Final Project

Predicting Stock Prices Of Amazon (AMZN) and Kroger (KR) Using Machine Learning Predicting Models

Phuong (Lucy) Doan

MRKTNG 8370 – Supply Chain Modeling & Analysis

Instructor: Dr. Suchithra Rajendran

May, 2020

Table of Contents

I. Problem Description

1.1. Business Objective

Stock pricing is an amusing topic and draws a huge interest from everyone. It provides analysts with valuable input for evaluating corporate and market performance, and gives investors a sense of what stocks to buy or sell.

The first quarter of 2020 observes multiple stock market crash that reflects the downturn of the US economy due to Coronavirus. However, some stocks stood out from the crowd when not following the downward trend. This project picks two out of such stocks, AMZN (Amazon) and KG (Kroger), that represent retail/ ecommerce industry which are also impacted from the Covid-19 pandemic. Each company has different operating and supply chain characteristics, Amazon thriving by selling on digital platforms while Kroger possessing the largest retailing network in the United States. These unique points combined with recent stock performance prove that their stock deserves public attention. Though both companies "survived" in early phase of the pandemic, no one can tell if things will remain the same in near future especially in case the outbreak lasts long. Therefore, forecasting price of the two stocks is an effort to disclose the uncertainty and understand the factors that affect each stock's performance.

1.2. Data Mining Objective

When it comes to forecasting future stock prices, there are two main streams, fundamental analysis and technical analysis. On the one hand, fundamental stream assesses the fundamental factors that influence a company's value and business, including macro perspectives such as overall economy and industry conditions, and such micro perspectives as company's financial health, management, and competition. On the other hand, technical stream analyzes statistical aspects of past trading activities, including price movement, volume and market data, to predict future price.

This project will apply the technical approach for the most part by using machine learning regression techniques that use each stock's indicators, such as open, high, low, close prices, as predictive variables. Especially, such forecasting methods as moving average, weighted moving average, and exponential smoothing, are applied to create independent variables accordingly. Besides, taking a reference on fundamental stream, the predicting models also employ stock prices of 30 companies within Dow Jones Industrial Average (DJIA) to reflect competition landscape and macro economy. After adopting two predicting algorithms, linear regression and regression forest, performance metrics are calculated to evaluate and select the optimal predicting model for each stock.

Data is collected from Yahoo Finance, on a daily basis (daily price) from 03-01-2000 to 08-05-2020. This range is large enough to cover such big events of the country as 11 September attack in 2001 and financial crisis 2007-2008. The Covid-19 crisis is compared to those critical times and the model can learn from stock performance in these period to predict prices during this coronavirus time.

II. Analytical Questions

2.1. Variable and Statistical Modeling identification

- **Statistical modeling identification**

As mentioned earlier, the report will build two machine learning models, linear regression and regression tree. After obtaining predicting values from each model, we will calculate three metrics, MSE (mean squared error), MAE (mean absolute error), and R^2 to evaluate and pick up the optimum predicting model for each stock.

- **Variable identification**

To predict close price (dependent var), we use 6 original variables, same-day open price and past-1-day performance of open price, high price, low price, close prices and volume.

Besides single-point value, investors usually look at historical price and volume over a period of time, such as past week, past month or past year. Therefore, we can create new variables by calculating average prices and volumes over past 5 days, past 21 days and past 252 days. On a side note, the time window is not 7 days, 30 days and 365 days because in fact, stock price is not collected 7/7 days per week. After that, we can also compute the ratios between each pair of these average values in three different time frames.

In addition to historical average values, investors also consider stock volatility which can be reflected through standard deviation of price and volume over three time windows, 7 days, 30 days and 365 days. In a similar manner, we can include the ratios between each pair of standard deviation values in different time frames.

