Forex Price Prediction using AI models

Melroy Elias Pereira
University of Adelaide
melroyelias.pereira@student.adelaide.edu.au

Abstract

The Forex market is very volatile market, and it is one of the most traded market in the world, People have been trading currencies for decades and trying to find various ways to trade in profit by minimizing the risk. In this literature, we are using Artificial Intelligence models to predict the price movements by using classification and regression models, our strategy is defined from scratch and we backtested with EURUSD daily data and it gave a promising result of profit and our models were able to predict the price movement, we achieved lowest RMSE of 0.003 and accuracy of 63% and turning our initial investment of $10000 USD to $12500 USD, these metrics were not possible without the data pre-processing and hyper-parameter tuning, therefore we used several processing techniques to clean the data and select the best features. Furthermore, our models were also able to predict better than the baseline models used in the project.

1. Introduction

In this project, we are predicting the forex signals using Artificial Intelligence (AI) models and we test our performance of our model with our own created backtester. In this project we are not solely dependent on the AI model for trade signals, we also combine human expertise methods to filter out the higher probability trades. We used EURUSD daily data from Dukascopy [1](Dukascopy).

In this project we used data processing techniques for cleaning the data and selecting the important features and we used both regression and classification models for prediction, classification model used for classifying the trade signals into Buy and Sell, and we used regression models for predicting the target values. We used following classification models:

- Support Vector Machine - Classification (SVM)
- Random Forest Classifier
- Logistic Regression
- Multi-layer perceptron (MLP)
- K-Nearest neighbor (KNN)

We also used following regression models for predicting the target levels:

- Support Vector Machine - regression (SVM)
- Random Forest Regression
- Linear Regression
- Polynomial Regression
- K-Nearest neighbor regression (KNN)

In order to learn from the data we used train and test split method and grid search method for hyper-parameter tuning for predicting the data using the best parameter. We later passed our prediction for human expertise to filter out the trades and we give the output of the strategy reports in dashboard, and we compare our prediction model with baseline models to check if our models are better than simple base models, after evaluation we see the results of all the models and strategy to consider if the strategy works or not.

In summary, my contribution to the literature can be listed as follows:

- We create our own backtester file, which only requires the users strategy function and rest of the calculation is automatically shown in the dashboard.
- We compare if the PCA data is worth to use in this type of dataset.
- We use various classification and regression models for predicting the price movements in the future.

- Finally, we show if the strategy is making the profit or loss using AI models.

The organization of the report is as follows: Section 2 is a materials and methods, section 3 is a Result and discussion, Section 4 is a Conclusion and future work, Section 5 is a reference and Section 6 is a Appendix.

2. Materials and Methods

Pre-Processing:

In this project we used many prediction models, however, those prediction models were not able to give better results if we had skipped the data pre-processing step. Data pre-processing step was crucial, and it was challenging to select the best pre-processing steps for our automatic data processing, and also because of time series data we didn't want to lose too much of information(patterns) from the data, however, we managed to select few pre-processing steps. Our pre-processing framework is shown in (see Appendix A) and it includes following steps:

1. cleaning the data from NaN to zero values.
2. we also encode the categorical values into integer number that is 1 for Buy and -1 for Sell.
3. We remove the outliers using Z-score method with boundary values of +2 and -2, we used these values because according to the normal distribution curve it says that within this range 95.4% values are in, so we considered rest of the 5% as outliers.
4. We normalize our data values to [0,1] scale using Min-Max scaler from sklearn library, we consider normalizing is important because if the data values are large then the coefficients to those features will be large and it causes the loss function to increase the loss and it will be difficult to converge the loss near to zero, therefore doing scaling it is faster to train and also faster to converge.

After the pre-processing step we do the feature selection.

