Using Historical Stock Data to Create Predictive Models

Shahbab Ahmed, Isaiah Erven, Amirul Miah, Tanbirul Miah

Statement of Purpose

We hope to analyze historical trends to give us some sort of guidance in forecasting future activity. We hope to synthesize newer machine learning algorithms with classic stock data to help people make investment decisions. We will divide our data based on which exchange the stock is traded on (NYSE vs. NASDAQ), to account for possible differences. We also hope to use several technical indicators to observe what correlates to an increase or decrease in stock price. Using our findings from analyzing the historical data, we will form predictive models to forecast future stock prices.

We will use the following tools for this project:

Technical indicators: This portion of the project will involve incorporating technical indicators into our dataset to use as features for our models.

Technical indicators source: https://www.investopedia.com/top-7-technical-analysis-tools-4773275

Linear Regression: We will train and test a linear regression model that uses historical stock data to predict stock prices.

Decision Tree: Create a model that uses historical stock data to make a decision whether to invest or not, based on technical indicators and any other features we may implement.

Comparative analysis of different markets: We will run some form of comparative analysis test (i.e. t-tests, ANOVA) to explore if there are any significant differences in the different stock markets: NYSE, NASDAQ, and NYSE MKT

Data Visualization: Data visualizations to help identify key identifiers that back up our models' predictions. Will utilize packages such as seaborn and matplotlib.

Source of data: https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

Data Cleaning

The stocks/etfs datasets contained two folders with .txt files that contained an individual stock's/etf's "Open", "High", "Low", "Close" prices along with "Volume" and "Date". The data for each stock/etf is contained within a .txt file named based on their ticker symbol. For example, the data for Apple is contained within "aapl.us.txt". Since the data was set up in this manner (making it very difficult to create a dataframe with all available stocks) and there are potentially millions of rows, we decided to train our models on individual stocks rather than the entire market.

To accomplish this, we decided to create a function that takes a stock ticker as input (string) and outputs a cleaned dataframe with that stock's data. The create_stock_df function takes in a stock's ticker, opens and reads the file for that stock, creates an unclean dataframe of all that text, then splits the text into their associated variables. The function appends the data into an empty dictionary then turns it into a pandas dataframe. It also sets the data as an index and creates another column in the dataframe called "Average Price", which is the average of Open, Low, Close, and High. Finally, the function returns the created dataframe. Below is an example of what the data looks like before and after the function is applied:

```
['Date,Open,High,Low,Close,Volume,OpenInt\n',
 '1999-11-18,30.713,33.754,27.002,29.702,66277506,0\n',
 '1999-11-19,28.986,29.027,26.872,27.257,16142920,0\n',
 '1999-11-22,27.886,29.702,27.044,29.702,6970266,0\n',
 '1999-11-23,28.688,29.446,27.002,27.002,6332082,0\n',
 '1999-11-24,27.083,28.309,27.002,27.717,5132147,0\n',
 '1999-11-26,27.594,28.012,27.509,27.807,1832635,0\n',
 '1999-11-29,27.676,28.65,27.38,28.432,4317826,0\n',
 '1999-11-30,28.35,28.986,27.634,28.48,4567146,0\n',
 '1999-12-01,28.48,29.324,28.273,28.986,3133746,0\n',
 '1999-12-02,29.532,30.375,29.155,29.786,3252997,0\n',
 '1999-12-03,30.336,30.842,29.909,30.039,3223074,0\n',
 '1999-12-06,30.547,31.348,30.505,30.883,2385046,0\n',
 '1999-12-07,30.883,31.052,29.909,30.547,2348161,0\n',
 '1999-12-08,30.547,30.795,30.249,30.505,2000481,0\n',
 '1999-12-09,30.547,31.012,30.547,30.924,2150096,0\n',
 '1999-12-10,30.842,31.012,30.209,30.209,1764043,0\n',
 '1999-12-13,30.713,31.221,29.958,30.713,4260349,0\n',
 '1999-12-14,30.635,30.635,28.391,29.027,2467856,0\n',
```

```
create_stock_df("a")
```

