STOCK PRICE PREDICTION

Date: 12th December 2019

Manpreet Kaur, 0731604
2019 Jie Gu, 0731409
Radhika, 0733926
Vinu Pradeep Menon, 0731928

1.Introduction

In today's world, machine learning has been widely used to find patterns from historical data, analyze it and predict what is to happen in the future with a level of accuracy which was once considered unachievable. One such example is predicting the stock price using historical stock data. For a stock, the owners, or the shareholders benefit when the stock price goes up. The potential profit for any stock is unlimited. This is one of the main reasons why stock price prediction has interested millions around the globe.

In this project we will try to come up with a model that is reasonable in predicting the stock prices of two companies namely AAPL (Apple Inc) and ADNT(Adient) with the help of previous years' data. The same model can be used to predict the stock price variation patterns of various other different companies. We will be using normal classification and regression techniques in the beginning and finally will be trying more complex and advanced such as the Artificial Neural Networks.

2.Related Work

1. **A Machine Learning Model for Stock Market Prediction** by Osman Hegazy, Omar S. Soliman, Mustafa Abdul Salam.
   - In this study the authors have chosen LSSVM (Least Squared Support Vector Machines), PSV-LS-SVM (Particle Swarm Optimization) and NN-BP (Back Propagation) to find the stock price patterns and calculate the MSE (Mean Squared Error) for 13 different companies. The best model for this study was ANN-BP which had the least MSE.
2. **How To Use Machine Learning To Possibly Become A Millionaire: Predicting The Stock Market?** by Jerry Xu.
   - This study can be considered more like a tutorial than a project on how to build an algorithm to predict stock price. GOOGL (Google) is the stock of interest in this study. The author have used various methodologies like moving averages, Stocker module, simple linear regression, k-nearest neighbors and multilayer perceptron. The author has concluded that the neural network is the worst algorithm to predict the prices given the stock that he is using. All other models have given out results that is not completely reliable.
3. **Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)** by Aishwarya Singh

   - This project is based on trying out different algorithms such as Moving Average,Linear Regression,k-Nearest Neighbors,Auto ARIMA,Prophet and LSTM on the stock price data of

Tata Global Beverages. The study finally concludes that LSTM is relatively the better model to do a time series prediction.

3.Methods

3.1 **Data Preparation:**

The data that we choose contained the stock prices of numerous stocks dated back till 1972 and had over 10 lakh observations. To make the project lot briefer given the time frame, the data was filtered out into two different datasets containing AAPL stock data over the past 5 years (2014-2018) and ADNT stock data over the past 3 years (2016-2018).

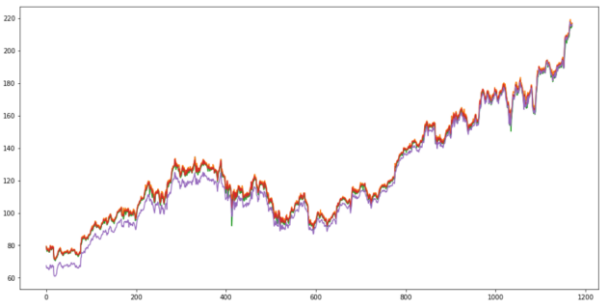
- Created two new columns by close price,
  - One is closePriceUp, goes up is 1, goes down 0;
  - Second is closePriceUpRate, which formula is as following,

closePriceUpRate=(close_price current day value-close_price previous day value)/close_price previous day value

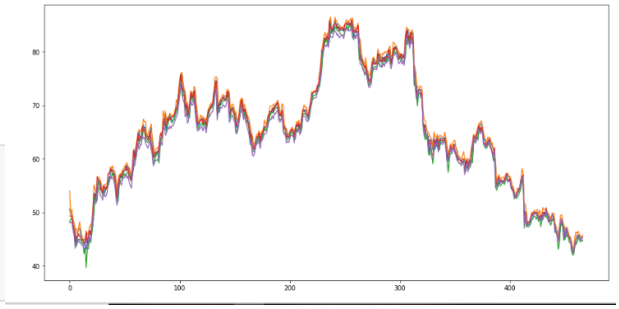
3.2 Data Exploration:

