DATA1030 Final Report:
Apple Stock Price Prediction

Li-Heng Pan

Data Science Initiative, Brown University

GitHub Repository Link

1 Introduction

The dataset of my choice is the Apple stock price dataset. It contains the stock prices and volumes of Apple from December 15, 1980 to September 19, 2022. The goal of this project is to predict future stock prices of Apple with its historical data.

For more than a century, stock has been a popular investment, and numerous scholars and analysts have conducted research on it. With the advancement of machine learning techniques, the prediction of stock prices can be more accurate nowadays. Hopefully, in the future, people can refer to my model when building their own investment portfolios.

The target variable is the **closing price** of the stock, i.e., the stock price recorded when the market closes. Since the closing prices are continuous, this is a regression problem. There are 10530 data points and 6 features in the original dataset.

The features include the opening price, the high price, the low price, the closing price, the adjusted closing price, and the volume. The opening price is the price recorded when the market opens. The high price refers to the highest price recorded for the day. The low price refers to the lowest price recorded for the day. The adjusted closing price is the modified closing price based on corporate actions. The volume is the number of stocks sold within the day. The price features are in the United States dollar (USD), and the volumes are in shares. All features are continuous.

The dataset is from Kaggle. Aman Chauhan scraped all the data from Yahoo Finance, compiled it into a dataset, and uploaded it to Kaggle. There are some projects conducted on this dataset. A person used the Seasonal-ARIMA technique and reached an R^2 of 0.704 on the test set (predicting from October 31, 2016 to December 31, 2019) [1]. Rahul Kumar did the same project with an LSTM model. He found that the LSTM model would predict lower values than the actual stock price for the most recent date stamps (the most recent 70 days) [2].

2 Exploratory Data Analysis

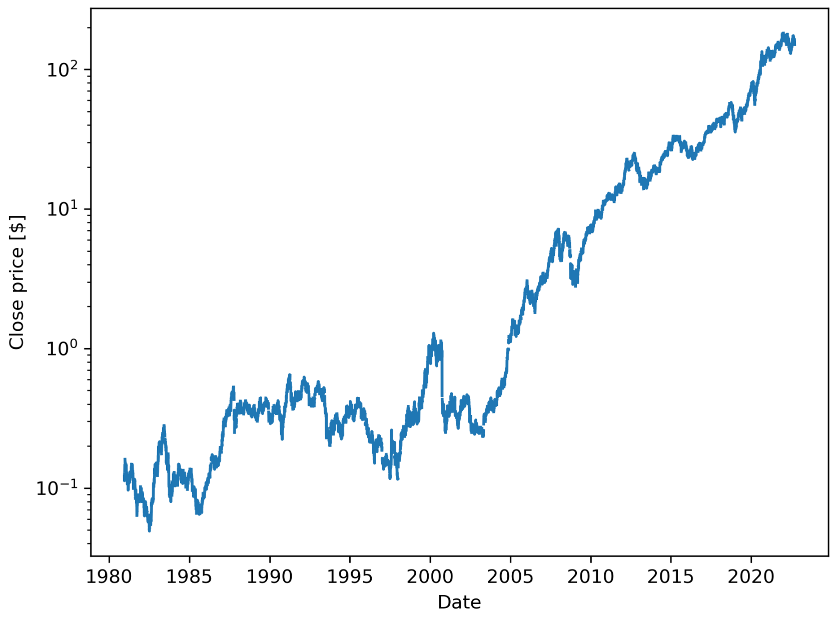

Figure 1: The closing price trend since 1980

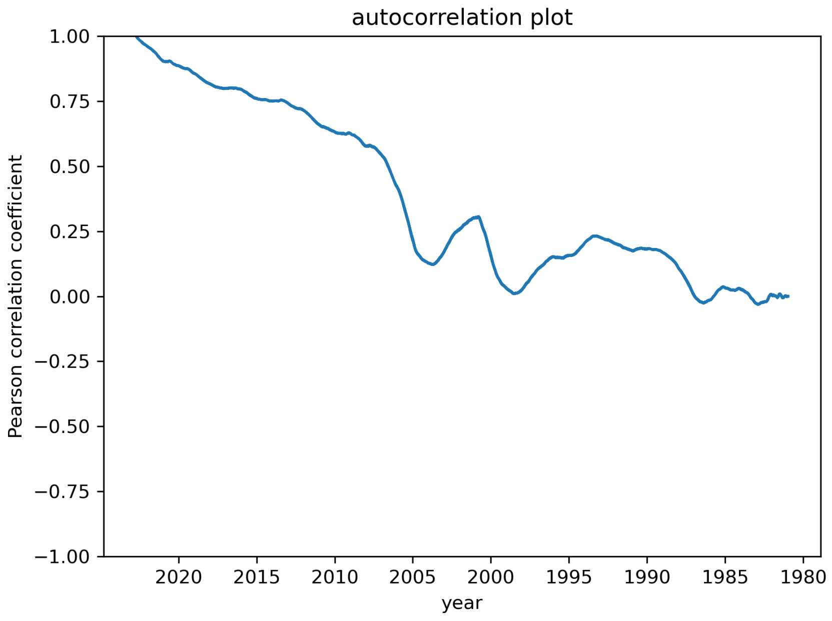

Figure 2: The autocorrelation plot of the closing price (lag days replaced by year)

