Stock Price Prediction using Machine Learning

Chris Lazarus clazar5@uic.edu
University of Illinois Chicago
Deepak Singhal dsingh39@uic.edu
University of Illinois Chicago
Sanjay Madesha smades2@uic.edu
University of Illinois Chicago

Abstract

The stock market is one of a country's most important economic sectors. Stock price prediction is difficult due to its non-stationary character. The goal of this paper is to discover the most accurate approach for predicting stock closing prices using a study of machine learning techniques in comparison. Machine learning techniques used in the study includes Linear Regression, K Nearest Neighbors (KNN), Support Vector Machine (SVM) and Long Short Term Memory (LSTM)Methods. Furthermore, a comparison study between machine learning methodologies have been made using performance evaluation metrics of root mean square error and r-square. Based on the rmse and r-square value SVM emerged to be the best prediction model for our stocks dataset and the most accurate method for predicting stock prices.. However for a very large dataset, neural network models would be a better fit taking into consideration the dependencies of our problem.

Introduction

Stock price prediction has always attracted people interested in investing in the share market and stock exchanges because of the direct financial benefits. However, prediction of stock market returns is a very complex issue since stock prices are affected due to many reasons like company related news, political, social economic conditions and natural disasters. Recently, the interest in applying Artificial Intelligence in making trading decisions has been growing rapidly with numerous research papers published each year addressing this topic. Considering the complexity of financial time series data, combining machine learning with financial market prediction is regarded as one of the most exciting topics of research.

The dawn of the COVID – 19 pandemic caused the stock markets to go on both bullish and bearish runs spanning over different periods of time which allowed some people to make huge profits and some

to incur huge losses. Our project tries to predict the stock prices of different multinational companies by using different machine learning and deep learning models. The dataset consists of stock prices of companies listed in the S&P 500 list for the past 5 years. The features of the stocks are Date, Opening price, Day High, Day Low, Close price, Volume, and Name of stock. The prediction of stock prices is done by making use of historical stock data. In addition to purchasing and selling stocks, each stock is not only characterized by its price, but also by other variables such as closing price which represents the most important variable for predicting next day price for a specific stock.

We start off with one of the most basic machine learning models for a regression problem – Linear Regression which acts as our baseline model due to its simplicity and ease of implementation. From there, we move to other models in the increasing order of complexity starting from KNN and then moving to Support Vector Machines (SVM) and Long Short-Term Memory (LSTM) which is a deep learning model. Each of the models have their own advantages and disadvantages when compared based on speed, ease of use, performance on large and small datasets, interpretability etc. To find which model suits the best for the task in hand, the predictions from all 3 models are compared using model evaluation metrics – Root Mean Square Error (RMSE) and R^2 value.

Related work

With the introduction of technological breakthroughs such as worldwide digitalization, stock market forecasting has entered a technologically enhanced era, reviving the traditional trading methodology. Stock trading has become a hub of investment for many people as market capitalisation continues to rise. Economic investors, several analysts and academics have devised tools and methods for predicting the future. Stock value changes and assists investors in making informed decisions. Trading models that are more advanced using non-traditional textual data from social platforms to forecast the market. The use of advanced machine learning techniques like text data analytics and ensemble learning have helped the cause. The accuracy of prediction has substantially improved thanks to these new methods.However, due to the volatile nature of stock markets, research in this subject remains one of the most difficult.

Traditional machine learning algorithms

Payal Soni, Yogya Tewari and Prof. Deepa Krishnan. Their work treated the stock price as a time series, avoiding the problems that the model faced during the training phase. Their article talks about employing normalized data and a Recurrent Neural Network model to make forecasts that are very close to the actual values, indicating that machine learning techniques are the best for forecasting stock prices.

Involving current news trend and sentiment analysis

Zhaoxia WANG, Seng-Beng HO and Zhiping LIN wrote an article on their work which examines the relationship between stock price and news sentiments in order to accurately predict stock price. The effect of news sentiments is taken into account in a revolutionary augmented learning-based system for stock price prediction. The efficiency of this enhanced learning-based method is demonstrated utilizing a real stock price data set with an improvement in performance in terms of minimizing the Mean Square Error when compared to previous learning-based methods (MSE)

Using a comprehensive deep learning system
Jingyi Shen and M. Omair Shafiq have explicitly worked on stock price prediction using deep learning systems. Their suggested approach was comprehensive because it incorporated stock market dataset pre-processing, several feature engineering techniques, and a proprietary deep learning-based system for stock market price trend prediction. They conducted extensive tests on commonly used machine learning models and found that their proposed solution outperforms due to the extensive feature engineering that they implemented. Their key contribution is a comparison of multi-layer perceptron (MLP) and SVM, which indicated that SVM outperformed MLP in the majority of scenarios, while the result was also influenced by different trading methods.

