Prathamesh Sai

Predicting Stock Market Prices Using Machine Learning

Using Python and Quandl

January 2021

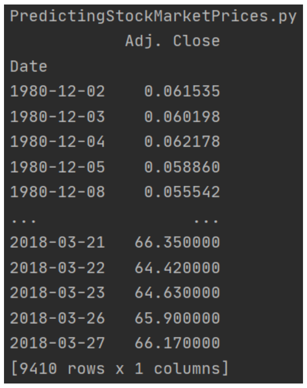
Quandl

The dataset used in this project is from Quandl [1], which is a marketplace for financial, economic and alternative data delivered in modern formats for today's analysts, including Python. This is an excellent resource to get real time data on the financial market. You can get data using an API, but Quandl is also a great way to access this data with ease using Python; I called this data using my API Key from my Quandl account. I then used the package manager for Quandle to produce this project.

I used the Quandl API, and the WIKI [2] data feed/service in particular. This contained data up to the 11th of April 2018.

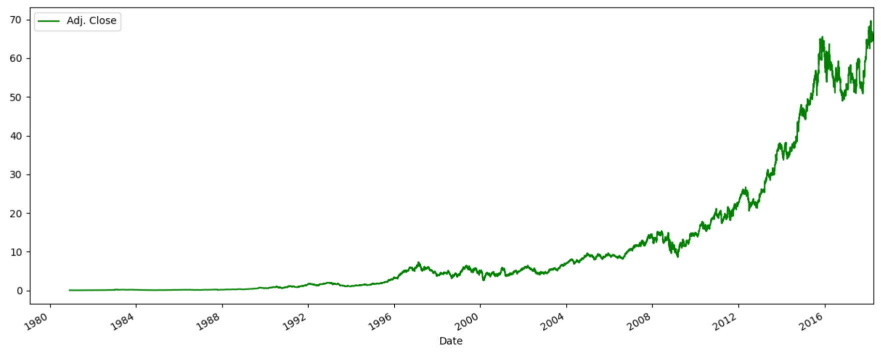
Using Specific Parts Of The Dataset

The stock which I used to predict was NKE ***(Nike)*** [3]. I used the "Adjusted Close" column from the dataset, because it accounts for inflation.



```
PredictingStockMarketPrices.py
            Adj. Close
Date
1980-12-02    0.061535
1980-12-03    0.060198
1980-12-04    0.062178
1980-12-05    0.058860
1980-12-08    0.055542
...                ...
2018-03-21   66.350000
2018-03-22   64.420000
2018-03-23   64.630000
2018-03-26   65.900000
2018-03-27   66.170000
[9410 rows x 1 columns]
```

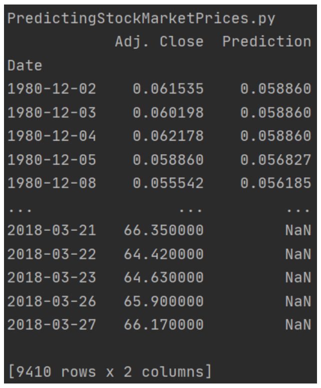
The adjusted close is the closing price after adjustments for all applicable splits and dividend distributions. With the help of Quandl, I had over 20 years of data for the "Adjusted Close" from 1998 to 2018.



Using Machine Learning To Predict Future Prices

To use machine learning, it was evident that I needed to format my test data; so I had to get the ”features” and ”labels” set up. In machine learning, in your dataset, you have something known as the ”features” which is the X variable i.e. the variables that are inputted into an imaginary function, in this case, a linear y = mx + c function. You also have the ”labels” which are the y resulting values.

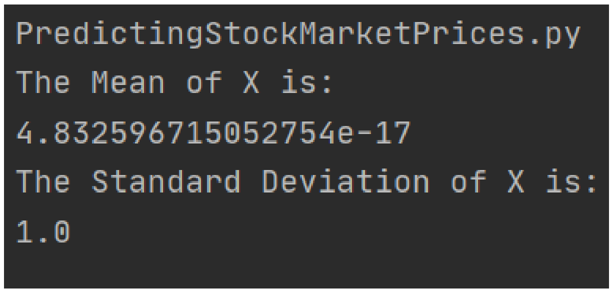
Therefore, I reformatted the data so that we'll take in the X input as a day 30 days prior to predict the stock price 30 days from a given X value. This means that I had to include an extra column in my data frame, that will shift the column of adjusted closed values up by 30; this allowed me to predict the price 30 days in the future.



```
PredictingStockMarketPrices.py
            Adj. Close  Prediction
Date
1980-12-02    0.061535    0.058860
1980-12-03    0.060198    0.058860
1980-12-04    0.062178    0.058860
1980-12-05    0.058860    0.056827
1980-12-08    0.055542    0.056185
...                ...         ...
2018-03-21   66.350000         NaN
2018-03-22   64.420000         NaN
2018-03-23   64.630000         NaN
2018-03-26   65.900000         NaN
2018-03-27   66.170000         NaN

[9410 rows x 2 columns]
```

It was important that I standardized the data that I was working with,because all the data has different ranges, different means, and different standard deviations. By standardizing the data, I ensured that our dataset has a mean of ***around 0***, and a standard deviation of ***exactly 1***. I used the module ”sklearn” [4] in Python to do this, and specifically imported ”preprocessing”.



```
PredictingStockMarketPrices.py
The Mean of X is:
4.832596715052754e-17
The Standard Deviation of X is:
1.0
```

Training and Testing Data

We need training data, and testing data because when we create a linear regression model, for example using a linear regression model using the module "sklearn" in Python, it will go ahead and train on your train data. Then, you can score and test the accuracy of your model by applying it on your test data.