Feature Selection:

Feature Selection is important to any Machine learning model because we don't want to include features that are not meaningful for target data because these data will act as noise and we want to filter those noise, therefore we use feature selection. Feature selection framework is shown in (see Appendix B). In the project we do semi-automatic feature selection process, we calculate all the coefficient's and we pass it to user to select the feature according to their requirement. We have following steps for feature selection:

1) We first pass data to correlation where we only select features with 80% above correlation.
2) We later pass it to linear regression selector for getting the coefficients of features, here we use coefficients because if the coefficients are zero we consider it as zero correlation with target data and if negative coefficients we will say negative correlation and if its positive coefficient's we consider it as positive correlation.
3) We again pass the features to random forest regression to get the coefficients and we use the same method above in step 2 for selection.
4) We later pass the selected strategy feature value into PCA for accumulation of values into one feature.
5) After all, above steps we pass one last time to correlation and we select only features with 80% above correlation.

Baseline model:

After all the pre-processing and feature selection step we compute the baseline model from selected features because we want to have some baseline prediction model for comparison of our predicted data thus we divided the results for models into category that is:

- Model without PCA
- Model With PCA

We first calculated the model without PCA, we used linear regression for regression model and random forest classifier for classification model as baseline models, and we obtained RMSE of 0.017 for regression model and accuracy of 46%

for classification model, however our predicted model has highest RMSE of 0.003 and accuracy of 63%, therefore we concluded that our models are predicting better than baseline models.

Later we used the same models with including PCA features, and we found that for baseline model we had RMSE of 0.04 and accuracy of 46%, therefore we can see that even baseline models are not well predicting with PCA for this data, however, we had computed the RMSE of prediction model and we found that polynomial regression had low RMSE of 0.01 but not low as when we didn't had PCA data, and classification accuracy were almost same. Therefore using PCA we got lower performance compared to when we didn't used PCA.

Hyper-Parameter tuning and Train-Test Split:

In this project we used Grid-Search method for hyper parameter tuning, this section comes under the strategy section and grid search method uses the pre-selected parameters for each model for the process with the train data and it returns the best parameter for predicting the test data, here we use sklean libraries train-test split method for splitting the data, in splitting the data we shuffle the 95% of train data for classification model and we do not shuffle rest 5% of test data of classification model because we want to keep the same sequence order therefore we didn't used K-Fold because it shuffles the data, and we also use the same library for splitting the data for regression model with 95% of train data and 5% of test data, here both data are not shuffled, because we don't want to shuffle the continuous value data of time series for mixing up the patterns, after train and test we pass the data for hyper parameter tuning and grid search method predicts the data from best selected parameter. After the prediction we pass that into strategy function.

Strategy:

In order to complete the project we have to do the trading strategy ourself and we need to feed our strategy function into the backtester class where the tester will evaluate the performance of our strategy and returns us with the dashboard containing P&L plots, the strategy framework is shown in (see Appendix C), In this project we used 5 models for classifying the trade signals into buy and sell that is, SVC, random forest, logistic regression, MLP and KNN and we used the signals that are maximum of all the prediction models as our trade signal, and we also used regression model for predicting the target levels and we used 5 models of Linear regression, Polynomial, random forest, KNN and SVR and we mean all the predicted target as our target level, after creating the AI model and its signal from hyper parameter tuning we pass the model signals and target level to the human expertise, where it will filters the trade signal according to the pips analyzed by the user, such that filtered signal is later passed into the backtester.

Back tester and Dashboard:

The Framework of the backtester is shown in (see Appendix D), In the backtester we receive the predicted signals and we compute the profit and loss for each trade at the end of the day close price, and we pass the profit and loss to our strategy metrics where it will evaluate its performance and send back the result to the user in dashboard format, and user can select the type of strategy from dropdown button in dashboard for their own strategy according to the profit and loss that is, "what would be the profit and loss if it was only long only based strategy/short only based strategy or both long and short strategy", the backtester will also return the performance of our prediction models for comparison with the baseline model.

