Final Project:

Stock Price Prediction Using Machine Learning Techniques

Phimonkae Wilairat

Analytics, Northeastern University

ALY6140: Analytics System Technology

Daya Rudhramoorthi

May 16, 2022

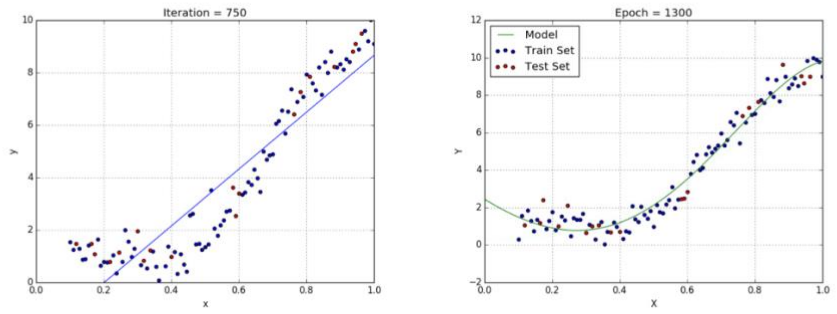
Introduction

Accurate prediction of stock returns is a challenging task today due to many uncertainty factors that affect financial stock's volatile and non-linear nature. Using the human ability to think critically by considering fundamental factors or looking at technical graphs to analyze stock may not be enough due to the vast amount of data to ponder in stock analyzing and forecasting. By taking advantage of the computational capabilities of machine learning and robust algorithms, the stock market prediction has become more efficient. Its ability in predictive analytics is suitable for the complex areas of human activity like the stock market, which reflect the thoughts and decisions of humans. The behavior of individual stocks sometimes does not always respond to economic factors, which is why there is the statement that the market status does not correlate with the health of the economy overall (Madrick, 2020). Therefore, the fundamental analysis technique considers economic data at a certain point in the past to predict the future may not provide accurate results. As the information coming into the market is getting closer to real-time, algorithms are used to learn investors' behavior through previous stock price performance and then provide a more accurate prediction of their future actions.

Machine learning can learn new things based on previous data, and its algorithms become better when data increase. The stock market generates tons of data daily, and thus, machine learning can become a perfect fit to analyze the stock markets (Joshi, 2020). For stock market prediction, Machine learning works in the same way as financial analysts do. It starts with studying previous stock market prices, then examining its trends, and using that information to predict the future movement of the stocks (Joshi, 2020). Machine learning deployed in predictive modeling is primarily expected to minimize the error of a model or make the most accurate predictions possible (Brownlee, 2016). We can check whether the algorithm performed accurately by comparing its result to the actual stock performances; as a result, more effective improvements will be applied to the algorithms.

In predictive modeling, statistical techniques using machine learning are utilized to predict possible future stock prices supported by historical and existing data. Three types of models have been chosen for this project : Simple Linear Regression, Quadratic Polynomial regression, and K Nearest Neighbor (KNN)

Simple Linear Regression: developed as a model for understanding the relationship between input and output numerical variables. The model predicts future stock prices from its price in the past and assumes a linear relationship between them. Therefore, the model can understand patterns that a given dataset fits in a simple linear equation.

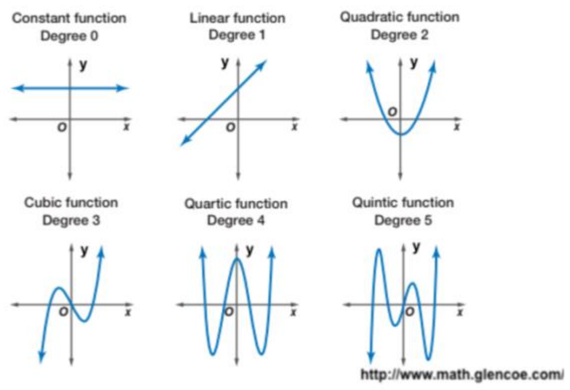


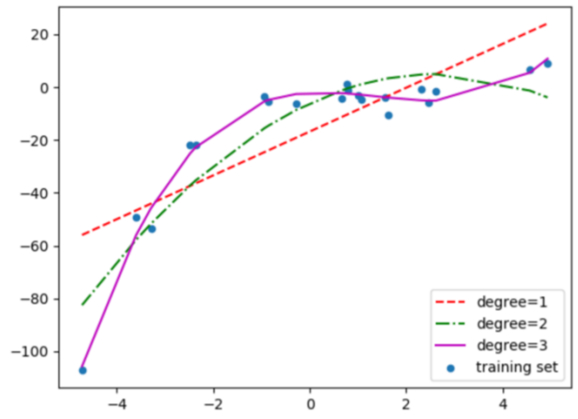
Left: Linear Regression, Right: Polynomial regression | GIF: Towards Data Science

Quadratic Polynomial Regression: developed as a model for dealing with complex data that applying linear regression model might not be accurate. The polynomial model produces curves that adjust with the data rather than the lines. The equation of the polynomial model would be :

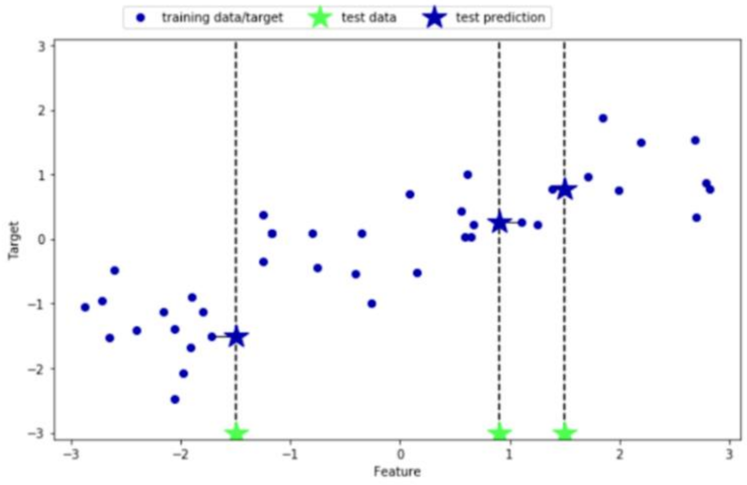
$$Y=\theta o + \theta_1 X + \theta_2 X^2 + \ldots + \theta \ \ X^m + \text{residual error}$$

This project includes higher degree polynomial terms: x^2 polynomial(2) and x^3 polynomial(3) that allows the fit of more flexible models representing the relation between the result and some continuous predictors.