On addition we took two more papers for our study to understand the LSTM in a more profound manner and understand how it works with the parameters.

In Stock Closing Price Prediction using Machine Learning Techniques, Mehar, Vijha, Deeksha, Chandolab and Vinay Anand Tikkiwala Arun Kumar used Artificial Neural Network and Random Forest techniques. Also for the evaluation of models used RMSE and MAPE. They showed that these two indicators have low values, indicating that the models are good at predicting stock closing prices.

In Stock Price Forecasting by a Deep Convolutional Generative Adversarial Network Alessio Staffini To deal with the challenge of anticipating stock closing prices, He developed a Deep Convolutional Generative Adversarial Network architecture which conducted both single step and multi step forecasting.

Model Design and Implementation

As mentioned above, we plan to use KNN, Linear Regression, SVM and LSTM models to make predictions of the stock prices.

Linear Regression

Linear Regression is a statistical procedure for calculating the value of a dependent variable from an independent variable. Linear regression measures the association

between two variables. It is a modeling technique where a dependent variable is predicted based on one or more independent variables. The linear regression analysis uses the mathematical equation, i.e., y = mx + c, that describes the line of best fit for the relationship between y (dependent variable) and x (independent variable). While correlation provides a quantitative way of measuring the degree or strength of a relation between two variables, regression analysis mathematically describes this relationship. Due to simple implementation and its good response to linearly varying data it was chosen as the baseline model for our project based on which the other models will be evaluated.

In this project, we have implemented the moving averages algorithm to predict the closing prices of stock of Apple shares. We have used the past 3 days and 9 days average price as two dependent variables. Hence the equation will look something like this:

$Y = A_1X_1 + A_2X_2 + B$

where, Y is the dependent parameter i.e., the closing price. X_1 is the average of the past 3 days' price, X_2 is the average of the past 9 days. A_1, A_2 are the slopes of the line for X_1 and X_2 respectively and B is the intercept. We kept the training testing proportion as 80-20.

Code Snippet:

```
from sklearn.linear_model import LinearRegression
linear = LinearRegression().fit(X_train,y_train)

model = linear.fit(X, y)

print('Intercept:', model.intercept_)
print('Coefficients:', model.coef_)
```

```
Intercept: 0.3255684517967552
Coefficients: [ 1.21289267 -0.21492898]
```

Fig 1. Code Snippet for linear regression

Model Validation: After building the model it is important to validate its performance. We can evaluate a model by looking at its **coefficient of determination** (R^2), **F-test**, and **t-test**. Before we continue, we will rebuild our model using the statsmodel library with the OLS() function. Then we will print the model summary using the summary() function on the model. The model summary contains lots of important values we can use to evaluate our model.

K Nearest Neighbors (KNN)

K Nearest Neighbors (KNN) is a simple algorithm that stores all available cases and predicts the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition as a non-parametric technique.

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors.

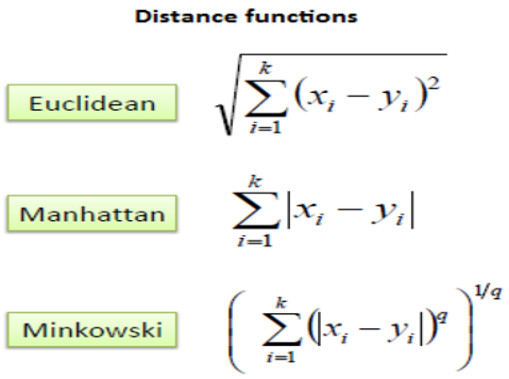

Fig 2. Distance functions

The above three distance measures are only valid for continuous variables which is the case in our project.