By simply plotting a line graph using matplotlib with close price as the x-axis and index (which is in a chronological order) as the y-axis for both the data, we came to know that stock prices followed a very fluctuating and visibly unpredictable trend throughout the timeline hence making the project more challenging. The following graph is the pattern that open, close and adj_close price that the AAPL and ADNT data follows.

```
#Features line plots
#Open Price/High/Low/Close?AdjClose
apple_1['open'].plot(x='date', figsize=(16,8))
apple_1['high'].plot(x='date', figsize=(16,8))
apple_1['low'].plot(x='date', figsize=(16,8))
apple_1['close'].plot(x='date', figsize=(16,8))
apple_1['adjClose'].plot(x='date', figsize=(16,8))
```



```
#Features line plots
#Open Price/High/Low/Close?AdjClose
ADNT['open'].plot(x='date', figsize=(16,8))
ADNT['high'].plot(x='date', figsize=(16,8))
ADNT['low'].plot(x='date', figsize=(16,8))
ADNT['close'].plot(x='date', figsize=(16,8))
ADNT['adjClose'].plot(x='date', figsize=(16,8))
```



3.3 SVM Model Creation

3.3.1 Why is SVM algorithm

SVM (Support Vector Machines) in machine learning is a supervised learning model. I analyze data for the classification and regression types. Given a set of training data, each marked as belonging to one or other of two categories, an SVM training algorithm builds a model that assigns new examples to categories, making the model as a non-probabilistic binary linear classifier. An SVM model is a representation of examples as points in space, mapped so that the examples are the separate categories are divided by clear gap that is as wide as possible. New examples are then mapped into

that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature space.

3.3.2 Target setting

To predict a stock , there are many indicators can be set as the target, here we select the simple one as ‘closePriceUp’, which means that the day stock price goes up or goes down. This is a classification type of problem. So we need find out what classifier could be used well.

3.3.3 Data Splitting and Features Selection

Here we did not use the normal sklearn test-train split randomly way, for the stock price is along with timeline. We select the latest year data as test data, other previous years data as training data.

Select’ open', 'close','adjClose','low','high','volume','closePriceUpRate' these features as training data and test data set features. Target feature is 'closePriceUp'.

3.3.4 Data Set Scaling

Before the SVM model creation, we noticed that our data set all the features are not scaled well, i.e. volume feature’s value are much larger than other prices. This will lead an unbalance between the features for the future model creation and prediction score. One preprocessing scale step could be down here for solving the problem. In this project we try both two standardizing methods, one is

MinMaxScaler(initial model tuning) and StandardScaler (within the pipeline).The latest was proved more effect to the model.

One point need be mentioned here, we scaled training data and test data separately. This is for the reason of to avoid test data set information leak into training data, to be overfitting for test data. But if we use pipeline class, this kind of mistake would be avoided automatically.

```
from sklearn.svm import SVC
svm = SVC(C=100)
# learning an SVM on the scaled training data
svm.fit(X_train_scaled, y_train)
```

3.4 Random Forest Model Creation

3.4.1 Why is Random Forest

Random forest algorithm is most popular classification algorithm. It is supervised classification algorithm too. Random forest algorithm applications area could be: Banking, Medicine, Stock Market, E-Commerce. In the stock market, random forest algorithm used to identify the stock behavior as well as the expected loss or profit by purchasing the stock.

Because our target is to predict the Apple and Adient stock close price goes up or down (up class 1, down class 0), this is a classification problem, so here we choose the second research algorithm as Random forest.

3.4.2 Model Creation

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=5, random_state=2)
forest.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=None,
            oob_score=False, random_state=2, verbose=0, warm_start=False)
```

3.5 Linear Regression

Note: For regression model we are using close price of these two stocks as the target variable along with index/date as the explanatory variable.

Linear regression can be considered as one of the simplest methods to predict continuous data that follows a linear relationship. For our dataset the linear regression model will try to fit a straight line to the close price of ADNT and AAPL, so that the pattern that the close price follows can be predicted.

3.5.1 Model Creation

```
lr = LinearRegression()
```

```
X_train, X_test = X[:n_train], X[n_train:]
y_train, y_test = y[:n_train], y[n_train:]
lr.fit(X_train, y_train)
y_pred_train = lr.predict(X_train)
y_pred = lr.predict(X_test)
```

3.6 Random Forest Regressor

Same as doing random forest can be used for classification problems, it can be used for regression problems as well. The model try to incorporate more noise in the data than basic linear regression.

3.6.1 Model Building

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100, random_state=0)
regressor.fit(X_train,y_train)
y_pred_train = regressor.predict(X_train)
y_pred = regressor.predict(X_test)
```

