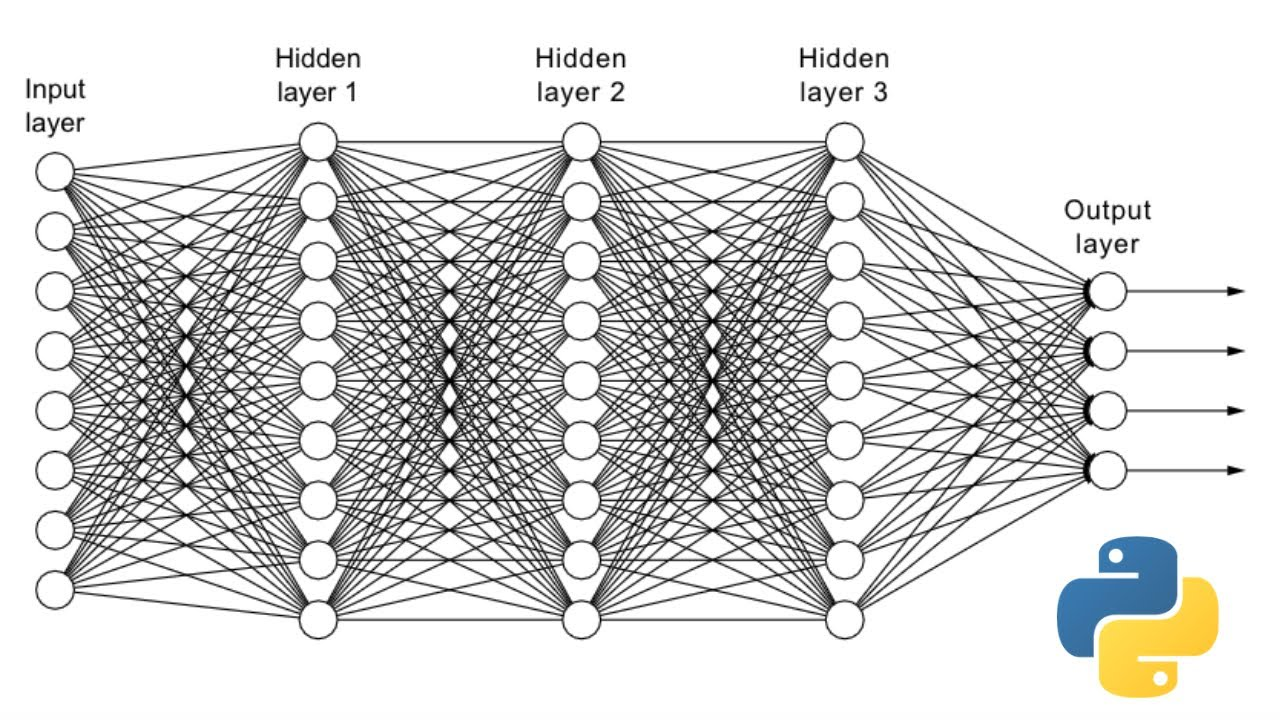
April 12, 2021

|  |  |
| --- | --- |
| **Group 6** | |
| Balsara, Sundeep | sbalsara |
| Erkaya, Selin | serkaya |
| Khan, Nalini | n242khan |
| McGillis, Riley | rmcgilli |
| Wang, Jing | wjing |
| Weigel, Reece | r2weigel |



**Data science 4 - Machine learning Predicting stock prices**

Table of Contents

[**Objective** 2](#_Toc68771479)

[**Data Preparation** 2](#_Toc68771480)

[Data procurement and processing 2](#_Toc68771481)

[Data engineering 2](#_Toc68771482)

[Data pre-processing 3](#_Toc68771483)

[Data visualization 3](#_Toc68771484)

[Challenges identified 4](#_Toc68771485)

[**Model Design** 4](#_Toc68771486)

[One stock, different models 4](#_Toc68771487)

[One model, different stocks 5](#_Toc68771488)

[**Model Evaluation** 5](#_Toc68771489)

[**Conclusions** 7](#_Toc68771490)

[**References** 9](#_Toc68771491)

[**Appendix** 10](#_Toc68771492)

# **Objective**

As more individuals personally invest in publicly traded stocks, there is a need to understand whether there is a predictability as to which stocks will have a high return. Our hypothesis for this research is that historical trends in price and other variables could reliably predict future stock prices through the right machine learning (ML) algorithm. This theory was tested using three different regression models to predict the stock price of Microsoft (MSFT), and additionally, we tested one of the models by seeing how well it predicted MSFT price compared with that of Tesla (TSLA). We will discuss the process and outcomes in this report, and the Python code is provided in a separate IPYNB file.