Date	Open	High	Low	Close	Volume	Average Price
1999-11-18	30.713	33.754	27.002	29.702	66277506	30.489667
1999-11-19	28.986	29.027	26.872	27.257	16142920	28.295000
1999-11-22	27.886	29.702	27.044	29.702	6970266	28.210667
1999-11-23	28.688	29.446	27.002	27.002	6332082	28.378667
1999-11-24	27.083	28.309	27.002	27.717	5132147	27.464667
...	...	...	...	...	...	...
2017-11-06	68.22	68.45	68.22	68.22	995731	68.296667
2017-11-07	68.32	68.64	68.04	68.25	966466	68.333333
2017-11-08	68.1	68.33	67.771	68.11	972616	68.067000
2017-11-09	67.92	67.98	66.91	67.47	1673083	67.603333
2017-11-10	67.35	67.58	66.7	66.81	1704549	67.210000

Feature Engineering

Technical indicators are commonly used in investigating trends and predicting the future of a stock. They are calculated using a stock's price and volume to measure various aspects of a stock's performance. We wanted to include some technical indicators in our model because we hypothesized that our models would be more accurate with these features included in addition to volume. There are a lot of different technical indicators available, but we chose the following:

Average True Range (ATR): Measures the volatility of a stock by looking at its true range in terms of its highs and lows.
Relative Strength Index (RSI): A momentum indicator that evaluates whether a stock is being overbought or oversold. RSI above 70% typically means a stock is being overbought and oversold when under 30%.
Simple Moving Average (SMA): The average price of a stock given a specified range of time which shows whether, on average, the stock is going up or down.
Exponential Moving Average (EMA): An extension of SMA, but puts a lot more weight on most recent data points.
On-Balance Volume (OBV): Measures the momentum of a stock using its volume rather than price.

Source for talib: https://mrjbq7.github.io/ta-lib/

These indicators were feature engineered into our dataframe using a package called talib, which contains a library of function that calculates various indicators. To incorporate them into our dataframe, we created a function for each of the technical indicators we're using. These functions require two parameters: the previously created dataframe and timeperiod. Timeperiod specifies x number of past days to use for calculation. Depending on this parameter, the first x number of rows would be NaN since data may be unavailable based on the timeperiod. We included this parameter in the case we wanted to look at lower frequencies such as weekly or monthly. Finally, the add_ta() function simply takes in the cleaned df and timeperiod then calls all ta functions to add indicators to the df. For training, we also dropped Open, High, Low, and Close since we're predicting the Average Price.

```
add_ta(stock_df,14)
```

Date	Open	High	Low	Close	Volume	Average Price	ATR_14	RSI_14	SMA_14	EMA_14	OBV
1999-11-18	30.713	33.754	27.002	29.702	66277506	30.489667	NaN	NaN	NaN	NaN	66277506.0
1999-11-19	28.986	29.027	26.872	27.257	16142920	28.295000	NaN	NaN	NaN	NaN	50134586.0
1999-11-22	27.886	29.702	27.044	29.702	6970266	28.210667	NaN	NaN	NaN	NaN	43164320.0
1999-11-23	28.688	29.446	27.002	27.002	6332082	28.378667	NaN	NaN	NaN	NaN	49496402.0
1999-11-24	27.083	28.309	27.002	27.717	5132147	27.464667	NaN	NaN	NaN	NaN	44364255.0
...	...	...	...	...	...	...	...	...	...	...	...
2017-11-06	68.22	68.45	68.22	68.22	995731	68.296667	0.324857	65.241183	67.574286	67.609917	644836243.0
2017-11-07	68.32	68.64	68.04	68.25	966466	68.333333	0.308796	66.045201	67.712857	67.704595	645802709.0
2017-11-08	68.1	68.33	67.771	68.11	972616	68.067000	0.302453	62.613751	67.831429	67.757316	644830093.0
2017-11-09	67.92	67.98	66.91	67.47	1673083	67.603333	0.293707	59.872804	67.882857	67.779007	643157010.0