Another significant financial indicator that investors keep an eye on is return, which is the gain or loss of close price over a particular period. This report will use daily return, weekly return, monthly return, and annual return. For instance, daily return is calculated as below:

$$\text{Return}_{i:i-1} = (\text{price}_i - \text{price}_{i-1}) / \text{price}_{i-1}$$

Based on daily return, we can produce moving average over a particular period of time, one week, one month, and one year. Besides moving average, we will utilize other methods, weighted moving average and exponential smoothing average over last 5 days. More specifically, our weighted moving average will use the weight of (1,2,3,4,5); closer a past value to same day value, the higher weight it will have. Concerning exponential smoothing average, with span of 5 days, alpha value equals 0.33 (=2/ (span + 1)), which is within popular range (0.1 – 0.4).

The last predictive variable is open price of Dow Jones Industrial Average (DJIA)

In total, there are 41 features/ variables described as following:

Variable Name	Description
open	The open price on the same day
open_1	The open price on the past day
close_1	The close price on the past day
high_1	The high price on the past day
low_1	The low price on the past day
volume_1	The volume on the past day
avg_price_5	The average close price over the past week
avg_price_30	The average close price over the past month
avg_price_365	The average close price over the past year
ratio_avg_price_5_30	The ratio between average price over past week and that over past month
ratio_avg_price_5_365	The ratio between average price over past week and that over past year
ratio_avg_price_30_365	The ratio between average price over past month and that over past year
avg_volume_5	The average volume over the past week
avg_volume_30	The average volume over the past month
avg_volume_365	The average volume over the past year
ratio_avg_volume_5_30	The ratio between average volume over past week and that over past month
ratio_avg_volume_5_365	The ratio between average volume over past week and that over past year
ratio_avg_volume_30_365	The ratio between average volume over past month and that over past year
std_price_5	The standard deviation of close prices over the past week
std_price_30	The standard deviation of close prices over the past month
std_price_365	The standard deviation of close prices over the past year
ratio_std_price_5_30	The ratio between standard deviation of close prices over past week and that over past month
ratio_std_price_5_365	The ratio between standard deviation of close prices over past week and that over past year
ratio_std_price_30_365	The ratio between standard deviation of close prices over past month and that over past year
std_volume_5	The standard deviation of volumes over the past week
std_volume_30	The standard deviation of volumes over the past month
std_volume_365	The standard deviation of volumes over the past year
ratio_std_volume_5_30	The ratio between standard deviation of volumes over past week and that over past month
ratio_std_volume_5_365	The ratio between standard deviation of volumes over past week and that over past year
ratio_std_volume_30_365	The ratio between standard deviation of volumes over past month and that over past year

return_1	Daily return of the past day
return_5	Daily return of the past week
return_30	Daily return of the past month
return_365	Daily return of the past year
moving_avg_5	Moving average of daily returns over the past week
moving_avg_30	Moving average of daily returns over the past month
moving_avg_365	Moving average of daily returns over the past year
weighted_moving_avg_5	Weighted moving average of daily returns over the past week, with the weights of (1,2,3,4,5)
exp_smoothing_5	Exponential smoothing average of daily returns over the past week, with a 5-day span and alpha = 0.33
DJIA_open	The open price of Dow Jones Industrial Average on the same day

2.2. Data Preparation

- **Feature generation**

  We create sub-functions then combine them all into a main feature generation function.

  - Sub-function for six original features

```
def add_original_feature (df, df_new):
   df_new["open"] = df["Open"]
   df_new["open_1"] = df["Open"].shift(1)
   df_new["close_1"] = df["Close"].shift(1)
   df_new["high_1"] = df["High"].shift(1)
   df_new["low_1"] = df["Low"].shift(1)
   df_new["volume_1"] = df["Volume"].shift(1)
```

  - Sub-function that generates 6 features related to average close prices

```
def add_avg_price (df, df_new):
   df_new["avg_price_5"] = df["Close"].rolling(5).mean().shift(1)
   df_new["avg_price_30"] = df["Close"].rolling(21).mean().shift(1)
   df_new["avg_price_365"] = df["Close"].rolling(252).mean().shift(1)
   df_new["ratio_avg_price_5_30"] = df_new["avg_price_5"] / df_new["avg_price_30"]
   df_new["ratio_avg_price_5_365"] = df_new["avg_price_5"] /
df_new["avg_price_365"]
   df_new["ratio_avg_price_30_365"] = df_new["avg_price_30"] /
df_new["avg_price_365"]
```