3.7 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used. *(Understanding LSTM Networks, August 27, 2015)*

3.7.1 Model Building

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))
```

Note: The metrics used to evaluate the above models are accuracy and RMS (root mean square) for classification and regression respectively.

4.Results

4.1 Classification:

For SVM model the test accuracy was found to be 0.93 whereas for the random forest classifier the accuracy was 0.947. Are these scores reliable? Here we tried the pipeline class to cross validate these algorithms by doing the best estimator, best parameters tuning.

4.1.1 Pipeline Class Cross Validations

Pipeline class is a general purpose tool to chain together multiple processing steps in a machine learning workflow, which still have same function as a single model , such as fit, predict, transform , and score etc. In particular when doing model evaluation, using cross-validation and parameter selection with grid search, using the pipeline class can capture all processing steps in essential for a proper evaluation.

Now we use the pipeline to tune the models for the project.

1. StandardScaler+ SVC, tune the paprameter C
2.StandardScaler+RandomForestClassifier. grid parameter n_estimator

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
pipe = Pipeline([('preprocessing', StandardScaler()), ('classifier', SVC())])
param_grid = [
{'classifier': [SVC()], 'preprocessing': [StandardScaler(), None],
'classifier__gamma': [0.001, 0.01, 0.1, 1, 10, 100],
'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100]},
{'classifier': [RandomForestClassifier(n_estimators=100)],
'preprocessing': [None], 'classifier__max_features': [1, 2, 3]}]
grid = GridSearchCV(pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best cross-validation accuracy: {:.2f}".format(grid.best_score_))
print("Test set score: {:.2f}".format(grid.score(X_test, y_test)))
print("Best parameters: {}".format(grid.best_params_))
```

The result is as follows for AAPL:

```
Best cross-validation accuracy: 1.00
Test set score: 0.99
Best parameters: {'classifier': RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features=2, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False), 'classifier__max_features': 2, 'preprocessing': None}
```

```
In [112]: grid.best_estimator_
```

```
Out[112]: Pipeline(memory=None,
     steps=[('preprocessing', None), ('classifier', RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features=2, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False))])
```

```
In [113]: grid.score(X_test, y_test)
```

```
Out[113]: 0.9868421052631579
```

The result is as follows for ADNT:

```
Best cross-validation accuracy: 0.97
Test set score: 1.00
Best parameters: {'classifier': SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False), 'classifier__C': 100, 'classifier__gamma': 0.01, 'preprocessing': StandardScaler(copy=True, with
_mean=True, with_std=True)}
```