Exploratory/Comparative Data Analysis

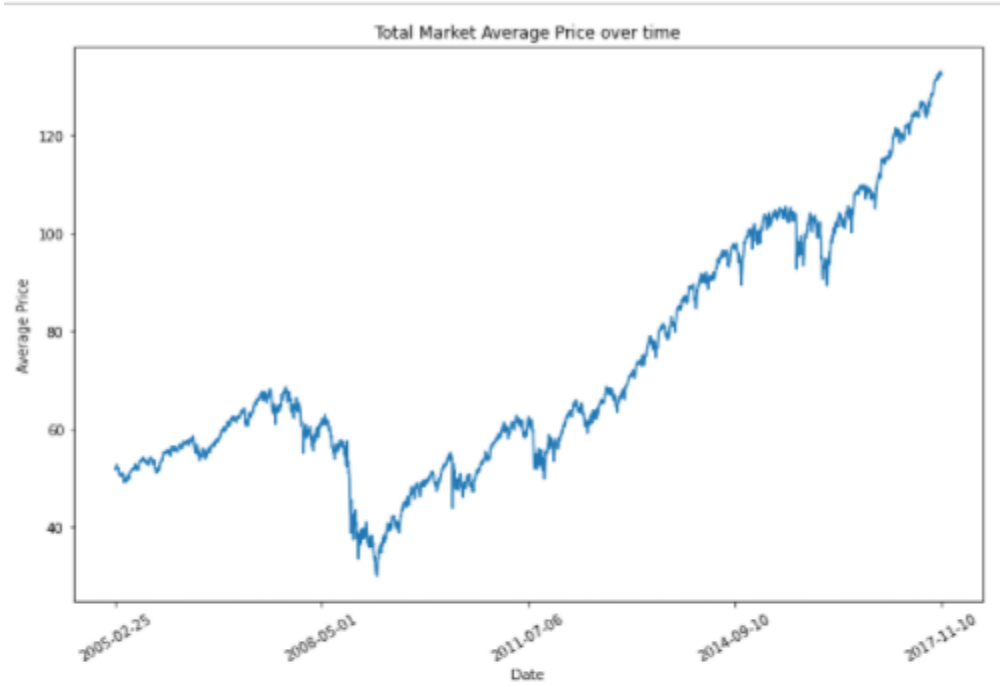
For the EDA, we used a few methods and plots to understand the data we cleaned:

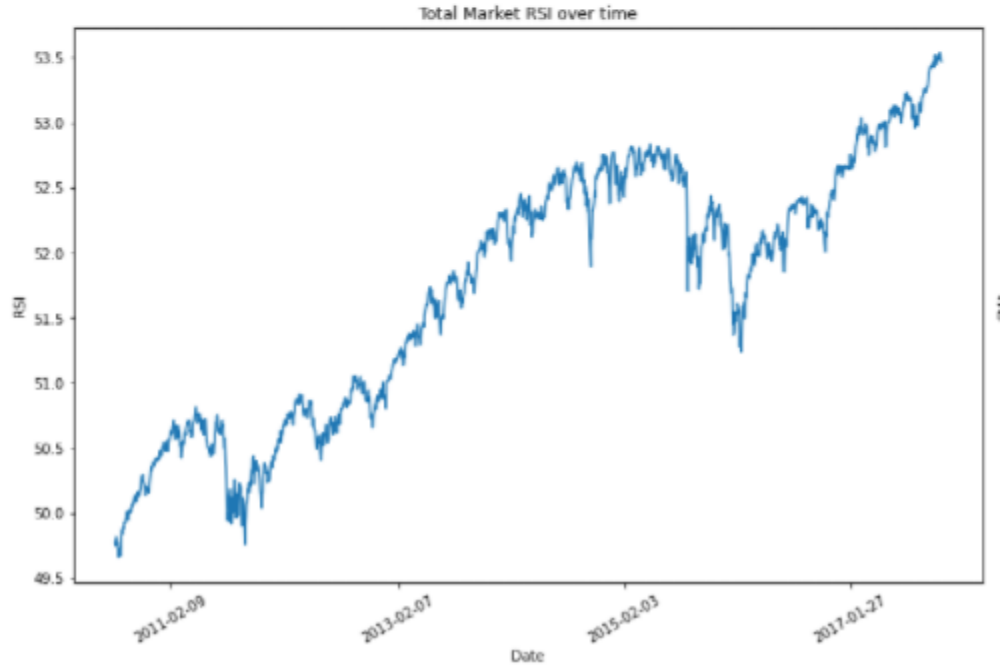
- dtypes to learn the types of columns in the dataframe
- shape to learn the dimensions of dataframe
- describe() to summarize the statistics of the dataframe
- pairplots to understand the correlation between the columns
- histplot to understand the distribution of frequencies of each volume value

Because we were looking at one stock at a time, the data wasn't too surprising. For example, most of the columns have positive correlations with one another. We also used the etf "vti" which tracks the performance of the entire market.

