Machine Learning Engineer Nanodegree

Capstone Project

Santhosh Kumar Rathode
June 30, 2020

Stock Price Prediction using Machine Learning

Domain Background

In Today's market investing in a stock is a biggest challenge to the investors. Stocks prices are unpredictable, it depends on various factors such as social media, political, financial, crisis, global relationship, demand, president and many more. Buying a stock means buying a part of the company. Making money or losing money depends on stock price up and down, so predicting price of the stock is the biggest achievement for the investors.

Number of companies from Small to large businesses are trading publicly on the world stock market with billions of investors. Platform is so simple that anyone can choose any stock exchange and invest with min amount. In order to get more profit out of stock market one should play smartly with good strategy. That can be achieved by building best Machine Learning models to predict the future trend by using past historical data and event driven indicators.

Herbart Alexander Simon says that "Machine learning is concerned with computer programs that automatically improve their performance through experience." The field of prediction is primarily focused on minimizing the errors and giving more accurate predictions.

Motivation

As like everyone, I also started investing in stock market and still I am not clear that how stock market is going up and down. Once I took this Machine Learning Engineer program I started analyzing data, models and methodologies, then I am able to understand. For this Nanodegree project I am choosing the project of my interest, so I can show more interest towards my ML program.

Problem Statement

Prediction stock market is not that easy and at the same time it's not that tough. Stock market is a complex and forecasting is characterized by data intensity, events, hidden relationships and nature of the market. Predicting fluctuating price with time in stock market is quite difficult.

Prediction in investment and trading is ongoing research, as part of this Nanodegree capstone project I am going to predict the stock price by using well-known machine learning models. I will be considering past 8 years of time-series historical data for selected stock.

Solution Statement

There are number of algorithms/models to predict time-series based problems. In this project I don't want to jump to top model with all possible indicators to predict stocks. As a beginner I am going to complete project by applying simple supervised learning methods, which I learned from this Nanodegree course. This time-series based project is the regression problem and proposed solution involved applying Long Short-Term Memory (LSTM) from recurrent neural network architecture. Decision Tree Regression model for the benchmarking. In addition to the proposal I am going to use two other models for this project Support Vector Regressor **(**SVR) and Random Forest.

For the model I am going to use 8 years of historical data with few technical indicators such as moving average convergence-divergence (MACD), relative strength index (RSI) and simple moving average (SMA).
I will compare both the benchmark and solution models and provide more insights.

Project Design

Project will be completed on the Jupyter Notebook with python language. This project requires pandas, numpy, scikit-learn, keras, matplotlib and seaborn libraries so I will be importing this before I start. Development code will be found in Final_Capstone_Stock_Prediction.ipynb notebook. I have defined implementation methods with proper documentation.

Machine Learning Work Flow:

1. Data Collection, Exploration and Preprocessing
2. Data visualization
3. Feature selection/Technical indicator selection
4. Normalize the data
5. Split the data into Train, Test and Validation sets
6. Implementing model prediction and evaluation
7. Finally, compare the result with benchmark model.

1. Data Collection, Exploration and Preprocessing

Datasets collection

Stock market historical data can be found from many places, but I choose to download from https://finance.yahoo.com/. There are many web scraping API's available to get the data based on selected dates. I downloaded CSV files excluding recent 2020 data because of COVID-19 market crash. I understand that model prediction is not based only on historical data, it also includes various factors like news, sentiments and etc. So, I am going to analyze 8 years of historical data from 2011-12-30 to 2019-12-30.

Data Exploration and Preprocessing

In data exploration part I am going to study the data and try to understand how data looks like and relation between each column. Data preprocessing is originated from data mining origin and this section is toughest challenge to everyone. ML model needs clean and processed data for the training. Need to filter out or fill null values in data.

For the stock prediction I am selecting two stocks from two different sectors one from Coffee house and another from Telecommunications.

1. Starbucks (SBUX)
2. ATT (T)

Dataset Characteristics

Columns	Description
Date	Calendar date of the trading day
Open	Opining price of the trading day
High	Highest price of the stock traded during the day
Low	Lowest price of the stock traded during the day
Close	Closing price of the trading day
Adj Close	Adjusted closing price of the trading day
Volume	Number of shares traded in exchange during the the day

Stock prediction is time-series analysis, so I am going to set index as date column and as I am going to predict price so using Adj Close column as target value. After reading the CSV, here I am going to provide sample data and their price trendline from 2012 to 2019.
Before starting any analysis, I am going to check for null values and basic data info. Current data set has no null values, below is basic info of data. It shows that Starbucks has 2011 non-null records and att has 2012 non-null records.

```
#Basic info
display(starbucks.info())
display(att.info())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2011 entries, 2012-01-03 to 2019-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       2011 non-null   float64
 1   High       2011 non-null   float64
 2   Low        2011 non-null   float64
 3   Close      2011 non-null   float64
 4   Adj Close  2011 non-null   float64
 5   Volume     2011 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 110.0 KB

None

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2012 entries, 2011-12-30 to 2019-12-30
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       2012 non-null   float64
 1   High       2012 non-null   float64
 2   Low        2012 non-null   float64
 3   Close      2012 non-null   float64
 4   Adj Close  2012 non-null   float64
 5   Volume     2012 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 110.0 KB

None
```

Also exploring at global statistics like count, mean, std, min and max. Based on the statistics no column has 0 records, so I am going to consider open, high, close and low columns for model feature.

