Machine Learning Nanodegree
Capstone Project: Stock Price Predictor.

Author: Oscar Martinez Rico

I. Definition

Project Overview

Stock is a type of security that express ownership in a company and represents a claim on the part of the corporation's assets and earnings (Hayes, 2017). Stocks are also known as "shares" or "equity." There are two main types of stocks:

There are two main types of stocks include:

1. A common stock usually gives the owner to vote at shareholders' meetings and to receive dividends.
2. A preferred stock does not have voting rights but has a higher claim on assets and earnings than the common shares.

A holder of stock, also known as a shareholder, has a claim to a part of the company's assets and earnings. In other words, a shareholder is an owner of a company. The ownership of a company is calculated by the number of shares a person has relative to the number of outstanding shares. Below is an example (Figure 1.0) of a stock information sheet.

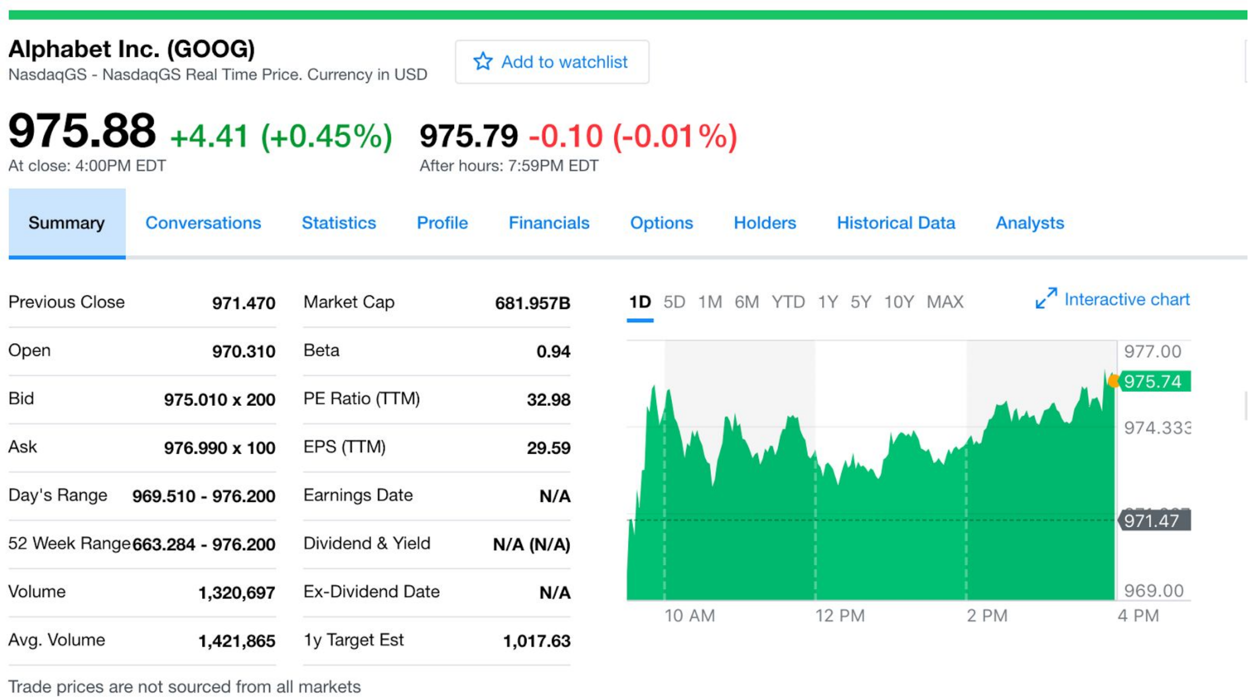
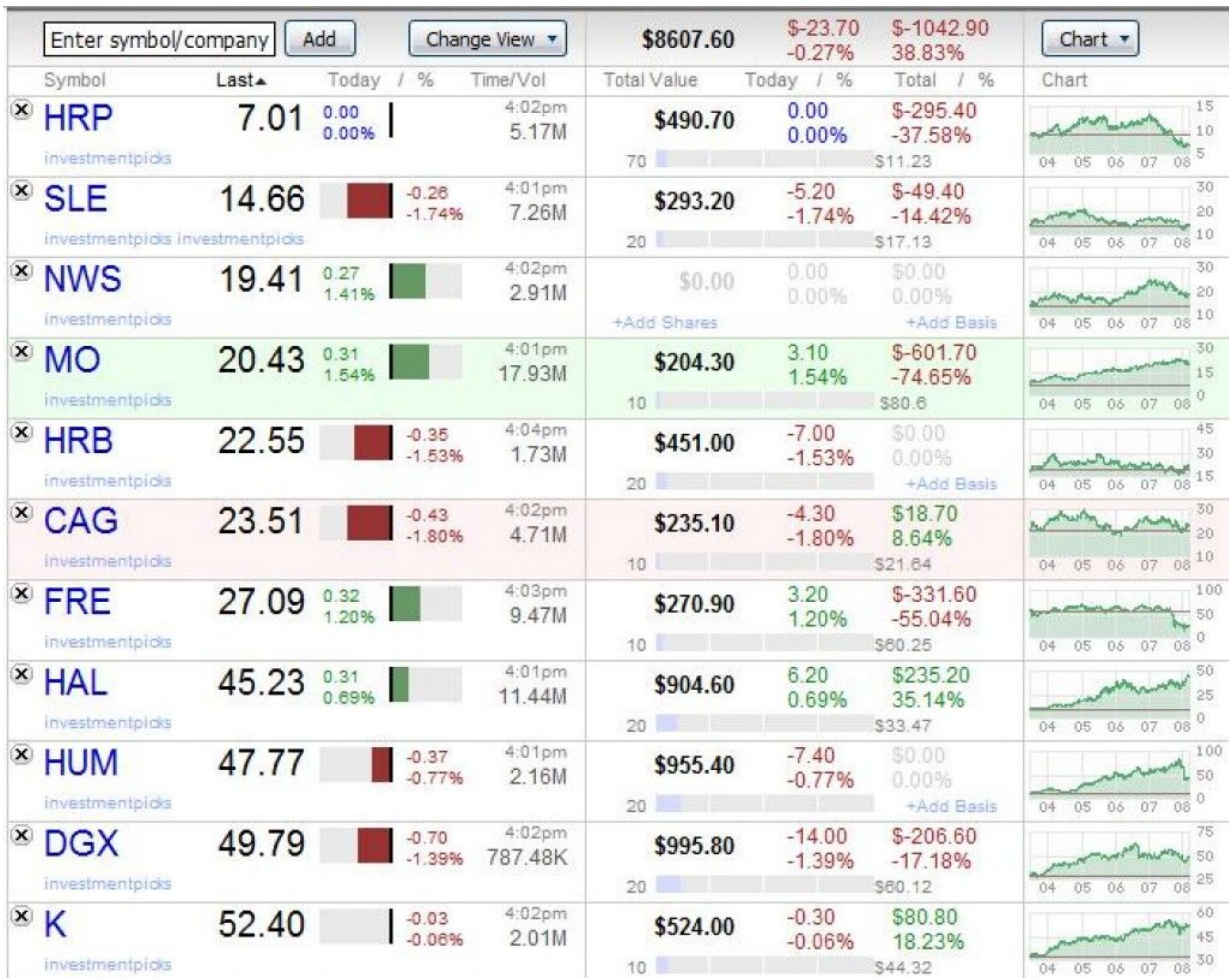