K Nearest Neighbor (KNN): its algorithm is developed for feature similarity to predict the values of any new data points. In other words, the new data point is assigned a value based on how closely it resembles the points in the training set (Singh, 2018).



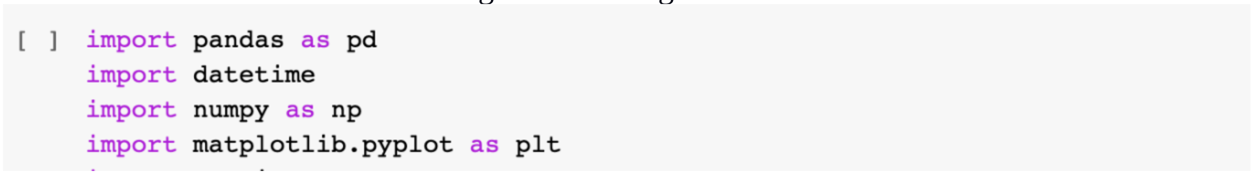
This report aims to understand stock analysis and price predictions using Python code. The initial process starts by selecting interested stocks to explore trading elements such as stock price and volume. The following process performs stock analysis by comparing performances between stocks, their sector, the market, and economic factor. Lastly, the stock price prediction is operated by using individual stock price data sets to predict its future price through the models mentioned above.

Exploratory Data Analysis

Data set used in this project

- Three common stocks : AAPL (Apple Inc.), GOOG (Alphabet Inc.), and AMZN (Amazon.com Inc.). The historical stock prices is kept on a daily basis in a file format .xlsx
- NASDAQ Composite index and NASDAQ 100 Technology Sector are selected to represent the performance of the market and sector, respectively. Both indexes will be used to compare the performance of individual stocks. The index price data are also collected on a daily basis in a excel file (.xlsx)
- US 10 Years Bond Yield historical data represent investors' confidence in the economy. The bond yield will be used as a proxy to compare the stock performance with the economic condition. Its historical data is gathered daily in an excel file.

These data sets will be extracted using the following codes:

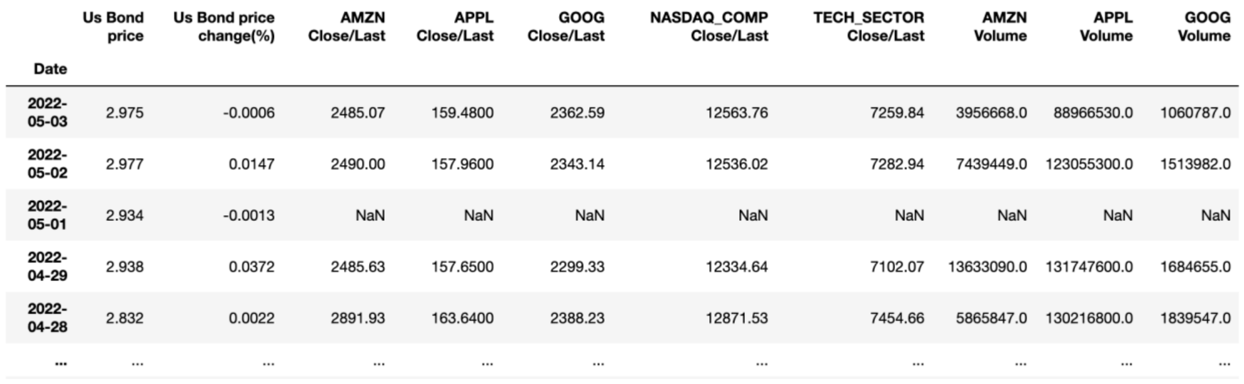


```
[ ] import pandas as pd
    import datetime
    import numpy as np
    import matplotlib.pyplot as plt
```

Pandas was implemented to manage time-series data that contains extensive tools for working with dates, times, and time-indexed data. In working with trading dates, the Python objects residing in the build-in datetime module will be executed to change different date formats into the same format. Then, convert the data frame column into an index. Numpy will be used for array oriented computing and Matplotlib for visualization.

Data cleanup

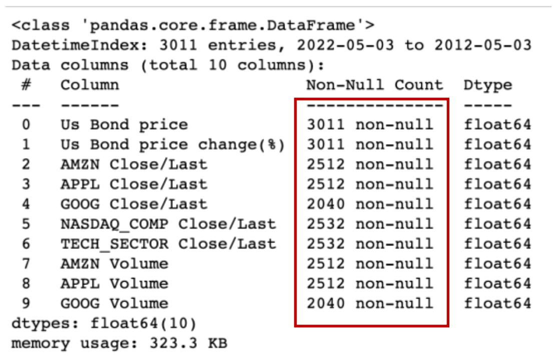
For stock and index data frame (Amzn, Appl, Goog, Nasd, techSec_df), the different date format in the date column is converted into the same format by creating a function that change one format to desired format (‘%Y-%m-%d’) through function datetime and strip time. And then make that date column to become the data frame's index. After dealing with the date format, data in other columns must be changed from string to float and remove $. However, US bond data frame (usBond_df) has a different way of dealing with date formats because the trade date is recorded in the form of the month name. Therefore, a dictionary is used to deal with this problem, and stripe time is still used to convert the format to ('%Y-%m-%d') same as the date format of stocks and indexs. The next step is merging data frame, which includes stocks, market index, sector index, and bond yield. Before putting all data together, make sure that the name of columns has been changed since each data frame has the same column name, such as "Close/Last". It should be specific which ones belong to which stock. The merged data frame will show as following