Starbucks - Global Statistics:

	Open	High	Low	Close	Adj Close	Volume
count	2011.000000	2011.000000	2011.000000	2011.000000	2011.000000	2.011000e+03
mean	50.385403	50.795077	49.969863	50.395219	47.058367	1.012686e+07
std	17.108990	17.232945	16.972948	17.103142	17.561368	5.493606e+06
min	21.799999	22.090000	21.520000	21.580000	18.873688	1.847800e+06
25%	37.052499	37.387501	36.780000	37.012501	33.200750	6.792600e+06
50%	54.130001	54.580002	53.799999	54.189999	50.682381	8.593500e+06
75%	58.625000	59.155001	58.050001	58.635000	54.989284	1.173130e+07
max	98.139999	99.720001	97.209999	99.110001	97.239372	6.209110e+07

ATT - Global Statistics:

	Open	High	Low	Close	Adj Close	Volume
count	2012.000000	2012.000000	2012.000000	2012.000000	2012.000000	2.012000e+03
mean	35.343226	35.573255	35.094896	35.341690	27.924208	2.737767e+07
std	3.074129	3.079648	3.074469	3.082351	4.261648	1.363271e+07
min	27.500000	28.100000	26.799999	27.360001	18.699926	6.862400e+06
25%	33.320000	33.580002	33.067500	33.337501	24.760416	1.971698e+07
50%	35.009998	35.180000	34.779999	35.000000	26.892488	2.422520e+07
75%	37.422499	37.689999	37.140000	37.419998	31.423689	3.090855e+07
max	43.500000	43.889999	43.330002	43.470001	38.430523	1.950827e+08

2. Data visualization

Visualization is best way to present the data to without going into the actual source and querying it. Anyone can easily understand the data if show them with good graphs.

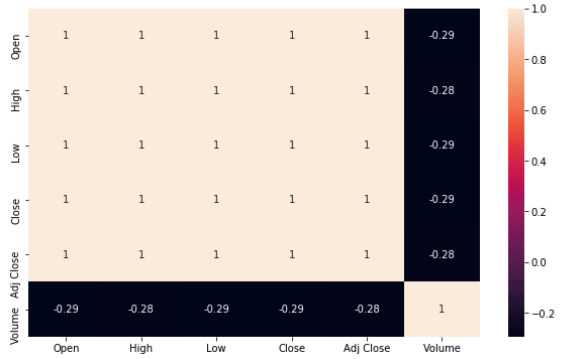
Correlation Analysis

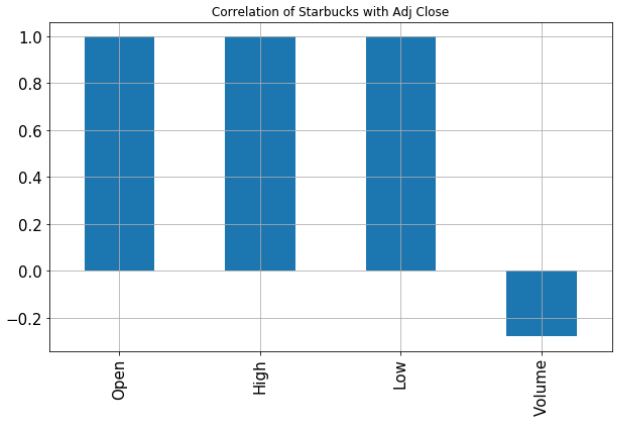
In statistical analysis correlation is nothing but relation between variables. If correlation coefficient is positive correlated then both the values are moving in one direction, if coefficient is exactly one it is called perfect positive correlation. In other way negative correlated values moves in different direction. Coefficient fluctuates in between -1 and 1, 0 means no relation.
Below we can observe how this data correlated with Adj Close price.

Starbucks

```
plt.figure(figsize=(10, 6))
sns.heatmap(starbucks.corr(), annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce2b0deed0>
```

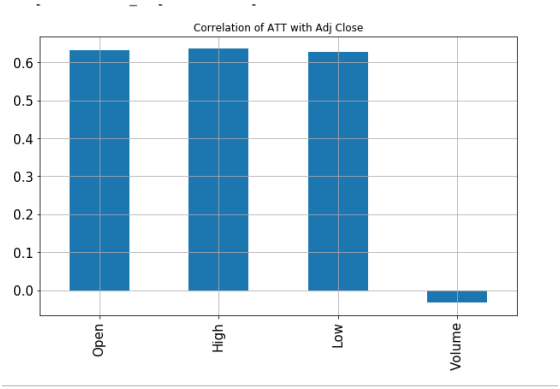




ATT

```
#Corr Matrix for ATT
plt.figure(figsize=(10, 6))
sns.heatmap(att.corr(), annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce2b2fd8d0>
```



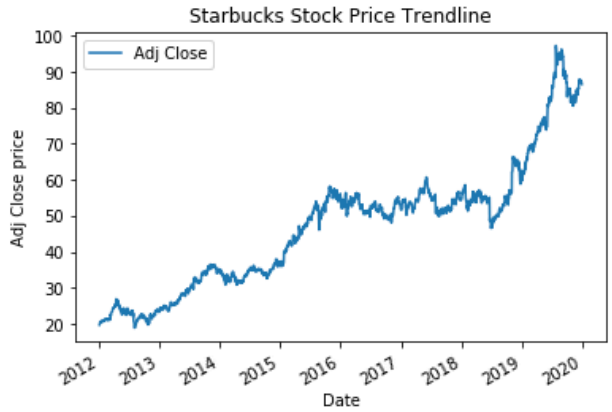
We can observe that both stocks Starbucks and ATT has positive correlation for open, high, low features and Volume alone has negative correlation.

So, I am going to consider these 3 columns as features for model prediction.

Until now, I explored data with some visual graphs and correlation matrix. Here I am going to present sample data and full trendline of the stocks in a graphical manner. I am using matplotlib to explore the visualization.

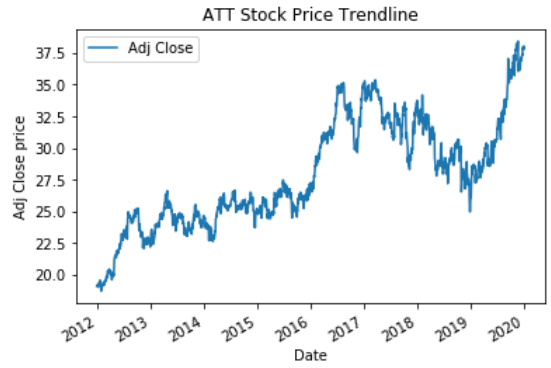
Starbucks

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03	23.424999	23.520000	22.639999	22.645000	19.675068	12922400
2012-01-04	22.705000	23.315001	22.639999	23.084999	20.057360	13886800
2012-01-05	23.094999	23.200001	22.775000	23.180000	20.139898	9731800
2012-01-06	23.190001	23.455000	23.115000	23.360001	20.296291	8105400
2012-01-09	23.365000	23.400000	23.135000	23.295000	20.239815	7261600



ATT

Date	Open	High	Low	Close	Adj Close	Volume
2011-12-30	30.100000	30.299999	30.080000	30.240000	19.111830	15687800
2012-01-03	30.459999	30.540001	30.299999	30.379999	19.200312	33005300
2012-01-04	30.580000	30.680000	30.350000	30.430000	19.231915	37213900
2012-01-05	30.490000	30.500000	30.180000	30.400000	19.212955	34457000
2012-01-06	30.070000	30.090000	29.600000	29.680000	19.033392	45580800



Trendlines shows that Adj Close price increased since 2012, that means investors share value become doubled if they would have invested in 2012.