KNN is often referred to as a lazy learner. This means that the algorithm does not use the training data points to do any generalizations. In other words, there is no explicit training phase. Lack of generalization means that KNN keeps all the training data. It is a learning algorithm because it doesn't assume anything about the underlying data.

```
# Training the model with K = 3 value and prediciting the y labels for test data
k = 3
model_final = neighbors.KNeighborsRegressor(n_neighbors = k)
model_final.fit(x_train, y_train)  #fit the model
pred_final=model_final.predict(x_test) #make prediction on test set
error_final = sqrt(mean_squared_error(y_test,pred_final)) #calculate rmse
rmse_val.append(error_final) #store rmse values
print('RMSE value for k= ' , k, 'is:', error_final)
r2 = r2_score(y_test, pred_final)
print("R square for k=", k, "is" ,r2)
```

```
RMSE value for k=  3 is: 0.9391094574200999
R square for k= 3 is 0.9990785071771179
```

Fig 3. Code snippet for KNN

In our project, we took the **opening, high and low price** of the stock as the independent parameters to predict the closing price of the Apple share. These parameters were normalized to avoid the influence of any singular parameter on the result. We used **Euclidean distance** to find the similarity between datasets. There is one hyperparameter: **K value** in KNN which needs to be tuned for our dataset.

Support Vector Machine (SVM)

Support Vector Machine (SVM): In machine learning, Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. In SVM, the straight line that is required to fit the data is referred to as hyperplane. Unlike Linear Regression, SVM and SVR give flexibility of defining how much error is acceptable in our model using C. The standard way of writing an SVM model is $z(x) = w^T x + b$ where w^T is equivalent to $\theta^T 1, 2, .. n$ and b is the value of θ^T when n=0. In general, we'd like to construct a hyperplane that is consistent with classifying the data while committing the least amount to the specific training dataset - to avoid overfitting. We want to have the most space possible between the decision boundary and the data points on each side of the line in order to increase the total confidence in our predictions**.** This "space" is called the margin. By maximizing the size of this margin, we can build an optimal classifier.

Support Vector Regression (SVR) is a supervised learning algorithm that is used to predict discrete values. SVR uses the same principle as the SVMs. In general, SVR is

quite similar to SVM, but there are some three notable differences.

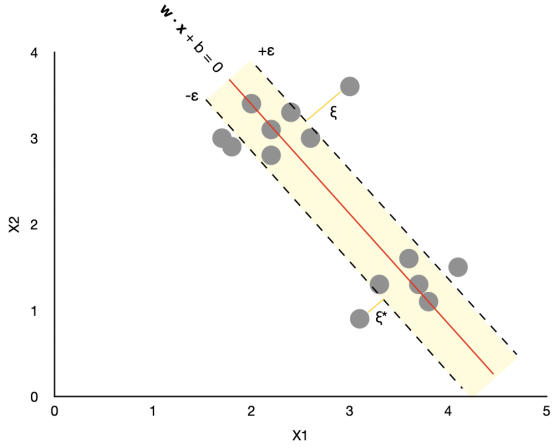

Fig 4. Illustration of SVM

1) SVR has an additional tunable parameter ε (epsilon). The value of epsilon determines the width of the tube around the estimated function (hyperplane). Points that fall inside this tube are considered as correct predictions and are not penalized by the algorithm. 2) The support vectors are the points that fall outside the tube rather than just the ones at the margin. 3) Finally, "slack" (ξ) measures the distance to points outside the tube, and you can control how much you care about it by tuning a regularization parameter C. The fit time complexity of SVR is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 10000 samples.

Long short-term memory (LSTM)

Long short-term memory (LSTM) was created as the solution to short-term memory. It has internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Core concepts of LSTM are the cell state, and its various gates. The cell state acts as a transport highway that transfers relative information all the way down the sequence chain. It acts as the memory of the network. The cell state can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information gets added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