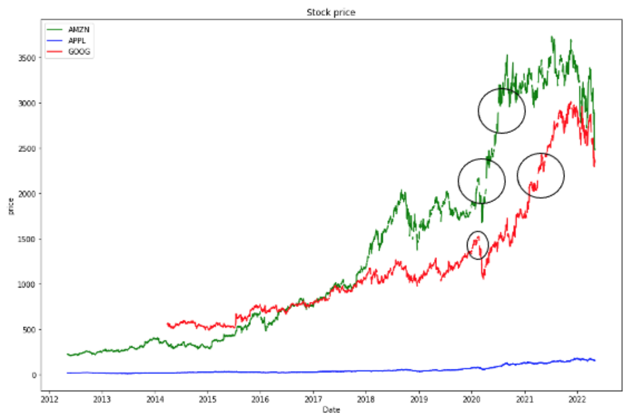
	Us Bond price	Us Bond price change(%)	AMZN Close/Last	APPL Close/Last	GOOG Close/Last	NASDAQ_COMP Close/Last	TECH_SECTOR Close/Last	AMZN Volume	APPL Volume	GOOG Volume
Date										
2022-05-03	2.975	-0.0006	2485.07	159.4800	2362.59	12563.76	7259.84	3956668.0	88966530.0	1060787.0
2022-05-02	2.977	0.0147	2490.00	157.9600	2343.14	12536.02	7282.94	7439449.0	123055300.0	1513982.0
2022-05-01	2.934	-0.0013	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2022-04-29	2.938	0.0372	2485.63	157.6500	2299.33	12334.64	7102.07	13633090.0	131747600.0	1684655.0
2022-04-28	2.832	0.0022	2891.93	163.6400	2388.23	12871.53	7454.66	5865847.0	130216800.0	1839547.0
...	...	...	...	...	...	...	...	...	...	...

Nevertheless, some data is missing after merged data frames due to mismatched operating dates between stock, index, and US bonds. Therefore, filling the missing value will be done in the next step. The function .interpolate(option='spline') was implemented to fix a problem based on fixed data points.

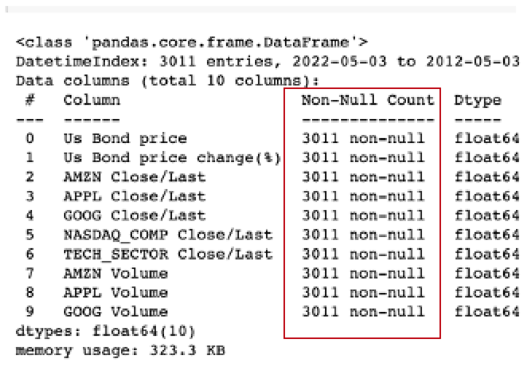
The data frame with missing value



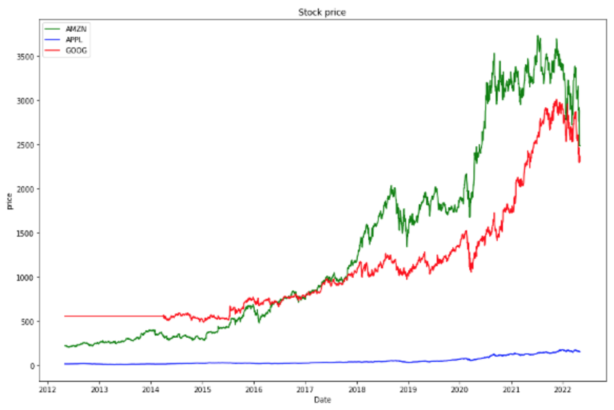
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3011 entries, 2022-05-03 to 2012-05-03
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Us Bond price             3011 non-null   float64
 1   Us Bond price change(%)   3011 non-null   float64
 2   AMZN Close/Last           2512 non-null   float64
 3   APPL Close/Last           2512 non-null   float64
 4   GOOG Close/Last           2040 non-null   float64
 5   NASDAQ_COMP Close/Last    2532 non-null   float64
 6   TECH_SECTOR Close/Last    2532 non-null   float64
 7   AMZN Volume               2512 non-null   float64
 8   APPL Volume               2512 non-null   float64
 9   GOOG Volume               2040 non-null   float64
dtypes: float64(10)
memory usage: 323.3 KB
```



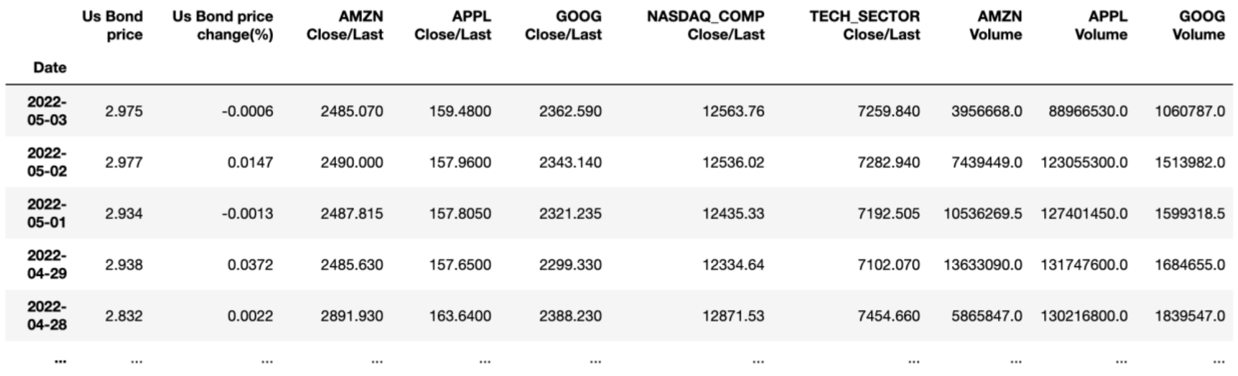
After fixed the problem with interpolate