3. Feature selection/Technical indicator selection

Technical indicators are factors that drives the stock price, it helps to understand price movement and helps to predict the future price, that leads investors buy or sell choices. Apart from above historical data features I am going to calculate few well know technical indicators and I will consider as features for prediction models.
To calculate I am going to use Adj Close as my target value.

1. Relative Strength Index (RSI)

It is widely used indicator to identify overbought and oversold conditions. It ranges between 0 and 100. If the value is 0 then oversold and 100 means overbought. Also used to chart the trendline of the stock (raise and fall) based on the closing price.
For the RSI calculation it considers 14 days' timeframe.

Formula:
RSI = 100 – (100/(1+(rUp/rDown))

rUp = average of upward price change
rDown = average of downward price change

2. Moving Average Cnvergence-Divergence (MACD)

This is simplest and powerful indicators to calculate the momentum of prices. It moves above and below the 0 line as the moving averages converge, cross and diverge.
MACD calculated based on 12 days and 26 days exponential moving average (EMA) difference.
Further this difference MACD is calculated which is 9 days EMA of difference. To calculate moving

averages, we use closing price (this case Adj Close). We can say that MACD line is positive if it is above signal line negative if it is below signal line.

Formula:

MACD Line: (12-day EMA - 26-day EMA)
Signal Line: 9-day EMA of MACD Lin
MACD Histogram: MACD Line - Signal Line

3. **Simple Mean Average (SMA)**

Simple mean average is very easy to calculate, it is the sum of all the closing price over selected time frame and divided by total number of prices in time frame. In short it is the average price over the time-frame.

Formula:

SMA = (P1+P2+……+Pn)/n

P = closing price
N =total number of prices.
I am using 15 day period to calculate SMA.

4. **Standard Deviation (STDEV)**

Standard deviation is used to measure stock market volatility. It shows how prices are dispersed from the average price. If prices fluctuate more then it indicates high volatility and if price trades narrow range, then indicates low volatility.

Formula:

STDEV = Sqrt [(Sum the ((Close for each of the past n Periods
- n Period SMA for current bar)^2))/ n]

5. **Bollinger Bands (BBand)**

BBand is a technical analysis tool designed to show opportunities to the investors. BBand identifies when asset it overbought or oversold.
BBand compose of 3 lines upper band, lower band and middle band(simple moving average). Upper band is 2 STDEV is above the middle line(SMA) and lower band is 2 STDEV below the SMA.

Formula:

upper_band = SMA + (2 * STDEV)
lower_band = SMA - (2 * STDEV)

To calculate all of the above technical indicators I have defined a simple method for each indicator and update final data to the table.

Technically, smart investors use 2 or more indictors to identify of buy or sell signals and enters into the market. Both MACD and RSI used to identify trade signals, and both are complement to each other due to their inherent nature of comparing the prices. MACD considers moving average of prices whereas RSI considers the ratio of the gains and losses. So, these two indicators work good together.

After adding the technical indicators, sample data looks like below. If you observe the data there are few indicators have 0 values, this is due to introduction of indicators.

Date	Open	High	Low	Close	Adj Close	Volume	RSI	MACD	SMA	STDEV	Upper_Band	Lower_Band	DIFF
2012-01-03	23.424999	23.520000	22.639999	22.645000	19.675068	12922400	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-04	22.705000	23.315001	22.639999	23.084999	20.057360	13886800	100.000000	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-05	23.094999	23.200001	22.775000	23.180000	20.139898	9731800	100.000000	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-06	23.190001	23.455000	23.115000	23.360001	20.296291	8105400	100.000000	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-09	23.365000	23.400000	23.135000	23.295000	20.239815	7261600	98.855975	NaN	NaN	0.245142	NaN	NaN	NaN

So, I am going to drop those 33 records from normalized data and remaining will consider further.

4. Normalize the data

In general data is in different ranges and it is very hard to use for further analysis. Normalization is used to organize or reduce the date in similar ranges. This technique will format each and every data into one format, mostly it ranges from 0 to 1.
For this project I am going to use sklearn's MinMaxScaler method and I defined one method to normalize the data using MinMaxScaler. Sample normalized data looks like below, first one has feature data.

Date	Open	High	Low	Volume	RSI	MACD	SMA	STDEV	Upper_Band	Lower_Band	DIFF
2012-02-21	0.031569	0.030207	0.033228	0.106963	0.679324	0.433086	0.015142	0.024692	0.001793	0.040310	0.426651
2012-02-22	0.030718	0.027438	0.031906	0.112902	0.686452	0.431018	0.015367	0.002657	0.001584	0.040967	0.424027
2012-02-23	0.031045	0.028146	0.032765	0.106329	0.700101	0.429092	0.015518	0.002354	0.001558	0.041292	0.422577
2012-02-24	0.031438	0.029112	0.033426	0.078110	0.666748	0.427049	0.015832	0.000756	0.000797	0.042686	0.420259
2012-02-27	0.029670	0.028211	0.031312	0.080759	0.651834	0.424888	0.015863	0.000631	0.000755	0.042790	0.417847
...	...	...	...	...	...	...	...	...	...	...	...
2019-12-23	0.875295	0.859977	0.880698	0.041530	0.635495	0.603963	0.885077	0.043004	0.878159	0.865940	0.616685
2019-12-24	0.869793	0.856756	0.878319	0.000000	0.656803	0.611387	0.888339	0.044991	0.878598	0.871934	0.615330
2019-12-26	0.876736	0.858817	0.875545	0.038675	0.601844	0.615051	0.890702	0.037279	0.878238	0.876966	0.605311
2019-12-27	0.868614	0.854953	0.874224	0.037871	0.606026	0.615855	0.893947	0.037051	0.873076	0.888627	0.595941
2019-12-30	0.867304	0.851604	0.865900	0.046608	0.521029	0.611998	0.894934	0.083161	0.871905	0.891769	0.576129

1978 rows × 11 columns

Before normalizing the data, I am going to select features data and target data into two different sets. Including technical indicators, I am going to select feature data includes ['Open', 'High', 'Low', 'Volume', 'RSI', 'MACD', 'SMA', 'STDEV', 'Upper_Band', 'Lower_Band', 'DIFF'] and target data frame will be only Adj Close. Finally, I am going to normalize features data frame using normalize_featuresDF() method.