3. Result and Discussion:

Strategy Result:

The core purpose of the strategy is to generate profits and as low as loss, In this project, we were able to generate the highest profit of $12500(USD) from $10000(USD) initial investment, we divided our strategy result into three strategy result that is long only, short only and both long and short. We were able to get $12500 from long only and $12000 from both long and short, however with short only we got $9800 which has a loss of $200 from initial investment. From both long and short, we were able to generate $7013 profit and loss of $-4869 with highest loss of -$224 and largest profit of $218, we had overall 130 profit trades and 100 loss trades. By looking at the profits and loss we know that by using short option for our strategy might have adverse effect on loosing money, so it would be better to not use short strategy for this type of strategy, thus we were able to get $12500 by using only long only strategy instead of $12000 and $9800. (see Appendix E).

Model Result:

In this project we are comparing our prediction model with the baseline models, our regression baseline model had RMSE of 0.017 and R2_score of -0.43 quality of fit and our classification baseline model had the accuracy of 0.46.

[Before I did test with other model I knew that simple regression is able to give better result, so I thought of giving example for model complexity Vs error] Our prediction model used 5 different regression models and out of 5, 2 models were better that is linear regression giving RMSE of 0.003 and R2_score of 0.973 quality of fit and polynomial regression model giving RMSE of 0.011 and R2_score of 0.743 quality of fit, and in our 5 classification models KNN was able to give accuracy of 0.63 and Logistic regression was able to give accuracy of 0.55 and SVM gave 0.55. By looking at the result we can see that our predicted model is performing better than the baseline model. It can be observed that our simple linear regression model is giving low RMSE and highest R2 score that is quality of fit, which we can also conclude that our simple model is also able to predict and we do not need complex model like polynomial because according to the occam's razor the simplest model giving better result is always the best one, and doing this we also overcome the problem of model complexity and it can be evident that as we increased our degree of polynomial we tried to fit more and more so we got lesser quality of fit for polynomial while testing the data, by using the simple model we also train fast and we get more inference.(see Appendix E).

4. Conclusion and Future Work

In this project, we were able to make profit of $2500 and our total capital is turned to aprx. $12500 from $10000 from long-only without using PCA data, however when we used PCA data we were able to get profit of aprx. $12000 from $10000 from long-only, but we had higher loss using PCA data than without PCA, so it might be not right to use PCA for this data, however, by using the above models for prediction we also outperformed the baseline models, therefore, we can say that our model is better than base model.

The downside of my back tester is it shows the output at the end, therefore For the future work, I want to improve my back testing frame work for event driven method than vectorized method, because using event driven we can see what is going on every data and every trade virtually, furthermore, I want to continue my core research on forex quantitative trading for making better strategies with better profits using data science methods, fundamental trading approach etc.

5. References:

[1] Dukascopy Historical data,

https://www.dukascopy.com/swiss/english/marketwatch/historical/

[2] Ernest P. Chan, "Quantitative Trading: How to build your own algorithmic trading business".