```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3011 entries, 2022-05-03 to 2012-05-03
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Us Bond price             3011 non-null   float64
 1   Us Bond price change(%)   3011 non-null   float64
 2   AMZN Close/Last           3011 non-null   float64
 3   APPL Close/Last           3011 non-null   float64
 4   GOOG Close/Last           3011 non-null   float64
 5   NASDAQ_COMP Close/Last    3011 non-null   float64
 6   TECH_SECTOR Close/Last    3011 non-null   float64
 7   AMZN Volume               3011 non-null   float64
 8   APPL Volume               3011 non-null   float64
 9   GOOG Volume               3011 non-null   float64
dtypes: float64(10)
memory usage: 323.3 KB
```



As a result, I got completed data frame which ready to be used in next step

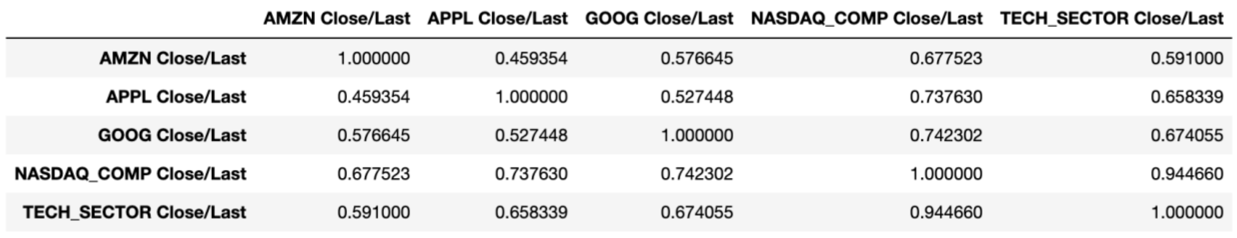


	Us Bond price	Us Bond price change(%)	AMZN Close/Last	APPL Close/Last	GOOG Close/Last	NASDAQ_COMP Close/Last	TECH_SECTOR Close/Last	AMZN Volume	APPL Volume	GOOG Volume
Date										
2022-05-03	2.975	-0.0006	2485.070	159.4800	2362.590	12563.76	7259.840	3956668.0	88966530.0	1060787.0
2022-05-02	2.977	0.0147	2490.000	157.9600	2343.140	12536.02	7282.940	7439449.0	123055300.0	1513982.0
2022-05-01	2.934	-0.0013	2487.815	157.8050	2321.235	12435.33	7192.505	10536269.5	127401450.0	1599318.5
2022-04-29	2.938	0.0372	2485.630	157.6500	2299.330	12334.64	7102.070	13633090.0	131747600.0	1684655.0
2022-04-28	2.832	0.0022	2891.930	163.6400	2388.230	12871.53	7454.660	5865847.0	130216800.0	1839547.0
...	...	...	...	...	...	...	...	...	...	...

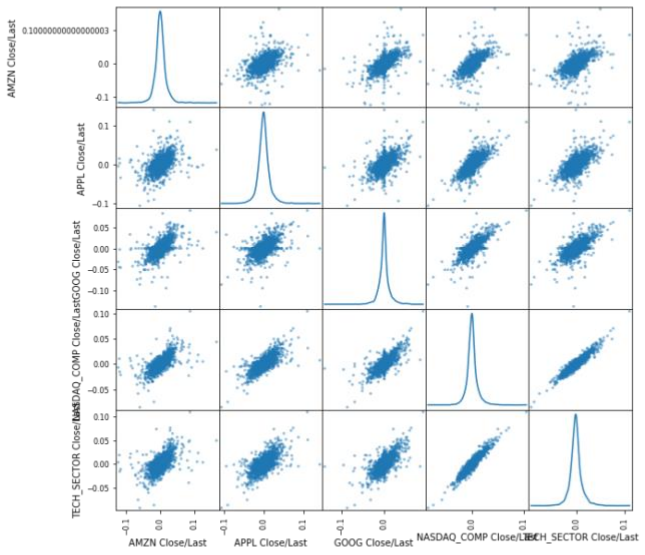
Data visualizations

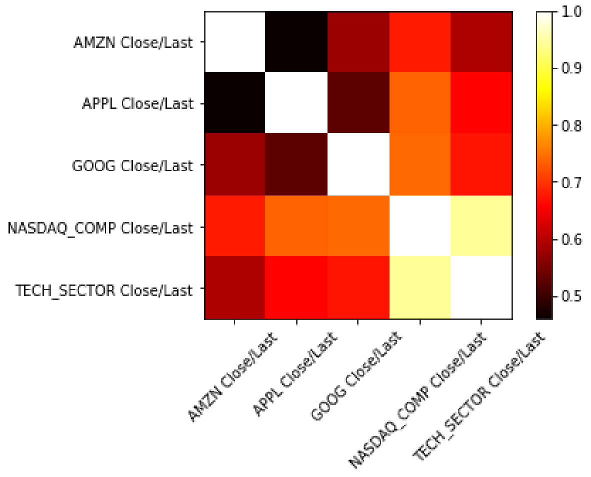
Competitor Analysis

Pandas provide the correlation function that can generate coefficients show how strong the relationship is and percentage change function, pct_change(), that can calculate stock return. The information from the completed data frame with these functions were utilized to analyze how one stock performed relative to other stocks, which are the same technology company, and how the stocks performed compared to their sector and the market. Therefore, the correlation value is used to see does one stock affect others or whether the sector and market affect those stocks. To further improve the analysis, the relationship is plotted as a scatter matrix to visualize possible correlations among stocks, the sector, and the market. Additionally, the heat map is used to visualize the correlation ranges. The lighter the color, the more correlated the two objects are.



	AMZN Close/Last	APPL Close/Last	GOOG Close/Last	NASDAQ_COMP Close/Last	TECH_SECTOR Close/Last
AMZN Close/Last	1.000000	0.459354	0.576645	0.677523	0.591000
APPL Close/Last	0.459354	1.000000	0.527448	0.737630	0.658339
GOOG Close/Last	0.576645	0.527448	1.000000	0.742302	0.674055
NASDAQ_COMP Close/Last	0.677523	0.737630	0.742302	1.000000	0.944660
TECH_SECTOR Close/Last	0.591000	0.658339	0.674055	0.944660	1.000000

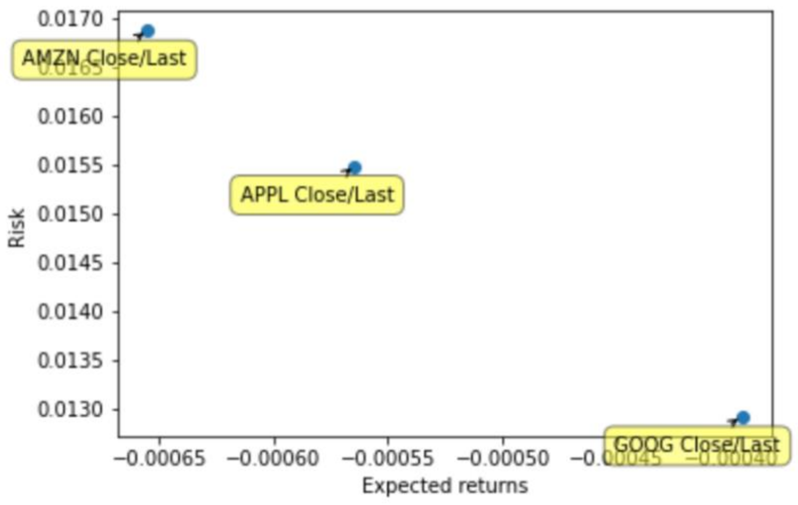