5. Split the data into Train, Test and Validation sets

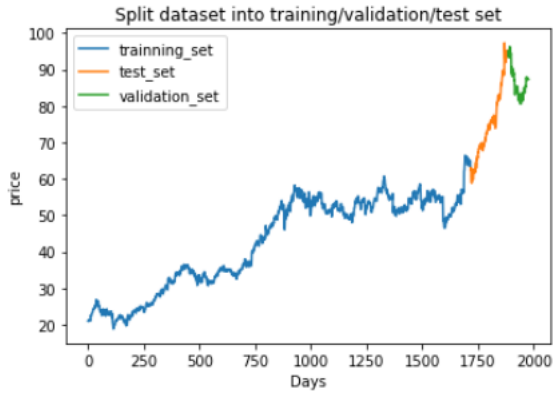
After normalizing the data, we need to split the data into 3 sets. Usually, data can be split into 80, 10, 10 and 75,15, 10.

Training set: to train the model
Testing set: to test the model, I am going to evaluate with best model

Validation set: to validate the model, used to calculate the evolution metrics.

I am going to split last 90 records for validation set and using sklearn's TimeSeriesSplit method to further split the training set and testing set.
By considering all above, to simply work I have defined simple methods to split train, test and validation sets. And also written simple method to visualize the train, test, validation sets in single trendline.
For the sanity check I am going to plot the graph which includes train, test and validation sets, because time-series set should not mix with different time frame.

```
#Visualizing the train, test and validation sets in single
#Starbucks
plot_3DataSets()
```



Until now, I have done first part of machine learning that is processing the data. It includes data collection, explore, preprocessing, normalization and data split. Now the actual part comes into picture that is modelling. This is the second part of the ML.

6. Implementing model prediction and evaluation

In this part, I will discuss about modelling it includes validation/evaluate the model and developing the benchmark and solution model. Complete code and methods used for this implementation can be found in **Final_Capstone_Stock_Prediction.ipynb**.

Metrics

To start with, metrics are very useful to assessing the models. In the machine learning regression experiments, forecasting analysis requires predicting future values based on set of historical data. Providing accurate prediction serves the business need. This metrics alone does not help investors or any users to determine the prediction, but it just helps us to forecasting best model for the given problem. In this project I am predicting stock future value, so I need to evaluate the model by comparing actual value with predicted value.

So, I am focusing on the error or deviation between the actual and predicted value. To validate the models, I am using two popular metrics Root Mean Square Error (RMSE) and R-Squared (R2). Both the metrics will define error between actual and forecasted. Both RMSE and R2 are good measures of errors and can serves as a loss functions to minimize. Loss function is distance between actual and predicted values.

Root Mean Square Error (RMSE)

RMSE is popular and it is mostly used for numerical predictions. It compares a predicted value and known value. As I explained above my target is to find forecasted value and calculate the error between actual and predicted values. Such cases RMSE fit well because it squares the errors to have positive values and puts more weight on large errors. Compare with other error metrics RMSE amplifies and severely punishes large errors.

$$RMSE = \sqrt{(f-o)^2}$$

f = forecast (expected values or unknown results)
o = observed values (known results)

RMSE is easy to calculate and best suitable for finding errors in time-series approach. Based on the RMSE error we can able to determine which time-series algorithm method is the best for the forecasting. I will consider model which has less RMSE error is the best model.

R-Squared (R2)

R2 is a statistical measurement of how close the data are to the fitted in regression line. It means it can provide how much variation in the dependent variable can be explained by the variation in the independent variables.

In other words, R-Squared is coefficient of determination and usually it used when there is a comparison of linear regression models to determine how variation is captures with set of observation data. I am forecasting stock, so majority of investors uses R-Squared to find out the coefficient. Coefficient helps them to compare performance of portfolio with the market and that result in predicting the future trend with that stock.

R-Square only works for linear regression model and gives the estimation of relation between actual trendline and predicted trendline based on the historical data. Prediction fits the data perfectly when R-Squared values is equal to 1.

There are other regression metrics like mean absolute error, mean forecast error, Pearson correlation and covariance used to find out error and coefficient, but I picked above two widely used metrics for this project.

To calculate RMSE and R2 score I am using sklearn's metric methods mean_squared_error, r2_score.
I will compare metrics with benchmark and solution models and choose the best solution model from them. Best model should have lower RMSE and higher R2 Scores. I have created methods **model_validateResult** and **bestModel_validateResult** to calculate metrics and plot the graph.

Algorithms and Techniques

The goal of the project to study stock market time series data and explore different models or algorithm to predict the stock price. Stock market prediction falls under time-series problem and the models I am using for this project comes under supervised learning algorithms.
There are number supervised machine learning algorithms were there, after researching a lot about time-series analysis I found that following models are widely used and they do not take hours to train. I am

going to study 4 different models, Decision Tree Regressor as benchmark model and support vector matrix (SVM), random forest and LSTM as solution models.

Un-tuned version of solution models will use the default parameters provided by scikit-learn and keras packages. Then parameter tuning will be used to improve the model performance. I will decide performance by evaluation the models with the RMSE and R2 squared metrics. I will use scikit-learn GridSearchCV method to perform tuning by giving set of dictionary parameters and values. GridSearchCV will test the all the possible combinations and produces the best result for the trained model. I will use best model for future price prediction.

All the models can learn complex relationship between features and target and provides continuous predicted output. The historical stock data I am using for these models are composed of continuous values, so no need of transformation from categorical to continuous data is needed.