We were unable to run comparative analysis on the different markets because of limitations of

our data. Since data was separated by individual stocks, we could not distinguish between markets.





FAANG

Since our project involves working with individual stocks rather than the entire market, we decided to report our findings and focus our project on FAANG. FAANG is an acronym representing Facebook (now called Meta), Amazon, Apple, Netflix, and Google. These stocks

have been labeled as the best performing stocks in terms of their rapid growth so we thought it would be interesting to investigate them.

Principal Component Analysis

We conducted PCAs on each stock but we did not exclude any features during training since we were not working with too many variables to begin with. Below shows the explained variance ratio for Facebook (results were similar for FAANG stocks). While we kept all of our features, we can see that PC1 and PC2 explain around 70% of our data. From later results, we saw that RSI seemed to be the most important feature.

```
: pca.explained_variance_ratio_

: array([5.40620879e-01, 1.98633984e-01, 1.64183326e-01, 7.62824060e-02,
         2.02750813e-02, 4.32426129e-06])
```

Train/Test Split

For our train/test split, we split the data by 80/20 and set the shuffle parameter equal to False since we are working with timeseries data. Shuffling stock data also does not make sense in the context of our project. As mentioned earlier, we used Volume + Technical Indicators as features and Average Price as target. Our training data is the first 80% of dates within our dataframe of a given stock and the test data is the last 20% of dates. In other words, we are using the first 80% of days to train our model then making predictions on the next 20% of days.

Multiple Linear Regression

We trained linear regression models for FAANG stocks and the results are shown below. For the purpose of saving space, we did not include all visualizations since we had similar results for FAANG stocks.

Red: Predicticted average price
Blue: Actual average price

Facebook

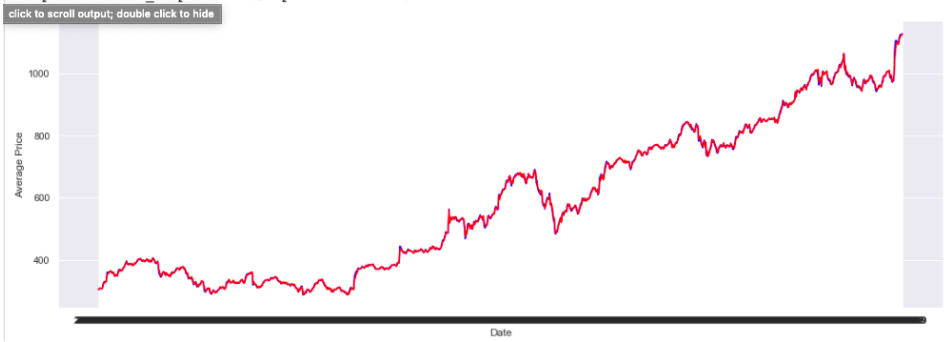
```
: from sklearn import metrics

  print("r2:", metrics.r2_score(y_true=y_test["Average Price"].values, y_pred=predictions))
  print("MSE:", metrics.mean_squared_error(y_true=y_test["Average Price"].values, y_pred=predictions))
  print("MAE:", metrics.mean_absolute_error(y_true=y_test["Average Price"].values, y_pred=predictions))

  r2: 0.99892308558025
  MSE: 0.40080462789093385
  MAE: 0.45918119946513175
```

Amazon

```
r2: 0.9997081651464661
MSE: 16.718717370765507
MAE: 2.792831171601175
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2746e210>
```

click to scroll output; double click to hide



Apple

```
r2: 0.9996726849774759
MSE: 0.3202908527975575
MAE: 0.4166866298672907
```

Netflix

```
r2: 0.998556091599789
MSE: 1.440724194545989
MAE: 0.8477507165800734
```

Google

```
r2: 0.9990708980678032
MSE: 15.205673747677256
MAE: 2.7434434231991767
```

Decision Tree

We also trained decision trees for each stock in order to classify whether a stock would go “Up” or “Down”. We first created a new column in the dataframe that tracked the percent change of a stock’s average price. Then we created a new variable called “Direction” which is 1 if the percent change is positive and 0 if it’s negative. We also removed the percent change