# **Data Preparation**

###### 

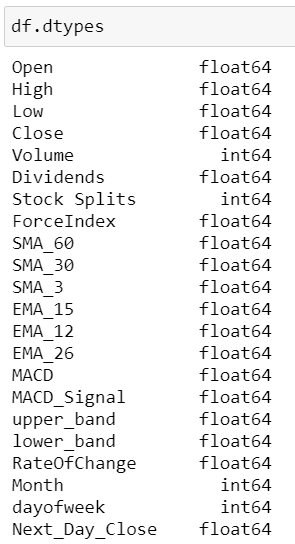
###### Data procurement

Open source daily stock data was downloaded from Yahoo Finance using Python’s yFinance library and imported into a pandas dataframe. The time series data with start date 2016 was clean and structured and no imputation was needed. The source data contained variables like opening price, volume traded, closing price, high and low price for the day.

###### 

###### Data engineering

To enrich the data, important stock-related features were engineered. These technical stock indicators included force index, moving averages (simple and exponential), moving average convergence divergence, rate of change. Please refer to the appendix for a description of the engineered technical stock features. The final list of features is shown in figure 1. Some of these calculations, averages for example, resulted in NaN values in the earlier timeframes, however, the algorithms did not use data older than 2017.



Original yFinance variables (7)

Engineered Features (15)

*Figure 1. Full List of Variables/Features*

###### 

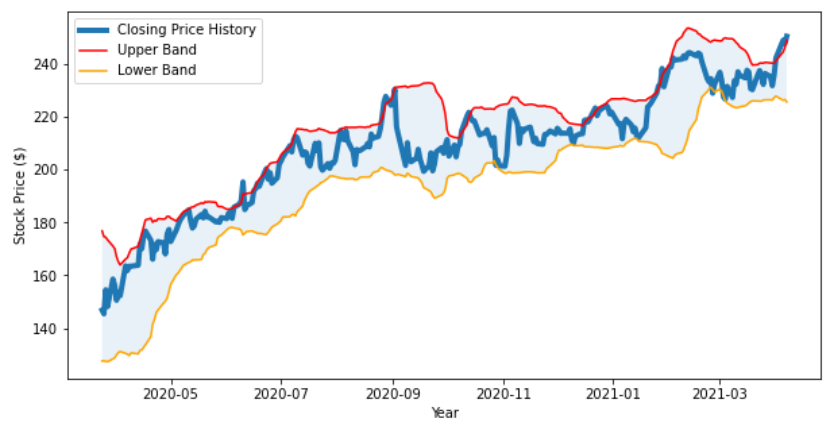
###### Data pre-processing

As mentioned, no imputation was needed as there were no missing values. RobustScaler and OneHotEncoder were used to scale and transform the data prior to running the ML algorithms. RobustScaler was chosen because it is not influenced by outliers in the sense that adding or removing outliers in the training set will yield approximately the same transformation. OneHotEncoder was used to encode the date variables into numeric types for use in the algorithms.

A ColumTransformer was used to prepare the pipeline for the linear and KNN regressors. For the neural net, a custom function was created to transform and sequence the data (the split\_sequence function is discussed in the Model Design section). After the pipeline was constructed, the data was divided into 20% for testing and 80% for training the algorithms. The predicted target variable was the current day’s closing price (“Close”).

###### Data visualization

Visualizations were created to examine how the data were distributed and how the different variables trended when plotted against one another. One additional trend graph that was created is the Bollinger Bands which is used in the trading industry to visualize how volatile the price is over a period. This is seen in figure 2. Please refer to appendix for more information on Bollinger Bands usage.



