DJIA Predictor

MSBD5012 Machine Learning - Term Project - 30/11/2019

Danish Alsayed, Kabir Rajput,
Ng Chi Ping and Ng Pui Yan

Abstract

Predicting stock prices using machine learning has always been a hot topic in this era. Established in 1885, the DJIA is a price weighted stock index that measures the performance of the stocks of 30 large companies listed on stock exchanges in the United States with a market cap of US$6.56 trillion. It is one of the most followed indexes for equities and if often used in benchmarking portfolios, predicting recessions, US economy performance among other indicators. In this article, we have explored the usage of multiple features with different transformation with multiple machine learning models in order to achieve the highest prediction accuracy for the DJIA index.

Keywords
DJIA; Log Returns; Autocorrelation; Partial Autocorrelation; Stationary; Technical Indicators; Moving Average Convergence Divergence; Bollinger Bands;

1. Introduction

1.1. Project Objective

This project aims to find a set of features, feature transformations and models that will best predict the direction of the Dow Jones Industrial Average (DJIA).

1.2. Project Description

We modelled the problem as a classification problem with 2 signals, Buy and Sell with daily frequency. They are defined as follows:

$$S_t = \begin{cases} 1\,(BUY) & \text{if}, P_{t+1} \geq P_t \\ 2\,(SELL) & \text{if}, P_{t+1} < P_t \end{cases}$$

where S_t is the signal at time t and P_t is the DJIA closing price at time t.

We explored the relationship between DJIA and multiple other assets and then applied various feature transformation to improve our predictions results.

Next, in this report we will describe our base set of features and justify as to why we decided to use them, then we will describe the time series analysis we conducted on our dataset, followed by some financial technical analysis feature transformations used in the project followed by the models that we decided to use and why we used them and finally we will then jump in the results. Due to a large amount of results collected from various features and feature transformation sets we will be dividing the results by the feature and feature transformations that produced them.

2. Feature Selection

1.1. Datasets

In addition to the Open, High, Low and Close prices and daily Volume of the DJIA, we used daily Close prices other indexes, currencies as well as commodities. We used both 5- and 10-year historical data in an attempt to examine the effect on our prediction of simply adding more data with the same feature transformations and models.

The indexes used are NYSE, Russell, NASDAQ, S&P500, FTSE and N225. For commodities we used gold closing prices, for currency exchange rates we used the pairs USDCNY, USDEUR, USD GBP and USDJPY.

2.2. Correlation Matrix

In *Figure 1*, "Close" refers to the close price of DJIA. As shown in Figure 1, There are strong correlations between the closing prices of the DJIA and the other assets. Clearly some assets are positively correlated and some are negatively which made use of as the report will later show. It is also worth noting that the above diagram is plotted based on the raw prices, which we later also transformed in order to make the data more uniform.

Our features set is not uniform in multiple ways. Not all of our features assets trade on the save trading venue which means that they are traded in different currencies e.g. N225 is a Japanese index and thus is traded in the Japanese Yen (JPY) while the FTSE is a British index that trades in GBP while the DJIA trades in USD. In order to take out the effect of the currencies in the prices we calculated the log returns of each of the assets and used them instead of the raw closing prices.

3. Feature Transformation

3.1. Log Returns

Given log returns are beyond the scope of this course, we will go ahead and explain it here.

Log returns are calculated as follows:

$$r_t = ln\,(1+R_t) = ln\,\frac{P_t}{P_{t-1}} = p_t - p_{t-1}$$

$$where\ p_t = ln\,(P_t),\ r_t = return\ at\ time\ t,\ P_t = price\ at\ time\ t$$

As an example, consider a 1 period gain of 5 dollars on a trade of 100 dollars. The percentage return for the trade is 5% while the log returns for the trade is 4.88%.

$$\begin{aligned} r_a &= \frac{FV-PV}{PV} \\ &= \frac{105-100}{100} = .05\ or\ 5\% \end{aligned}$$

$$r_l = \ln\left[\frac{105}{100}\right] = \ln(1.05) = 0.0488\ or\ 4.88\%$$

For smaller returns, arithmetic and logarithmic returns will be similar. However, as these functions move further away from 0, the returns will be further apart as can be seen from the *Figure 2*.

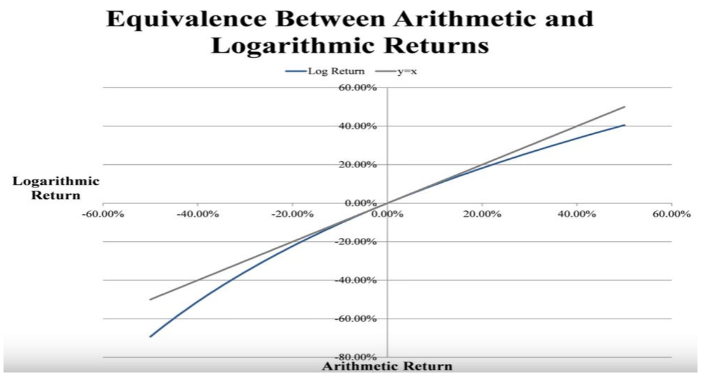
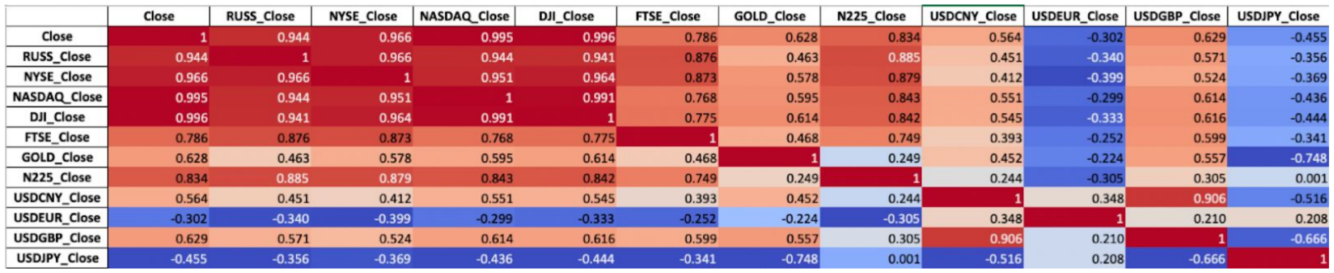


Figure 2: Equivalence Between Arithmetic and Log Returns



	Close	RUSS_Close	NYSE_Close	NASDAQ_Close	DJI_Close	FTSE_Close	GOLD_Close	N225_Close	USDCNY_Close	USDEUR_Close	USDGBP_Close	USDJPY_Close
Close	1	0.944	0.966	0.995	0.996	0.786	0.628	0.834	0.564	-0.302	0.629	-0.455
RUSS_Close	0.944	1	0.966	0.944	0.941	0.876	0.463	0.885	0.451	-0.340	0.571	-0.356
NYSE_Close	0.966	0.966	1	0.951	0.964	0.873	0.578	0.879	0.412	-0.399	0.524	-0.369
NASDAQ_Close	0.995	0.944	0.951	1	0.991	0.768	0.595	0.843	0.551	-0.299	0.614	-0.436
DJI_Close	0.996	0.941	0.964	0.991	1	0.775	0.614	0.842	0.545	-0.333	0.616	-0.444
FTSE_Close	0.786	0.876	0.873	0.768	0.775	1	0.468	0.749	0.393	-0.252	0.599	-0.341
GOLD_Close	0.628	0.463	0.578	0.595	0.614	0.468	1	0.249	0.452	-0.224	0.557	-0.748
N225_Close	0.834	0.885	0.879	0.843	0.842	0.749	0.249	1	0.244	-0.305	0.305	0.001
USDCNY_Close	0.564	0.451	0.412	0.551	0.545	0.393	0.452	0.244	1	0.348	0.906	-0.516
USDEUR_Close	-0.302	-0.340	-0.399	-0.299	-0.333	-0.252	-0.224	-0.305	0.348	1	0.210	0.208
USDGBP_Close	0.629	0.571	0.524	0.614	0.616	0.599	0.557	0.305	0.906	0.210	1	-0.666
USDJPY_Close	-0.455	-0.356	-0.369	-0.436	-0.444	-0.341	-0.748	0.001	-0.516	0.208	-0.666	1

Figure 1: Correlation matrix of all the assets against each other

In stock market modelling, it is common to assume returns are normally distributed. Log returns are far superior to arithmetic returns due to the following reasons [7]:

3.1.1 Log-normality

The sum of repeated samples from a normal distribution is normally distributed. However, the product of repeated samples from a normal distribution is not normally distributed.

