**Millenium Data Project Report**

Yanfei Cui

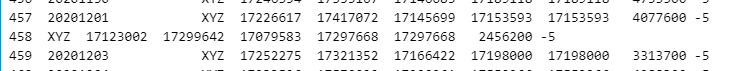
Note: All my codes and results are available in code.ipynb.

**Part1 Data handling**

1. Question 1: What are exceptions?

After my observation, I found a total of 10 different kinds of exceptions as follows:

1. Date is empty, such as



1. Ticker is not XYZ, such as



1. Open, High, Low, Close, Volume are less than or equal to 0, such as



1. Open, High, Low, Close, Volume contain garbled characters or spaces, such as



1. Adj\_close contains garbled characters or spaces, such as



1. Adj\_close does not match Close, such as



1. Magnitude does not equal -5, such as



8. Due to copying errors in data duplication cases where an extra column appears, such as



9. Duplicate dates occur and by comparing them we can observe that almost all duplicate date data are identical, such as



10. The data format changed from YYYYMMDD to MMDDYYYY for YYYYDDMM，such as



1. Question 2: How to handle exceptions?

1. Date is empty, which can be inferred based on the nearest data rows.

2. Ticker is not XYZ, and it can be observed that this row has significant differences compared to other rows, so the approach is to delete this row.

3. Open, High, Low, Close, Volume are less than or equal to 0; the approach is to set them as -999.

4. Open, High, Low, Close, Volume contain garbled characters or spaces; the approach is to set them as -999.

5. Adj\_close contains garbled characters or spaces; the approach is to set it as -999.

6. Adj\_close does not match Close; the approach is to set it as the value of Close.

7. Magnitude does not equal -5; the approach is to set it as -5.

8. Due to copying errors in data duplication cases where an extra column appears; the approach is to find and remove duplicated numbers.

9. Duplicate dates occur and by comparing them we can observe that almost all duplicate date data are identical; therefore they can be merged together.

10. The data format changed from YYYYMMDD to MMDDYYYY or YYYYDDMM; correcting it back will suffice.

1. Question 3: List out what kinds of rules you put into the data handling and why?

First, I will read the file line by line. For each line in the file, after removing leading and trailing spaces, I will read each field one by one according to the specified positions in the table. If there is an exception, I will handle the abnormal field and merge it into the correct position to ensure that there are no errors when reading the next field based on its position.

Next, for the Date field, I will first check if it contains "XYZ". If it does, then it means that this field is empty. Then I will use a try-except statement to read it. If an exception occurs, it means that there is an error in the time format and it needs to be corrected.

Then, for the Ticker field, I check if it is XYZ. If not, then this entire row should be discarded.

Afterwards, for the following fields: Open High Low Close AdjClose and Volume; I will individually check for exceptions. First of all, I'll check if their length is zero. If so,assign -999 as their value.Then try converting them to int.If an exception occurs,it means there are special characters,and they should be changed to -999.If no exceptions occur,I'll further check if these values are less than or equal to 0.If so,assign -999.Furthermore,I'll also compare this value with its adjacent previous field's value.If they are equal and total length of this row exceeds 87,it indicates extra copying error.In such case,the value should be deleted and re-read.As for Adj\_close,I also need to compare whether it's equal to Close.If not,equalize them.

Then for Magnitude field,I'll check if its value equals -5.If not,set it as -5.

Finally,before writing this corrected row into a file,I need to check whether its date has already been written before.If true,I'll compare with previously written rows.Merge the normal values and exclude -999.

**Part2 Model training & testing**

1. Question 4: Compare the 3 results and comment.

According to the question, we need to predict the future price of stocks. First, we need to further clarify the problem.