Figure 1.0

A stock quote provides information that includes: the current bid and offer prices, and the last price the stock was traded at. The highest price that an individual is willing to pay at a given time is the bid price. If an individual is interested in buying stocks, they have to make a bid. When the price of a bid and offer coincide, a trade is affected.

There is more information about a stock for example, trading volume, this is number of shares traded. Most of the time stock information is obtained online. To obtain the stocks information, we can search by their symbol. A stock symbol is formed using between one to four capital letters, which corresponds to the company name. Below are a picture(**Figure 2.0)** of some stocks symbols and their prices.



Enter symbol/company	Add	Change View		$8607.60	$-23.70 -0.27%	$-1042.90 38.83%	Chart
Symbol	Last	Today / %	Time/Vol	Total Value	Today / %	Total / %	Chart
HRP investmentpicks	7.01	0.00 0.00%	4:02pm 5.17M	$490.70 70	0.00 0.00%	$-295.40 -37.58% $11.23	
SLE investmentpicks investmentpicks	14.66	-0.26 -1.74%	4:01pm 7.26M	$293.20 20	-5.20 -1.74%	$-49.40 -14.42% $17.13	
NWS investmentpicks	19.41	0.27 1.41%	4:02pm 2.91M	$0.00 +Add Shares	0.00 0.00%	$0.00 0.00% +Add Basis	
MO investmentpicks	20.43	0.31 1.54%	4:01pm 17.93M	$204.30 10	3.10 1.54%	$-601.70 -74.65% $80.6	
HRB investmentpicks	22.55	-0.35 -1.53%	4:04pm 1.73M	$451.00 20	-7.00 -1.53%	$0.00 0.00% +Add Basis	
CAG investmentpicks	23.51	-0.43 -1.80%	4:02pm 4.71M	$235.10 10	-4.30 -1.80%	$18.70 8.64% $21.64	
FRE investmentpicks	27.09	0.32 1.20%	4:03pm 9.47M	$270.90 10	3.20 1.20%	$-331.60 -55.04% $60.25	
HAL investmentpicks	45.23	0.31 0.69%	4:01pm 11.44M	$904.60 20	6.20 0.69%	$235.20 35.14% $33.47	
HUM investmentpicks	47.77	-0.37 -0.77%	4:01pm 2.16M	$955.40 20	-7.40 -0.77%	$0.00 0.00% +Add Basis	
DGX investmentpicks	49.79	-0.70 -1.39%	4:02pm 787.48K	$995.80 20	-14.00 -1.39%	$-206.60 -17.18% $60.12	
K investmentpicks	52.40	-0.03 -0.06%	4:02pm 2.01M	$524.00 10	-0.30 -0.06%	$80.80 18.23% $44.32	

Figure 2.0

Problem Statement

There is no doubt that investing in the stock market can be one of the most exciting ways to invest money, especially when you can see your money multiply and grow.

Building a stock price predictor takes daily trading data over a certain date range as input, and outputs projected estimates for given query dates. By looking at the historical data of a given stock as an input, the stock predictor application will train the model to predict the Adjusted Close value for any given stock in the future. The model will be created using regression algorithms as we are attempting to predict the stock price. Having such a software system will benefit individuals and companies to make more educated decisions managing their stock portfolio.