3.1.2 Approximate Raw-Log Equality

The following equation holds true for smaller returns. However for larger returns the returns diverge as shown in *Figure 2*.

$$\log(1+r) \approx r\,,r \ll 1$$

3.1.3 Time Additivity

The total return over 2 periods is the same as the sum of the 2 periods added together for log returns. The same is not true for arithmetic returns.

3.1.4 Mathematical Ease (Stochastic Process)

This identity is tremendously useful as shown in the following equation, as much of financial mathematics is built upon continuous time stochastic processes which rely heavily upon integration and differentiation.

$$e^x = \int e^x dx = \frac{d}{dx}e^x = e^x$$

Most of the results we will show later used log returns instead of raw prices unless otherwise indicated.

3.2. Time Series Analysis

Given that we are working with time series, we went ahead and visualized each of our features and the DJIA for a better understanding. The below is a diagram of the decomposed time series of the DJIA close price, we made similar plot for each of our features which you can find in the Charts directory of our project:

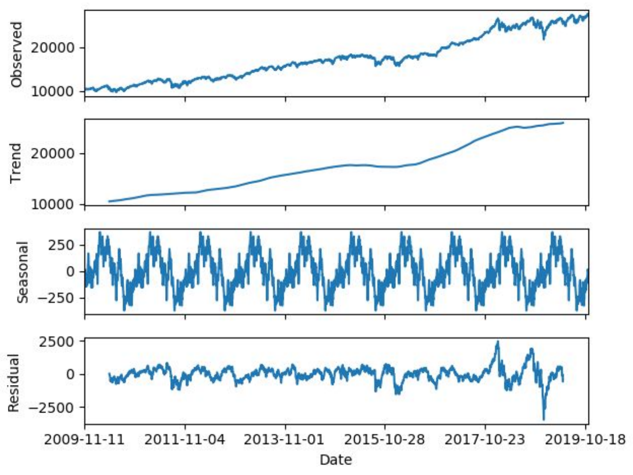


Figure 3: decomposed time series of the DJIA close price

Next we will introduce some concepts that are beyond the scope of the course but were used for feature transformations in our project as will later be shown.

3.2.1 Stationarity

A stationary time series is a time series whose mean, variance, autocorrelation, etc. are all constant with time [5]. Most statistical forecasting techniques operate on the assumption that a time series can be stationarized which makes them easy to predict. Note that measure like means, variances, and correlations are only useful if the series is stationary. E.g, if the time series is increasing over time, the sample mean and variance will increase with the size of the sample, and will always underestimate the mean and variance for the future.

However, as we can see in the decomposed plot above, the DJIA as well as all of our features are far from stationary when expressed in their original units of measurement i.e. US dollar values, and even after deseasonalizing it will still show trends, cycles or even random-walking, among other non-stationary behavior. The DJIA and most of our features show a stable long-term trend and moreover they tend to return to the trend line after a disturbance as we can see above round about the 2015 mark. So we went ahead and tred-stationarized our features by including the time index as an independent variable in our models (which, surprisingly did not improve our results). It was clear that just de-trending wasn't enough to make our series stationary, so we performed a period-to-period differencing i.e. transformed our series to a difference-stationary to see the effect, as will be shown later.

3.2.2 Seasonality

Seasonality is the presence of variations that occur at certain and noticeable intervals that are more or less regular e.g. monthly, quarterly or even yearly etc.

As we can observe, there is clearly a seasonal factor in the prices of the past 10 years of the DJIA. We removed the seasonal factor and collected results (as later will be shown) using additive seasonality [1] which is described as:

$$S = Y - (T + C + I)$$

where S is the seasonal values, Y is the actual data values, T is the trend values, C is for the cyclical values and finally I is for the irregular values.

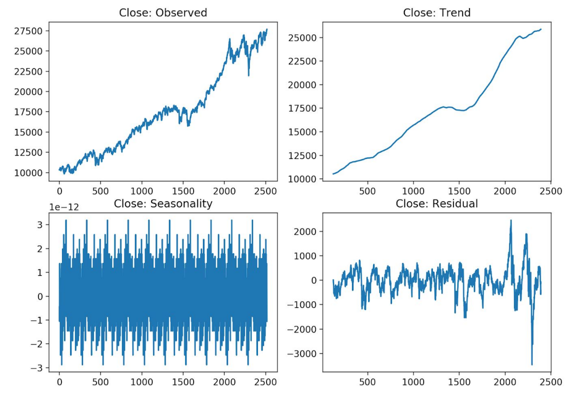


Figure 4: decomposed time series after deseasonalizing

From Figure 4, As we can see, the seasonal component is nearly eliminated.

3.2.3 Autocorrelations and partial autocorrelation

Autocorrelation function (ACF) or serial correlation, is the correlation of a value with a delayed copy of itself as a function of time delay [3]. Informally, it is the similarity between observations as time progresses.

Partial autocorrelation function (PACF) describes the partial correlation of a stationary time series with its own lagged values but regressed the values of the time series at all the shorter lags; note that ACF does not care about other lags [3].

Note that these concepts were only used in order to fit ARIMA model. We plotted the ACF and PACF for the DJIA as shown below (note that this part was done in R):

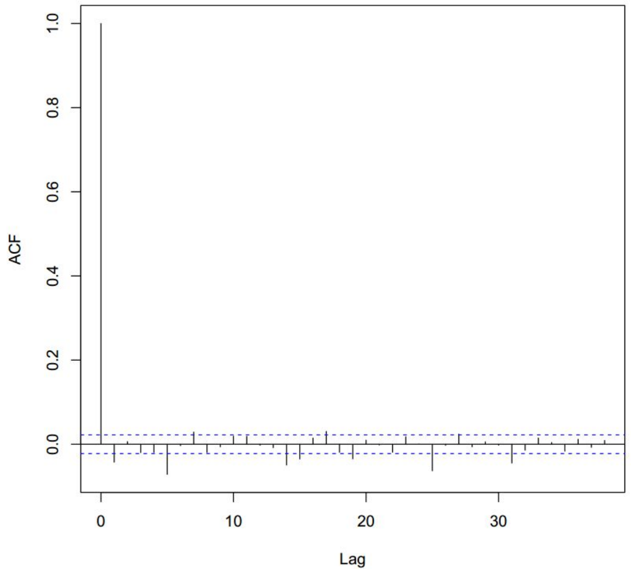


Figure 5: ACF for the DJIA

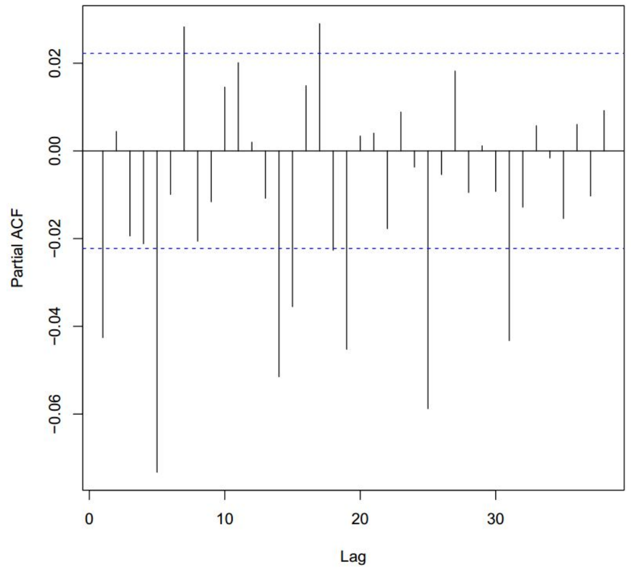


Figure 6: PACF for the DJIA

3.2.3 White Noise

Time series that show no to little autocorrelation are called white noise. For a white noise series, we expect 95% of the spikes in the ACF to lie within ±2/√T where T is the length of the time series [3]. We plotted these bounds on the ACF chart above using the blue dashed lines around the x axis. If substantially more than 5% of spikes are outside these bounds, then the series is not white noise. So we concluded that DJIA is not white noise, however, some of our features were white noise and we made a data set without them to see the difference on the predictions as will be shown later. Here are the ACF plots for our white noise features:

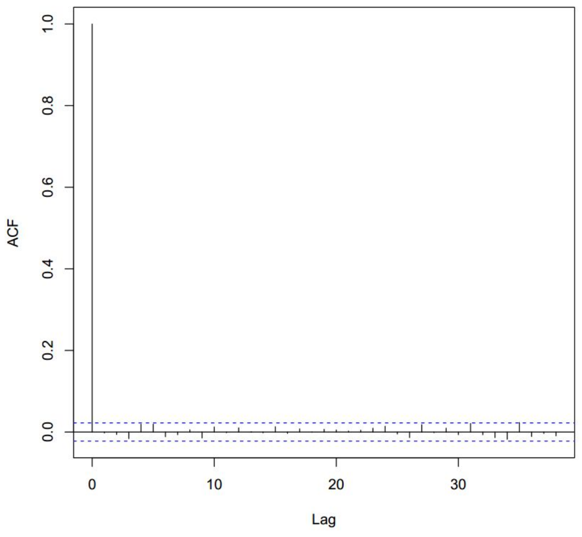


Figure 7: ACF Plot for USDEUR feature

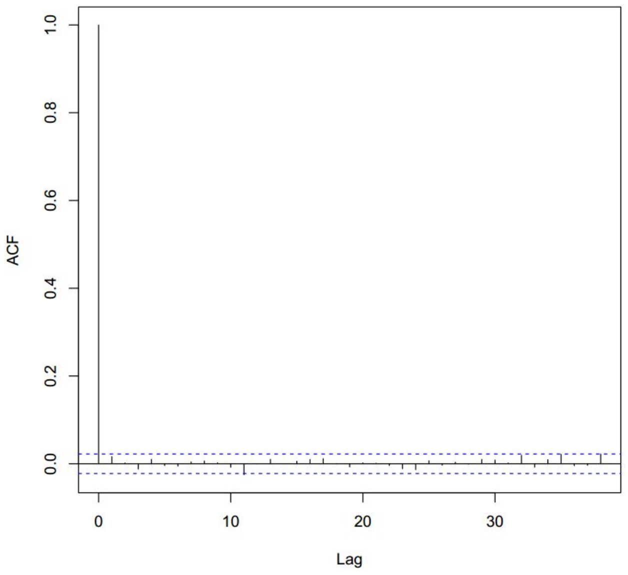


Figure 8: ACF Plot for USDGBP feature

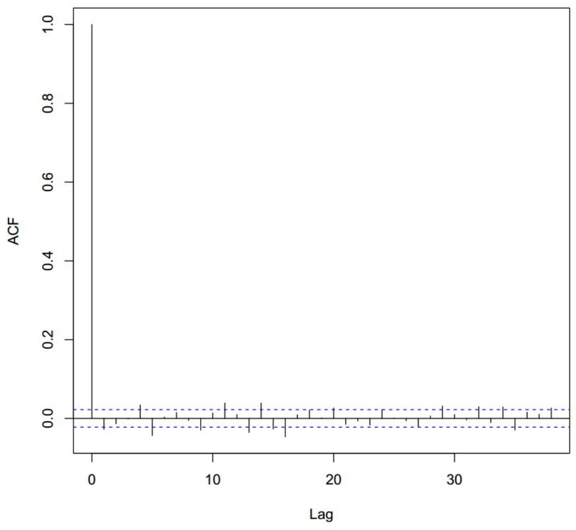


Figure 9: ACF Plot for USDJPY feature

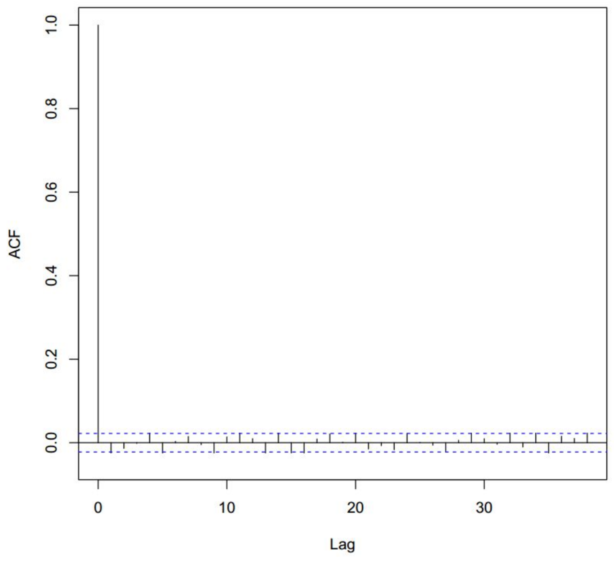


Figure 10: ACF Plot for USDCNY feature

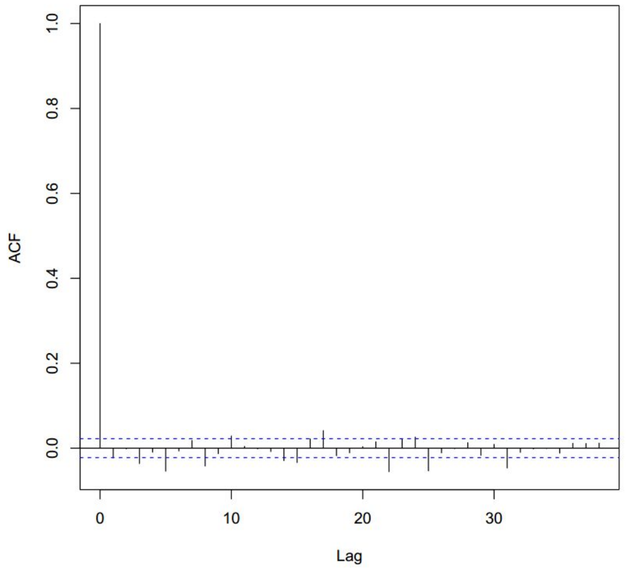


Figure 11: ACF Plot for NASDAQ Close feature

We confirmed our results via the Ljung Box test and the above conclusion was endorsed that USDEUR, USDJPY, USDCNY, USDGBP and NASDAQ close are white noise.

3.3. Technical Analysis

Technical analysis (TA) is a form of analysis used by analysts who believe they can predict future stock performance based on past trends and patterns. Well-known technical indicator includes Moving Average Convergence Divergence (MACD) and Bollinger bands.

3.3.1 Moving Average Convergence Divergence

The MACD indicator depends on three time parameters, namely the time constants of the three EMAs. The notation "MACD(a,b,c)" usually denotes the indicator where the MACD series is the difference of EMAs with characteristic times a and b, and the average series is an EMA of the MACD series with characteristic time c. These parameters are usually measured in days.

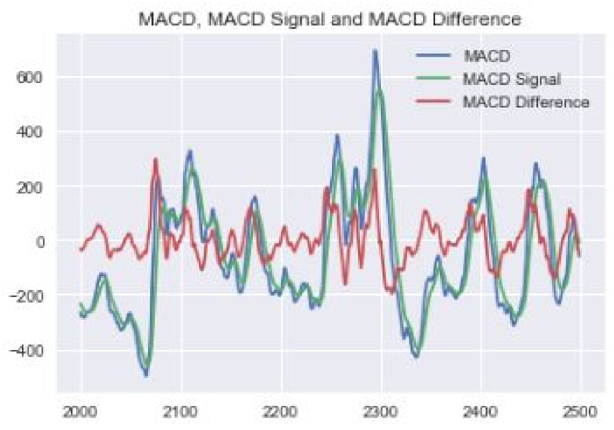


Figure 12: MACD, MACD Signal and MACD Difference

3.3.2 Bollinger bands

Bollinger Bands display a graphical band (the envelope maximum and minimum of moving averages) and volatility (expressed by the width of the envelope) in one two-dimensional chart.

Bollinger Bands consist of an N-period moving average (MA), an upper band at K times an

N-period standard deviation above the moving average (MA + Kσ), and a lower band at K times an N-period standard deviation below the moving average (MA − Kσ). The chart thus expresses arbitrary choices or assumptions of the user, and is not strictly about the price data alone.

The Bollinger bands specify underbought and overbought areas for the asset class. In case the stock is trading below the lower range of the Bollinger bands, it is a good idea to buy the stock. In case the stock is trading above the Bollinger bands, it makes more sense to sell the stock as it is in the overbought range.

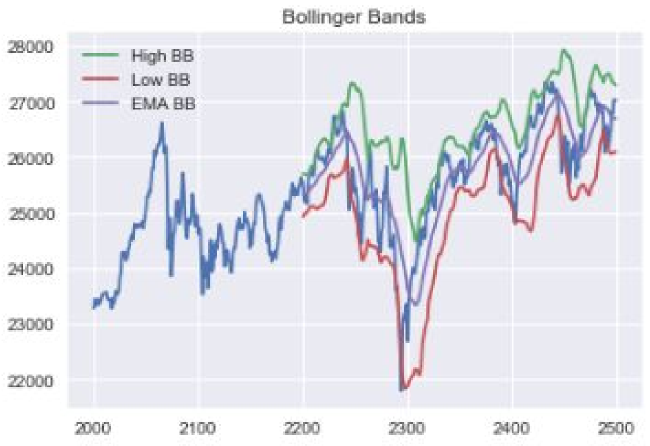


Figure 12: Diagram of Bollinger bands

We used a list of 50+ technical indicators. Some of the indicators are listed below:

momentum_rsi	volatility_kcl
momentum_mfi	volatility_kchi
momentum_tsi	volatility_kcli
momentum_uo	volatility_dcl

Table 1: List of Technical Indicators used

3.3.3 Feature Importance of Technical Indicators

After selecting the features we ran multiple models to test out which model performs best for the dataset. We also ran a feature importance test on the model. Feature importance for the whole dataset was testing using the XGboost algorithm and the final results show the following indicators for the feature importance.