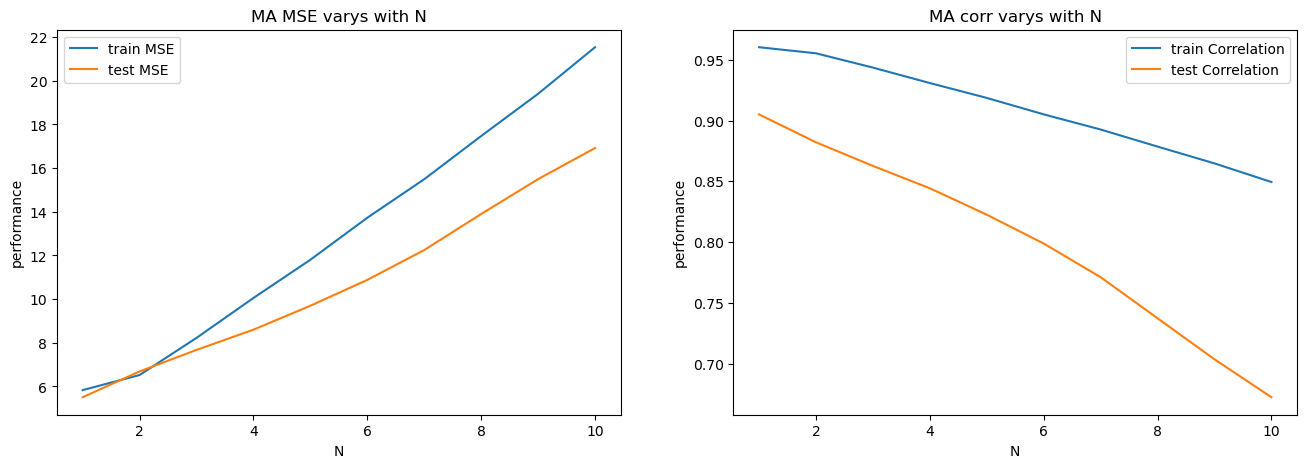
What is the target of prediction? There are four prices each day: open, close, high, and low. Here, let's choose the close as the target for prediction.

What data is available? If we want to predict the price on day i, a reasonable method is that we can use all data from day i-1 and before. We can also use existing data to create more features for prediction.

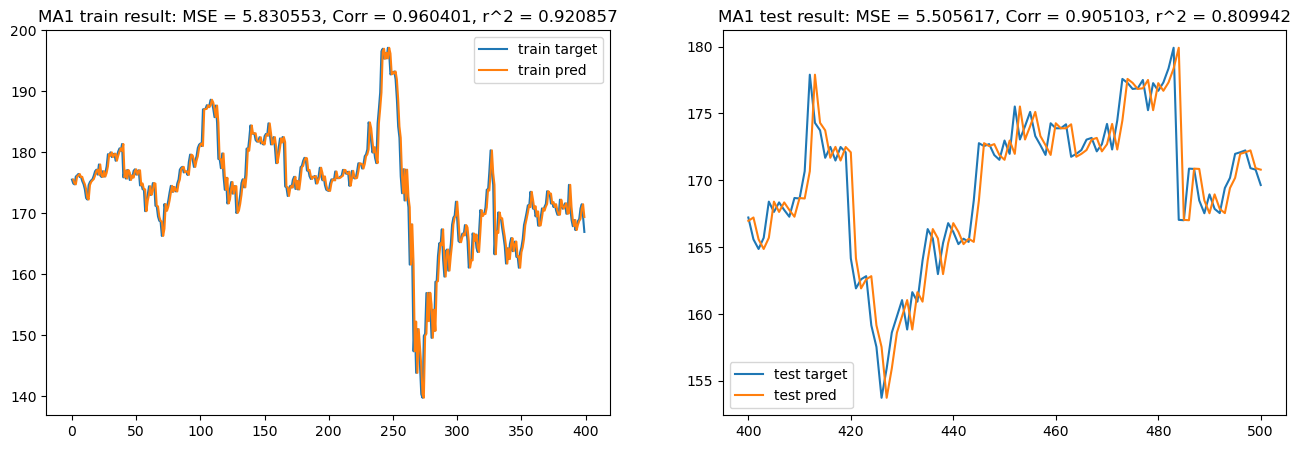
Which indicators are used to evaluate prediction performance? Since this is a regression problem, we can use mean squared error (MSE), Pearson correlation coefficient, and score as evaluation indicators for prediction performance.