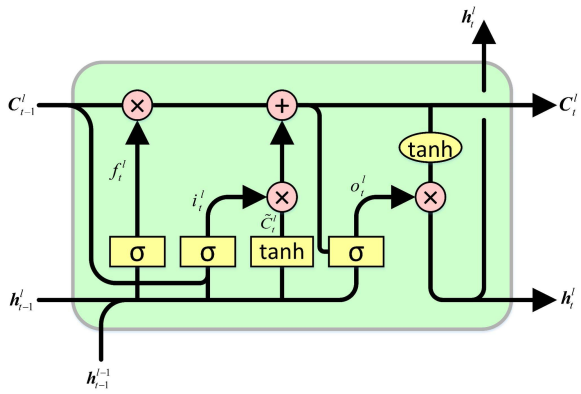

Fig 5. Gate structure of LSTM

$$f_t = \sigma_g\ (W_f \times\ x_t + U_f \times h_{t-1} + b_f)$$
$$i_t = \sigma_g\ (W_i \times\ x_t + U_i \times h_{t-1} + b_i)$$
$$o_t = \sigma_g\ (W_o \times\ x_t + U_o \times h_{t-1} + b_o)$$
$$c'_t = \sigma_c\ (W_c \times\ x_t + U_c \times h_{t-1} + b_c)$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$$
$$h_t = o_t \cdot \sigma_c(c_t)$$

f_t *is the forget gate*
i_t *is the input gate*
o_t *is the output gate*
c_t *is the cell state*
h_t *is the hidden state*

Also, Wf, Wi, Wo, Wc, Uf, Ui, Uo, Uc are the weight matrices and bf, bi, bo, bc are our biases.

The forget gate determines what information requires attention and what information can be ignored. The sigmoid function is used to transmit information between the current input X(t) and the hidden state h(t-1). Sigmoid generates values ranging from 0 to 1. It determines if a portion of the previous output is required (by giving the output closer to 1). The cell will utilize this value of f(t) for point-by-point multiplication later.

To update the cell status, the input gate performs the following processes. The second sigmoid function receives the current state X(t) and the previously hidden state h(t-1). The values are changed from 0 (important) to 1 (not important) (not-important).

The tanh function will then be used to pass the identical information from the hidden state and current state. The tanh operator will construct a vector (C(t)) with all the possible values between -1 and 1 to regulate the network. The activation functions generate output values that are ready for point-by-point multiplication.

The forget gate and input gate have provided enough information to the network. The information from the new state must then be decided and stored in the cell state. The forget vector f is multiplied by the prior cell state C(t-1) (t). If the result is 0, the values in the cell state will be dropped. The network then executes point-by-point addition on the output value of the input vector i(t), updating the cell state and giving the network a new cell state C. (t).

The value of the next hidden state is determined by the output gate. This state stores data from earlier inputs.

The current state and previous concealed state values are first sent to the third sigmoid function. The tanh function is then used to construct a new cell state from the old cell state. Both of these outputs are multiplied by a factor of two. The network selects which information the hidden state should convey based on the final value. Prediction is based on this concealed state.

The new cell state and hidden state are then passed forward to the next time step.

Finally, the forget gate selects which past stages' relevant information is required. The input gate determines what relevant information from the previous phase can be contributed, while the output gates complete the next hidden state.

Pseudo code for LSTM Model:

```
model = Sequential()
# Adding a LSTM layer with 5 internal units
model.add(LSTM(5,input_shape=(None,1),a
ctivation='relu'))
# Adding a Dense layer with 1 unit.
model.add(Dense(1))
# Loss function + optimizer
model.compile(loss='mean_squared_error',o
ptimizer='adam')
history =
model.fit(X_train,y_train,validation_data=(
X_test,y_test),epochs=100,batch_size=5,ver
bose=1)
```

Dataset and Exploratory Data Analysis

The dataset consists of information related to the stock prices of stocks listed in the S&P 500 list for the past 5 years. For exploratory data analysis and the future modeling process we focus on only the stock prices of Apple. To observe the trend, we generate a new column named price which is the average of all the prices which account for a stock i.e. open, close, high and low.

Firstly we start our data analysis by detecting null values for which we used the Missingno library. Missingno is a Python library that is Pandas compatible and which visualizes the distribution of NaN values.

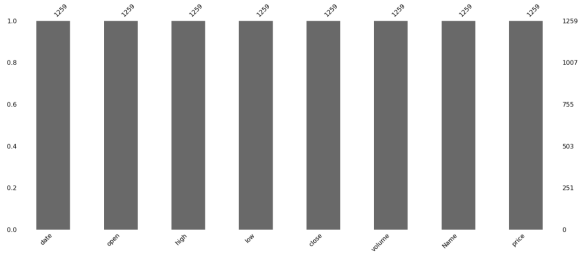

Fig 6. Distribution of values to detect NULL

Through this particular graph below, since all the bars are complete and they are identical in nature we can conclude that we have no null values.