For all the models, I will use training set to train the model and validates the model with validation sets. Second time I will tune the model with different parameters and train and validates the model. I will compare RMSE and R2 metrics for each case and select best model out of those. Finally, I will evaluate the best trained model with testing set to predict the stock price. I will explain more about models and their performance below.

First, I am going to explain about each model and implementing models for the Starbucks along with every implementation step. In second part I am giving to repeat same steps and show you my analysis.

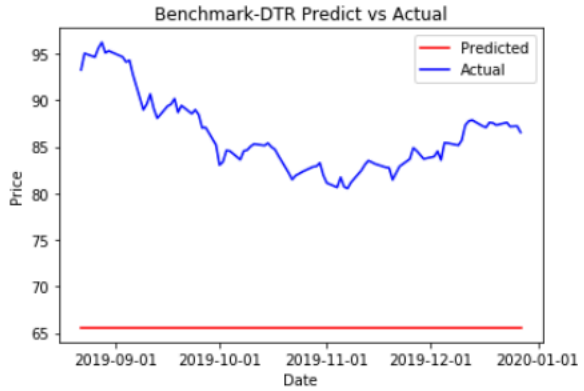
Benchmark Model - Decision Tree Regressor

After going through lot of materials I have decided to use Decision Tree Regressor (DTR) as my benchmark model with default hyperparameters. DTR is powerful and simple algorithm in supervised machine learning.

It uses flow chart like tree structure to predict the target value and it produces/predicts the continuous output by training features of an object. If the target value is continuous, model can be used to fit sine curve with addition noisy observation. It tries to maximize the gain information by learning local regression approximating the sine curve.

DTR is part of scikitlearn's tree. Once I trained the model with default parameters I will validate the results. After validating the model, I got RMSE and R2 score as below.

```
#Benchmark - DTR model RMSE and R2 score with plot
RMSE_Score, R2_Score = model_validateResult(model_Benchmark_DTR, model_name = "Benchmark-DTR")
```




```
Benchmark-DTR RMSE:  21.062661876991406
Benchmark-DTR R2 score:  -26.443680317986924
```

Solution Models

I am going to use 3 different solution models for this project. For each model I am going to use tune little bit and compare the metrics with initial model. After validating the result, I will consider best model for this project.

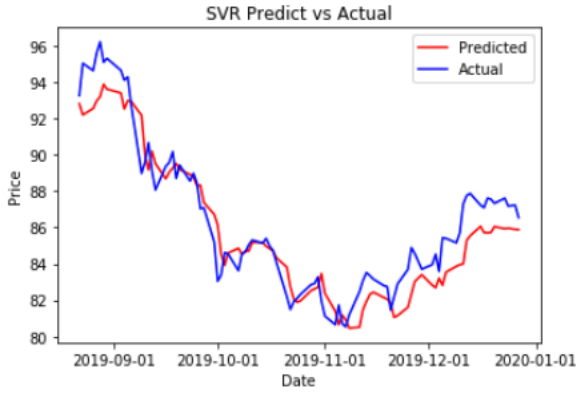
1. Support Vector Regressor (SVR)

Support Vector Regressor is very popular machine learning algorithm for classification and regression analysis. SVM works based on maximum margin hyperplane which maximizes separation between data classes. All the values that are close to hyperplane are called the support vectors. SVR builds upon this hyper plane and use kernels to specify epsilon and provides the output that is support vector regression. SVR relies on kernel feature and this is very important for the model. I am going to use 'linear' kernel for the model.

Prediction depends only of subset of training data, cost function for building the model ignores the training data which is close to the prediction.

For tuning model, I am using set of parameters of C and epsilon and used GridSearchCV method to fit the best parameters.
SVR model

```
#SVR model
RMSE_Score, R2_Score = model_validateResult(model_SVR, model_name = "SVR")
```



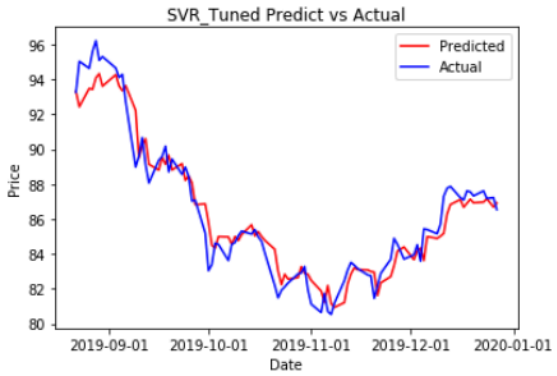

```
SVR RMSE:  1.4055763757946307
SVR R2 score:  0.8777849364184385
```

I am also tuning few parameters and trained the model, result looks like below. I am using set of C and Epsilon and used GridSearchCV to select best hyperparameters.
Tuned model got improvement in score with C =300 and epsilon =0 is.

```
hyperparameters_linearSVR = {
    'C':[0.5, 1.0, 10.0, 50.0, 100.0, 120.0,150.0, 300.0, 500.0,700.0,800.0, 1000.0],
    'epsilon':[0, 0.1, 0.5, 0.7, 0.9],
}
```

```
#SVR model Tuning
RMSE_Score, R2_Score = model_validateResult(model_SVRTuning, model_name = "SVR_Tuned")
```

```
{'C': 300.0, 'epsilon': 0}
```




```
SVR_Tuned RMSE:  1.0378068621009788
SVR_Tuned R2 score:  0.9333732060549099
```

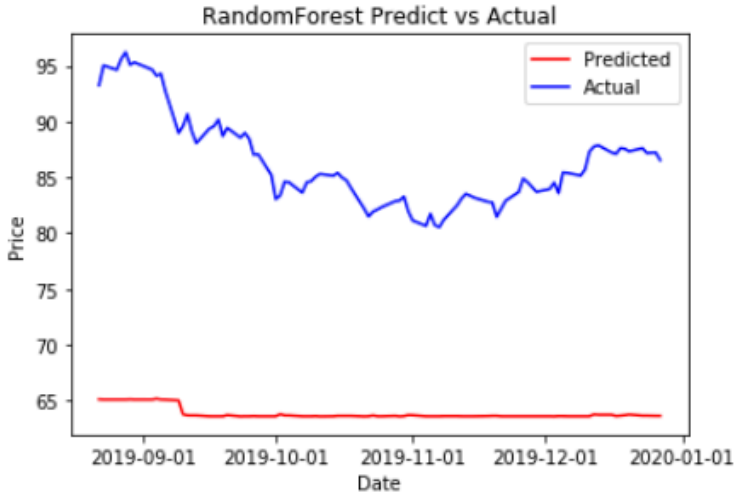
2. Random Forest (RF)

Random Forest algorithm also used for both classification and regression challenges. RF regressors are estimators that fits number of classifying decision trees on samples of dataset. Then they uses averaging technique to improve prediction accuracy.

