Final project

Yuan Gao

August 2024

1 Introduction

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, these systems learn from data, identify patterns, and make decisions or predictions based on that information.

Machine learning is used across a wide range of industries and applications, providing powerful tools to analyze data, make predictions, and automate decision-making processes.

2 Methodology

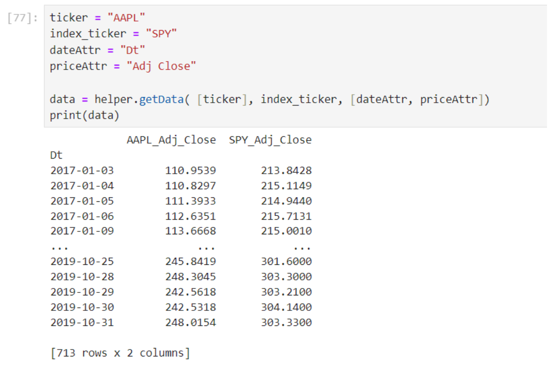
Among all the methods related to machine learning, I chose linear regression to cope with the project. Since linear regression is a fundamental statistical technique used in machine learning and data analysis to model and analyze the relationship between a dependent variable (often called the target or response variable) and one or more independent variables (also known as predictors or features). The primary goal of linear regression is to fit a linear equation to the observed data, which can be used to predict the value of the dependent variable based on the values of the independent variables.

3 Project content

The final project requires us to find the correlation between returns on APPL and SPY 500.

3.1 Get the data

To deal with the project, getting the data should be the first step. Python code : 'helper' can be used to import the data and the data of adjusted close price is used to do the calculation. These data are divided into three columns which can be displayed clearly.



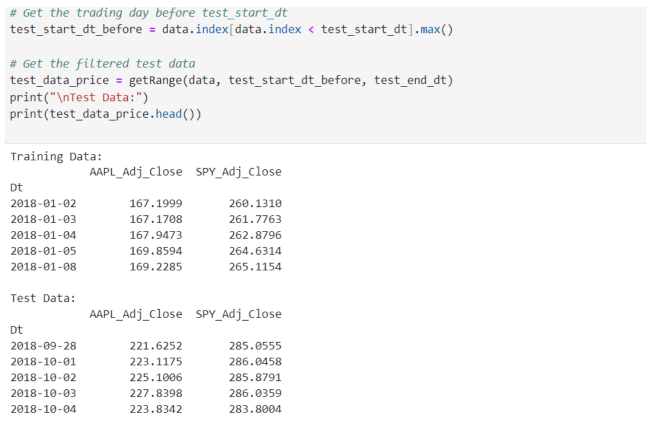
```
[77]: ticker = "AAPL"
      index_ticker = "SPY"
      dateAttr = "Dt"
      priceAttr = "Adj Close"

      data = helper.getData( [ticker], index_ticker, [dateAttr, priceAttr])
      print(data)
```

```
            AAPL_Adj_Close  SPY_Adj_Close
Dt
2017-01-03        110.9539       213.8428
2017-01-04        110.8297       215.1149
2017-01-05        111.3933       214.9440
2017-01-06        112.6351       215.7131
2017-01-09        113.6668       215.0010
...                    ...            ...
2019-10-25        245.8419       301.6000
2019-10-28        248.3045       303.3000
2019-10-29        242.5618       303.2100
2019-10-30        242.5318       304.1400
2019-10-31        248.0154       303.3300

[713 rows x 2 columns]
```

Figure 1: This is the code related to importing the data.

Since we have approximately 700 hundred data, we simplified them. We just choose a subset of 5 data to form the correlation, we then pick other 5 data randomly to test whether the correlation will be suitable for us to predict.