- Sub-function that generates 6 features related to average volumes

```
def add_avg_volume (df, df_new):
   df_new["avg_volume_5"] = df["Volume"].rolling(5).mean().shift(1)
   df_new["avg_volume_30"] = df["Volume"].rolling(21).mean().shift(1)
   df_new["avg_volume_365"] = df["Volume"].rolling(252).mean().shift(1)
   df_new["ratio_avg_volume_5_30"] = df_new["avg_volume_5"] /
df_new["avg_volume_30"]
   df_new["ratio_avg_volume_5_365"] = df_new["avg_volume_5"] /
df_new["avg_volume_365"]
   df_new["ratio_avg_volume_30_365"] = df_new["avg_volume_30"] /
df_new["avg_volume_365"]
```

- Sub-function that generates 6 features related to standard deviation of close prices

```
def add_std_price (df, df_new):
   df_new["std_price_5"] = df["Close"].rolling(5).std().shift(1)
   df_new["std_price_30"] = df["Close"].rolling(21).std().shift(1)
   df_new["std_price_365"] = df["Close"].rolling(252).std().shift(1)
   df_new["ratio_stdg_price_5_30"] = df_new["std_price_5"] / df_new["std_price_30"]
   df_new["ratio_std_price_5_365"] = df_new["std_price_5"] / df_new["std_price_365"]
   df_new["ratio_std_price_30_365"] = df_new["std_price_30"] /
df_new["std_price_365"]
```

- Sub-function that generates 6 features related to standard deviation of volumes

```
def add_std_volume (df, df_new):
   df_new["std_volume_5"] = df["Volume"].rolling(5).std().shift(1)
   df_new["std_volume_30"] = df["Volume"].rolling(21).std().shift(1)
   df_new["std_volume_365"] = df["Volume"].rolling(252).std().shift(1)
   df_new["ratio_stdg_volume_5_30"] = df_new["std_volume_5"] /
df_new["std_volume_30"]
   df_new["ratio_std_volume_5_365"] = df_new["std_volume_5"] /
df_new["std_volume_365"]
   df_new["ratio_std_volume_30_365"] = df_new["std_volume_30"] /
df_new["std_volume_365"]
```

- Sub-function that generates 9 return-based features

```
weights = np.arange(1,6)
weights
```

```
def add_return_feature (df, df_new):
   df_new["return_1"] = ((df["Close"] - df["Close"].shift(1))/df["Close"].shift(1)).shift(1)
   df_new["return_5"] = ((df["Close"] - df["Close"].shift(5))/df["Close"].shift(5)).shift(1)
   df_new["return_30"] = ((df["Close"] -
df["Close"].shift(21))/df["Close"].shift(21)).shift(1)
   df_new["return_365"] = ((df["Close"] -
df["Close"].shift(252))/df["Close"].shift(252)).shift(1)
   df_new["moving_avg_5"] = df_new["return_1"].rolling(5).mean().shift(1)
   df_new["moving_avg_30"] = df_new["return_1"].rolling(21).mean().shift(1)
   df_new["moving_avg_365"] = df_new["return_1"].rolling(255).mean().shift(1)
   df_new["weighted_moving_avg_5"] = df_new["return_1"].rolling(5).apply(lambda
wma_return: np.dot(wma_return, weights)/weights.sum(), raw=True)
   df_new["exp_smoothing_5"] =
df_new["moving_avg_5"].ewm(span=5).mean().shift(1)
```

- The main feature generating function that combines all sub-functions

```
def generate_feature(df):
   df_new=pd.DataFrame()
   add_original_feature(df, df_new)
   add_avg_price(df, df_new)
   add_avg_volume(df, df_new)
   add_std_price(df, df_new)
   add_std_volume(df, df_new)
   add_return_feature(df, df_new)

   df_new = df_new.dropna(axis=0)

   df_new["close"] = df["Close"]
   df_new["DJIA_open"] = df["DJIA_Open"]
   return df_new
```