There are multiple companies, hedge funds,and researchers that are currently investing resources in order to try to predict stock prices (Jigar, 2015). In order to help with the problem of predicting stock prices, companies and individuals can develop software systems using machine learning techniques and algorithms.

Metrics

To evaluate the trained, supervised learning model $\mathbf{R^2}$ will be used. The adequacy of the regression specification is often assessed by reference to the coefficient of (multiple) determination, commonly denoted R2 (**Figure 3.0**), a statistic, whose value is bounded by zero and unity, which measures the proportion of the variation in the response variable explained by the regression. As $\mathbf{R^2}$ declines from its maximum attainable value, it is perforce the case that the variance of the predicted, or fitted, values declines relative to that of the response variable itself (Dancer, 2005).

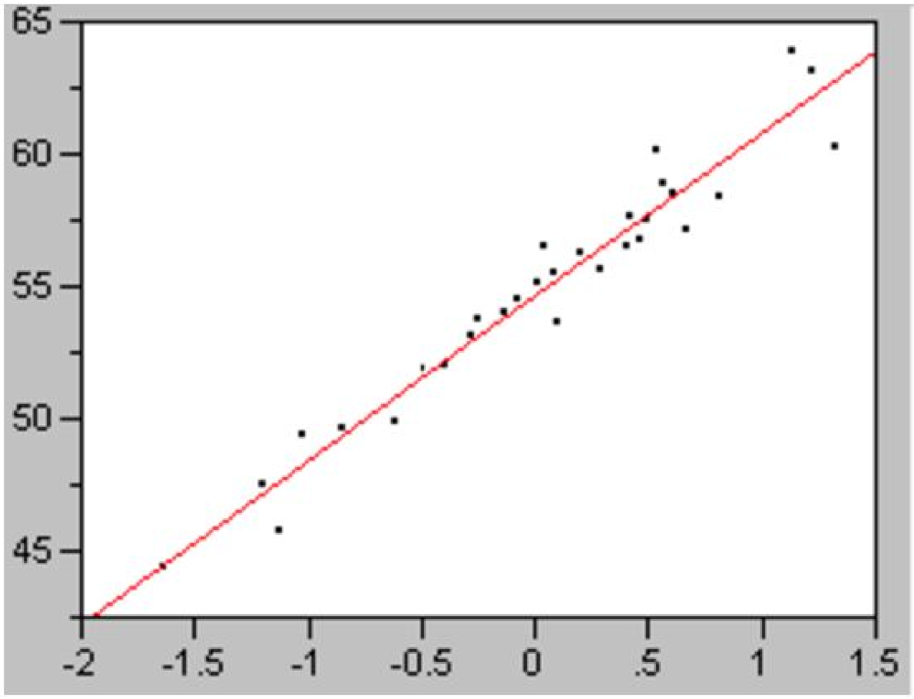


Figure 3.0

R^2 is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination.

The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:

$$R^2 = \frac{Explained\ variation}{Total\ variation}$$

R^2 is always between 0.0 and 1.0:

- 0 indicates that the model explains none of the variability of the response data around its mean.
- 1.0 indicates that the model explains all the variability of the response data around its mean.

In general, the higher the R-squared, the better the model fits your data.

The best possible score is 1.0. If the **R^2** score is negative, it shows that the model can be arbitrarily worse. A constant model that always predicts the expected value of y, disregarding the input features, would get an $\mathbf{R^2}$ score of 0.0.

The R^2 (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination (Brownlee, 2016).

II. Analysis

Data Exploration

Yahoo finance was used to obtain the historical dataset for this project. The dataset contains six features as it can bee seen in the following picture(**Figure 4.0**):

- *Open*: The opening price is the price at which a security first trades upon the opening of an exchange on a given trading day.
- *High*: the highest price at which a stock traded during the course of the day.
- *Low*: the lowest price at which a stock traded during the course of the day.
- Close: generally refers to the last price at which a stock trades during a regular trading session.
- *Volume*: the number of shares or contracts traded in a security or an entire market during a given period of time.
- *Adjusted Close*: is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open.

Alphabet Inc. (GOOG)
NasdaqGS - NasdaqGS Real Time Price. Currency in USD

Add to watchlist

966.95 +2.09 (+0.22%)
At close: 4:00PM EDT

967.99 1.04 (0.11%)
After hours: 7:51PM EDT

Summary | Conversations | Statistics | Profile | Financials | Options | Holders | **Historical Data** | Analysts

Currency in USD

Download Data

Date	Open	High	Low	Close	Adj Close*	Volume
Jun 01, 2017	968.95	971.50	960.01	966.95	966.95	1,403,700
May 31, 2017	975.02	979.27	960.18	964.86	964.86	2,448,100
May 30, 2017	970.31	976.20	969.49	975.88	975.88	1,466,700
May 26, 2017	969.70	974.98	965.03	971.47	971.47	1,252,000
May 25, 2017	957.33	972.63	955.47	969.54	969.54	1,660,500
May 24, 2017	952.98	955.09	949.50	954.96	954.96	1,024,800
May 23, 2017	947.92	951.47	942.58	948.82	948.82	1,270,800