For observing the overall trend and looking at the outliers. We use histogram and line graph. As seen from the distribution, the data approximately follows a normal distribution. The line chart shows the increasing trend in the stock prices of Apple.

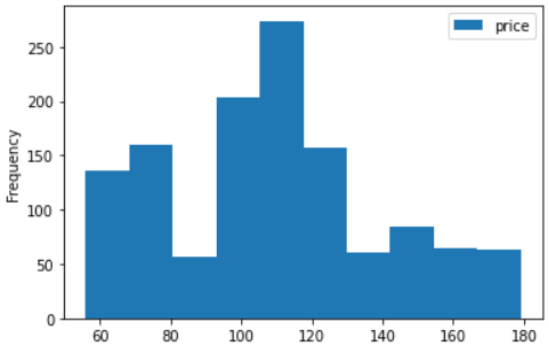

Fig 7. Histogram of price variation

Next, we calculate the log difference. The advantage of this computation of evaluating change is that the computations are symmetrical in both directions.

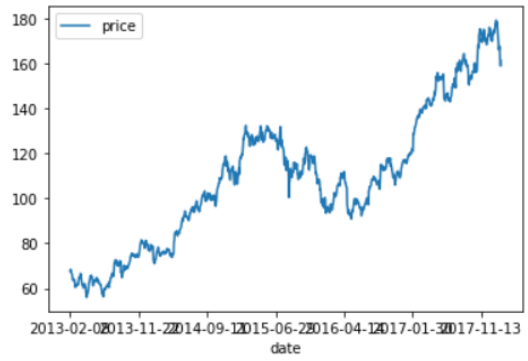

Fig 8. Scatter plot of price variation

For more detailed understanding we will plot a graph of the logarithmic difference to understand the symmetry of our data.

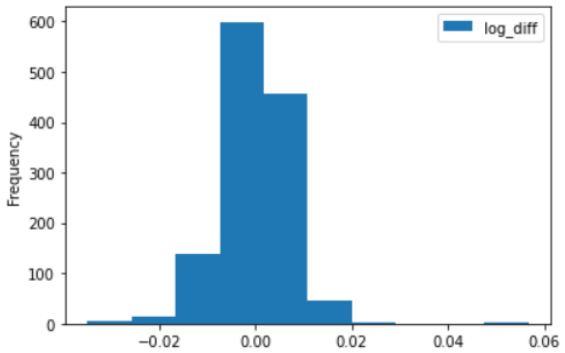

Fig 9. Symmetry of price data

Next we will build a correlation matrix using a heat map which can seen as follows:

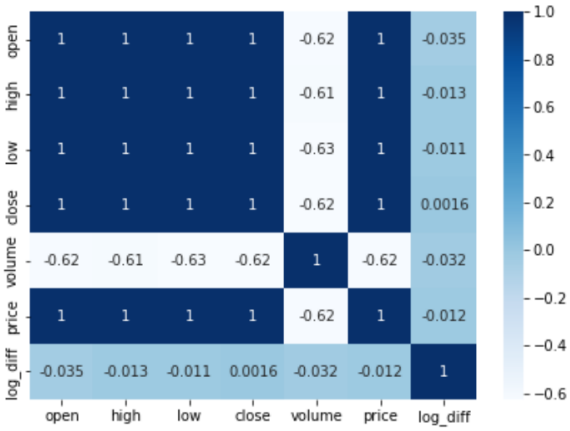

Fig 10. Correlation matrix

On observing, we see how much influence the other parameters have on the stock price and each other. To understand the parameters even further we compute the differences among these parameters and plot them on a heat map again.

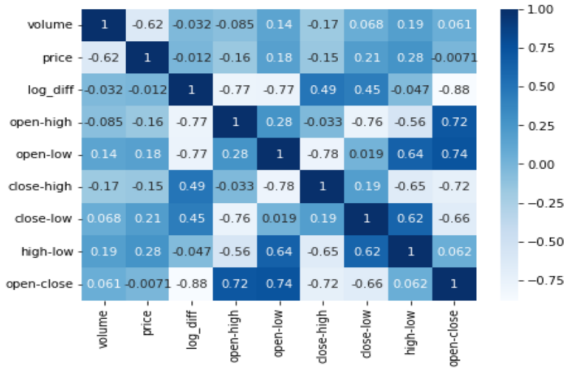

Fig 11. Correlation matrix for the calculated parameters

We see that the parameters open, high and low have a greater correlation value. Therefore, we use these parameters to analyze our dependent price in the future. The volume feature has the highest negative correlation with the open-high. This means that the wider the gap between a stock's opening value and its greatest value on a given day, the more stocks will be traded that day. Because there is no information on the number of buyers and sellers, we can only speculate that a large gap between the stock's opening and peak value will attract more buyers. To understand this trend we plot graphs to see how the price fluctuations vary according to the computed parameters.