We then fix different start date and an end date for both training data and test data, by using the code 'get range'. Moreover, we need the code 'data.head()' to help us choose the first 5 days of the selected range. Applying these codes, we then get 5 training data and 5 test data as the following:



```
# Get the trading day before test_start_dt
test_start_dt_before = data.index[data.index < test_start_dt].max()

# Get the filtered test data
test_data_price = getRange(data, test_start_dt_before, test_end_dt)
print("\nTest Data:")
print(test_data_price.head())
```

```
Training Data:
            AAPL_Adj_Close  SPY_Adj_Close
Dt
2018-01-02        167.1999       260.1310
2018-01-03        167.1708       261.7763
2018-01-04        167.9473       262.8796
2018-01-05        169.8594       264.6314
2018-01-08        169.2285       265.1154

Test Data:
            AAPL_Adj_Close  SPY_Adj_Close
Dt
2018-09-28        221.6252       285.0555
2018-10-01        223.1175       286.0458
2018-10-02        225.1006       285.8791
2018-10-03        227.8398       286.0359
2018-10-04        223.8342       283.8004
```

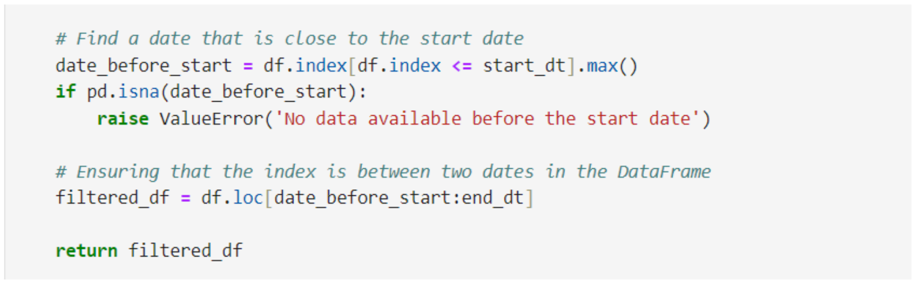
Figure 2: 5 training data and 5 test data.

3.2 Prepare the data

Since we need to find the correlation between two returns, we need to do the calculation using this formula :

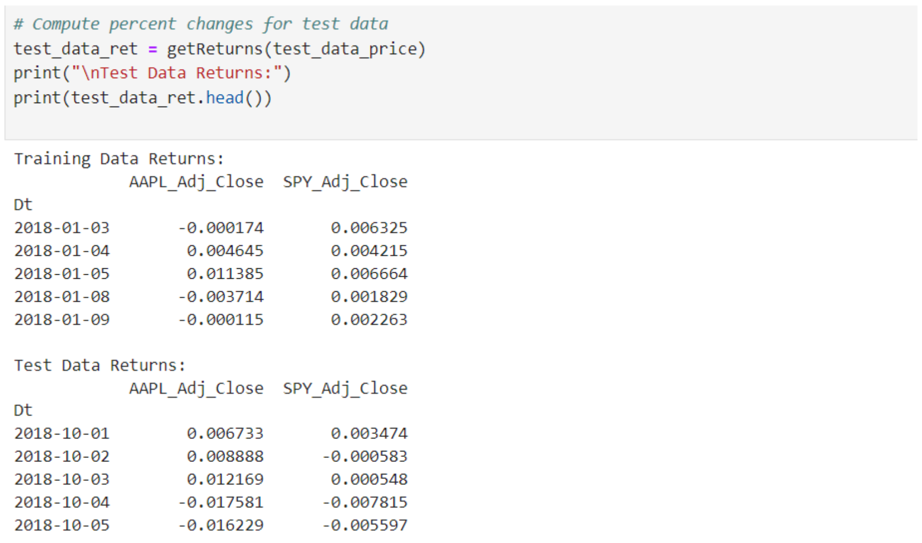
$$r_{AAPL}^{(t)} = \frac{p_{AAPL}^{(t)}}{p_{AAPL}^{(t-1)}} - 1$$

where r means the return on APPL, which is obtained by dividing today's stock price by yesterday's stock price and subtracting 1. Because of this, the date before the starting date is also required during the process. Then we can get the following returns on training data and test data:



```
# Find a date that is close to the start date
date_before_start = df.index[df.index <= start_dt].max()
if pd.isna(date_before_start):
    raise ValueError('No data available before the start date')

# Ensuring that the index is between two dates in the DataFrame
filtered_df = df.loc[date_before_start:end_dt]

return filtered_df
```