The analysis result found that the most positive correlations occurred between Google and Amazon returns compared with other stock relations. Google and Apple return also has the highest correlations with the NASDAQ index and technology sector among the stocks. In other words, the correlation coefficient ranges close to 1 showing two stocks or a stock and a benchmark market index are moving in the same direction. For visualization, the scatter matrix shows that most stocks' distributions have approximately positive correlations, and the heat map clearly shows that Apple and Google have strong correlation with the market and technology sector.

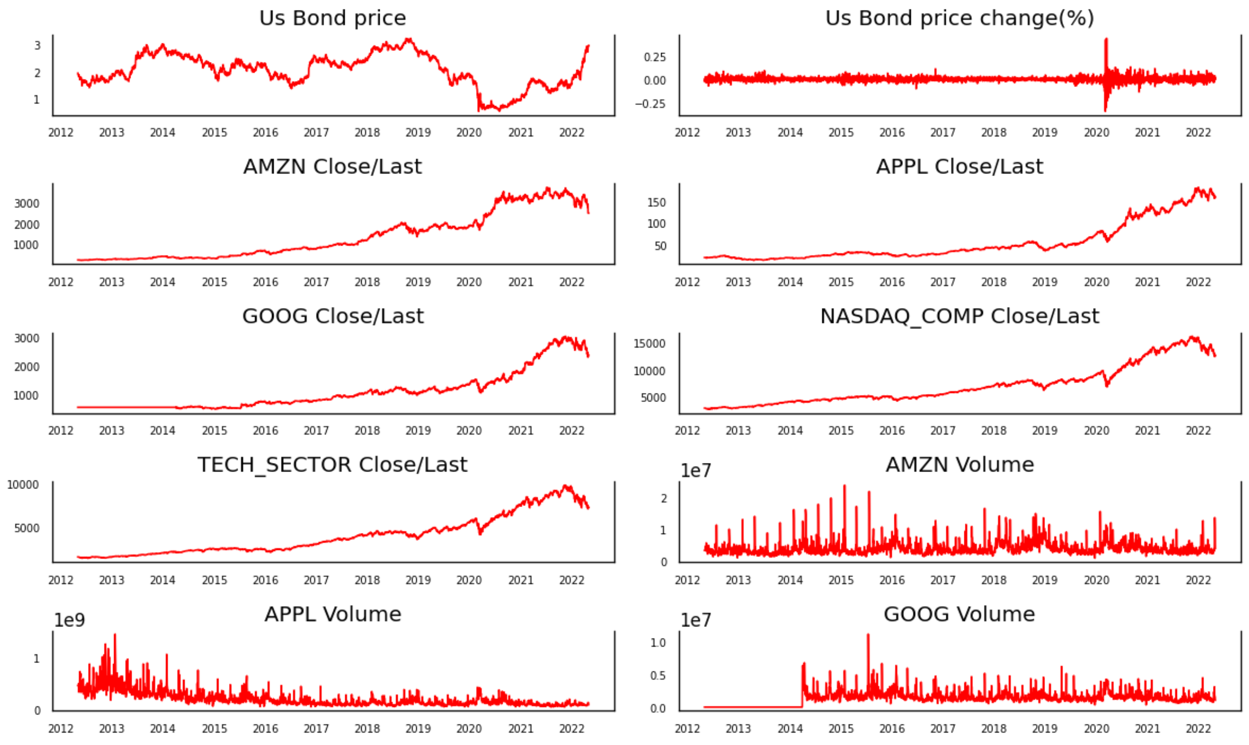
Stock and Return Analysis

The individual stock risk and return were analyzed and presented by the neat chart to understand which stocks are worth buying. In this analysis, the average of returns (Return Rate) and the standard deviation of returns (Risk) were extracted to plot in the graph. The chart of risk and return comparisons show that Google is attractive stock because it gives the highest return with the lowest risk.



Time Series Analysis

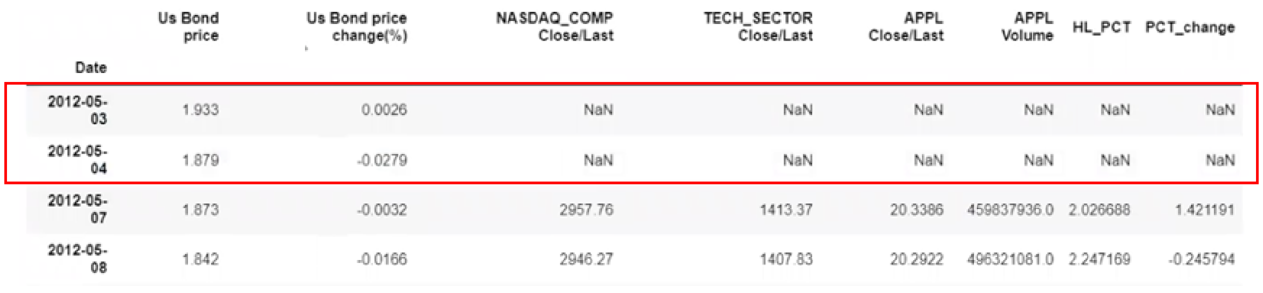
The stock market data need Time Series Plot because the data were collected over a period of time, and it can observe that the data are made at evenly spaced intervals throughout time (Kumar, 2022). The necessary libraries such as pandas and matplotlib.pyplot play an essential role in creating a visualization of this type of data. plt.subplot is used to create multiple subplots which generate a figure and a grid of subplots and also manage how the individual plots are created in the frame.



The multiple graphs above showed the closing value in ($) of the NASDAQ and NASDAQ 100 Technology sector indexes in relation to the year, the closing value in ($) of individual stocks in relation to the year and their volume, and the closing value of US bond price as a percent of the bond's face value in relation to the year and its percentage change over the period.

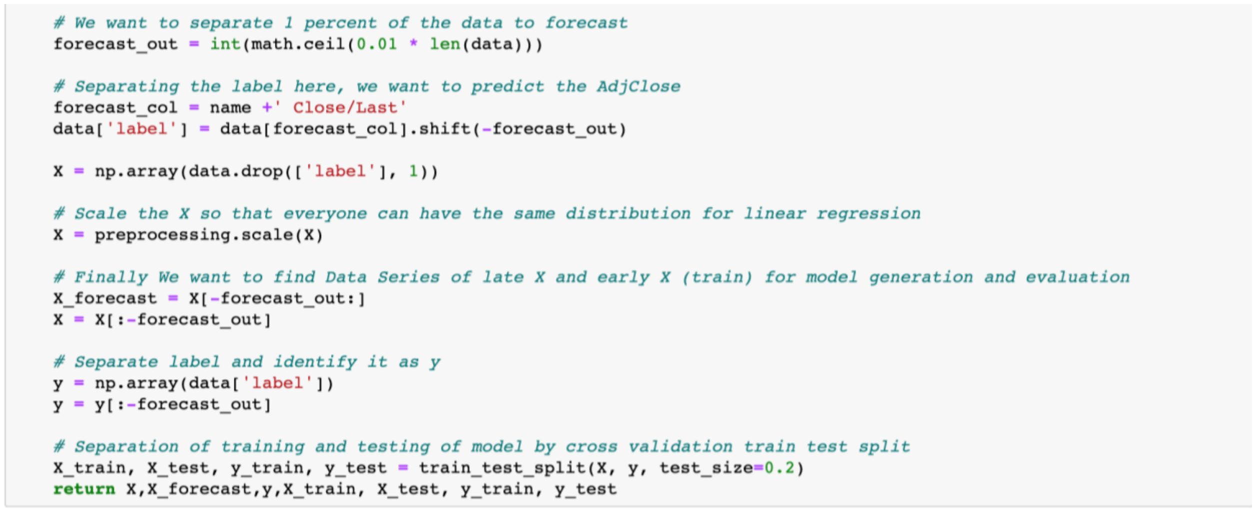
Predictive Models

For prediction part, the stock closing price of individual stock will be used which remark the final price in which the stocks are traded by the end of the day. Three models mentioned earlier (Simple linear regression, Quadratic Polynomial Regression and K Nearest Neighbor) were applied to predict the stock price and train for the accuracy of the models. The data frame applied to the machine learning models was adjusted to be specific for each stock before going through the training and prediction process because the previously completed data frame includes information from many stocks in one data frame. However, when data is recombined, it need to be filled the missing value by using interpolate again.

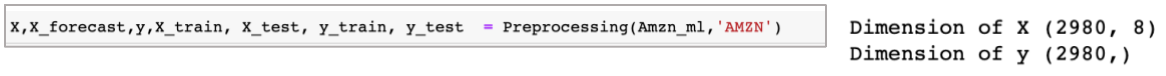


	Us Bond price	Us Bond price change(%)	NASDAQ_COMP Close/Last	TECH_SECTOR Close/Last	APPL Close/Last	APPL Volume	HL_PCT	PCT_change
Date								
2012-05-03	1.933	0.0026	NaN	NaN	NaN	NaN	NaN	NaN
2012-05-04	1.879	-0.0279	NaN	NaN	NaN	NaN	NaN	NaN
2012-05-07	1.873	-0.0032	2957.76	1413.37	20.3386	459837936.0	2.026688	1.421191
2012-05-08	1.842	-0.0166	2946.27	1407.83	20.2922	496321081.0	2.247169	-0.245794

Preprocessing