[3] Quantconnect Discussion Forum,

< https://www.quantconnect.com/forum/>

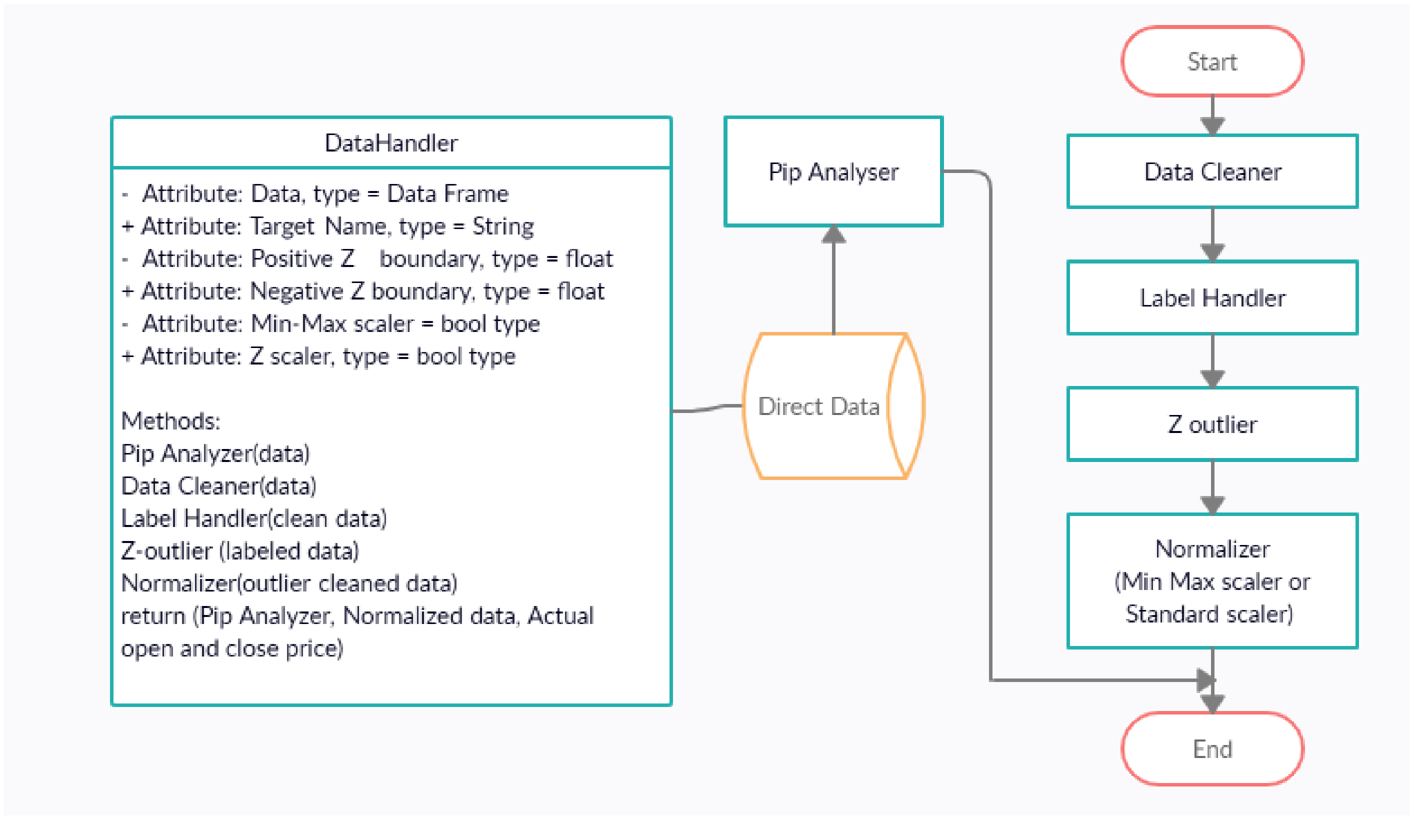
[4] Quantstart, Algorithmic trading articles,

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

Appendixes

Appendix A

In this Appendix we show the framework of pre-processing the dataset, and the framework is shown below:


DataHandler
- Attribute: Data, type = Data Frame
+ Attribute: Target Name, type = String
- Attribute: Positive Z boundary, type = float
+ Attribute: Negative Z boundary, type = float
- Attribute: Min-Max scaler = bool type
+ Attribute: Z scaler, type = bool type
Methods:
Pip Analyzer(data)
Data Cleaner(data)
Label Handler(clean data)
Z-outlier (labeled data)
Normalizer(outlier cleaned data)
return (Pip Analyzer, Normalized data, Actual open and close price)
Pip Analyser
Direct Data
Start
Data Cleaner
Label Handler
Z outlier
Normalizer (Min Max scaler or Standard scaler)
End

DataHandler is the class name for the data processing function

Step 1: Data flow is divided into two parts, first the data flow to the pip analyzer and gives the output as plots.