A common technique in Machine Learning is to split your data into 80 and 20. 80% should be training data, and 20% should be testing data. To do this, I used the "sklearn" module and imported the "train_test_split" function.

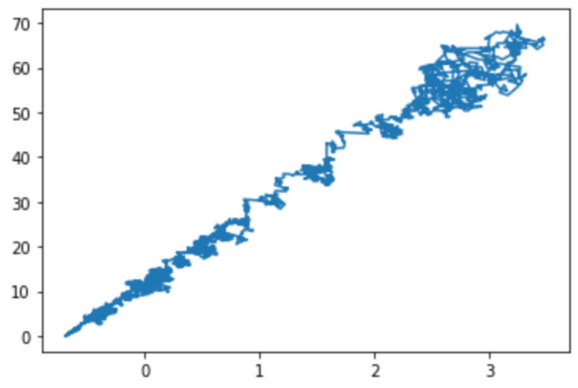
Linear Regression

A good way to illustrate how linear regression is used in this project is to plot the adjusted closed prices, by passing in the X and y variables.

```
plt.plot(X, y)
```

This provides this graph, which is essentially a scatterplot:

```
[<matplotlib.lines.Line2D at 0x1758fa7a310>]
```



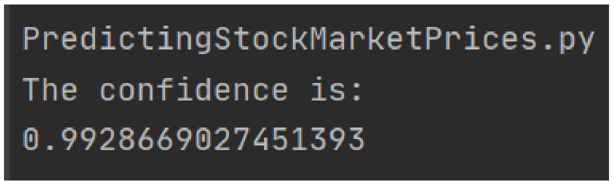
The goal of linear regression in this project is to find the line of best fit that minimizes the distance of each point to that line.

The model in this project will take the train data, and find the two parameters, the slope (m) and the y-intercept from the equation y=mx+c, and give us the line of best fit.

Calculating Confidence Scores

To see if our model is accurate, we score our data and get the ***confidence***, and store it into a variable. The value of ***confidence*** should be between 0 and 1 and therefore it should be close to 1 if it's an accurate model. The higher the value, the more we can trust the model.

```
confidenceScore = estimatorInstance.score(test_X, test_y)
print("The confidence is: ")
print(confidenceScore)
```



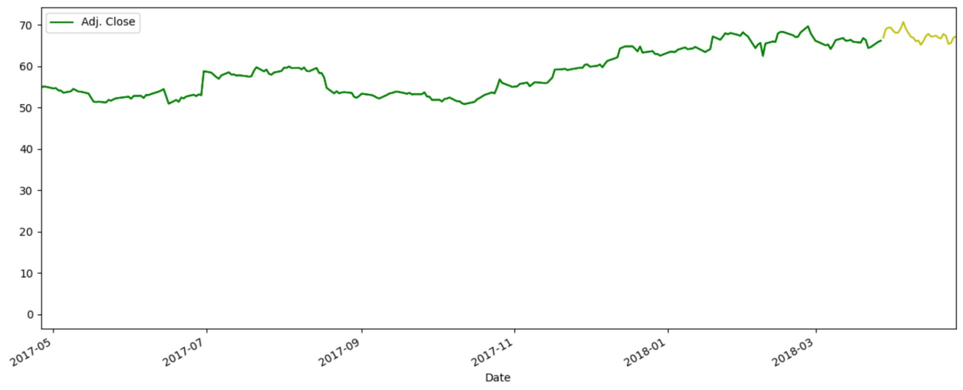
```
PredictingStockMarketPrices.py
The confidence is:
0.9928669027451393
```

Predicting Stock Prices

Now that we have confidence in our model, we can finally predict the stock prices in the future.

```
predicted_forecast = estimatorInstance.predict(forecast_X)
```

Then we can plot both the actual stock prices alongside our predicted stock prices on the same graph



The green line represents the actual prices, the yellow line represents our future predictions.

Conclusion

It is evident that Linear Regression is ***not the best model to predict stock market prices***, because stocks can be ***volatile and unstable***. However, this project ***shows the power of Machine Learning*** and how it can use past data to predict the prices of stable stocks that have been increasing at a stable rate.

References and Links

[1]. https://www.quandl.com/
[2]. https://www.quandl.com/databases/WIKIP/documentation
[3]. https://www.marketwatch.com/investing/stock/nke
[4]. https://scikit-learn.org/stable/