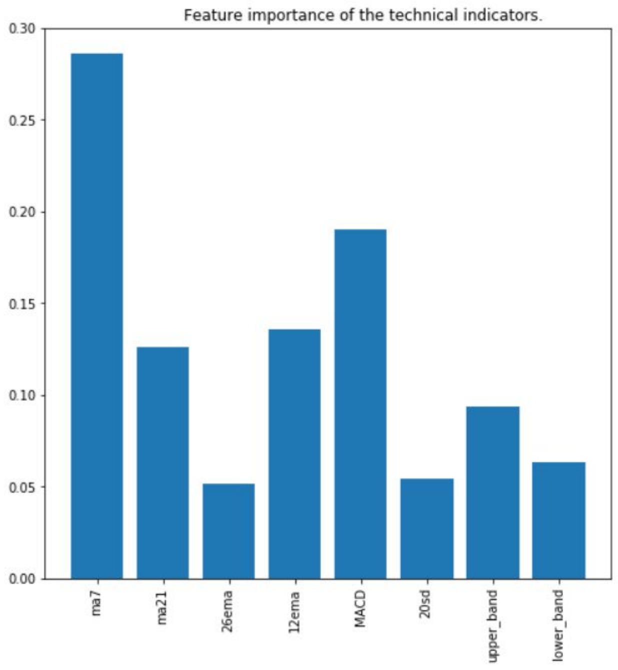


Figure 13: Feature Importance of the Technical Indicators

It is not a surprise that moving averages are the best features in the dataset. In the real world as well, moving averages including EMA and MACD are constantly used as the most popular tools for technical analysis. This feature importance graph reinstates the above-mentioned point about the practical applications of technical indicators like moving averages. Another great approach to further look into it ARIMA GARCH, we will further look into this approach as well.

4. Models

4.1. Adaboost Classifier

Adaboost was another ensemble classifier, which combined weak classifier algorithms to form a strong classifier [2]. The model relied on dataset previously trained, and the weights generated according to the accuracy it returned. The model then determined the updates of dataset using these weights for the input of next sub-model.

Configuration of our adaboost classifier model is as follows:

Cross-validation*: 3 folds*

Epoch*: 10*

Parameter grid:

Loss function: Linear, Square, Exponential

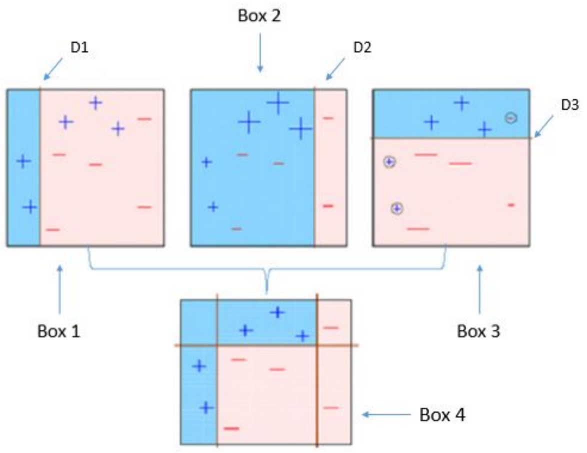


Figure 14: Adaboost Classifier [2]

4.2. Random Forest Classifier

Random Forest was an ensemble model made of a multitude of decision trees, generated through the use of bootstrapping [4]. Predication was often made from average voting.

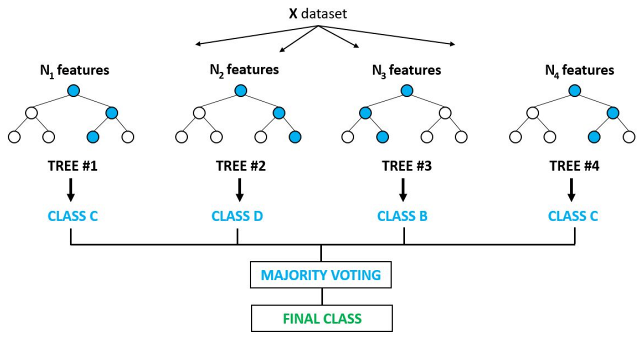


Figure 14: Random Forest Classifier [4]

Configuration of our random forest classifier model is as follows:

Number of Iteration*: 100*

Cross-validation*: 3 folds*

Parameter grid*:*

Depth of tree: 10 to 100 with a step of 10, and unrestricted

Minimum number of leaves: 1, 2, and 4

Minimum number of samples for a node split:

2, 5, 10

Number of trees: 600 to 2,000, with a step of 200

Number of features to consider for best split: square root

Use bootstrap for tree generation: both

4.3. Gradient Boosting

Gradient boosting produced a prediction model that ensembled weak decision trees. Unlike Random Forest, it combined the trees at the beginning of the process. In the iterative processes the decision tree considered the residual error, instead of updating the weight of data points produced in the previous iteration.

Configuration of our gradient boosting model is as follows:

Cross-validation*: 5 folds*

Minimum number of samples for a node split: *2*

Minimum number of leafs*: 1*

Number of features to consider for best split: *square root*

Parameter grid*:*

Maximum depth of tree: 2 to 7

Number of trees: 100, 250 to 1,750, with a step of 250

Learning rate: 0.001, 0.005, 0.01, 0.05, 0.1, 0.15

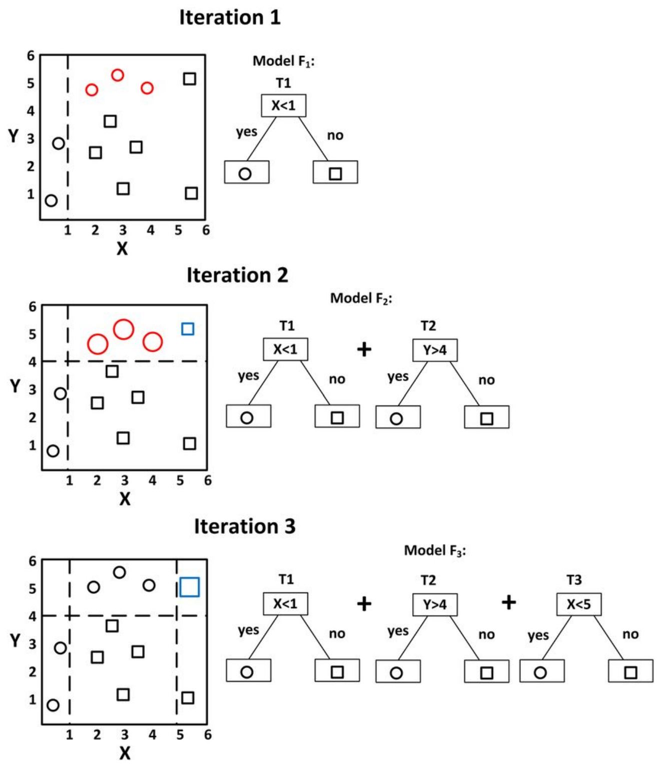


Figure 15: Gradient Boosting [8]

4.4. K-Nearest Neighbors (KNN)

K Nearest Neighbors was a supervised learning algorithm; classification was based on the distance between the data points and its nearest neighbors.

Configuration of our KNN model is as follows:

Cross-validation: *2 folds*

Parameter grid:

Number of neighbors: 1 to 30

Power parameter for the Minkowski metric: 1 to 9

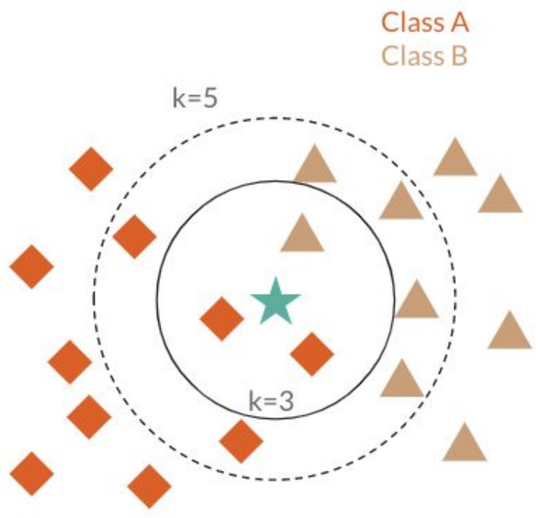


Figure 16: KNN

4.5. Support Vector Machine (SVM)

Support Vector Machine applied kernel function to perform discriminative classification by finding the hyperplane that maximised the margin between the two classes.