Step 2: After analyzing the pips from the plots, we pass the data to the clean data, that is if there is any nan values it will turn to zero.(In the framework I forgot to add the extra line for "start" from data, I have only showed one, but it also flows to the start)

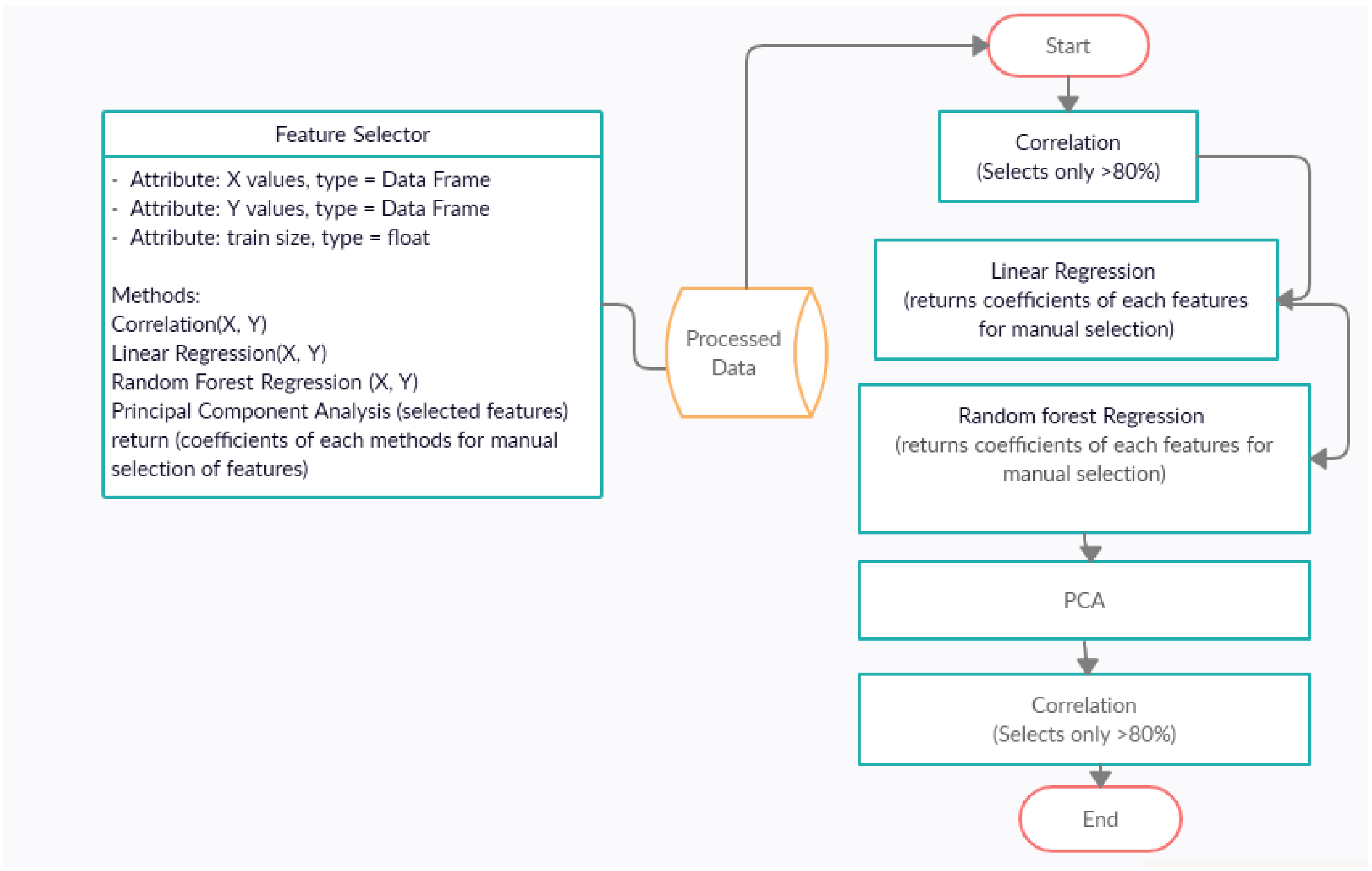
Step 3: Label handling, the signal feature has categorical values that is Buy and Sell, we turn them into 1(Buy) and -1(Sell).

Step 4: we use z score outlier methods that is if the values are more than positive and negative boundary we remove them, here we are using +2 and -2 as boundary values, we know that within these boundaries we have 95% of the values according to the normal distribution curve.