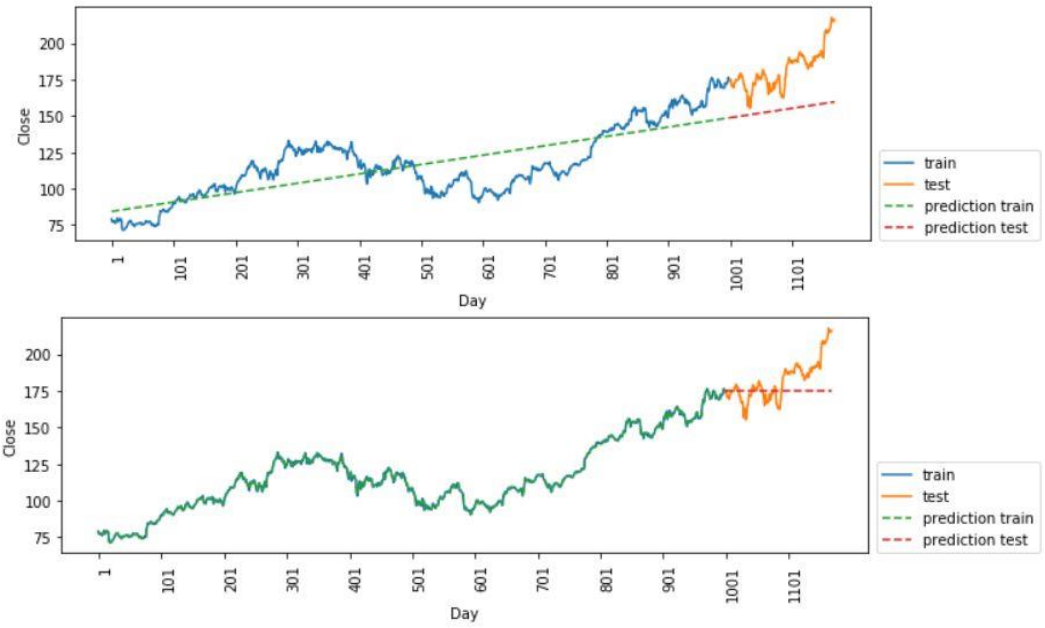
```
In [39]: grid.best_estimator_

Out[39]: Pipeline(memory=None,
         steps=[('preprocessing', StandardScaler(copy=True, with_mean=True, with_std=True)), ('classifier', SVC(C=100, cache_size=2
00, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False))])
```

```
In [40]: grid.score(X_test, y_test)

C:\Users\manpr\Anaconda3\envs\DAB300\lib\site-packages\sklearn\pipeline.py:511: DataConversionWarning: Data with input dtype in
t64, float64 were all converted to float64 by StandardScaler.
  Xt = transform.transform(Xt)

Out[40]: 0.9966996699669967
```

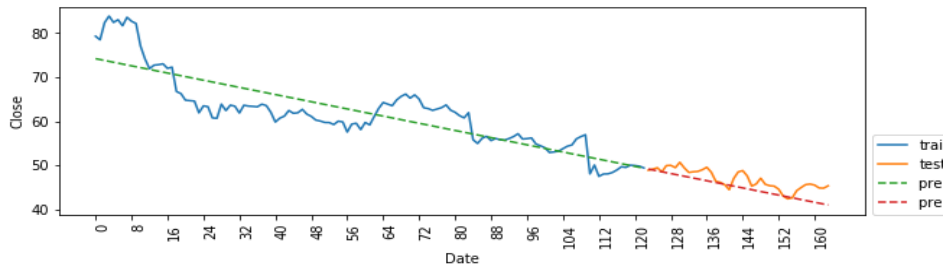
From the above results we can say that predicting if the stock price will go up or down for both the stocks was successful. Deep study is required to understand this result is reliable or not.

4.2 Regression:

The following is the graph of the linear regression and random forest regression model of the AAPL data



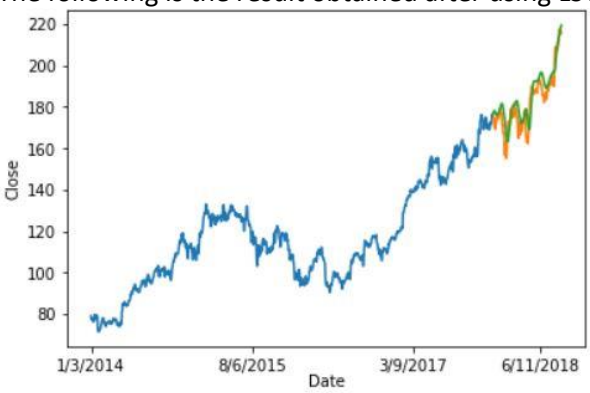
As we can probably see both the models does not capture the necessary variance in the data. The RMS value for both for linear regression and random forest regression is 29.9 and 15.3 which is quite large. For the ADNT data the linear regression model fits better than AAPL data which is shown below;



This is obviously due to the difference in data and we can surely say that linear regression model is not so much reliable when it comes to time series (in this case stock) prediction.

4.3 Long Short Term Memory (LSTM):

The following is the result obtained after using LSTM on AAPL data



Clearly, we can see that the model's prediction (shown in green) have learned most of the variance of the data. This have helped in predicting stock price fluctuations of the test data with very little error. The RMS value for this model was found to be as small as 6.11, which in this context is a good value to have. It is to be noted that increasing the layers of the LSTM model or changing the epoch can result in better outcomes.