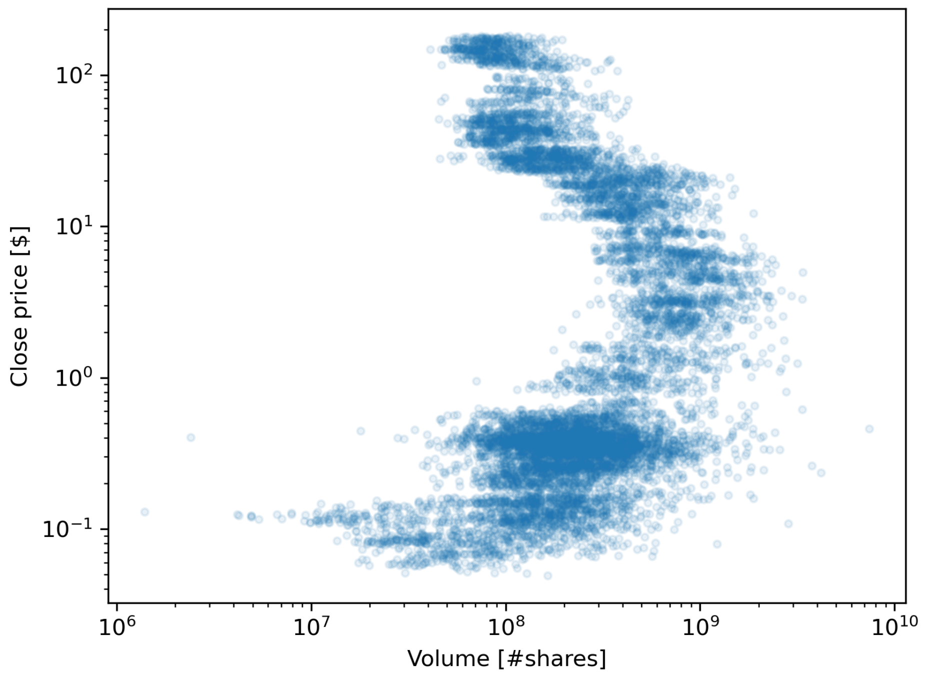

Figure 3: The scatter plot of volume and closing price

In Figure 1, Apple's stock price has an overall increasing trend over the past four decades. The curve between the late 1990s and the early 2000s perfectly depicts the entire process of the tech bubble. Ever since the end of the bubble, Apple's stock price has skyrocketed.

In Figure 2, the Pearson correlation coefficient of the closing price decreases as lag days increase. That is, prices from the past have less to do with the prices in recent years. Another point worth mentioning is that there is a plunge in the correlation coefficient between 2000 and 2005 because of the tech bubble.

In Figure 3, we can see that the volume tends to be lower when the closing price is either high or low. The volume tends to be higher when the price is within the range of 1 and 100.

3 Methods

In the original dataset, stock prices are only recorded when the stock market is open. Therefore, there are no stock prices on weekends and holidays, making the time intervals of the data points different. Before splitting and preprocessing the data, the missing dates in the dataset are filled in to make the time interval between every pair of consecutive rows equal. In this project, I experimented with two different approaches–an uninterpolated dataset and a linearly-interpolated dataset. For the uninterpolated dataset, the missing values on weekends and holidays are kept; for the linearly-interpolated dataset, the missing values are interpolated.

Next, a feature matrix is made with autoregression. For each feature, 10 extra columns are created, i.e., the prices or the volumes from the past 10 days. After the feature matrix is compiled, there are 60 columns in it. The number of lag days of 10 is merely an example here. To be rigorous, different values of lag days should be experimented, and the value that gives the best model performance should be chosen.

Since this is a non-IID, time-series dataset, the data is kept in chronological order and split without shuffling to avoid information leakage. The first 60 percent of the data (data from 1980 to 2004) is split as the training set, the following 20 percent of the data (data from 2004 to 2014) works as the validation set, and the remaining 20 percent of the data (data from 2014 to 2022) constitutes the test set.