```
# We want to separate 1 percent of the data to forecast
forecast_out = int(math.ceil(0.01 * len(data)))

# Separating the label here, we want to predict the AdjClose
forecast_col = name +' Close/Last'
data['label'] = data[forecast_col].shift(-forecast_out)

X = np.array(data.drop(['label'], 1))

# Scale the X so that everyone can have the same distribution for linear regression
X = preprocessing.scale(X)

# Finally We want to find Data Series of late X and early X (train) for model generation and evaluation
X_forecast = X[-forecast_out:]
X = X[:-forecast_out]

# Separate label and identify it as y
y = np.array(data['label'])
y = y[:-forecast_out]

# Separation of training and testing of model by cross validation train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
return X,X_forecast,y,X_train, X_test, y_train, y_test
```



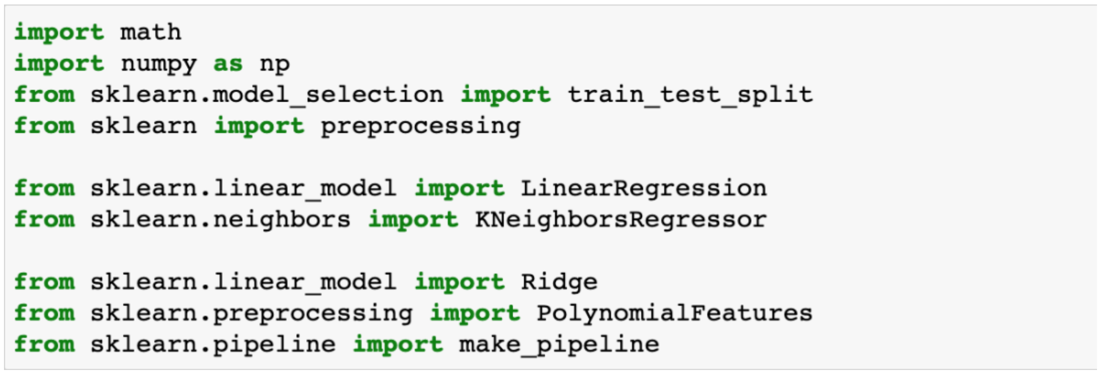
```
X,X_forecast,y,X_train, X_test, y_train, y_test  = Preprocessing(Amzn_ml,'AMZN')
```