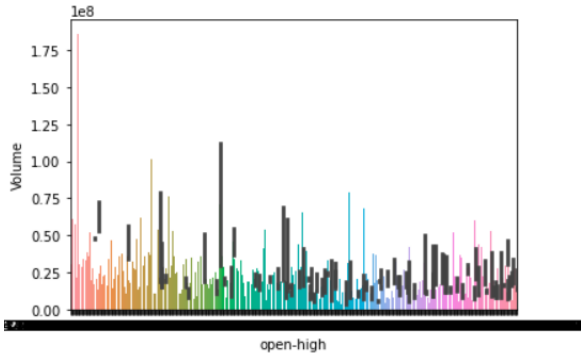

Fig 12. Fluctuations in opening and highest price difference

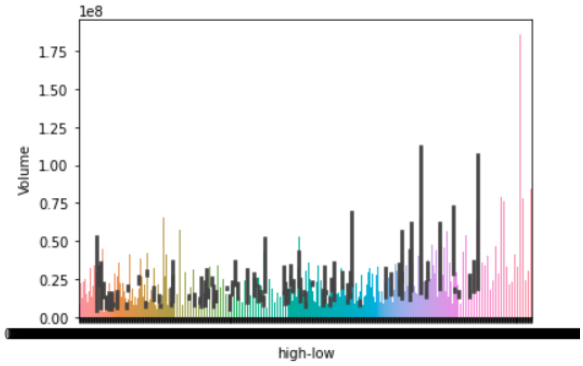

Fig 13. Fluctuations in highest and lowest price difference

Results

Baseline model : Linear regression

Since we need to predict prices of stocks using the historical data, we make use of the moving averages concept and create two different moving average variables. One which stores the moving average of the stock from the last 3 days, and another which stores the moving average of the stock from the last 9 days. Using these newly created variables, we predict the prices of the stocks by fitting a linear regression line. The prediction from the linear regression model closely resembles the actual values in the test dataset as seen in the figure below.

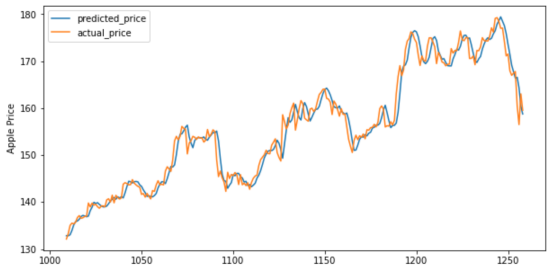

Fig 14. Results for Linear regression

After data modeling, the results were:
A1 = 1.213 (Coefficient for average of past 3 days)
A1 = -0.214 (Coefficient for average of past 9 days)
B = 0.32557 (Intercept)
Hence, the final equation is:
$Y = 1.213X_1 - 0.214X_2 + 0.32557$

The Root Mean Square error (RMSE) and R^2 metric are used as an evaluation metric to check the performance of the model. For an ideal model, the RMSE value would be zero indicating perfect prediction. The Linear regression model gives a RMSE score of 2.24 and R^2 value of 0.97. We got R^2 score of about 0.97 which means 97% of our dependent variable can be explained using our independent variables.

F test (Anova)

```
print('F-statistic:', olsmod.fvalue)
print('Probability of observing value at least as high as F-statistic:', olsmod.f_pvalue)
```

```
F-statistic: 154271.44191808262
Probability of observing value at least as high as F-statistic: 0.0
```

Fig 15. F-test code snippet

As per ANOVA test, our f_pvalue is lower than 0.05, thus we can conclude that our

model performs better than another simpler model.

```
print(olsmod.pvalues)

const     1.109259e-01
S_3      1.931957e-229
S_9       1.453057e-12
dtype: float64
```

Fig 16. Moving averages

As per T-test, both of our independent variables, S_3 and S_9, have p-value less than 0.05 which shows that there is sufficient evidence that they affect the closing price.