- **Data splitting and pre-processing**

  - <u>Splitting data</u>

After deleting the rows with missing values, the data now begins on 09-01-2001. We split it into training set (09-01-2001 – 31-12-2018) and testing set (02-01-2019 – 08-05-2020)

```
#Splitting data and counting training samples

start_train = '2001-01-09'
end_train = '2018-12-31'
start_test = '2019-01-02'
end_test = '2020-05-08'
data_train = data.loc[start_train:end_train]
X_train = data_train.drop('close', axis=1).values
y_train = data_train['close'].values

print(X_train.shape, y_train.shape)
```

```
(4522, 40) (4522,)
```

```
#Counting testing samples

data_test = data.loc[start_test:end_test]
X_test = data_test.drop('close', axis=1).values
y_test = data_test['close'].values

print(X_test.shape)
```

```
(341, 40)
```

We have 4522 training samples and 341 testing samples.

  - <u>Data pre-processing</u>

Except for regression forest, the three remaining models are sensitive with data with features at largely different scales. Therefore, we need to normalize features into comparable scales.

```
#Rescaling both training and testing datasets

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled_train = scaler.fit_transform(X_train)
X_scaled_test = scaler.transform(X_test)
```

2.3.Learning from Datasets and Making Predictions

```
#Install predicting algorithms and performance-evaluating metrics

from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

- **SGD-based Linear Regressor**

  First, we will build the predicting models for Amazon's stock (AMZN). Then we apply the same feature generating process and predicting algorithms for Kroger's stock within same time period. Refer to Appendix 1 for optimal parameters and performance measurements across 4 models for KR stock.

  The following are results from AMZN stock data.

```
#Find out optimal set of parameters for SGD-based Linear Regression model

param_grid = {
    "alpha": [1e-5, 3e-5, 1e-4],
    "eta0": [0.01, 0.03, 0.1],
}

lr = SGDRegressor(loss='squared_loss', learning_rate='constant', penalty='l2',
max_iter=1000)
grid_search = GridSearchCV(lr, param_grid, cv=5, scoring='r2')
grid_search.fit(X_scaled_train, y_train)

print(grid_search.best_params_)
```

```
{'alpha': 1e-05, 'eta0': 0.01}
```

```
#Use the optimal model to make predictions of the testing samples

lr_best = grid_search.best_estimator_
predictions_lr = lr_best.predict(X_scaled_test)
```

```
#Measure predicting performance of Linear Regression model

MSE_lr = mean_squared_error(y_test, predictions_lr).round(decimals=3)
MAE_lr = mean_absolute_error(y_test, predictions_lr).round(decimals=3)
Rsquared_lr = r2_score(y_test, predictions_lr).round(decimals=3)
print("For Linear Regression model,",
     "Mean squared error:", MSE_lr, ", Mean absolute error:", MAE_lr, ", R_squared:",
Rsquared_lr)
```

```
For Linear Regression model, Mean squared error: 1272.451 , Mean absolute error: 24.983 , R_squared: 0.956
```

- **Regression Forest**

```
#Find out optimal set of parameters for Regression Forest model

param_grid = {
    'max_depth': [50,70,80],
    'min_samples_split': [5,10],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [3,5]
}
rf = RandomForestRegressor(n_estimators=500, n_jobs=-1)
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='r2', n_jobs=-1)
grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
```

```
{'max_depth': 50, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}
```

```
#Use the optimal model to make predictions of the testing samples

rf_best = grid_search.best_estimator_
predictions_rf = rf_best.predict(X_scaled_test)
```

```
#Measure predicting performance of Regression Forest model

MSE_rf = mean_squared_error(y_test, predictions_rf).round(decimals=3)
MAE_rf = mean_absolute_error(y_test, predictions_rf).round(decimals=3)
Rsquared_rf = r2_score(y_test, predictions_rf).round(decimals=3)
print("For Regression Forest model,",
     "Mean squared error:", MSE_rf, ", Mean absolute error:", MAE_rf, ", R_squared:",
Rsquared_rf)
```