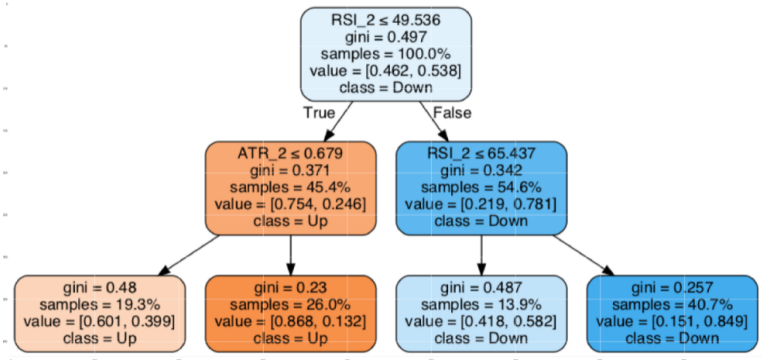
column after adding the Direction column. For the decision tree, we used Direction as the target and Volume, Average Price, and Technical Indicators as features. Below is an example of the dataframe used.

Date	Volume	Average Price	ATR_2	RSI_2	SMA_2	EMA_2	OBV	percent_change	Direction
2005-03-02	14459396	1.545267	0.034675	60.850757	1.53575	1.541067	4.608446e+07	0.590201	1
2005-03-03	7353878	1.527600	0.025238	41.802873	1.53500	1.531756	3.873059e+07	-1.143276	0
2005-03-04	5499270	1.519533	0.019019	27.735787	1.52070	1.520119	3.323132e+07	-0.528061	0
2005-03-07	4238311	1.524267	0.015909	56.806128	1.52070	1.524773	3.746963e+07	0.311499	1
2005-03-08	7677705	1.488533	0.015805	28.591201	1.51925	1.515858	2.979192e+07	-2.344297	0
...	...	...	...	...	...	...	...	...	...
2017-11-06	5845318	200.190000	0.891249	87.339097	199.90000	199.677942	2.859226e+09	0.372698	1
2017-11-07	6462740	198.266667	0.445624	87.339097	200.00000	199.892647	2.859226e+09	-0.960754	0
2017-11-08	4233611	195.943333	2.222812	8.754774	198.00000	197.297549	2.854993e+09	-1.171822	0
2017-11-09	5964989	193.253333	1.881406	5.171722	195.23000	195.405850	2.849028e+09	-1.372846	0
2017-11-10	6620920	191.233333	2.365703	2.056494	193.03500	192.875283	2.842407e+09	-1.045260	0

One of the parameters of a decision tree is max_depth, which defines the number of levels the tree will iterate through to arrive at a decision. The ROC AUC measures the performance of the model. Higher the AUC, the better the model is at predicting “Up” classes as “Up” and “Down” classes as ”Down”. We used this to determine the best max_depth to use, which varied based on the stock but typically it was 2-3 for FAANG.

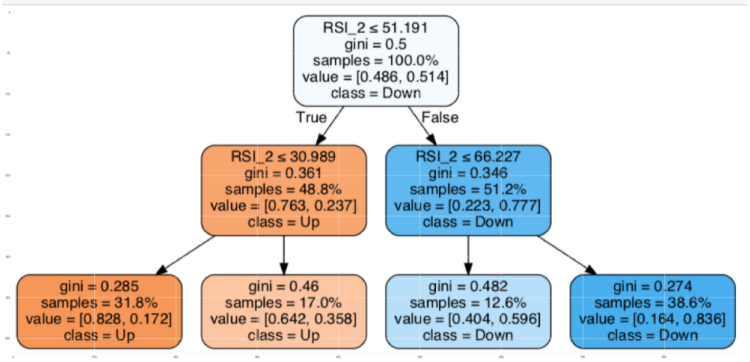
Facebook

```
{'Volume': 0.0,
 'Average Price': 0.0,
 'ATR_2': 0.09158765866828045,
 'RSI_2': 0.9084123413317196,
 'SMA_2': 0.0,
 'EMA_2': 0.0,
 'OBV': 0.0}
```



Amazon

```
{'Volume': 0.0,
 'Average Price': 0.0,
 'ATR_2': 0.0,
 'RSI_2': 1.0,
 'SMA_2': 0.0,
 'EMA_2': 0.0,
 'OBV': 0.0}
```



Apple

```
print(metrics.confusion_matrix(y_test, predictions))
```

```
[[557 199]
 [150 767]]
```

