORTANCES| *****
log_1 : 0.07404880025946274
log_2 : 0.12290223905906761
pos_1 : 0.10456425266166411
pos_2 : 0.12112143148872109
neg_1 : 0.24231828025524188
neg_2 : 0.13086798695909022
neu_1 : 0.11788958649223147
neu_2 : 0.08628742282452098

  MSE : 1.852114314590402e-05
```

2.3.4 Question 3.4

Compare the performance of these 3 random forest regressions. Compare also to the performance from `Homework 3` when only the `compound` sentiment score was used.

The random forest regressions do better than the linear regressions across the board. This one does the best when it uses the individual sentiment scores and NO lagged returns. This is consistent with the findings from HW3.

2.4 Question 4 (10pt)

2.4.1 Question 4.1

Compare the performance of the various regressions utilized. Do you find the text data to be a useful feature in your analysis. Explain why or why not.

Text data is definitely useful in the analysis. The error is significantly lower from both HW3 and HW4 when text data is incorporated in the model. It is generally more important than the lagged returns, in terms of the error as well as the feature importances of the random forest regression.