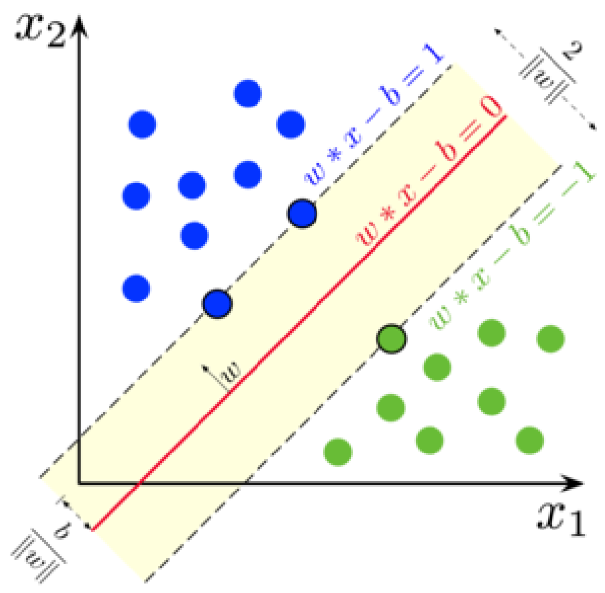


Figure 17: SVM

Configuration of our SVM model is as follows:

Parameter grid:

Penalty parameter: 0.001 to 1,000 with a step of 10

Kernel: RBF kernel

Kernel coefficient: 0.001& 0.0001

4.6. Logistic Regression

Logistic regression is a binary regression model that used a logistic function to model/describe a dichotomous dependent variable.

Configuration of our logistic regression model is as follows:

Parameter grid:

Inverse of regularization strength: 0.001 to 1000 with a step of 10

Optimisation solver: Newton-conjugate gradient, L-BFGS, Lib-linear, SAG, SAGA

4.7. Forward Feed Neural Network (FNN)

Forest

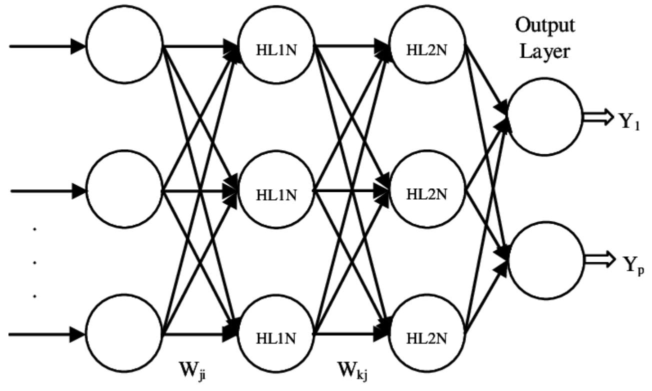


Figure 18: FNN

A Forward Feed Neural Networks was the simplest neural network constructed with layers of hidden units called perceptions. Information moved only in one forward direction from input nodes to the output nodes.

Configuration of our FNN model is as follows:

***Epoch**: 100*

***Sequential Model**:*

2 Rectified Linear Unit (ReLU) Layers

1 Sigmoid Layer

***Loss Function:** Binary Cross-entropy*

***Early stopping** with 10-epochs applied*

4.8. Long Short-Term Memory (LSTM)

LSTM was a variation of RNN, which was invented for predicting sequential data. It implemented Input, Output, Forget gates and internal memory across consecutive states, which allowed valuable information be retained until a signal triggered the model to discard it.

Configuration of our LSTM model is as follows:

***Epoch:** 15*

Steps per epoch: *10*

Optimiser: *Adam*

Loss function: *Mean Absolute Error*

LSTM Model:

1 LSTM Layer

2 ReLU Layers

1 Linear activation Layer

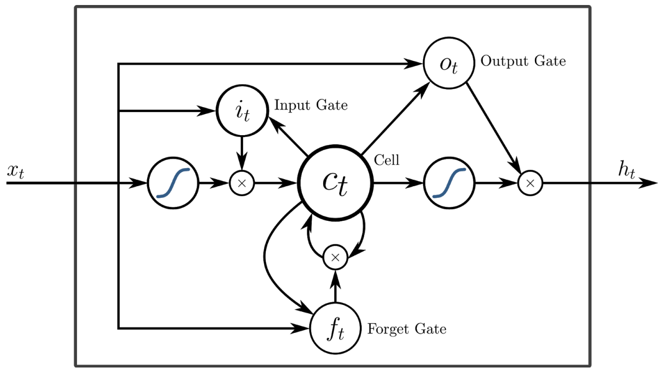


Figure 19: LSTM

4.9. Autoregressive Integrated Moving Average (ARIMA)

ARIMA - Autoregressive integrated moving average was a regression analysis that predicted future values of a series based entirely on its own inertia, lagged or prior values.

p, d, q Parameters :

p, number of time lags: 2

d, degree of differencing: 1

q, order of moving average model: 2

‘p’ is the order of the ‘Auto Regressive’ (AR) term. It refers to the number of lags of Y to be used as predictors. And ‘q’ is the order of the ‘Moving Average’ (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.

The first step to build an ARIMA model is to make the time series stationary. Therefore we applied differencing which made the data more stationary as described in the stationarity section.

The value of d, therefore, is the minimum number of differencing needed to make the series stationary. And if the time series is already stationary, then d = 0.

AR Model - Yt depends only on its own lags. That is, Yt is a function of the ‘lags of Yt’.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

MA Model- Yt depends only on the lagged forecast errors.

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

ARIMA Mode - Predicted Yt = Constant + Linear combination Lags of Y (upto p lags) + Linear Combination of Lagged forecast errors (upto q lags)

While we were differencing to make the time series stationary, we were careful to not over-difference the series. Because, an over differenced series may still be stationary, which in turn will affect the model parameters. The right order of differencing is the minimum differencing required to get a near-stationary series which roams around a defined mean and the ACF plot reaches to zero fairly quick. In the event, if we can’t really decide between two orders of differencing, then we go with the order that gives the least standard deviation in the differenced series.

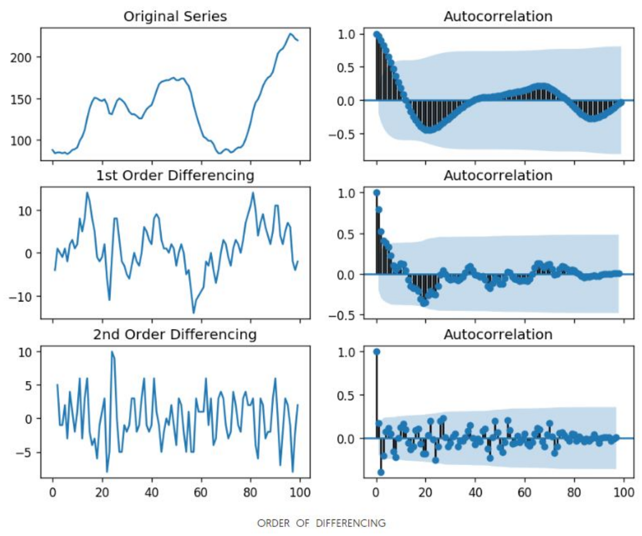


Figure 20: Autocorrelations of Time Series

For the above series, the time series reaches stationarity with two orders of differencing. But on looking at the autocorrelation plot for the 2nd differencing the lag goes into the far negative zone fairly quick, which indicates, the series might have been over differenced.

The next step is to identify if the model needs any AR terms. We found out the required number of AR terms by inspecting the Partial Autocorrelation (PACF) plot.

Partial autocorrelation can be imagined as the correlation between the series and its lag, after excluding the contributions from the intermediate lags. So, PACF sort of conveys the pure correlation between a lag and the series.

5. Results

In the previous section, important features were identified and extracted to run on the models introduced. Results were summarised in the below section.