Figure 4.0

The Yahoo finance website was used to extract the historical datasets for the stocks symbols listed below from 2012-05-22 to 2017-05-22:

- GOOG - Google
- AAPL - Apple
- AMZN - Amazon
- MSFT - Microsoft

Each dataset includes 1,257 entries, and this shows each day that the market was open. The features needed to train the model are: 'Open', 'High', 'Low', 'Close', and 'Volume', 'Adj Close'. 'Tomorrows Date' is the target variable, this is the value that the model is trying to predict. This column is included in the dataset by getting the 'Adj Close' from the following day, using the following:

```
dataset['Tomorrows Date'] = dataset['Adj Close']
dataset['Tomorrows Date'] = dataset['Tomorrows Date'].shift(-1)
```

The original dataset is split into training and testing datasets. The size of the training dataset is 10% of the size of the original dataset.

- Size of the training set has 1131 samples.
- Size of the resting set has 126 samples.

Statistical Information

The table below shows the statistical information of the dataset.

	Open	**High**	**Low**	**Close**	**Adj Close**	**Volume**
count	1257.000000	1257.000000	1257.000000	1257.000000	1257.000000	1.257000e+03
mean	574.670656	579.002621	569.857201	730.132065	574.584454	2.845960e+06
std	157.438846	158.364195	156.473257	165.820660	157.466377	2.065757e+06
min	279.123779	281.205963	277.220917	491.201416	278.481171	7.900000e+03
25%	442.665771	446.16766	440.354431	577.594238	442.675751	1.488100e+06

50%	556.532043	559.813049	551.981018	721.688660	556.573853	2.194200e+06
75%	719.469971	724.479980	713.000000	812.479370	718.359985	3.761500e+06
max	940.000000	943.109985	937.580017	1216.829224	943.000000	2.497790e+0

A new column is added to the previous dataset that represents the '**Adj Close'** price of the stock the next day, the column added is **'Tomorrow's Date'**, this is the target column.

Exploration Visualization

To better understand the dataset, a scatter matrix is created (**Figure 5.0**). The graph shows each of the features present in the data. For the features 'Open', 'High', 'Low', and 'Close', the graph shows the prices for the stock against at each trading date. For the 'Volume' feature we can see the number of shares traded each date. Although it is hard to see at this point, the graph below slightly shows that there might be a correlation between "Low" and "High" features. It is also important to note the data distribution of the graph below. The graph shows the features that are presented are left-skewed and we can see this at the diagonal of the scatter matrix plot; this indicates the skewness of the data.