Netflix

```
[[196  91]
 [ 69 284]]
```

Google

```
[[203  92]
 [ 53 318]]
```

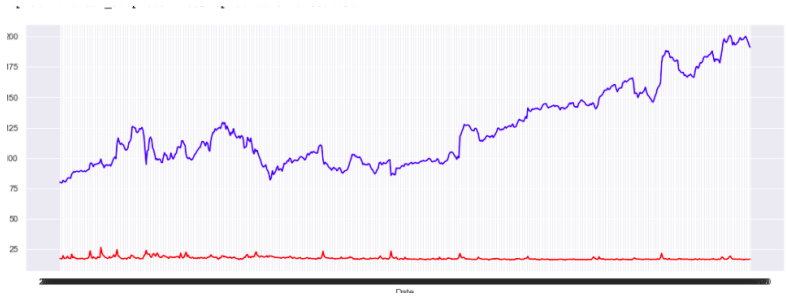
Results

Linear Regression:

Our regression models were very accurate for all of the FAANG stocks. All of our coefficients of determination (r2) were around .99, indicating that our models performed very well on the test data. In addition, our mean squared errors were also fairly low. For example, our predicted average prices for Amazon deviated from the actual average prices by around 16 (on average), whereas, it was around .30 for Apple. In addition, the graphs also display the accuracy of our model. You can see that the predicted line (red) trails extremely close to the actual line (blue).

We also trained and tested our model without using any of the technical indicators and only used volume. The results are shown below:

```
r2: -11.246361410455542
MSE: 12235.751822035692
MAE: 105.85602881986456
```



From this, we concluded that the technical indicators played a very significant role in the accuracy of our models.

Decision Trees:

Our decision trees also performed very well. The dictionaries under Facebook and Amazon show feature importances in the tree. We can clearly see that RSI seems to be the only measure used to make decisions. This is also reflected in the visuals because we can see that the tree mostly uses RSI to create branches (i.e. root node branches out based on whether RSI is <= around .50).

In addition, the confusion matrix conveys the following for Apple:

- 557 cases were correctly identified as “Up”
- 150 cases were incorrectly identified as “Up”
- 767 cases were correctly identified as “Down”
- 199 cases were incorrectly identified as “Down”
- 1324 cases were correctly identified out of 1673 observations

While results varied for each stock, we achieved around an 80% accuracy rate which was sometimes higher or lower.

Also, the gini impurity measure is high in some decisions (i.e. .5 in root node). This measure is the probability of misclassifying a winning team in each decision node. This is something to keep in mind when determining how well the model can perform.

Limitations/Challenges

The main limitations of these forecasts relates to the variety of factors that contribute to stock prices in the modern stock market. The stock market was previously based primarily on performance metrics that companies kept track of, but this has not been the case in recent times. The stock market always represents perceived values in companies and many things can affect this perception. This includes the pandemic, news stories, reddit events, and corporate branding. There are also newer platforms like Robinhood that are allowing even more people access to the stock market. The results of this project is limited to the scope of this project and the features we used. There are an unknown number of variables that affect a stock’s price and controlling for them is beyond the scope of this project.

Another limitation is that we trained our models only on individual stocks rather than the entire market. This may not be the best approach since the trajectory of the entire market has a significant impact on the performance of individual stocks. Another approach to this project would be to use the average price of all other stocks within a market to + technical indicators of a stock to predict its price. However, with how much data that would be (we estimated at least 2 million rows but could be a lot more), we would not have the computing power to train and test our models within a reasonable timeframe.

Future Recommendations

Future work surrounding each of the contributing factors could be extremely promising. Extensive research on what causes stock market crashes (panic, pandemics, politics) could be added to account for even more situations. Even smaller events like news stories and viral events on reddit can now affect stock prices. Individual influencers like Warren Buffet and Elon Mask can also cause stock market shifts. Using models that attempt to take into account these smaller events could also aid in providing recommendations and projections surrounding the stock market.

Future work on analysis of individual companies representing larger portions of the market would also be very effective. This would need a lot of work surrounding accounting analysis of companies in addition to the machine learning model. The pure number of companies listed on the stock exchange make this a massive project. This would mean looking into the financial health of companies and analyzing what factors contribute to the stock price at a company level.

We also recommend utilizing the entire market to predict the price of a singular stock (use all stocks as features and target stock as y).