5.1. Datasets with Technical Features

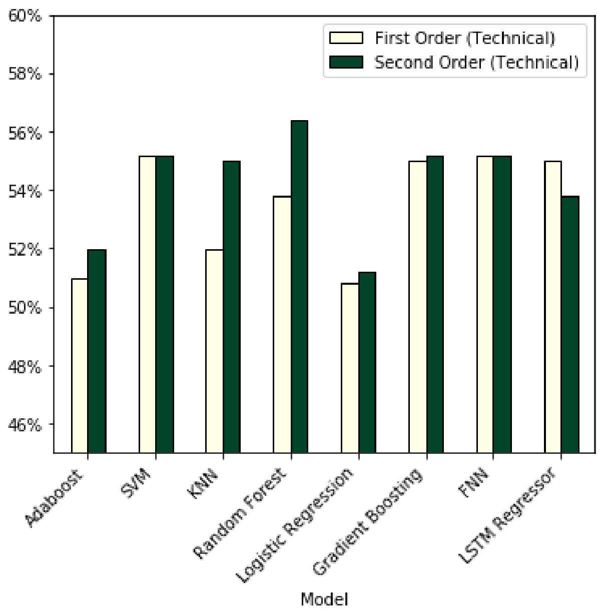
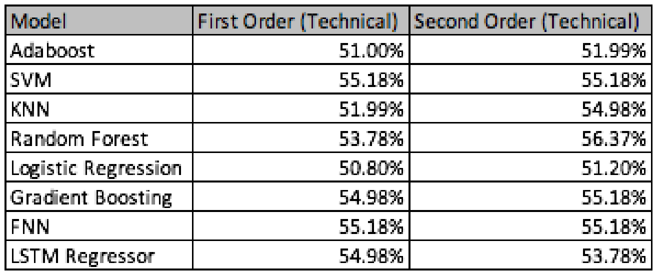


Figure 21: Results between order-differencing with technical features

The use of technical indicators for prediction was not rare in the finance industry. In real life, these features were chosen by analysts and experts, while in this project, these importance were learnt through the use of machine learning approaches.



Model	First Order (Technical)	Second Order (Technical)
Adaboost	51.00%	51.99%
SVM	55.18%	55.18%
KNN	51.99%	54.98%
Random Forest	53.78%	56.37%
Logistic Regression	50.80%	51.20%
Gradient Boosting	54.98%	55.18%
FNN	55.18%	55.18%
LSTM Regressor	54.98%	53.78%

Table 2: Results between order-differencing with technical features

The average accuracy of running technical data with first order differencing was 53.49% versed the score of 54.23% using second-order differencing. The highest accuracy achieved was 56.37% by Random Forest on data with second-order differencing; however, this top score became trivial compared with the scores produced using base features.

5.2. Training/Testing Datasets

Daily closing prices of the mentioned currencies and indices across 10 years were used for making the prediction for the next trading action. This dataset produced the 80/20 training/testing split where various transformations were applied, including: First- and Second-order differencing, which helped make the data more stationary as described in the previous sections. Seasonal adjustment was applied to the dataset for producing Stationary values as one of the feature transformations.

Instead of using features of a single day for prediction, data was reconstructed to include features in the previous n days, where n is the window size. This experiment exploited the signal/pattern potentially hidden in a series, rather than a data point.

These transformations of dataset were fitted into models described in the previous section, and the results were explained below.

			(NASDAQ & Currencies Excluded)	
Model	1st Order	2nd Order	1st Order	2nd Order
Adaboost	59.56%	59.76%	62.17%	57.14%
Random Forest	51.11%	53.72%	51.11%	54.12%
SVM	55.93%	55.53%	55.94%	55.94%
KNN	54.33%	55.13%	54.33%	53.32%
Gradient Boosting	56.30%	55.90%	55.70%	55.90%
Logistic Regression	56.13%	55.13%	56.14%	54.73%
FNN	55.94%	55.94%	55.94%	55.94%
LSTM Regressor	54.84%	52.02%	52.82%	52.82%

Table 3: Results between different order-differencing and features removal

5.3. Feature Transformations - Order Differencing

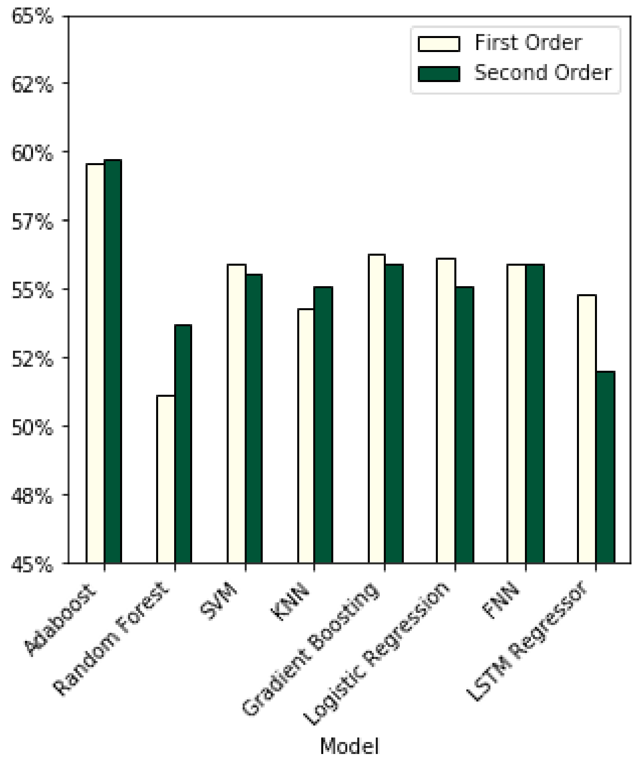


Figure 22: Results between order-differencing

The results indicated that half of the eight models performed better with second order differencing and the average improvement is 0.90%. Among these, Random Forest gave a 2.61% of improvement with second order differencing. The remaining models showed an average of 1.16% decrement in their accuracy.

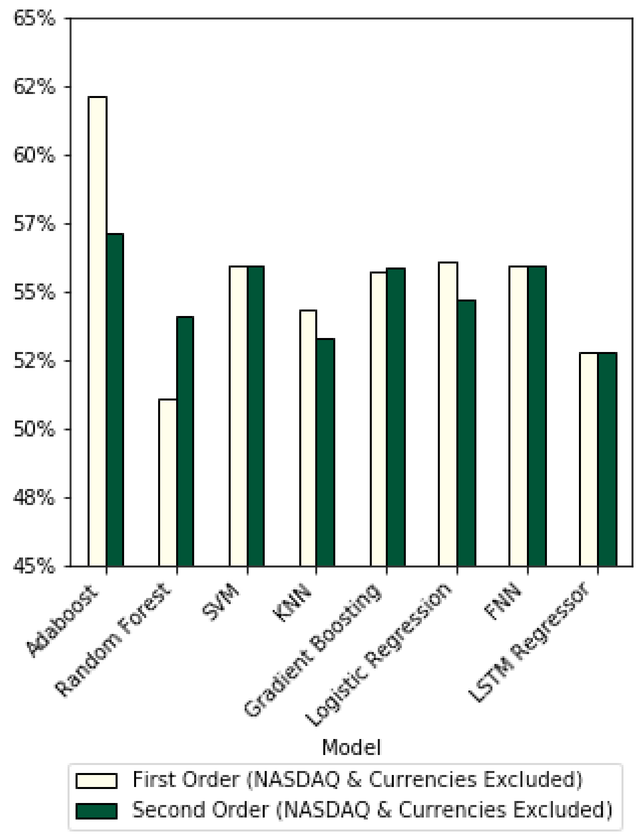


Figure 23: Results between order-differencing with features removal

Considering datasets with NASDAQ and all currencies removed, the average improvement of taking a 2nd order differencing dropped to 0.80%; however, Random Forest gave a higher improvement of 3.01% with second order differencing. A stronger negative influence of 1.86% on Adaboost, KNN and Logistic Regression was observed. Among them, Adaboost's performance was largely deteriorated with a 5.03%.

5.4 Features Removal

Certain features ("X Features") were removed from the dataset for, or with a view to reducing the potential "white noise" that reduced model accuracy; in this subsection, the NASDAQ and closing price of all currencies were excluded. A column that emphasised the orderal property of the records was appended, since the date column was removed.