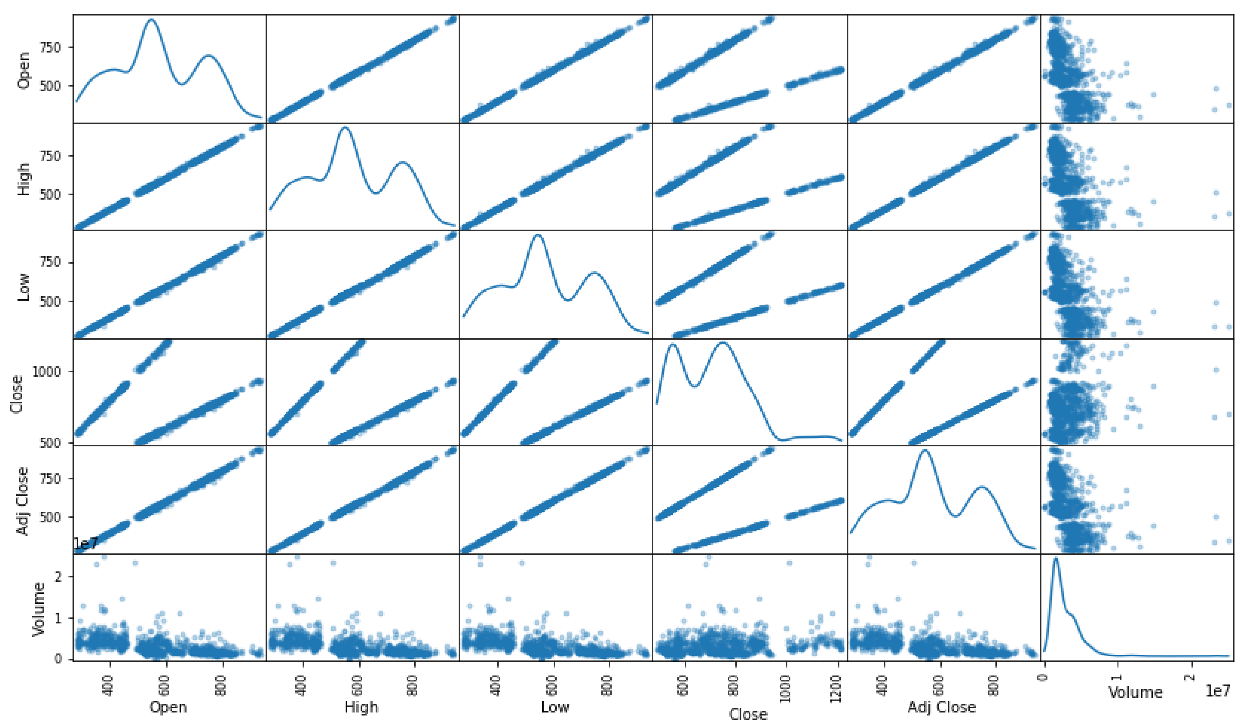


Figure 5.0

Algorithms and Techniques

Supervised machine learning algorithms will be used to create different models. In this problem the supervised machine learning model will try to predict stock prices, this is a good candidate for using a regression algorithm (Ericson, 2017). Supervised learning algorithms make predictions based on a set of examples. The historical stock prices will be used to predict future stock prices. A subset of data from the dataset will be used for training.The supervised learning algorithm will look for patterns in those value labels. After the algorithm has found the best pattern it can, it uses that pattern to make predictions for unlabeled testing data, for instance, future prices (Ericson, 2017). Regressions are helpful for estimating the relationships among variables. Regressions are widely used for prediction and forecasting(Penn State, 2017). Sklearn provide different Regression implementations and algorithms that we will discuss and explore in the following sections.

Linear Regression.

A linear regression is a statistical method that allows to condense and to study relationships between two continuous variables:

- One variable, denoted x, is regarded as the predictor, explanatory, or independent variable.
- The other variable, denoted y, is regarded as the response, outcome, or dependent variable.

Because the other terms are used less frequently today, we'll use the "predictor" and "response" terms to refer to the variables encountered in this course. The other terms are mentioned only to make you aware of them should you encounter them in other arenas.

KNeighborsRegressor.

KNeighborsRegressor is a regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors (Scikit-learn, 2017).

Epsilon-Support Vector Regression.

The method of Support Vector Classification can be extended to solve regression problems. This method is called Support Vector Regression. The model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. The free parameters in the model are C and epsilon (Scikit-learn, 2016).

Lasso.

Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. The implementation in the class Lasso uses coordinate descent as the algorithm to fit the coefficients.

Benchmark

From sklearn, we will use an out-of-the-box DummyRegressor for the benchmark. We will compare the $\mathbf{R^2}$ scores between the DummyRegressor and the scores obtained from the selected algorithms (Linear Regression, KNeighborsRegressor, Epsilon-Support Vector Regression, and Lasso). The $\mathbf{R^2}$ scores obtained from models created need to show that significantly outperform the DummyRegressor results so that we can assume that the model can be useful. Below is the implementation of the benchmark for this project:

```
from sklearn.dummy import DummyRegressor

def benchmark(X_train, X_test, y_train, y_test):

    print "*******************DummyRegressor Model*****************"

    model = DummyRegressor()

    model.fit(X_train, y_train)

    print '{}'.format(model.score(X_test, y_test))

    return model

benchmark(X_train, X_test, y_train, y_test)
```

```
*******************DummyRegressor Model*****************
-35.2392292353
DummyRegressor(constant=None, quantile=None, strategy='mean')
```

III. Methodology

Data Preprocessing

Once the dataset was obtained from the Yahoo Finance website the following steps were performed to prepare the data before creating the models:

- In the dataset, we used the 'Date' feature as the index column for the dataset.
- The target variable ('Tomorrows Date') was added to the dataset. This column is

included in the dataset by getting the ‘Adj Close’ from the following day, the code below shows how this variable was added.

  - dataset['Tomorrows Date'] = dataset['Adj Close']
  - dataset['Tomorrows Date'] = dataset['Tomorrows Date'].shift(-1)

- Split the data into two different sets, training and testing datasets respecting the order of the entries in the dataset. The testing dataset consists of 10% of the size of the original dataset.
- Feature scaling of the data. A natural logarithm was applied. **Figure 6.0** shows the data after its preprocessing.