While running this model it builds multiple decision trees and finally merges them into one to get stable and accurate output. Random forest has function to control the overfitting by setting dropout layers, with help of this model generate better results. The scikit-learn has very good version of model to combine classifiers by averaging their probabilistic prediction instead of letting each classifier vote for a single class.

Initially I am going to use 50 trees in the forest and then for tuning model I am going to use set of hyper parameters. Those are n_estimators, max_features and max_depth and I will use GridSearchCV to select best parameters for the tuned model.
Random forest model:

```
#RF model
#Validate result for randomForest
RMSE_Score, R2_Score = model_validateResult(model_randomForest, "RandomForest")
```



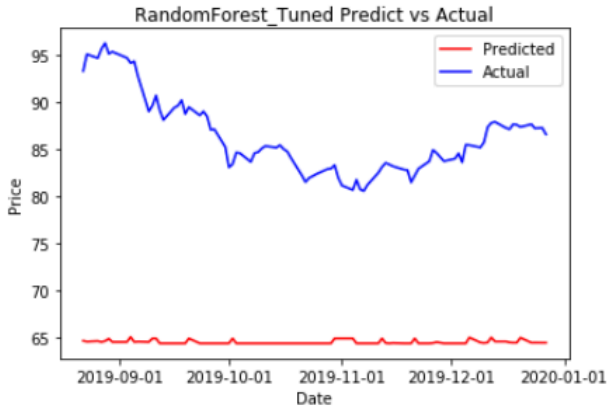

```
RandomForest RMSE:  22.735701793642082
RandomForest R2 score:  -30.976620916210354
```

After tuning model with set of max_depth, max_feaures and n_estimators there is no big difference in metrics.

```
hyper_parameters = {
'n_estimators':[10,15,20, 50, 100],
'max_features':['auto','sqrt','log2'],
'max_depth':[2, 3, 5, 7,10,13, 15, 20],
}
```

```
#Tuned RF model
RMSE_Score, R2_Score = model_validateResult(model_randomForestTuning, "RandomForest_Tuned")
```

```
{'max_depth': 15, 'max_features': 'auto', 'n_estimators': 10}
```




```
RandomForest_Tuned RMSE:  22.107111575448627
RandomForest_Tuned R2 score:  -29.232902678305255
```

3. Long Short-Term Memory (LSTM)

Long Short-Term Memory is one of the deep learning algorithms. It is recurrent neural network and ability to process the entire sequence of data.

Since, I am working on forecasting data with time series data, where each point has a temporal characteristic attached to it, LSTM network is one of best algorithm to predict the temporal data. LSTM works better for time-series based approach because it has memory element. Unlike other model's LSTM is capable of automatically learning features from provided sequence data, support multiple-variate data, and also outputs a variable length sequences that can be used for multi-step forecasting.

It consists of cell, input gate, output gate and forget gate. LSTM cell remembers the data over the timeframe whereas 3 gates regulates the flow of information in and out of the cell. The difference with other algorithm is that, LSTM networks are capable of auto learning features from sequence of data, supports multiple variable data.

Unlike all other solution models, LSTM uses different input format to train the model. Input data needs to be in the format of **(number of records, number of times steps, feature dimension).** So, I will use **convert_LSTM_data**() method to convert to 3D data.

For the LSTM model training I am using few input parameters.
Units : dimension of the inner cells in LSTM
Return_sequence: it checks whether adding more layers to the model or not and ensures that next layers receives sequenced and not random data
Input_shape: to provide number of steps and number indicators (Input_shape(number of time steps, number of indicators/features).
Dropouts: dropout coefficient to avoid overfitting
Dense: number of neurons in the layers.

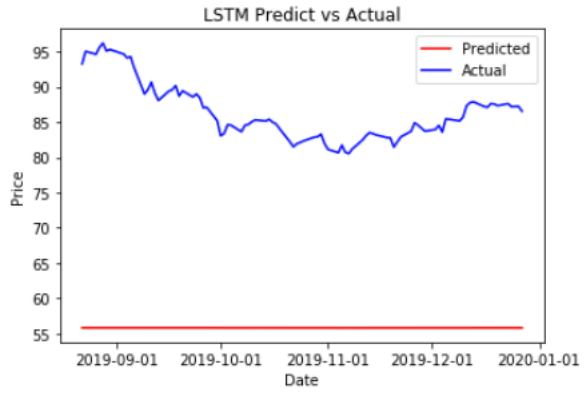
Using Keras packge for the LSTM model. LSTM model is stateful, that means we have to reset the state of the network at the end of each training epoch. To train LSTM model need to instantiate sequential class and designing multiple LSTM layers and dropout layers.

First, I will define a layer with 50 units and return sequence true. I will repeat adding layers to the model 3 more times and I will apply dropout layers after each layer to avoid over fitting of the model. Finally, I will compile the model with mean_squared_error loss function and adam optimizer. First model is fit with 200 epoches and 32 batch size. For the tuned model I increased dropout rate to 0.5 and used batch_size 16.

```
K.clear_session()
x_train_t, x_test_t, validation_x_t, validation_y_t = convertion_model(x_train, x_test, validation_x, validation_y)
# instantiate the Sequential class
model = Sequential()
#Create LSTM and Drop out Layers, return_sequences = True because we will add more layers to the model
#input_shape(number of time steps, number of indicators/features)
model.add(LSTM(units=50, input_shape=(1, x_train.shape[1]), activation='relu', return_sequences=True))
#dropout layter is to avoid over-fitting
model.add(Dropout(0.2))
#Add few more laters and dropouts to the model
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50))
model.add(Dropout(0.2))

#Create Dense layer, number of neurons in dense layer  = 1 becasue we want to predict one value
model.add(Dense(units = 1))
#need to compile before training the data
#
model.compile(loss='mean_squared_error', optimizer='adam')

early_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)

history_model = model.fit(x_train_t, y_train, epochs=200, batch_size=32,callbacks=[early_stop])
  prediction = model_lstm.predict(validation_x_t)
```