Step 5: we normalize the values using the Zscore method (standard scaler) or MinMax sclaer method.

Appendix B

In this Appendix we show the framework of feature selection of the dataset, and the framework is shown below:


Start
Correlation (Selects only >80%)
Linear Regression (returns coefficients of each features for manual selection)
Random forest Regression (returns coefficients of each features for manual selection)
PCA
Correlation (Selects only >80%)
End
Processed Data
Feature Selector
- Attribute: X values, type = Data Frame
- Attribute: Y values, type = Data Frame
- Attribute: train size, type = float
Methods:
Correlation(X, Y)
Linear Regression(X, Y)
Random Forest Regression (X, Y)
Principal Component Analysis (selected features)
return (coefficients of each methods for manual selection of features)

FeatureSelector is the class for feature selection:

Step 1: The data goes into Correlation function, and we select only the features with above 80% correlation(positive)

Step 2: We use the Linear regression coefficients to select the features, if the features coefficients are zero, then its zero correlated, and if its negative its negatively correlated and if its positive its positively correlated.

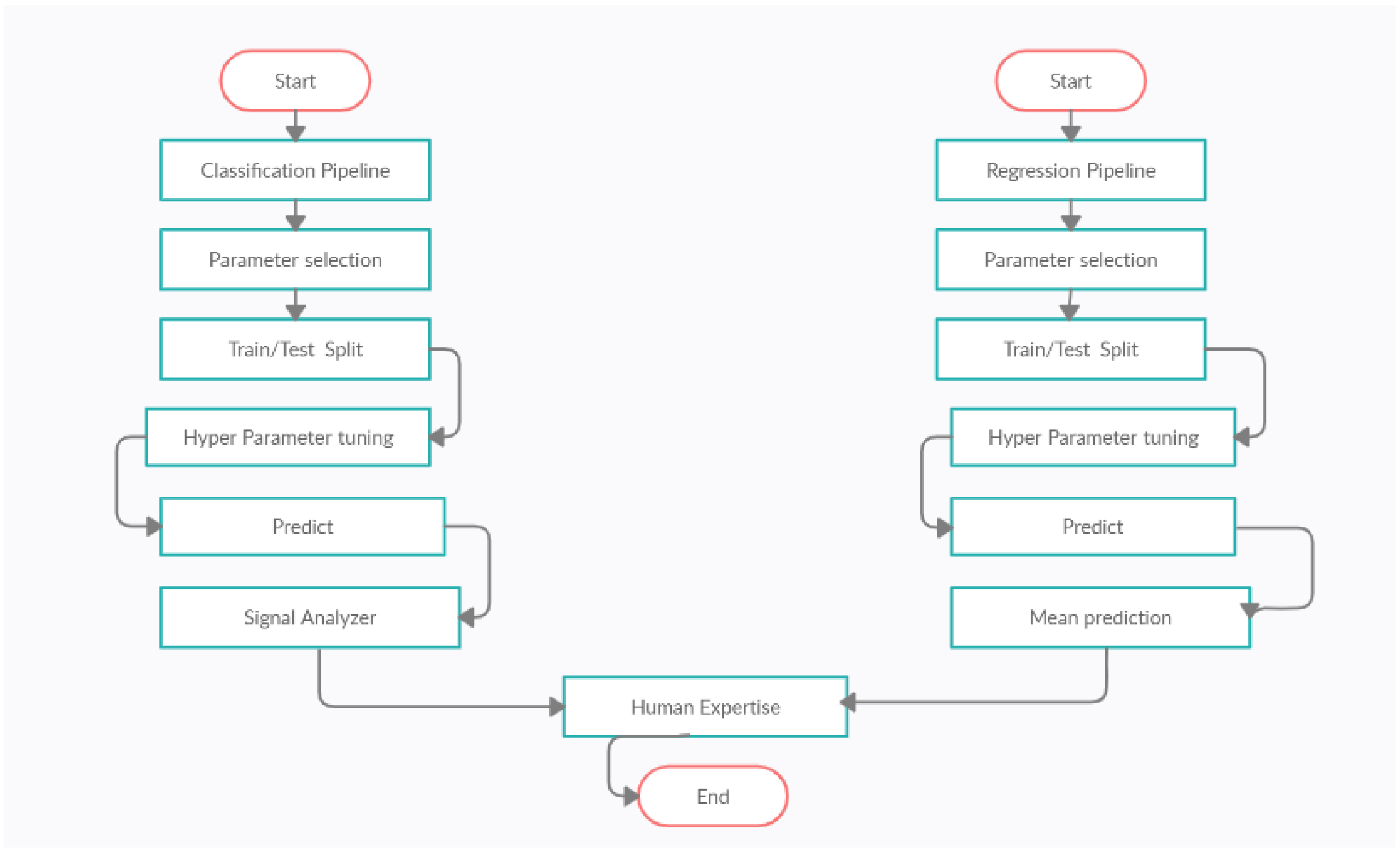
Step 3: We use the Random Forest regression coefficients to select the features, if the features coefficients are zero, then its zero correlated, and if its negative its negatively correlated and if its positive its positively correlated.