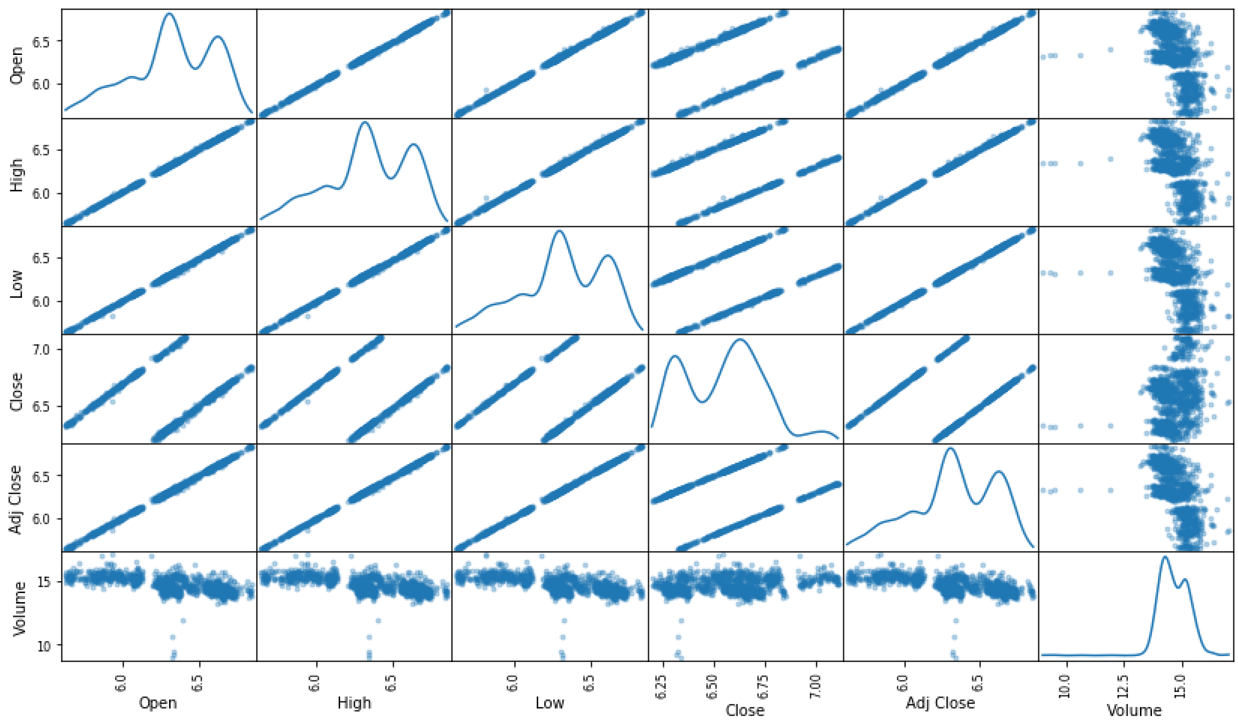


Figure 6.0

Implementation

The implementation of the stock predictor was performed executing the following steps:

1. The application receives the stock symbol of the stock as an input parameter. The supported symbols correspond to the downloaded datasets(GOOG, AAPL, AMZN, and MSFT).
2. Split historical data collected into two datasets, training data, and test data.
3. Training the model using the historical data.
4. Apply different supervised learning algorithms.
5. Measure the score and the performance of each algorithm.
6. Tweak the model, if necessary, to get better scores results.
7. Select the algorithm that provides a better score.

Supervised Learning Algorithms

Different algorithms were applied to select the algorithm that provides a better score. The following supervised learning models that were used are available in scikit-learn:

- Linear Regression.
- KNeighborsRegressor.
- Epsilon-Support Vector Regression.
- Lasso.

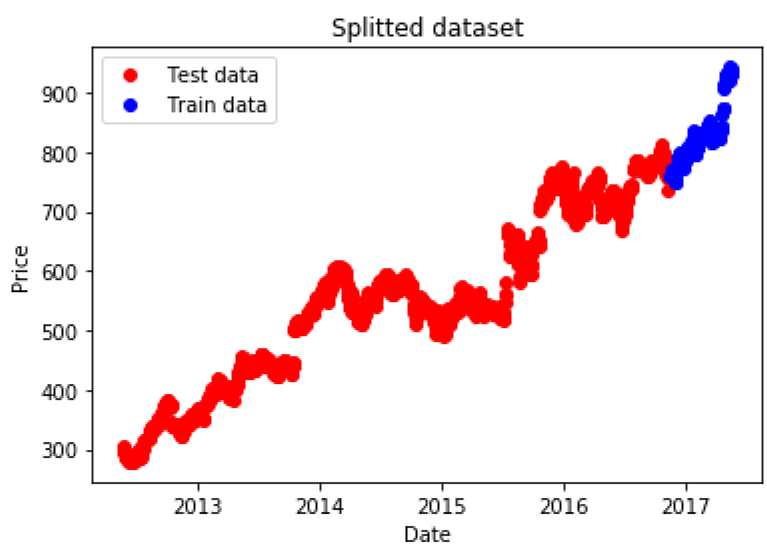


Figure 7.0

Figure 7.0 shows the data after it was split respecting the temporal order of the observations. The impletencion splits the original dataset by selecting an arbitrary split point in the ordered list of observations and creating two new datasets. The dataset was split in the following way:

- Training set has 1131 samples, which corresponds to approximately 90% of the size of the dataset.
- Testing set has 126 samples, this is approximately 10% of the size of the original dataset.

Each of the algorithms were trained with the same training dataset and also were evaluated against the same testing set. $\mathbf{R^2}$ was used to measure the coefficient of determination. In $\mathbf{R^2}$ the best possible score is 1.0 and it can be negative.The following table(**Figure 8.0)** describes the performance of the regressors and their corresponding scores.

Algorithm	R^2 Score	Trained model time.
Linear Regression.	0.976145747214	0.0210 seconds.
KNeighborsRegressor.	-19.0346707517	0.0021 seconds.
SVR Regression.	-36.6662215986	0.0792 seconds.
Lasso.	0.973000112941	0.0174 seconds.

Figure 8.0

It is easy to see that KNeighborsRegressor and SVR Regressor performed poorly as their $\mathbf{R^2}$ is close to 0.0 or produces a negative value. A model that produces a negative $\mathbf{R^2}$ indicates that the model is arbitrarily worse than one that always predicts the mean of the target variable (Scikit-learn, 2016). $\mathbf{R^2}$ provides a measure of how well future samples are likely to be predicted by the model, a score closer to 1.0 is always preferable. Linear Regression and Lasso produced a better score. Therefore, these algorithms are the best candidates. This is described in detail in the refinement section.

Refinement

After the default model of Linear Regression and Lasso regressors obtained an **R^2 score** of **0.976145747214** and **0.973000112941** respectively. Both models were tuned using different parameters values of each regressor using grid search. The grid search process of each regressor is discussed in detail in the subsequent sections.