*Figure 2. Bollinger Bands*

###### 

###### Challenges identified

The only challenge faced in working with the data was ensuring the yFinance library was installed in order to read the data into the program. Otherwise, the data was ready for use once it was enriched and pre-processed.

# **Model Design**

The following describes how the two scenarios introduced above were implemented:

###### One stock, different models

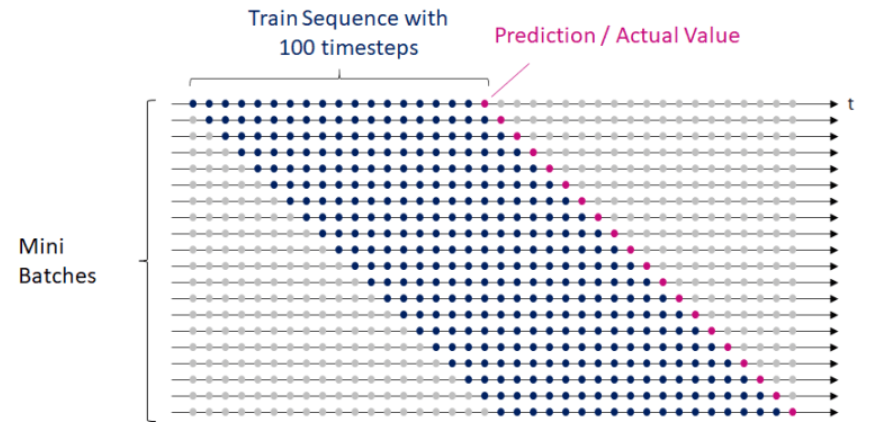
Three models were created to predict MSFT stock price: Linear Regression (LR)Model, k Neighbours Regressor (KNN) Model, and Neural Network (NN) Model. The models selected to conduct the experiments were based on their ability to predict time series data effectively.

The LR model from Python’s Scikit Learn Library followed the standard ordinary least squares (OLS) methodology and did not require use of hyperparameters.

The KNN algorithm, also from Scikit Learn, was trained and tested using GridSearchCV which allowed the model to find and use the best threshold for the n\_neighbours (nearest neighbours) hyperparameter.

The NN was developed using the Tensor Flow Library and the model consisted of two branches using Long Short-Term Memory (LSTM) and Dense inputs. The two-input structure was designed as follows:

Input 1 (Time Series Features) - Time Series data such as Open price, High, Low, Volume, was processed in mini batches by a LSTM cell. The data was sequenced in an additional preparation step using the split\_sequence custom function:



*Figure 3. Mini Time Series Batches for LSTM branch of NN*

Input 2 (Technical Indicators) – Features that were not time series data were processed using the dense branch of the neural net.

A function was built to perform hyperparameter tuning for the NN predictor. Standard parameters (epoch and batch size), along with a custom parameter for Look Back Period were tested and tuned, and the best results were shown for the following combination:

best Values:

Look back period: 92

Epochs: 50

BatchSize: 5

RMSE: 1.4731942090465557

###### One model, different stocks

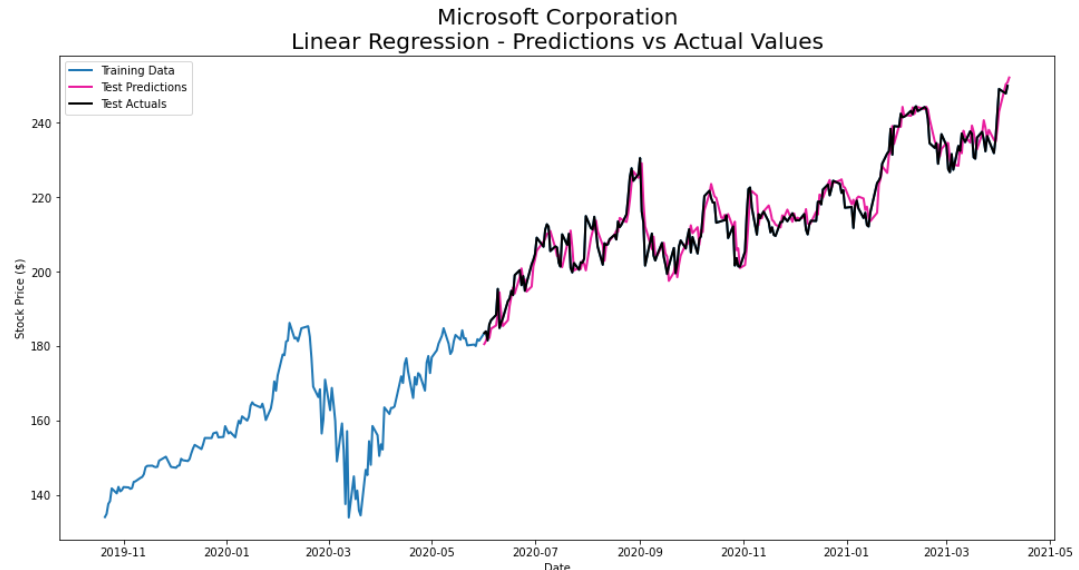
After the models were fitted and trained, the Close price values were predicted for MST. Then the NN model was run with the same input parameters but using TSLA historical data. All of the same preprocessing steps and functions were carried out on both sets of data, and the results were compared to see how well the NN was able to predict the prices across different stocks.

# **Model Evaluation**

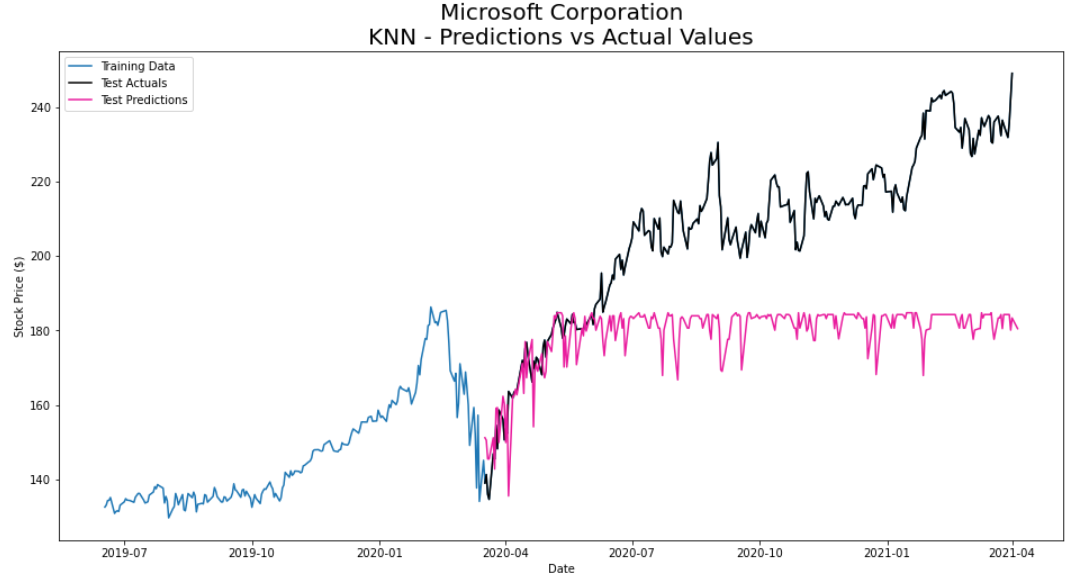
Evaluation of the models was assessed using root mean square error (RMSE). The Neural Net and Linear Regression models performed well on predicting the Close price, scoring very low RMSE values, Table 1. The output values are also visually displayed in the graphs where the Predicted and Actual True values for the test data appear to be superimposed on one another in the LR and NN models. Even with grid search, the KNN algorithm was not impressive and this could be attributed to the noise seen the historical training data.

|  |  |
| --- | --- |
| **Model Name** | **RMSE Value** |
| Linear Regression | 4.006 |
| KNN Regressor with GridSearchCV | 46.908 |
| Neural Network with LSTM and Dense | 1.4731 |