```
Dimension of X (2980, 8)
Dimension of y (2980,)
```

1% of data was chosen to use in the prediction. In other words, there are totally about 3000 observations, and they were taken off around 30 days. The nature of the Time series prediction is that the previous data are used to describe the following data. If the data was taken too much, fewer data would be left to train with the model. As a result, there is an increasing chance of the model going wrong.

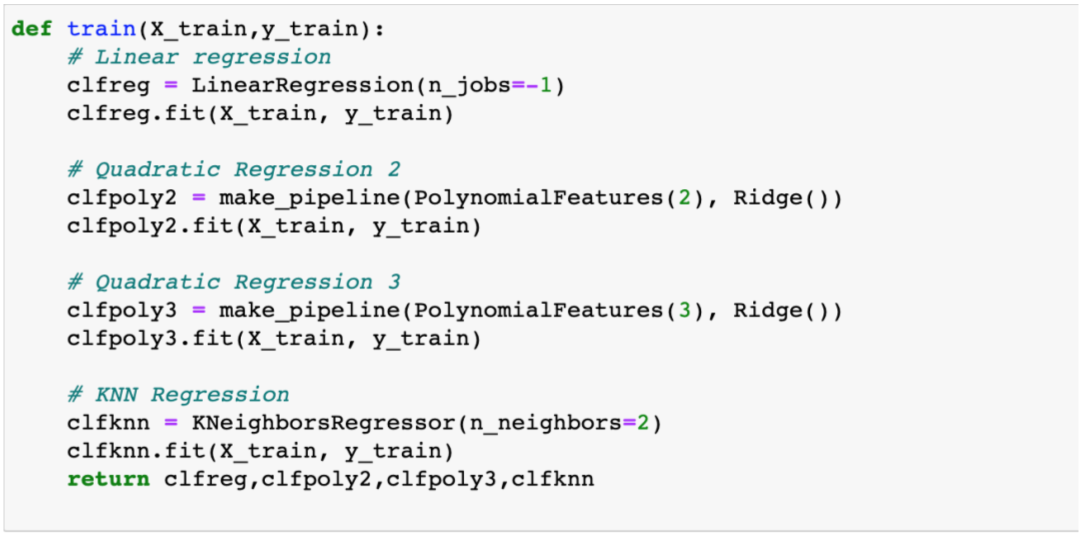
The stock price observation of approximately 30 days is used to predict the future stock price in the next one month. Therefore, the data have the amount of 3011, 2980 will be trained, and it was tested by predicting if for 30 days. Executing the function below allows the models to have 2980 data to train. However, the models need to be tested before the actual prediction that the data is divided to keep 20% (around 600 observations; about two years) to see how well the training performance can do.

The libraries imported to apply in machine learning session as following:

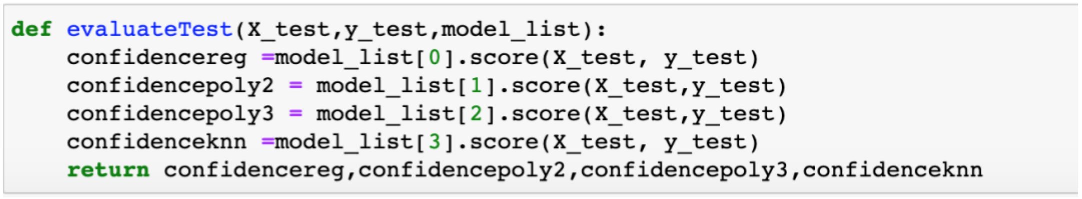


```python
import math
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor

from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

Next step, the train function is created to train the data with four models: Simple linear regression, Quadratic Polynomial Regression degree 2, degree 3, and K Nearest Neighbor. Once the training is complete, the result is evaluated with evaluateTest function to see the confidence value of the model or the discrepancies that occurred.

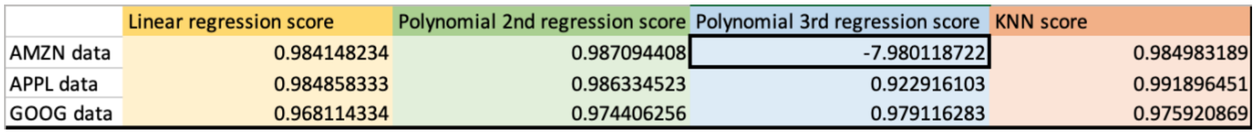


```python
def train(X_train,y_train):
    # Linear regression
    clfreg = LinearRegression(n_jobs=-1)
    clfreg.fit(X_train, y_train)

    # Quadratic Regression 2
    clfpoly2 = make_pipeline(PolynomialFeatures(2), Ridge())
    clfpoly2.fit(X_train, y_train)

    # Quadratic Regression 3
    clfpoly3 = make_pipeline(PolynomialFeatures(3), Ridge())
    clfpoly3.fit(X_train, y_train)

    # KNN Regression
    clfknn = KNeighborsRegressor(n_neighbors=2)
    clfknn.fit(X_train, y_train)
    return clfreg,clfpoly2,clfpoly3,clfknn
```



```python
def evaluateTest(X_test,y_test,model_list):
    confidencereg =model_list[0].score(X_test, y_test)
    confidencepoly2 = model_list[1].score(X_test,y_test)
    confidencepoly3 = model_list[2].score(X_test,y_test)
    confidenceknn =model_list[3].score(X_test, y_test)
    return confidencereg,confidencepoly2,confidencepoly3,confidenceknn
```

The result showed that Almost model used for training prediction stock price has minor discrepancies. A model's confidence values for each stock are about 95% or higher. However, the Quadratic Polynomial Regression degree 3 is not suitable to use with Amazon stock because it showed a negative number at around -7.98. In other words, this model did not fit with the data set.