Linear Regression Fine Tune

For the Linear Regression model, also known as ordinary least squares, a few parameters were considered to improve the **R^2 score** of the model (Scikit-learn, 2016). The following table (**Figure 9.0)** describe the parameters and the values that grid search identified.

Parameter	Description	Value
fit_intercept	Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations .	False.
normalize	If True, the regressors X will be normalized before regression. This parameter is ignored when fit_intercept is set to False. When the regressors are normalized, note that this makes the hyperparameters learned more robust and almost independent of the number of samples.	True.

Figure 9.0

After applying grid to the Linear Regression model the initial **R^2 score** was slightly improved from **0.976145747214** to **0.976615311921**.

Lasso Regression Fine Tune

Lasso is a linear model that estimates sparse coefficients (Scikit-learn, 2016). For this model, some parameters were considered to improve the **R^2 score** of the model. The following table (**Figure 10)** describe the parameters and the values that were applied to perform the grid search.

Parameter	Description	Value
fit_intercept	Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations.	True.
normalize	If True, the regressors X will be normalized before regression. This parameter is ignored when fit_intercept is set to False. When the regressors are normalized, note that this makes the hyperparameters learned more robust and almost independent of the number of samples.	True.
max_iter	The maximum number of iterations.	10000
alpha	Constant that multiplies the L1 term. Defaults to 1.0. alpha = 0 is equivalent to an ordinary least square, solved by the **LinearRegression** object.	0.1
selection	If set to ‘random’, a random coefficient is updated every iteration rather than looping over features sequentially by default.	random

Figure 10

After applying the grid search to the Lasso Regression model, the initial **R^2 score** was slightly improved from **0.973000112941** to **0.976310473071**.

IV. Results

Model Evaluation and Validation

This project explored further Linear Regression and Lasso Regression models because these models provided a far better **R^2 score** than the other models explored, KNeighborsRegressor and SVR Regression, even before that the tuning process was applied.

A grid search was used to tune Linear Regression and Lasso Regression models and obtained the better estimators; we can see a summary in the following table (**Figure 11).**

Model	Parameters	Final R^2 Score	Time to Train
Linear Regression.	{'copy_X': True, 'normalize': True, 'n_jobs': 1, 'fit_intercept': False}	0.976615311921	7.6564 seconds
Lasso Regression.	{'normalize': False, 'warm_start': False, 'selection': 'cyclic', 'fit_intercept': True, 'positive': False, 'max_iter': 10000, 'precompute': False, 'random_state': 42, 'tol': 0.0001, 'copy_X': True, 'alpha': 1.0}	0.976310473071	20.3263 seconds.

Figure 11

The difference between the two different models is minimal. The Lasso Regression model **R^2 score** is slightly above the Linear Regressor model by **~0.0003%**. As the purpose of this project is to predict the stock price I believe that having a more precise model is a better idea. In this case, I decided to use the Linear Regression because it produces a **R^2 score** that it closer to 1.0. This model was tested using different stocks:

- GOOG - Google
- AAPL - Apple
- AMZN - Amazon
- MSFT - Microsoft

I believe that this model needs to be improved to be used to make real life trading decisions. For instance, similar stocks could be grouped together to create a model that would discover new features that can impact the stock prices for similar companies. Stock prices predictions are so complex that we need to build a more robust dataset to be able to build a model that it can predict the prices more accurate. Having a machine learning model like this is a good starting point, however, it needs improvement.

Justification

After tuning both models, Linear Regression and Lasso Regression, the **R^2 score** obtained significantly outperform the **R^2 score** obtained by the DummyRegressor.

Model	Parameters	R^2 Score
Dummy Regressor	constant=None, quantile=None, strategy='mean'	-35.2392292353
Linear Regression.	{'copy_X': True, 'normalize': True, 'n_jobs': 1, 'fit_intercept': False}	0.976615311921
Lasso Regression.	{'normalize': False, 'warm_start': False, 'selection': 'random', 'fit_intercept': True, 'positive': False, 'max_iter': 10000, 'precompute': False, 'random_state': 42, 'tol': 0.0001, 'copy_X': True, 'alpha': 0.1}	0.976310473071

Figure 12

As we can see in the table(**Figure 12**) above, both models were able to outperform the DummyRegressor used for the benchmark for this project. However, while building this project, it was noted that to create a model that predicts the stock prices more accurately, it is necessary to obtain a more robust dataset to captures other important features that help to predict a stock price.

V. Conclusion

Free-form Visualization

After tuning both models, Linear Regression and Lasso Regression models, the **R^2 score** obtained was close to 1.0, which the possible score for the **R^2** score.

<table>
<tr><th>Linear Regressor</th><th>Lasso Regressor</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Figure 13

We can see in the **Figure 13** that after approximately 100 training points the **R^2 score** were close to 1.0 . However, this high **R^2 score** does not mean that the model has a good fit. After obtaining these scores for both models, I believe that scores values were too high, in the Reflection section I will explore some possibilities that can be causing these high scores.

Reflection

Creating a regression to help with the stock price prediction is a hard problem to solve. This project only shows a first step in the process of creating a more robust model. Also it's important to realize that the split process in different for this type of regressions where time series data is present. Splitting the data respecting the order of the entries is a key important task to solve the stock price. For this type of problem we can not split it randomly using sklearn train_test_split. When dealing with time series data, we have to split the data maintaining the order in the data, and split at a certain point in time. In this case we used the last 10% of the data for testing.