Figure 3: Data before start data was used.



```
# Compute percent changes for test data
test_data_ret = getReturns(test_data_price)
print("\nTest Data Returns:")
print(test_data_ret.head())
```

```
Training Data Returns:
            AAPL_Adj_Close  SPY_Adj_Close
Dt
2018-01-03       -0.000174       0.006325
2018-01-04        0.004645       0.004215
2018-01-05        0.011385       0.006664
2018-01-08       -0.003714       0.001829
2018-01-09       -0.000115       0.002263

Test Data Returns:
            AAPL_Adj_Close  SPY_Adj_Close
Dt
2018-10-01        0.006733       0.003474
2018-10-02        0.008888      -0.000583
2018-10-03        0.012169       0.000548
2018-10-04       -0.017581      -0.007815
2018-10-05       -0.016229      -0.005597
```

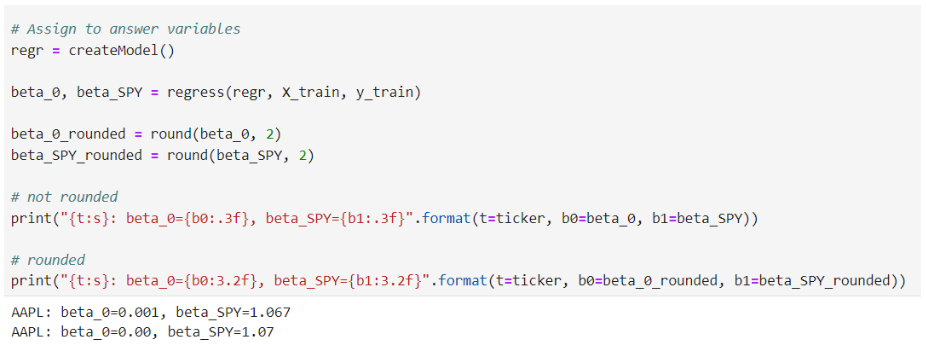
Figure 4: Returns on APPL and SPY500.

3.3 Train a model

We the use the Linear Regression to predict the return of a ticker from the return of the market proxy SPY500. We know that the formula between two returns is :

$$r_{AAPL}^{(t)} = \beta_0 + \beta_{APPL,SPY} * r_{SPY}^{(t)}$$

Then we use the data above to calculate the coefficients of the linear regression equation:



```
# Assign to answer variables
regr = createModel()

beta_0, beta_SPY = regress(regr, X_train, y_train)

beta_0_rounded = round(beta_0, 2)
beta_SPY_rounded = round(beta_SPY, 2)

# not rounded
print("{t:s}: beta_0={b0:.3f}, beta_SPY={b1:.3f}".format(t=ticker, b0=beta_0, b1=beta_SPY))

# rounded
print("{t:s}: beta_0={b0:3.2f}, beta_SPY={b1:3.2f}".format(t=ticker, b0=beta_0_rounded, b1=beta_SPY_rounded))
```

```
AAPL: beta_0=0.001, beta_SPY=1.067
AAPL: beta_0=0.00, beta_SPY=1.07
```

Figure 5: The coefficients calculated based on the observed data.