```
For Regression Forest model, Mean squared error: 41862.093 , Mean absolute error: 149.186 , R_squared: -0.434
```

2.4.Evaluating Models' Performance

To make it easy for comparison, we generate a table with performance measurements across 4 predicting models

- **Amazon's stock (AMZN)**

Performance measure	SGD-based Linear Regression	Regression Forest
MSE	1272.451	41862.093
MAE	24.983	149.186
R^2	0.956	-0.434

It is clear that SGD-based Linear Regression is better than Regression Forest, because its MSE and MAE metrics are smaller than the those of the latter, while its R^2 is higher.

- **Kroger's stock (KR)**

Performance measure	SGD-based Linear Regression	Regression Forest
MSE	0.36	0.309
MAE	0.415	0.373
R^2	0.962	0.967

In an opposite manner, for forecasting Kroger's stock price, Regression Forest model outperforms SGD-based Linear Regression model on all three performance metrics.

III. Conclusions and Recommendations

3.1. Conclusions

Although we apply the same feature generating process and predicting algorithms for Amazon's stock and Kroger's stock with the same time period, they result in different optimal parameter package, thus different optimal predicting models.

For AMZN, the SGD-based Linear Regression proves to be the optimum model to predict Amazon's stock price. In contrast, for KR, the Regression Forest possess better performance evaluation on all three metrics, MSE, MAE and R^2.

In a comparison between two stocks, it seems that both two predicting algorithms are better fit with Kroger's stock price, because three performance measurements of the worse model predicting KR stocks are even better than those of the optimal model predicting AMZN stocks. The lack of some features related to macro- and micro-economics could be a reason for insufficiency in models predicting Amazon's stock.

3.2. Recommendations

There are multiple ways to improve the predictive models, such as increasing sample size, adding more features.

More specifically, we can consider extending the beginning point further before 03-01-2000. For the ending point, we can wait a bit later to collect data, so that our stock price during early days of coronavirus will be included in testing dataset. By that way, the predicting model can learn from this event and will be better in predicting the stock prices in later stages of the outbreak.

Regarding adding more features, we can relate to the fundamental analysis approach to add in more variables reflecting macro conditions, such as world oil price, S&P 500 index, stock price of major competitors in the same industry, the US's GDP, and micro conditions, e.g. ROA, ROE, operating profit of the company. However, one precaution is many of these indicators are not collected on a daily basis, but rather monthly, quarterly or even yearly, thus resulting in distorting predictive values if we do not have appropriate solutions. Furthermore, we can dive deeper into technical stream to create more features. For example, the weighted moving average and exponential smoothing average can also be applied to weekly, monthly and yearly returns. Besides these two methods, we can employ the forecasting method that employ seasonality, such as stocks getting quite between May and August and performing the best during October – December period.

APPENDICES

Appendix 1. Optimal parameters and performance measurements for Kroger's stock

- **SGD-based Linear Regressor**

```
#Find out optimal set of parameters for SGD-based Linear Regression model
```

```
{'alpha': 1e-05, 'eta0': 0.01}
```

```
#Measure predicting performance of Linear Regression model
```

```
For Linear Regression model, Mean squared error: 0.36 , Mean absolute error: 0.415 , R_squared: 0.962
```

- **Regression Forest**

```
#Find out optimal set of parameters for Regression Forest model
```

```
{'max_depth': 80, 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 10}
```

```
#Measure predicting performance of Regression Forest model
```

```
For Regression Forest model, Mean squared error: 0.309 , Mean absolute error: 0.373 , R_squared: 0.967
```

REFERENCES

Yahoo Finance (2020, May 09), Amazon.com, Inc. (AMZN). Retrieved from: https://finance.yahoo.com/quote/AMZN/history?p=AMZN

Yahoo Finance (2020, May 09), The Kroger Co. (KR). Retrieved from: https://finance.yahoo.com/quote/KR/history?p=KR

Yahoo Finance (2020, May 09), Dow Jones Industrial Average (^DJI). Retrieved from: https://finance.yahoo.com/quote/%5EDJI/history?p=%5EDJI