The steps taken to create the model are the following:

- Explored different Supervised Machine Learning algorithms.
- Obtain the datasets from Yahoo finance for each of the following symbols:
    a. Google.
    b. Amazon.
    c. Microsoft.
    d. Apple.
- Explore the dataset using different plots such as scatter matrix plot.
- Dataset preprocessing:
    a. Extract the target variable from the original dataset.
    b. Perform feature scaling using natural logarithm.
- Split the dataset into training and testing datasets respecting the ordered list of observations.
- Use and calculate the benchmark for the dataset, DummyRegressor from sklearn.
- Evaluate the following regressor models from sklearn.
    a. Linear Regression.
    b. KNeighborsRegressor.
    c. Epsilon-Support Vector Regression.
    d. Lasso.
- Calculate the **R^2 scores** and determine the top models.
    a. Linear Regression.
    b. Lasso.
- Tune the top models using Grid Search.
- Determine the best model base of the score results.
    a. Linear Regression.

As we discuss earlier high **R^2 scores** doesn't necessarily mean that the model is more precise. Some factors that could be causing these high **R^2 scores** are:

- **Overfitted model**: An overfitted model is a model that is too complicated for a data set. Some reasons for these could be including too many terms in your model compared to the number of observations. When this happens, the regression model becomes tailored to fit the quirks and random noise in the sample dataset rather than reflecting the overall population. Overfitting occurs when model describes random error or noise instead of the underlying relationship (Multicollinearity, 2017).
- **Highly correlated data**: Correlation between the features can lead to overfitting. The best regression models are those in which the predictor variables each correlate highly with the dependent (outcome) variable but correlate at most only minimally with each other. Such a model is often called "low noise" and will be statistically robust (Multicollinearity, 2017).

The table below shows the correlation between the all the features and the target variable in the original dataset used to train the models described earlier.

	Open	**High**	**Low**	**Close**	**Adj Close**	**Volume**
Open	1.000000	0.999696	0.999512	0.150242	0.999182	-0.520535
High	0.999696	1.000000	0.999474	0.148506	0.999559	-0.516828
Low	0.999512	0.999474	1.000000	0.154026	0.999689	0.527687
Close	0.150242	0.148506	0.154026	1.000000	0.151542	0.229581
Adj Close	0.999182	0.999559	0.999689	0.151542	1.000000	-0.523261
Volume	-0.520535	-0.516828	-0.527687	0.229581	-0.523261	1.000000

Figure 15

By looking at the table(**Figure 15)** above we can see that the features “Open”, “High”, and “Low” are highly correlated. Since multicollinearity causes imprecise estimates of coefficient values in a regressor, the resulting predictions will also be imprecise. These are the reasons why I believe the current dataset needs to be improved.

Improvement

Stock price prediction has become more popular as more individuals have stocks as part of their overall equity. In the tech industry, it is common for companies to offer a stock equity as part of the compensation package for their employees. Finding the right time to sell or buy a certain stock to maximize the capital gains can be a hard problem to solve.

After the creation of the model for this project, it was clear to see that the original dataset needs to be extended to capture more features that can help to create a more realistic model. There can be a significant correlation between the changes in weekly stock prices based on important current events, news, and weather occurring around the world. It is important to review and consider a company's financial health, the value of a company's assets, debts, cash, revenues, expenses, profitability and plans of development.

References

- Brownlee, Jason. (2016). Metrics To Evaluate Machine Learning Algorithms in Python. Retrieved June 8, 2017, from http://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/.
- Dancer, D., & Tremayne, A. (2005). R-Squared and Prediction in Regression with Ordered Quantitative Response. *Journal Of Applied Statistics*, *32*(5), 483-493.
- Ericson, G. & Franks, L. (2017). How to choose algorithms for Microsoft Azure Machine Learning. (2017). Retrieved June 3, 2017, from https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice
- Hayes, Adam. (2017). Stocks Basics: What Are Stocks? Retrieved June 2, 2017 from http://www.investopedia.com/university/stocks/stocks1.asp
- Jigar, Patel. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques, Expert Systems with Applications, Volume 42, Issue 1, January 2015, Pages 259-268, ISSN 0957-4174, Retrieved June 7, 2017, from https://doi.org/10.1016/j.eswa.2014.07.040.
- Multicollinearity. (2017). In Wikipedia, The Free Encyclopedia. Retrieved June 5, 2017, from https://en.wikipedia.org/w/index.php?title=Multicollinearity&oldid=776794468
- Penn State. (2017). What is Simple Linear Regression? Retrieved June 8, 2017, from https://onlinecourses.science.psu.edu/stat501/node/251
- Scikit-learn developers. (2016). Generalized Linear Models. Retrieved June 4, 2017, from http://scikit-learn.org/stable/modules/linear_model.html#lasso
- Scikit-learn developers. (2016). Model evaluation: quantifying the quality of predictions. Retrieved June 4, 2017, from http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score
- Scikit-learn developers. (2016). Nearest Neighbors Classification. Retrieved June 4, 2017, from http://scikit-learn.org/stable/modules/neighbors.html#regression.
- Yahoo Finance. (2017). Alphabet Inc. Retrieved June 1, 2017, from https://finance.yahoo.com/quote/GOOG/history?ltr=1