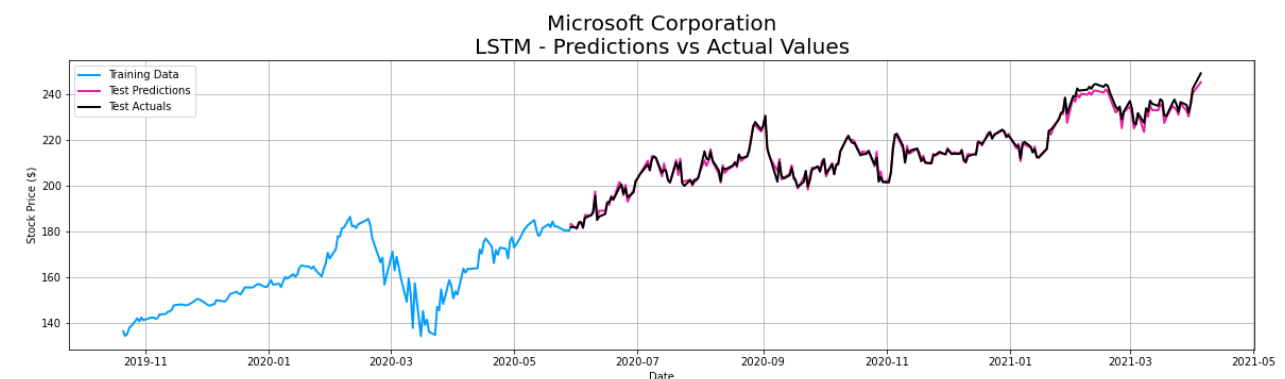
*Table 1. RMSE Values for Regression Models*



*Figure 4. Linear Regression for MSFT*



*Figure 5. K Nearest Regression for MSFT*

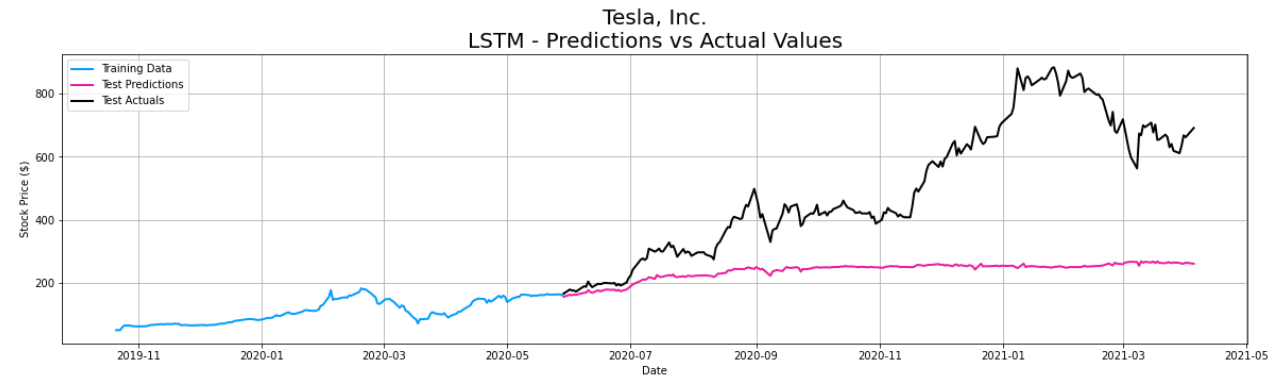


*Figure 6. Neural Net Prediction for MSFT*

The second way we assessed stock prediction was to use the top performing model, the neural net, to predict the stock price for TSLA. Comparison of the model in predicting MSFT and TSLA was made using a few different results from the regression. Using the same parameters as the MSFT stock does not result in a good model for TSLA. This demonstrates that there is not a single model that will work for all stocks and that each stock will need to be tuned and trained differently.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MAPE** | **MPE** | **RMSE** | **Diff to Actual** |
| MSFT | 0.51 | 0.13 | 1.47 | ~2% below last reported price |
| TSLA | 44.71 | 44.71 | 320.48 | ~60% below last reported price |