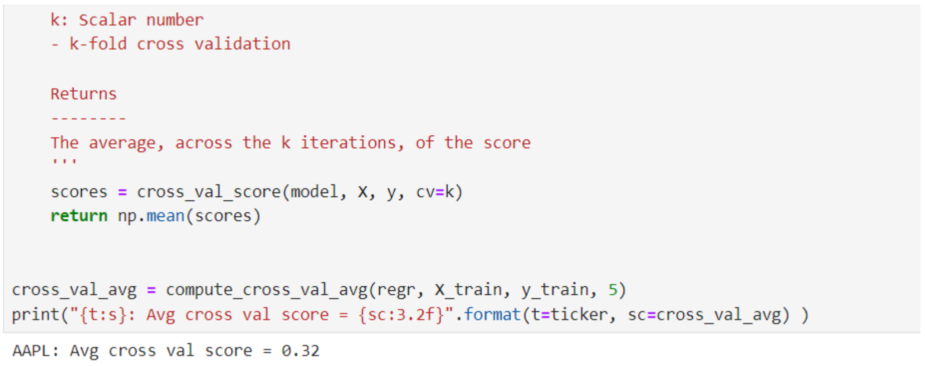
Since I didn't put any restrict on the data, the outcomes seem to be little different from the given output : beta for APPL and SPY = 1.071, but I think it is feasible. And I tried to keep these data to two decimal places, and I find that the result tend to be similar. And i think the reason for the difference is the sample selection bias, and it is acceptable. Besides, the intercept is the same, which is 0.001.

The outcome means with an intercept of 0.001 and a coefficient of 1.067, as the return of the SPY500 increases by one unit, the expected change in the return of AAPL is 1.067. The intercept of 0.001 indicates that when the return of the SPY500 is 0, the expected return of AAPL is 0.001.

3.4 Assess the model

5-fold cross validation is a commonly used method for evaluating machine learning models. In this type of cross validation, the data is divided into 5 approximately equal subsets. The model is then trained 5 times, each time using 4 of the subsets for training and 1 subset for validation. This process is repeated 5 times, with each subset being used for validation once. Finally, the performance evaluation of the model is the average of the 5 validation results.

This method effectively assesses the stability and performance of the model across different subsets of the data. To ensure the randomness of the data, we use Python for calculations:



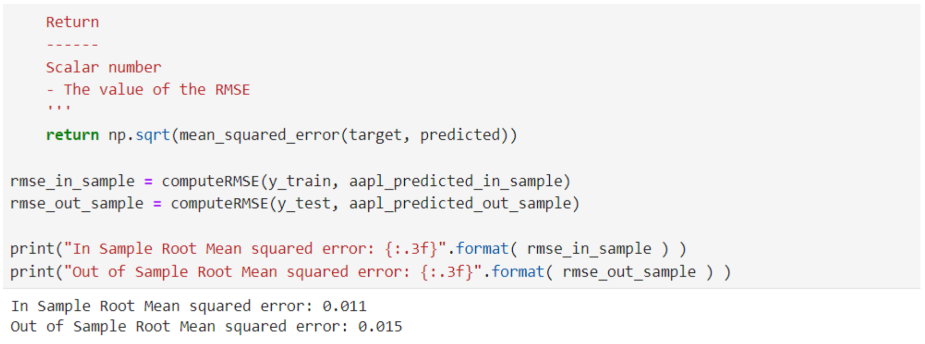
```
    k: Scalar number
    - k-fold cross validation

    Returns
    --------
    The average, across the k iterations, of the score
    '''
    scores = cross_val_score(model, X, y, cv=k)
    return np.mean(scores)


cross_val_avg = compute_cross_val_avg(regr, X_train, y_train, 5)
print("{t:s}: Avg cross val score = {sc:3.2f}".format(t=ticker, sc=cross_val_avg) )
```

```
AAPL: Avg cross val score = 0.32
```

Figure 6: Average cross validation score.

According to the result, we can see that the average cross validation score is 0.32. From my perspective,in the field of finance, or more specific, in the field of stock return prediction, the average cross validation score means R square in linear regression mathematically, and the number 0.32 is a relative good outcome, since the market is volatile, uncertain, and random.