In terms of preprocessors, StandardScaler is applied on all of the columns since prices and volumes are continuous features that do not have clear boundaries. For the uninterpolated dataset, the entries where the target variable is missing are omitted, and there are 10530 rows and 60 columns in the feature matrix after preprocessing. For the linearly-interpolated dataset, there are 15244 rows and 60 columns in the feature matrix.

For the uninterpolated dataset, I trained an XGBoost model since it is one of the models that allow missing values. The hyperparameter values are listed in Table 1. The evaluation metric is root-mean-square error (RMSE).

Hyperparamters	Values
learning rate	0.05
n_estimators	100
missing	np.nan
max_depth	[1,3,10,30,60]
colsample_bytree	0.9
subsample	0.66

Table 1: Hyperparamter values for XGBoost

For the linearly-interpolated dataset, I trained linear regression models with L1 regularization, support vector machine models, random forest models, and XGBoost models. For random forest models, each model is trained 5 times with 5 different random states to measure uncertainties. The evaluation metric is also root-mean-square error (RMSE). The hyperparameter values are listed from Table 1 to Table 4 (XGBoost hyperparameters in linearly-interpolated and uninterpolated scenarios are the same).

Hyperparamters	Values
alpha	np.logspace(-7,0,19)
max_iter	10^8

Table 2: Hyperparamter values for Linear Regression (L1 Regularization)

Hyperparamters	Values
gamma	$10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}$
C	$10^3, 10^4, 10^5, 10^6, 10^7$

Table 3: Hyperparamter values for Support Vector Machine

Hyperparamters	Values
max_depth	$1, 3, 10, 30, 60$
max_features	$0.25, 0.5, 0.75, 1$

Table 4: Hyperparamter values for Random Forest

4 Results

For the linearly-interpolated dataset, the linear regression model with L1 regularization and the support vector machine model outperform the baseline score. The linear regression model performs the best with a test RMSE of 1.26. The tree-based models perform poorly on the dataset due to the property of the dataset and the splitting approach. They always give a constant prediction when encountering an outlier. To be more specific, the tree models are trained with data from 1980 to 2004, whereas all of the data points in the test set (from 2014 to 2022) are well beyond the scope of the training set. This is the reason why both models give constant predictions in the test set. In the uninterpolated scenario, the XGBoost model also suffers from the problem mentioned above and fails to reach the baseline score. The test scores are listed in Table 5, and the predictions of the test set are in Figure 4.

Algorithm	RMSE
Linear Regression (L1)	1.26
Support Vector Machine	43.71
Random Forest	79.98
XGBoost	80.01
Baseline	47.75