5.Discussion

5.1 Reliability of the classification models.

Trying to predict if the stock price goes up or down using a numeric feature can be a little misleading. This can be due to the fact that there can be multiple values for the explanatory variable that is mapped to a single target variable. Even though the accuracy that we found was really high, deep study is required to understand this result is reliable or not.

5.2 Concern about the feature 'closeUpRate'

For the case of Apple, the feature 'closeUpRate' is a feature we explored from the original data set. Its sample values have very strong correlation with then target 'closePriceUp'.
We did a step in coding with "how much would the feature 'closeUpRate' effect for the model".For SVM model drop off the "closeUpRate" , the score for both train and test data set goes down to 0.58, 0.43.Samilar when we take the "closeUpRate" off from RandomforestClassifier model , train data model score is good as 0.935, but not good for test data is only 0.395.Assume this is not a good way to drop off "closeUpRate"feature. So still to keep it.

5.3 Project target setting more realistically

We choose two type of targets for our project machine learning basing on our data set, one is the stock price goes up or goes down('closePriceUp') and the other one is real stock price prediction for the next days. Choosing these relative simply targets is mainly for the reason of stock marketing prediction which is a fairly complex area and we want to make sure we can have a definite output for the project. But in real world stock marketing prediction area, there should be more complicated target, the shareholders getting more profit. These are what we want to try , but due to the limitation of the project term, we left it to the future machine learning research.

5.4 Concern about the date utilization

From the beginning we thought that the stock price should be a task relative with timeline. So we did some work on splitting date value from '3/17/2017' format to year, month, day, and quarters information to individual features.

But in actually project progressing, we did not find a good method to accommodate the time element within our machine learning scope. So, we drop them off when we created the models. Still this is good topic to practice timeline features with other features in the machine learning cases in future.

6.Conclusion

This project was started with a simple goal to analyze and predict the future stock price using simple machine learning techniques like classification and regression. As the project progressed, we understood that predicting the stock prices is not something a beginner in the field of machine learning could do. Hence, we had to think out of the box to come up with a model like LSTM. The best way to predict the stock prices is by using ANN and various models associated with it.

The classification models (SVM and Random Forest Classifier) which predicted if the stock price will go up or down yield good results having said that the process and results need to be studied thoroughly to be termed as reliable. On the other hand, regression models (Linear Regression and Random Forest Regression) was poor in predicting the stock prices. The relatively successful model was LSTM which, unlike other regression models, predicted the output with minimum amount of error. Hence, we can expect to get a similar kind of result when LSTM is used for other stocks.

Finally, we must admit that due to the unpredictable nature of this field, stock price prediction is a very though task and not a single model can be considered 100% reliable. Deep research and extensive knowledge about Neural Network are needed to build a model that has the ability to predict future stock price accurately with consistency.

7.Contributions

Jie Gu: Made the SVM, Random Forest Classifier and pipeline to tune the model parameters for AAPL dataset
Radhika: Did the SVM, Random Forest Classifier and the pipeline to tune the model parameters for ADNT dataset
Manpreet Kaur: Did the Linear regression model for the ADNT dataset, pipeline to tune the data.
Vinu Pradeep Menon: Made the Linear Regression model, Random Forest Regression model and the LSTM model for the AAPL data. Also, compiled the report.

Note: Each member did their respective parts for the report and was finally compiled. Also, the meeting minutes for every week was done in the same manner.

8.References

1. https://towardsdatascience.com/how-to-use-machine-learning-to-possibly-become-a-millionaire-predicting-the-stock-market-33861916e9c5
2. https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/

3. https://www.researchgate.net/publication/259240183_A_Machine_Learning_Model_for_Stock_Market_Prediction
4. https://en.wikipedia.org/wiki/Long_short-term_memory
5. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
6. https://www.wisebread.com/9-ways-to-tell-if-a-stock-is-worth-buying
7. Introduction to Machine Learning with Python by *Andreas C. Müller and Sarah Guido* (Text book)

9.Appendices

- Final_Adient_File.ipynb - 3.3,3.4,4.1
- LINEARMODELA.ipynb - 3.5,3.6,4.2
- FinalProjectVinu.ipynb - 3.5,3.6,3.7,4.2,4.3
- APPL_EDA_1.ipynb – 3.2,3.3,3.4,4.1
- Datasets: apple_clean.csv, ADNT_clean.csv, ADNT_2018.csv, AAPL.csv