Also, we calculate the root of mean square error, It reflects the average error between the predicted values and the actual observed values of the model. RMSE can be used to evaluate the accuracy of a stock price prediction model.



```
    Return
    ------
    Scalar number
    - The value of the RMSE
    '''
    return np.sqrt(mean_squared_error(target, predicted))

rmse_in_sample = computeRMSE(y_train, aapl_predicted_in_sample)
rmse_out_sample = computeRMSE(y_test, aapl_predicted_out_sample)

print("In Sample Root Mean squared error: {:.3f}".format( rmse_in_sample ) )
print("Out of Sample Root Mean squared error: {:.3f}".format( rmse_out_sample ) )
```

```
In Sample Root Mean squared error: 0.011
Out of Sample Root Mean squared error: 0.015
```

Figure 7: RMSE in and out of sample.

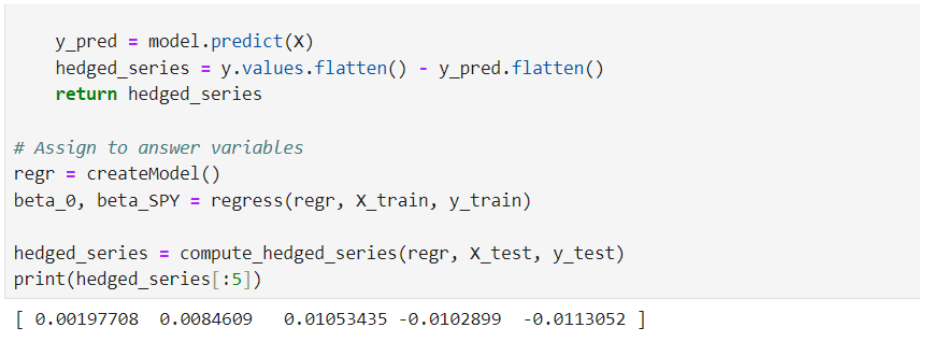
We notice that the rmse is 0.011 in sample and 0.015 out of sample, which are rather small indicating that the selection of sample data can be considered reasonable, as the RMSE of the in-sample and out-of-sample data is very close.

This implies that this linear regression model is relatively appropriate and can effectively reflect the relationship between the two returns.

Furthermore, hedged returns reflect the difference between the predicted values and the actual observed valueswhich can be obtained from the formula :

$$r'_{AAPL}{}^{(t)} = r^{(t)}_{AAPL} - \beta_{APPL,SPY} * r^{(t)}_{SPY}$$

Also, with the help of Python, we get:



```
    y_pred = model.predict(X)
    hedged_series = y.values.flatten() - y_pred.flatten()
    return hedged_series

# Assign to answer variables
regr = createModel()
beta_0, beta_SPY = regress(regr, X_train, y_train)

hedged_series = compute_hedged_series(regr, X_test, y_test)
print(hedged_series[:5])
```

```
[ 0.00197708  0.0084609   0.01053435 -0.0102899  -0.0113052 ]
```

Figure 8: Hedged returns.

We get the first 5 hedged returns based on the predicted series, we can find that the differences between observed value and predicted value are quite small and we can come to the conclusion that the model has high model prediction accuracy, and the prediction error is within acceptable range.

4 Conclusion

Machine learning in finance involves the application of advanced computational techniques to analyze and make predictions based on financial data. This includes using algorithms and statistical models to identify patterns, trends, and relationships within financial markets, asset prices, risk assessment, and investment strategies.

Through this project, I learnt that the linear regression model is utilized for stock price prediction. By leveraging large datasets and complex algorithms, the model enables me to gain insights, automate decision-making processes, and improve the accuracy of the predictions.

However, it is important to note that the use of machine learning in finance also presents challenges related to data quality, model interpretability, and regulatory compliance. It is often the case that we need to double check the results of both Python outcomes and mathematical results when using linear regression model in this project.

Additionally, it is also suggested to use more mature and comprehensive mathematical models for prediction such as Neural Network, as the results would be more reasonable and accurate.