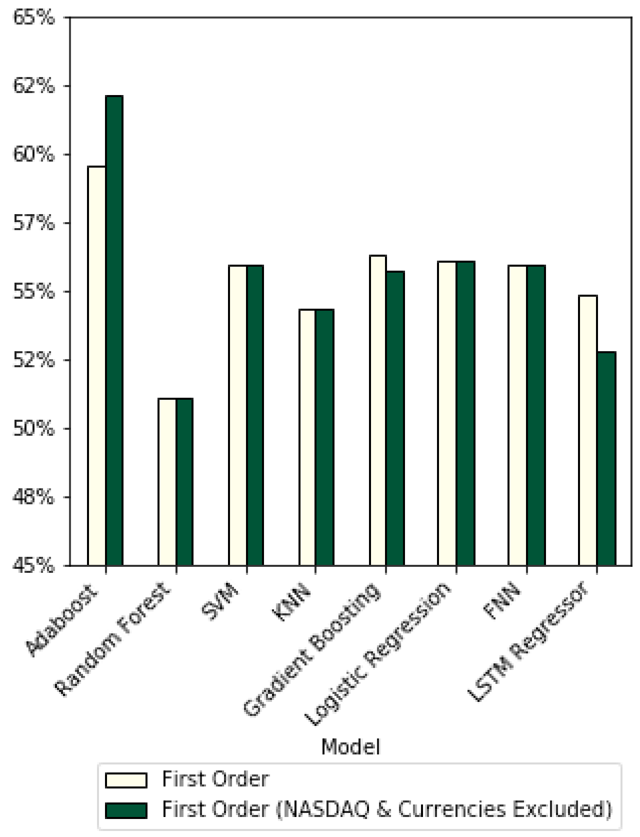


Figure 24: Comparison between features removals of first order-differencing

While taking first order differencing on the values, the removal of X Features did not show much impact on most models; only Adaboost and LSTM Regressor were reported to have +2.61% and -2.02% influence on their accuracies.

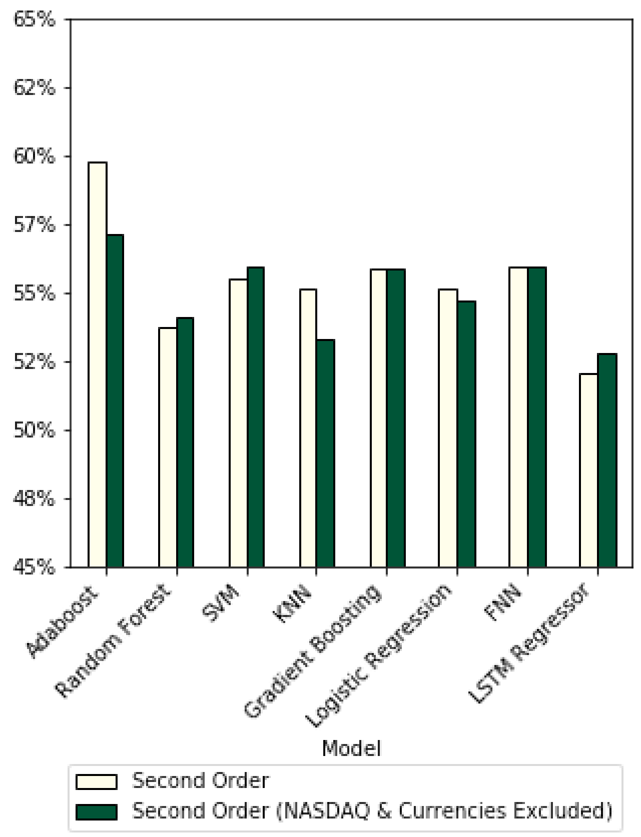


Figure 25: Comparison between features removals of second order-differencing

While applying data value with second order differencing, accuracies of Adaboost and KNN dropped by 2.62% and 1.81% respectively. Other models remained relatively stable against the removal of X Features.

Models behaved differently to feature transformations. From the result, features transformations appeared to have little influence on Forward Feed Neural Networks and Gradient Boosting, their variances in accuracy percentage were 0.00% and 0.06% respectively. Adaboost was found very sensitive to feature transformations, and the variance is approx. 4.22%.

5.5. Stationary Features

By fitting stationary features into the models, the accuracies obtained were lower than the average of other transformations. Adaboost achieved the highest accuracy of 59.80%, but the score was still 2.37% lower than the top runner. The removal of seasonality did not appear to help the prediction.

Model	First Order	First Order (NASDAQ & Currencies Excluded)	Stationary
Adaboost	59.56%	62.17%	59.80%
Random Forest	51.11%	51.11%	50.40%
SVM	55.93%	55.94%	54.00%
KNN	54.33%	54.33%	52.20%
Gradient Boosting	56.30%	55.70%	54.00%
Logistic Regression	56.13%	56.14%	54.00%
FNN	55.94%	55.94%	54.00%

Table 4: Comparison between order-differencing and stationary sets

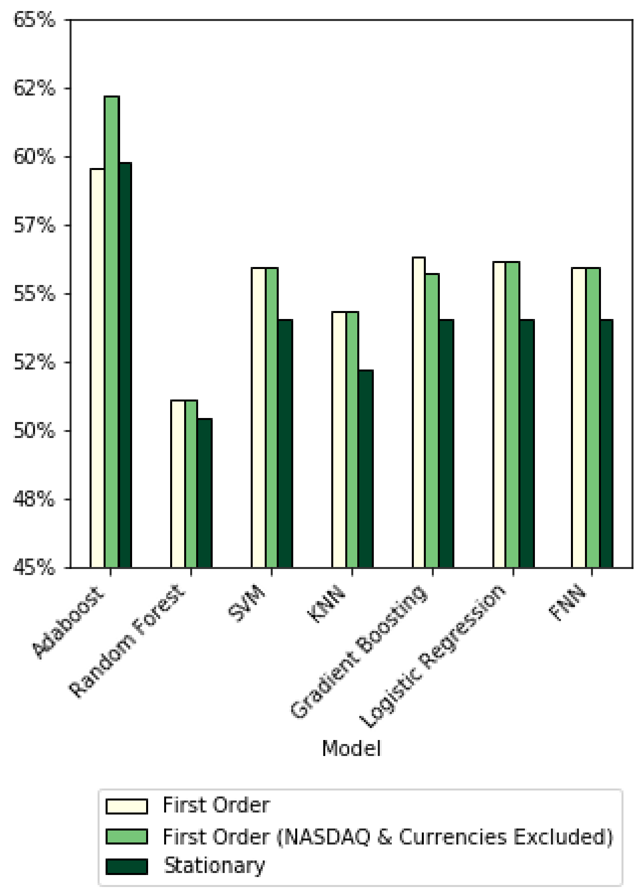


Figure 26: Comparison between order-differencing and stationary sets

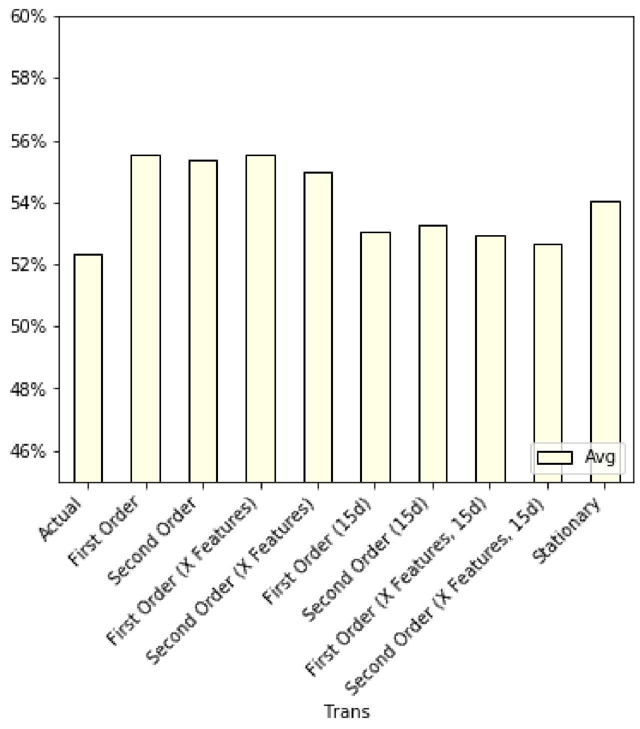


Figure 27: Results with a 15-day window

5.6. Additional Experiment & Results

Instead of fitting data of a particular day into the models, each data point was restructured to include data of the past n consecutive days. The number of features considered would become n times the original feature size, e.g. with a 15-day window, a data point expanded to contain 240 features from the original 16. From this experiment, however, the accuracies of the models dropped significantly; it was suspected that a larger window size may represent pattern for a long run better. Tuning of window size was the next topic to explore in the future.

6. Conclusion

Figure 28 summarises the accuracies each model obtained. Only the 5 transformation categories with significant accuracy were compared.