* 1. Moving Average

The result of Moving average is as follows



We can see that the smaller N is, the better the effect of moving average prediction. Therefore, the best result should be a one-day moving average, and its predicted results are as follows:

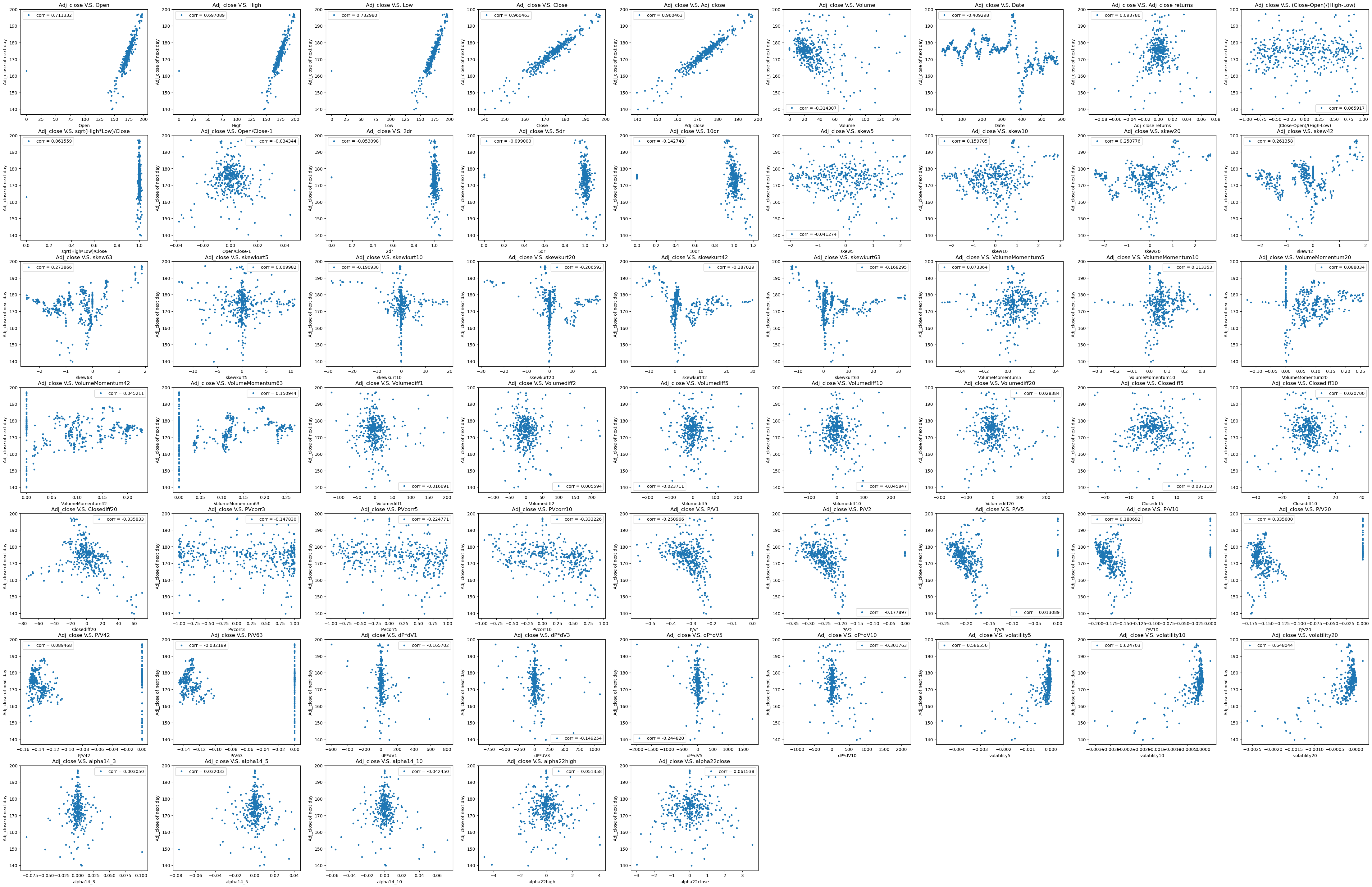


It can be seen that the fitting effect of the one-day moving average is very good, with a very small MSE and high correlation coefficient and . However, there is a serious problem: it essentially uses yesterday's price as today's price, which results in significant lag. This may not be a big issue in one-sided market conditions, but in volatile markets, it will lead to substantial losses. Therefore, this is not a good forecasting method.

* 1. Linear Regression

In order to achieve good performance of the linear regression model, I tried many methods. First, to solve the problem of strong collinearity in the original data, I manually created a large number of low correlated features. Secondly, some of these features that I created are useful while others are not. Having too many useless features will drag down the performance of the model. Therefore, I used many methods including PCA for data dimensionality reduction, but none of them yielded satisfactory results. In the end, by individually examining the correlation between each feature and the target variable, I manually selected out highly correlated features and achieved decent results. Then I set a parameter N representing the number of days for model retrospective analysis and also set a decay period weight. However, through experiments, I found that 1 is the best number for retrospective analysis days as other values easily lead to overfitting. Below are my model results:

* + 1. Feature’s correlation with price target



* + 1. Selected Features

"Open", "Volume", "Adj\_close", "2dr", "5dr", "10dr", "P/V1", "P/V2", "P/V5", "P/V20", "PVcorr3", "PVcorr5", "sqrt(High\*Low)/Close", "skew5", "skew10", "skew20", "skew42", "skew63", "skewkurt10", "skewkurt63", "Closediff5", "Closediff10", "Volumediff20", "VolumeMomentum5"

* + 1. Results