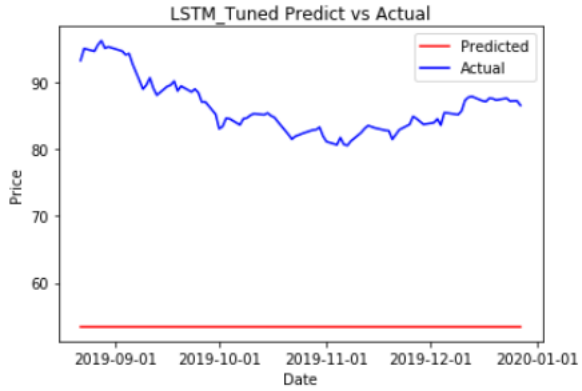
This is how LSTM solves sequential task and performed the validation with validation set.

In my proposal I choosed LSTM model as solution model, sbut after looking at the metrics and trendline I changed my mind.




```
LSTM RMSE:  30.73246315786804
LSTM R2 score:  -57.426595261266364
```

Looks like that LSTM is not fit for this project and also, I tried to improve model by tuning dropouts to avoid over fitting, but still no improvement.




```
LSTM_Tuned RMSE:  33.06680499534911
LSTM_Tuned R2 score:  -66.63948748124066
```

For each solution model, I will validate the model and generate RMSE and R2 scores. Same way I will try to tune those models by setting few sets of hyper parameters and compare the RMSE and R2 score with original model. Finally, I am going to select less RMSE and high R2 score model as best model.

7. Model Comparison

Based on the RMSE and R2 scores of benchmark model and 3 solution models, plot clearly shows that only SVR model has better RMSE and R2 score. When I tuned SVR model with different set of C and Epsilon and used Gridsearch to pick best hyper parameter.
Hyper parameters improved the predication accuracy. So I am going to use SVR with hyperparameters as my best model and going to validate again with test data.

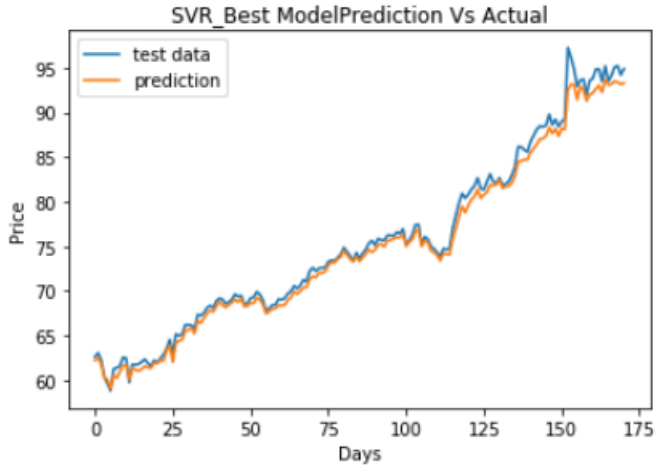
For the Starbucks I got the RMSE and R2 scores as below. I have taken the best score of models after tuning. This validation is based on validation set.

Model	RMSE Score	R2 Score
Benchmark-DTR	21.0626	-26.443
SVR	1.037	0.933
Random Forest	22.10711	-29.23
LSTM	30.73	-57.426

When we observe the RMSE and R2 scores – SVR model has less RMSE and high R2 scores. So, this is the best model.
Validated SVR and benchmark model with test data set and results are as follows.

Model	RMSE Score	R2 Score
Benchmark-DTR	13.95	0.99114
SVR	0.944	-0.93

```
{'C': 300.0, 'epsilon': 0}
SVR_Best Model RMSE:  0.9440338471546559
SVR_Best Model R2 score:  0.9911445502771433
```



And the final prediction price with true value looks like below.

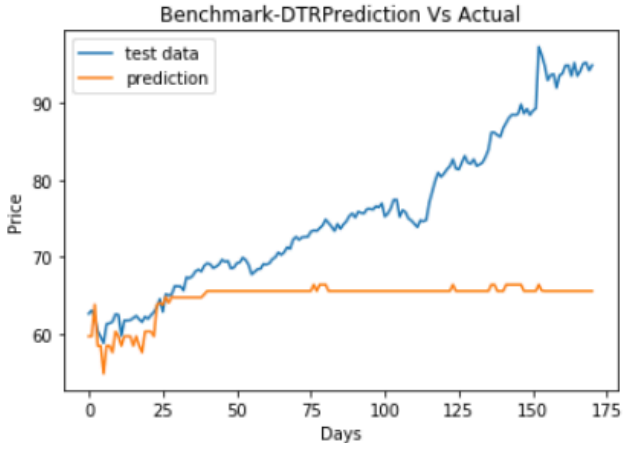
```
{'C': 300.0, 'epsilon': 0}
```

	True_value	Predicted_value
0	62.634216	62.231409
1	63.071400	62.503296
2	62.235886	61.798903
3	60.380272	60.281159
4	59.641914	60.095261
...	...	...
166	94.082199	93.165138
167	95.057190	93.478712
168	95.195076	93.365399
169	94.180672	93.142042
170	94.860222	93.282034

Same way I have validated the benchmark model too and got the better result compared to earlier benchmark model with validation set but it's not best for this project.

```
#Benchmarking model validation with test data
RMSE_Score, R2_Score = bestModel_validateResult(model_Benchmark_DTR, model_name = "Benchmark-DTR")
```

```
Benchmark-DTR RMSE:  13.954225082357135
Benchmark-DTR R2 score:  -0.9348483797483156
```



And the final prediction of benchmark-DTR model with true price is looks like below.