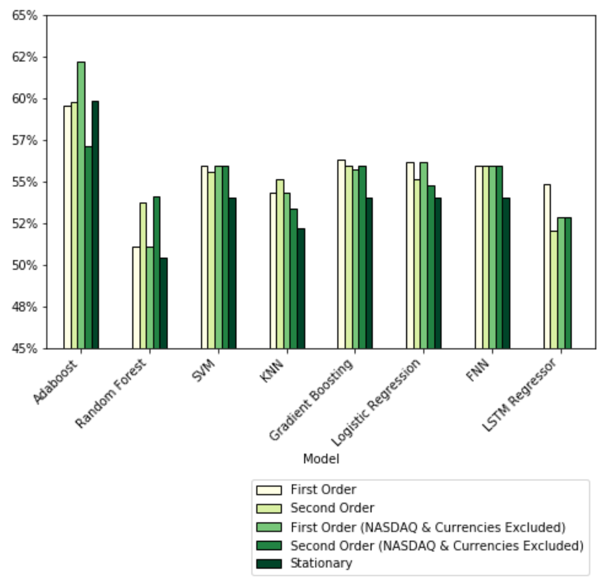


Figure 26: Summary of all feature transformations

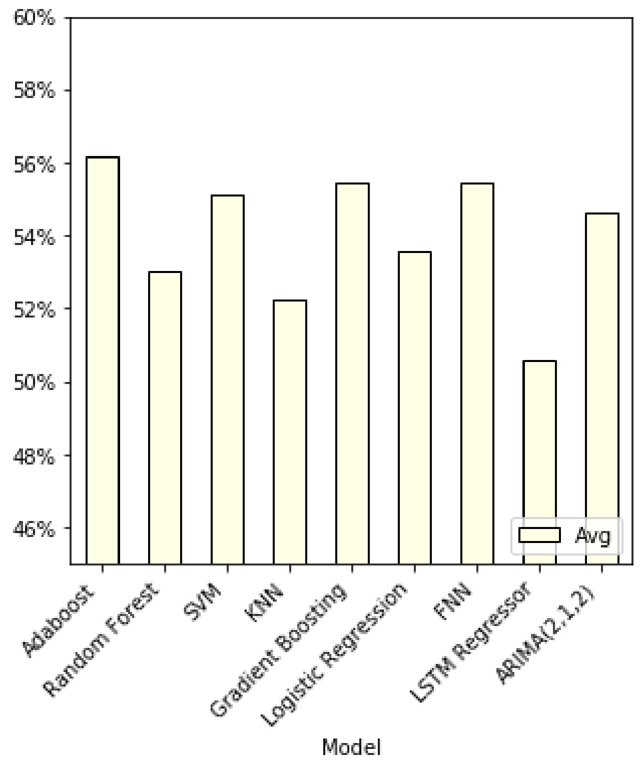


Figure 27: Summary of the average results of all models

Adaboost outplayed other models across all data transformation categories. Even considering it's

worst prediction using data with second order value differencing, NASDAQ and all currencies column removed, the accuracy of 57.14% was still above all other models.

Using the ARMIA(2,1,2) model, results were transformed back to binary signal using the transformation applied to the input dataset, and it scored an average of 54.6% accuracy, still outplayed almost half of the other models.

LSTM was designed to make predictions based on sequential data but it did not perform very well in our experiments. It was suspected that only daily closing values may not be sufficient for the neural network to predict the next trading action; and the sampling interval may overgeneralise meaningful signal and the volatility of the index throughout the day.

In this study, the predicted labels were discrete values of either BUY or SELL action - this value was disjointed and unordered; since Adaboost was an ensemble office multiple decision trees, this dichotomous nature matched the capability of such classifiers. The high accuracy of the model also indicated the potential drawback of overfitting; as one key objective of Adaboost was to minimise training error.

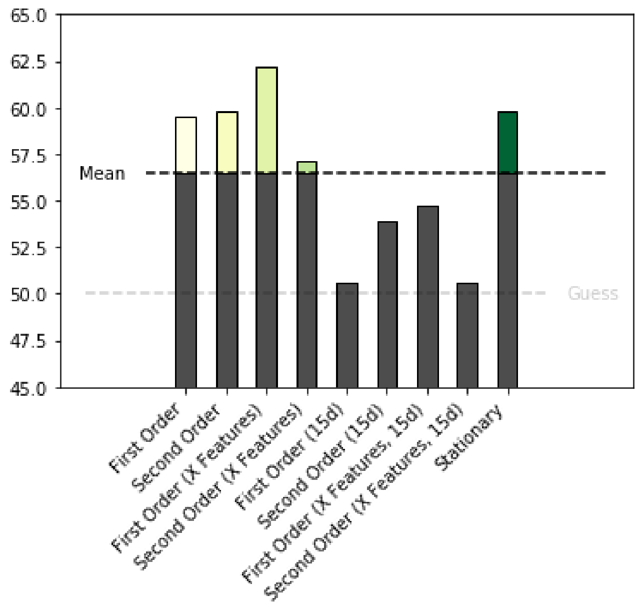


Figure 28: Comparison of prediction accuracy of Adaboost among feature transformations

Before claiming Adaboost the winning model, an experiment was performed to verify its accuracy. Since cross validations could always mitigate overfitting problems[], the model then ran the dataset with various splits, starting from the minimal 2-fold to 15-fold. Only dataset with X Features removed and with first order value differencing was used, and the results were shown in the figure below.

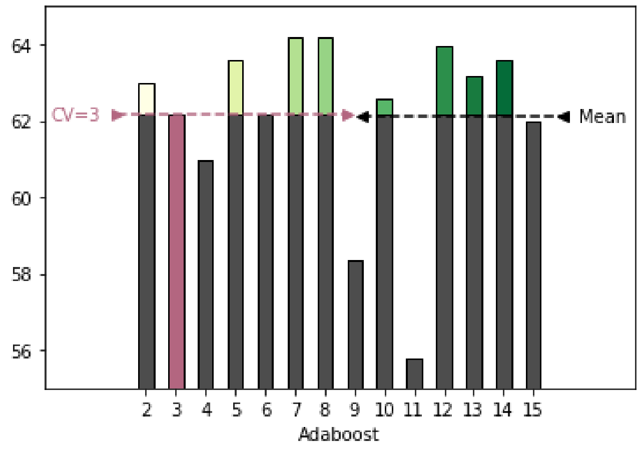


Figure 29: Results of cross validation parameter search for Adaboost

Among the 14 test cases using different cross validation splits, the average accuracy was 62.11%, while the winning score of the model was very close, at 62.17% when cv=3. With other cv values, e.g. cv=11 the accuracy dropped drastically to only 55.75%, and when cv=7 and 8, the value soared to 64.19%. This test results implied that, despite the overfitting nature of Adaboost, the number of cross validation chosen was appropriate and the result tended to be reliable. So the winning model was Adaboost using first order differenced dataset with NASDAQ and Currency values removed, running with a 3-fold Cross Validation that scored a 62.17% accuracy.

7. Teamwork Distribution

Member	**Contribution**
Danish Alsayed	Collecting data and constructing datasets, Choosing and Coding the models & feature transformations, time series analysis, Report writing, feature transformations, presentation, overall group management
Ng Pui Yan	Generating stationary dataset, Coding models, collecting results, presentation, video making
Ng Chi Ping	Execute models, collecting results, implement 15day window, CV test, result analysis and visualisation, report writing, part of the model explanation, presentation
Kabir Rajput	Collecting data for technical and fundamental features, fine tuning models to fit technical features, feature importance, report writing, presentation

8. Reference

[1] Additive models & Multiplicative models. (n.d.). http://www-ist.massey.ac.nz/dstirlin/CAST/CAST/Hmultiplicative/multiplicative1.html.

[2] Desarda, A. (2019, January 17). Understanding AdaBoost. Retrieved from https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe.

[3] Forecasting: Principles and Practice. (n.d.). https://otexts.com/fpp2/autocorrelation.html.Fuqua

[4] Random Forest Classifier - Machine Learning. (2019, October 9). Retrieved from https://www.globalsoftwaresupport.com/random-forest-classifier/.

[5] Stationarity and differencing. (n.d.). https://people.duke.edu/~rnau/411diff.htm.

[6] Tom Bylander, Lisa Tate. (2006) Using Validation Sets to Avoid Overfitting in AdaBoost, https://pdfs.semanticscholar.org/bf0f/470fe94c78b8af9c512b47b9cc20d2f4c16c.pdf

[7] Why Log Returns. (2012, November 29). https://quantivity.wordpress.com/2011/02/21/why-log-returns/.

[8] Zhang, Zhongxing & Mayer, Geert & Dauvilliers, Yves & Plazzi, Giuseppe & Pizza, Fabio & Fronczek, Rolf & Santamaria, Joan & Partinen, Markku & Overeem, Sebastiaan & Peraita-Adrados, Maria & Silva, Antonio & Sonka, Karel & Río, Rafael & Heinzer, Raphael & Wierzbicka, Aleksandra & Young, Peter & Högl, Birgit & Bassetti, Claudio & Manconi, Mauro & Khatami, Ramin. (2018). Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning. Scientific Reports.