It can be seen that the model's prediction performance is slightly better than the one-day moving average, especially in terms of corr and on the test set, which are 0.910581 and 0.813358 respectively, higher than the moving average’s 0.905103 and 0.809942. And the test MSE is 5.105692, higher than the moving average’s 5.505617. This indicates that my feature engineering is effective.

* 1. My model ()

The selection of the model is a major issue. On one hand, the data size is too small (only 500 days of data and only one stock), which makes it impossible to use deep learning models. On the other hand, through my experiments, I found that as long as appropriate feature engineering is done, linear models such as Ridge and Lasso generally perform worse than linear regression. And furthermore, the performance of tree models in this task is also poor. The predictive ability of a single decision tree is insufficient, and tree models like random forests have too much randomness. Moreover, gradient boosting decision trees like XGBoost have poor parameter stability. Even with a difference of only 1 in n\_estimator, there can be significant differences in performance between the training set and the test set. The fundamental reason for this is that the data volume is too small to support overly complex models.

Therefore, I have decided to combine multiple linear regression models together to create a two-stage model. In the first stage, three linear regression models will be trained separately to predict 1-day returns, 2-day returns, and 5-day returns respectively. In the second stage, these three model predictions along with Adj\_close from the original data will be concatenated together and used to train another linear regression model for predicting 1-day returns (which is equivalent to training a model that combines all three types of returns). Then by multiplying the last predicted 1-day return with each day's stock price, we can obtain the predicted value for stock prices.

* + - 1. Results



It can be seen that the prediction performance of XGBoost is not very good. On one hand, overfitting occurs in the training set, and on the other hand, there is a significant error in the test set. This is understandable because price sequences are inherently highly autocorrelated.

* 1. Predict the returns, then calculate the prices.