	True_value	Predicted_value
0	62.634216	59.706837
1	63.071400	59.706837
2	62.235886	63.819469
3	60.380272	58.475086
4	59.641914	58.475086
...	...	...
166	94.082199	65.577927
167	95.057190	65.577927
168	95.195076	65.577927
169	94.180672	65.577927
170	94.860222	65.577927

Part-2: ATT

All the above analysis is for the Starbucks and I have used the same flow and done the same implementation for ATT. Implementation development code for ATT is in **Final_Capstone_Stock_Prediction.ipynb** notebook. Finally, I got the best scores for the SVR tuned model.
So, I am considering SVR for validating test set..

If we observe the scores, SVR has less RMSE and high R2 score.

Model	RMSE Score	R2 Score
Benchmark-DTR	3.379	-6.141

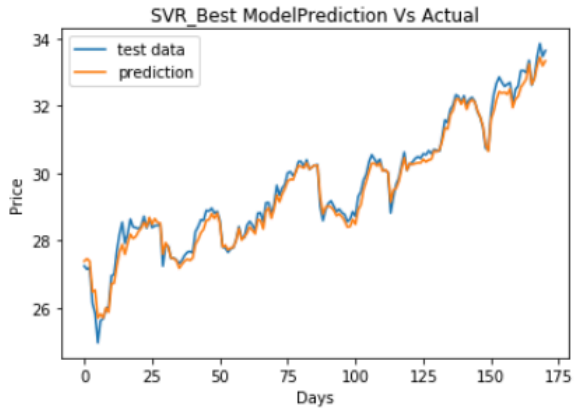
SVR	0.528	0.825
Random Forest	3.489	-6.611
LSTM	4.313	-10.63

Same way after validating best model and benchmark model with test sets result as follows.

Model	RMSE Score	R2 Score
Benchmark-DTR	0.752	0.839
SVR	0.224	0.985

```
#SVR_Tuning model validation with test data
RMSE_Score, R2_Score = bestModel_validateResult(model_SVRTuning, model_name = "SVR_Best Model")
```

```
{'C': 700.0, 'epsilon': 0.1}
SVR_Best Model RMSE:  0.22409840651574023
SVR_Best Model R2 score:  0.9857352265813928
```



Below I am showing df that has actual value along with predicted value for the final model.

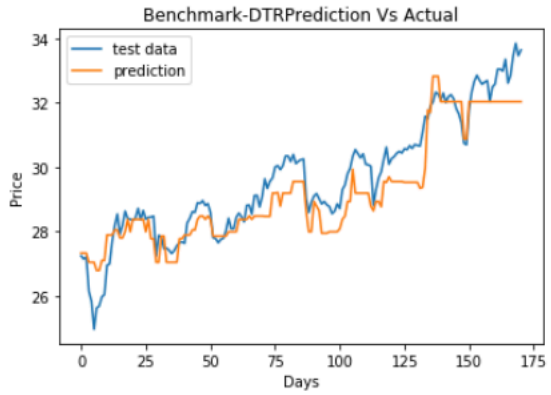
```
{'C': 700.0, 'epsilon': 0.1}
```

	True_value	Predicted_value
0	27.242346	27.384466
1	27.141987	27.453404
2	27.205853	27.380427
3	26.138416	26.469648
4	25.828224	26.526224
...	...	...
166	32.847504	32.795403
167	33.450123	33.172082
168	33.842304	33.441133
169	33.459682	33.182074
170	33.631863	33.327428

And the validation of benchmark model with test data set looks like below.

```
#Benchmarking model validation with test data
RMSE_Score, R2_Score = bestModel_validateResult(model_Benchmark_DTR, model_name = "Benchmark-DTR")
```

```
Benchmark-DTR RMSE:  0.7520435533583852
Benchmark-DTR R2 score:  0.8393527386061143
```



For ATT also benchmark model with testing data set performed well compared to the initial validation. And the predicted value for benchmark looks like below when I compared with actual value.

	True_value	Predicted_value
0	27.242346	27.324455
1	27.141987	27.324455
2	27.205853	27.324455
3	26.138416	27.041630
4	25.828224	27.041630
...	...	...
166	32.847504	32.031246
167	33.450123	32.031246
168	33.842304	32.031246
169	33.459682	32.031246
170	33.631863	32.031246

171 rows × 2 columns

Conclusion

I have implemented stock price prediction using different supervised learning algorithms. After comparing the metrics with other models, I can say that SVR models predicted good result for this project. Actual trendline and predicted trendline clearly shows that SVR model is good fit for both the stocks. After tuning SVR model with C and epsilon prediction is improved. I observed that benchmark model predicted good values on testing set compared to validation set.

I can say that it is impossible to get 99% model prediction because there are too many factors can affect the stock prices. But we can at least predict the general trendline by taking various factors into consideration.

Improvement

There are number of algorithms to predict the time-series analysis and various technical indicators. No model works really well, we just have to improve the models. We can also predict by considering sentiments from news, social media and different sources. For this project I have just selected basic methods, as I move forward I will add more useful inputs to models.

Stock prediction is time-series analysis instead 8 years of historical data, I should have used more. And also, by increasing the epochs and trying different activation function and model functions I would have got better result.

Challenges

I spent most of the time on data exploration, feature extraction. As I am new to ML, it was very hard to decide good model for this project and also, I spent whole 2days to decide technical indicators. Formatting the data for LSTM model was another difficult step for me.

Lesson Learned

Gathering useful data from various sources, preprocessing techniques, researching effective models for the problems, AWS Sagemaker deployment and applying machine learning models to ML problems. Finally got confident that anyone can learn any technology with time and efforts.

This is not the end for me to learn Machine Learning, it's just beginning...

References

https://www.diva-portal.org/smash/get/diva2:1019373/FULLTEXT01.pdf
https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/
https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/
https://en.wikipedia.org/wiki/Stock_market
https://finance.yahoo.com/
https://blog.quantinsti.com/machine-learning-trading-predict-stock-prices-regression/
https://www.youtube.com/watch?v=JuLCL3wCEAk&feature=youtu.be
https://analyticsindiamag.com/hands-on-guide-to-lstm-recurrent-neural-network-for-stock-market-prediction/
https://www.investopedia.com/terms/r/rsi.asp
https://www.dummies.com/personal-finance/investing/stocks-trading/top-technical-indicators-for-stock-investors/
https://www.profit.co/blog/kpis-library/financial/r-squared-r2/#:~:text=R%2Dsquared%20is%20mainly%20used,in%20the%20(near)%20future.