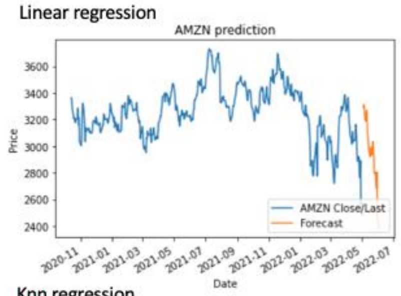


	Linear regression score	Polynomial 2nd regression score	Polynomial 3rd regression score	KNN score
AMZN data	0.984148234	0.987094408	-7.980118722	0.984983189
APPL data	0.984858333	0.986334523	0.922916103	0.991896451
GOOG data	0.968114334	0.974406256	0.979116283	0.975920869

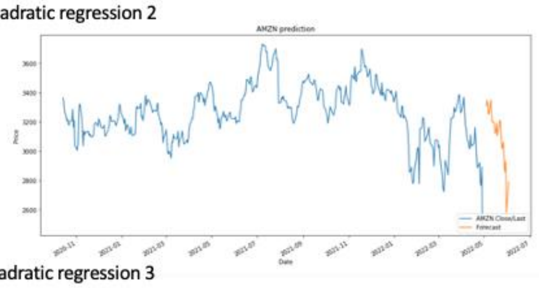
After training and evaluating stage, these models predict three stock prices ; AMZN, APPL, GOOG for 1 month later. The models used 1% of data extracted earlier for forecasting. The forecasted values were added over the last date for the next 30 days, as we can see in the stock price visualization as following:

Amazon stock prediction

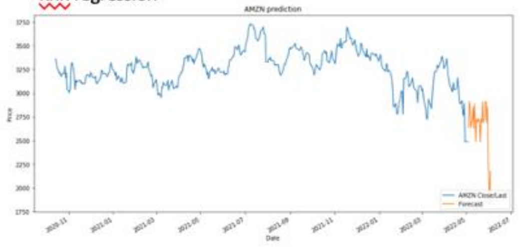
Linear regression



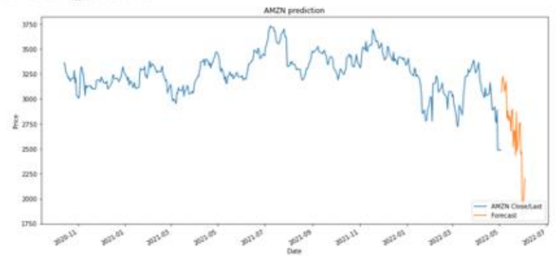
Quadratic regression 2



Knn regression

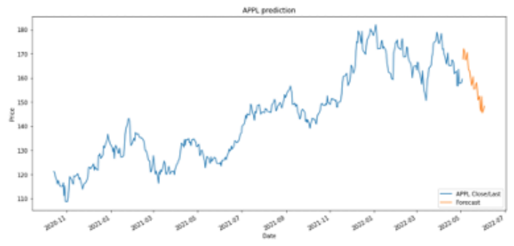


Quadratic regression 3

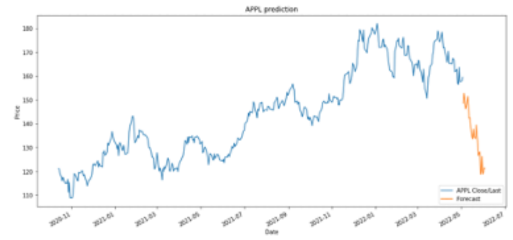


Apple stock prediction

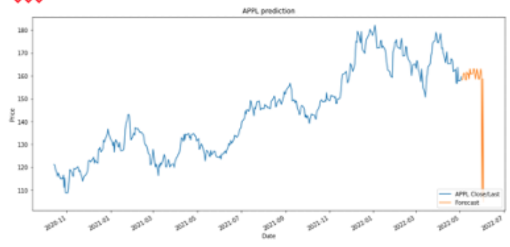
Linear regression



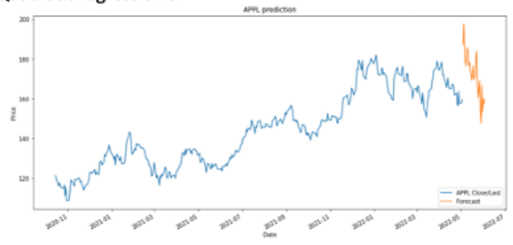
Quadratic regression 2



Knn regression

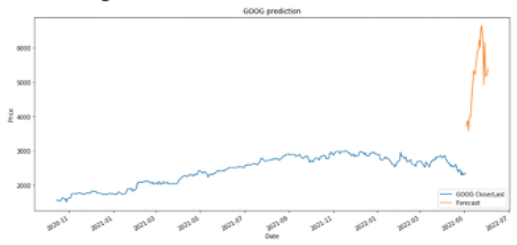


Quadratic regression 3

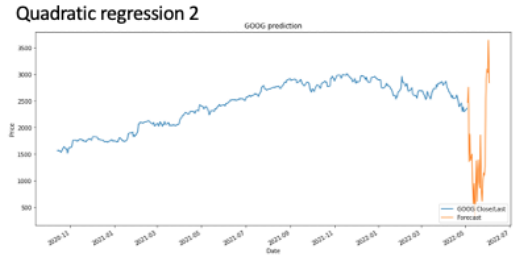


Google stock prediction

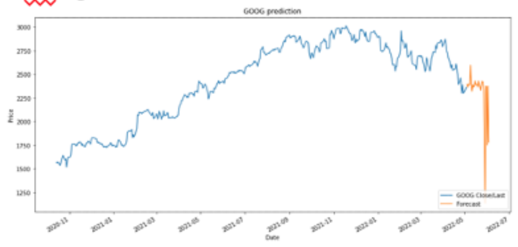
Linear regression



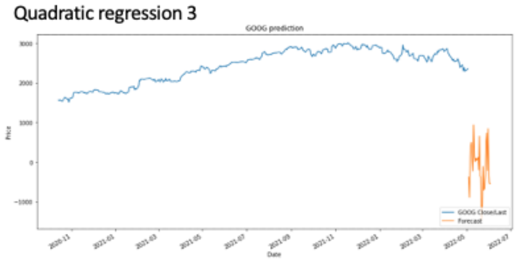
Quadratic regression 2



Knn regression



Quadratic regression 3



References

Kurama, V. (2022, February 1). Regression in machine learning: What it is and examples of different models. Built in. https://builtin.com/data-science/regression-machine-learning

Pythoninvest. (n.d.). *Macroeconomic Indicators Affecting Stock Market*. https://pythoninvest.com/long-read/macro-indicators-affecting-stock-market

Sharma, P. (2021, October 13). *Machine learning for stock market prediction with step-by step implementation*. Analyticsvidhya. https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/

Tatan, V. (2019, May 26). *In 12 minutes: Stocks Analysis with Pandas and Scikit-Learn.* Towardsdatascience. https://towardsdatascience.com/in-12-minutes-stocks-analysis-with-pandas-and-scikit-learn-a8d8a7b50ee7

Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). *Stock closing price prediction* using machine learning techniques. Precedia Computer Science. 167. 599-606.