Table 5: Test scores of the models

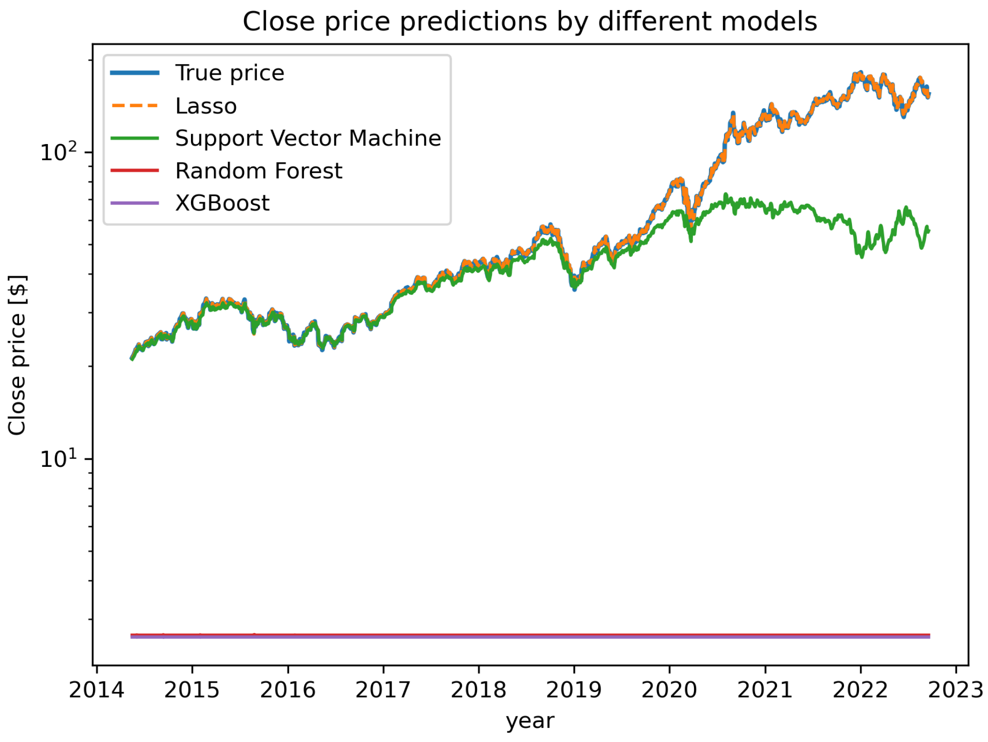

Figure 4: Model predictions of the test set

For global feature importance, I used the coefficients of the linear regression model, the total cover metric of the XGBoost Model, and the summary plot of SHAP package as metrics. For the linear regression model, only the coefficients of the features from the previous day are non-zero. The summary plot for the random forest model in Figure 5 shows that the features from the previous day are the most important. In fact, all the metrics show exactly the same thing: **only the price features from the previous 1 or 2 days play important roles in predicting the closing price on a certain date**.

For local feature importance, SHAP values are calculated for certain points in the test set. The SHAP values are shown in Figure 6. The inference remains the same: the price features from the previous day are the most important.

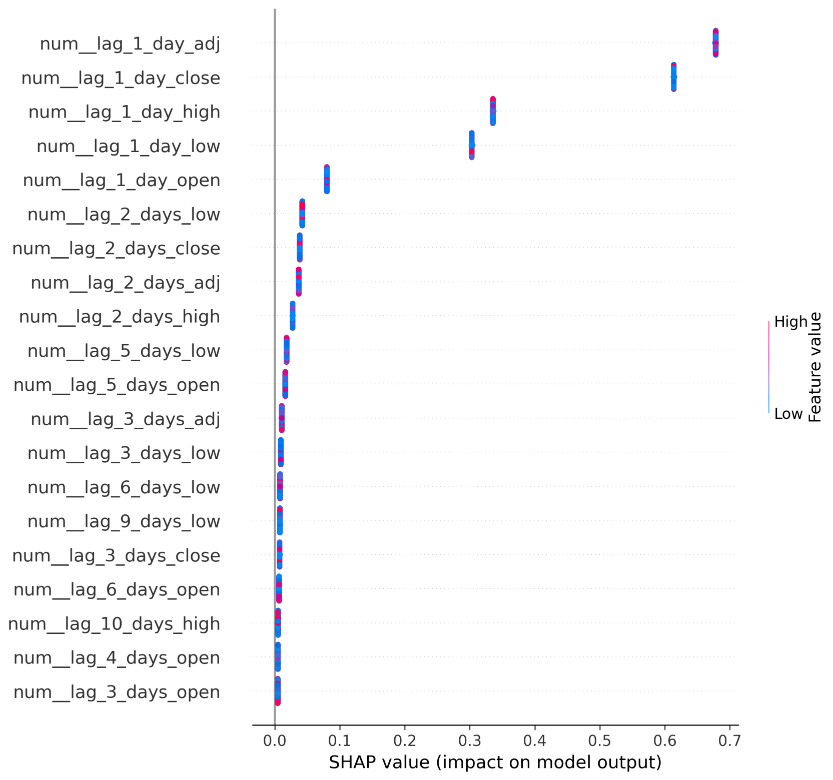

Figure 5: Summary plot by SHAP

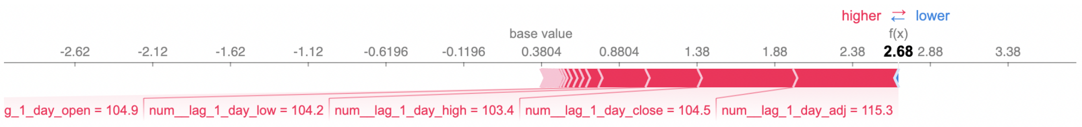

Figure 6: SHAP values for local feature importance

5 Outlook

As mentioned in **4 Results**, only the features from the previous 1 or 2 days are important. Thus, the number of lag days can be decreased to reduce noises in the data. The linear model is already performing brilliantly in the problem. However, it only means that stock prices do not change drastically overnight, i.e., the stock prices for two consecutive days are close. To make the project more practical, we should try to predict the stock price trend in the following **months or years**. At that point, more sophisticated and powerful techniques, e.g., ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average), and deep learning models, should be considered.

References

[1] nikhils10 (2020) *Time-Series-Forecasting-Apple-Stock-Price-Using-SARIMA-Prophet.* https://github.com/nikhils10/Time-Series-Forecasting-Apple-Stock-Price-Using-SARIMA-Prophet

[2] Rahul Kumar (2021) *Predicting-Apple-Stock-Price-Using-An-LSTM* https://github.com/rahkum96/Predicting-Apple-Stock-Price-Using-An-LSTM

[3] Aman Chauhan (2022) *Apple Inc. Stock Market Analysis since Founding Years* https://www.kaggle.com/datasets/whenamancodes/alphabet-inc-google-founding-years-analysis