Main Model 1: KNN

The KNN algorithm is heavily dependent on the value of K. Depending on the value of K, the model performance changes. Hence, to allow changing the K value and selecting the best K value for the model before applying it to the test data, we split the data into 3 parts - Training, Validation and Testing datasets. The dataset is divided as Training dataset which contains 80% of the original records which is further subdivided into Validation dataset consisting of 20% of training data. To avoid any effect of features having different ranges, we normalize the data and implement KNN for values ranging from 1 to 20.

The RMSE Value for each value of K is calculated to understand the effect of changes in K. Using Elbow curve, RMSE value and GridSearchCV, we get the best value of K as 3, which produces the least value of RMSE as 0.94 and R^2 value of 0.99

```
RMSE value for k=  1 is: 1.0178233324060217
RMSE value for k=  2 is: 1.0158019961323832
RMSE value for k=  3 is: 0.9648268105017221
RMSE value for k=  4 is: 0.9455661509846863
RMSE value for k=  5 is: 0.9360855145067172
RMSE value for k=  6 is: 0.9525186851087472
RMSE value for k=  7 is: 0.9709903015968334
RMSE value for k=  8 is: 0.978187972844754
RMSE value for k=  9 is: 0.9920962625475711
RMSE value for k=  10 is: 1.004498091402953
RMSE value for k=  11 is: 1.017753393758283
RMSE value for k=  12 is: 1.0313609204063818
RMSE value for k=  13 is: 1.0393952141497897
RMSE value for k=  14 is: 1.0609399254255347
RMSE value for k=  15 is: 1.0738311483973735
RMSE value for k=  16 is: 1.071668529195449
RMSE value for k=  17 is: 1.0758410886462892
RMSE value for k=  18 is: 1.0896057473596281
RMSE value for k=  19 is: 1.102341483191008
RMSE value for k=  20 is: 1.1167408807718686
```

Fig 17. Finding out the best K value

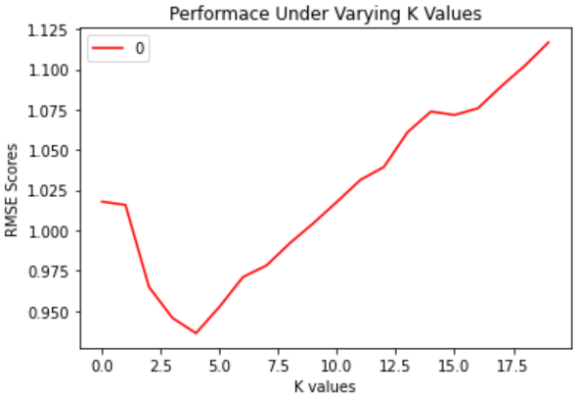

Fig 18. Model performance under different K values

Comparing the RMSE score, we can conclude that our 1st Main model which is KNN, performs better than a baseline model which in this case is a Linear Regressor.

Main Model 2: SVR

SVR can be implemented using different kernels such as Linear, Radial Basis Function and Polynomial Function. To select the best kernel from linear and RBF and best C value, we make use of the GridSearchCV package. Linear kernel was found to be the best kernel with a C value of 1000 after using GridSearch with 5 fold cross validation. The performance of SVR was measured using the RMSE and R^2 value.

```
# Declaring the SVM parameters
parameter_candidates = [
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]

# Using GridSearchCV to find best parameters employing K-fold cross-validation with K=5
svr = SVR()
clf = GridSearchCV(svr, param_grid=parameter_candidates, cv=5 )
clf.fit(x_train, y_train)
print('Best score for data:', clf.best_score_)
print('Best C:',clf.best_estimator_.C)
print('Best Kernel:',clf.best_estimator_.kernel)
print('Best Gamma:',clf.best_estimator_.gamma)
```

```
Best score for data: 0.9996698028752066
Best C: 1000
Best Kernel: linear
Best Gamma: 0.001
```

Fig 18. SVM Code Snippet

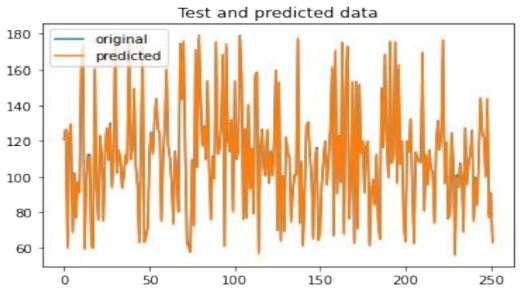

Fig 19. Model performance for SVM

For SVR, we got a RMSE value of 0.73 and a R^2 value of 0.99. As seen from the plot above of the original and predicted stock prices. The SVR model has been able to predict the prices correctly to a large extent. This is mainly because the kernel chosen was Linear and not RBF. When the RBF kernel was used, the results deteriorated and the SVR model was not able to accurately predict the stock prices.