*Table 2. Neural Net Regression Values to Compare MSFT and TSLA*



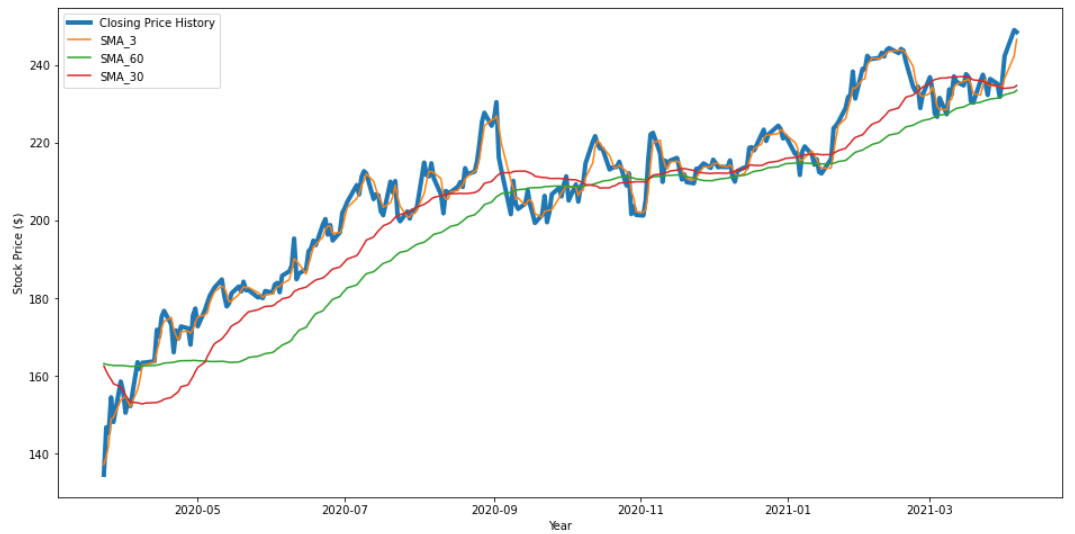
*Figure 7. Neural Net Prediction for TSLA*

# **Conclusions**

We were able to use the models to predict stocks where history was stable and where there was time to perform hyperparameter tuning. In our experiment, we have illustrated that the neural net proved to be the most accurate, closely followed by the linear regression model. K nearest regressor performed the worst in this setting.

Some steps we might have taken to improve KNN’s predictions are possibly narrowing down the inputs to the smoothed SMA\_30 or SMA\_60 variables, reducing the historical noisiness in the data, shown in Figure 8. Although the neural net reliably predicted MSFT, our thinking is that the TSLA stock would need to be trained on more recent history as the price has significantly inclined over the past two years (Figure 7).

Given the results we have seen, we believe models can be developed with more specificity and trained to predict stock prices in the future, helping to make the jobs of professional and amateur traders easier.



*Figure 8. MSFT Closing Price Historical Data – Actual and Smoothed*

# **References**

<https://www.ig.com/en/trading-strategies/10-trading-indicators-every-trader-should-know-190604>

<https://heartbeat.fritz.ai/classification-with-tensorflow-and-dense-neural-networks-8299327a818a>

<https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/bollinger-bands>

<https://www.investopedia.com/>

# **Appendix**

API - Application Programming Interface

The force index is used for trend and breakout confirmation, as well as spotting potential turning points by looking for divergences

Exponential Moving Average (EMA) is similar to Simple Moving Average (SMA), measuring trend direction over a period of time. However, whereas SMA simply calculates an average of price data, EMA applies more weight to data that is more current.

MACD is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA. When the MACD falls below the signal line, it is a bearish signal that indicates that it may be time to sell. Conversely, when the MACD rises above the signal line, the indicator gives a bullish signal, which suggests that the price of the asset is likely to experience upward momentum

Bollinger Bands use 2 parameters, Period and Standard Deviations. Bollinger Bands are envelopes plotted at a standard deviation level above and below a simple moving average of the price. Because the distance of the bands is based on standard deviation, they adjust to volatility swings in the underlying price. Tight bands mean low volatility, and vice versa.

RMSE- Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.

MAPE- The mean absolute percentage error (MAPE) is the percentage equivalent of MAE, the absolute difference between the data and the model’s predictions

MPE- Mean percentage error (MPE) is useful to us because it allows us to see if our model systematically underestimates (more negative error) or overestimates (positive error)