Main Model 3 : LSTM

Since our dataset is extremely small for a neural network we chose our number of internal units as 2. Typically for a larger dataset the number of internal units stands between 5 and 10. But taking into consideration our dependencies such as the closing price for the previous day, the influencing factors and the difference between the closing prices between two consecutive days, LSTM should be the best model.

Post prediction we plotted a graph on how close our train and test prices are compared to the original price, which in this case comes out as almost identical. We have chosen the number of epochs as 100. Increasing the number of Epochs generally increases the accuracy of our model.

Fig 20. Prediction Graph for LSTM

For LSTM, we got a RMSE value of 1.73 and a R^2 value of 0.99

Conclusion

Table 1 Rmse and R-square values

MODEL	RMSE	R^2 VALUE
Linear Regression	2.248	0.96
KNN	0.93	0.99
SVR	0.73	0.99
LSTM	1.78	0.99

As we can see from the above table, comparing RMSE and R square value, SVR turns out to be the best model considering our dataset.

To summarize, using machine learning models, we can predict the stock prices. While Linear regression and KNN perform relatively well, SVR proves to be the best machine learning model to predict the stock prices due to its flexibility and its usage of Linear kernel to fit a hyperplane. LSTM, a model which is traditionally used for time series data, was expected to perform the best. However, since it is a deep learning model and deep learning models require large data to learn the parameters, which was not available in our case, LSTM performance comes only second best to SVR.

Failure analysis: We predicted in the beginning that LSTM would be our best model as LSTM behave very efficiently in case of time series data, but as we have limited dataset ie 1259 rows (5 years of stock pricing of Apple), that's why the RMSE value for LSTM model is not that promising.

As we are getting 0.99 as R^2 value for KNN, SVR and LSTM. There is a need to compare both of them, both measure a model's goodness of fit, but they have different ideas about what "good" means. For RMSE, "good" means that the model generates accurate predictions (small residuals). For R^2, "good" means that it's the predictor variable doing the actual predictive work, as opposed to the response variable simply having low variance and being easy to predict even without the predictor variable.

In this project, we tried to predict the stock prices using the historical data only. While this approach has given optimal results but in real world scenarios there are daily events which trigger a dip or peak. Daily events ranging from the release of the companies financial statements to news of national and global politics which directly or indirectly affects the stock prices. We can future extend the scope of this project by making use of tweets which acts the most common source of information in current times as an additional factor in deciding the stock prices. This would be done by scrapping the tweets from twitter and performing sentiment analysis on these tweets to find the sentiment of people towards a particular stock or event which will then be weighted using an appropriate factor and considered as a feature while predicting the stock price of a company along with the historical data about the stock. We can simply the process on stock price prediction to a classification one by making the result into a binary class that is predict if the stock price is going to be higher than the previous day price or lower than it.

References

[1] Machine Learning Approaches in Stock Price Prediction: A Systematic Review Payal Soni , Yogya Tewari and Prof. Deepa Krishnan AICECS 2021Journal of Physics: Conference Series

[2] Stock Market Prediction Analysis by Incorporating Social and News Opinion and Sentiment .Zhaoxia WANG, Seng-Beng HO and Zhiping LIN. 2018 IEEE International Conference on Data Mining Workshops (ICDMW)

[3] Short-term stock market price trend prediction using a comprehensive deep learning system. Jingyi Shen & M. Omair Shafiq Springer.

[4] Stock Closing Price Prediction using Machine Learning Techniques Mehar, Vijha, Deeksha, Chandolab and Vinay Anand Tikkiwala Arun Kumar Procedia Computer Science Volume 167, 2020

[5] Stock Price Forecasting by a Deep Convolutional Generative Adversarial Network Alessio Staffini 04 February 2022