Step 4: We pass the selected features to the Principal component analysis (PCA), here we only pass strategies features and we get output as accumulated feature.

Step 5: We pass all the features again for correlation to remove if any zero or negative correlated data

Appendix C

In this Appendix we show the framework of Strategy, and the framework is shown below:



Strategy used in this project is as follows:

<u>The strategy has two parts:</u>

- Classifying the trade signal into Buy and Sell
- Predicting the level that the price can go using the regression models.

1) Classifying the trade signal into Buy and Sell

Step 1: First we create a pipeline where all the classifier models will be put in.

Step 2: We create parameters for hyper tuning the models

Step 3: We split the data into train and test with 95% train size and we pass the shuffle = True, since the data are now shuffled we can use the shuffled data for training, however, we want the ordered sequence data for testing because we are not using the shuffle in regression thus the predicted values for features data need to be same in classification and regression, thus we use dummXtestC and dummytestC which are shuffle and will not be used, we train 95% data and we pass the last 5% data for testing that is XtestC, ytestC.

Step 4: We pass the trained data into gridsearch method for hyper parameter tuning, which will select the best parameter of all the given parameters.

Step 5: We now predict the data using the parameter selected from grid search method, and we put all our prediction into dataframe with model name. total predicted data are now 232.

Step 6: We now pass the precited dataframe for signal analyser, where it selects the maximum predicted signal into the precited signal columns.

for example: If, SVC = -1

Random forest = -1

MLP = 1

KNN = 1

Logistic regression = -1

Now we know that three models predicted -1 and two models predicted 1 , according to the rules I set, I choose maximum predicted signal that is -1, because now i have higher probability of getting signal -1 than 1.

Step 7: Now we have the predicted signals.

2) Predicting the level that the price can go using the regression models

Step 8: Now we use the regression model for predicting the target levels for those predicted signals, we pass all regression models into the pipeline.

Step 9: We create parameters for hyper parameter tuning.

Step 10: we split the data into train and test, now we do not shuffle the data because we need the sequence data for predicting the continuous output, using shuffle it will shuffle the patterns for my strategy hence we do not shuffle, and we use 95% training data.

Step 11: We pass the train data into the grid search method for hyper parameter tuning for selecting the best parameter for testing.

Step 12: We predict using the parameter selected from grid search method

Step 13: Since now we have predicted values we will pass the predicted valued into dataframe and we create new columns where it predict the mean value of each model as its target value.

for example:

If linear regression = 1.1145

Polynomial regression = 1.1148

Random forest regression = 1.1143

Knn = 1.1146

We mean the values in row wise Mean predicted = (1.1145+1.1148+1.1143+1.1146)/(4)

The mean predicted is 1.11455, so we use this as our predicted target.

Step 14: Now before we send our prediction into backtester we pass the strategy for human expertise, where it again selects the signals according to the following rules:

If signal is 1 and if the (open price+ 0.0010) is less than the predicted mean target then we consider it as a buy signal.

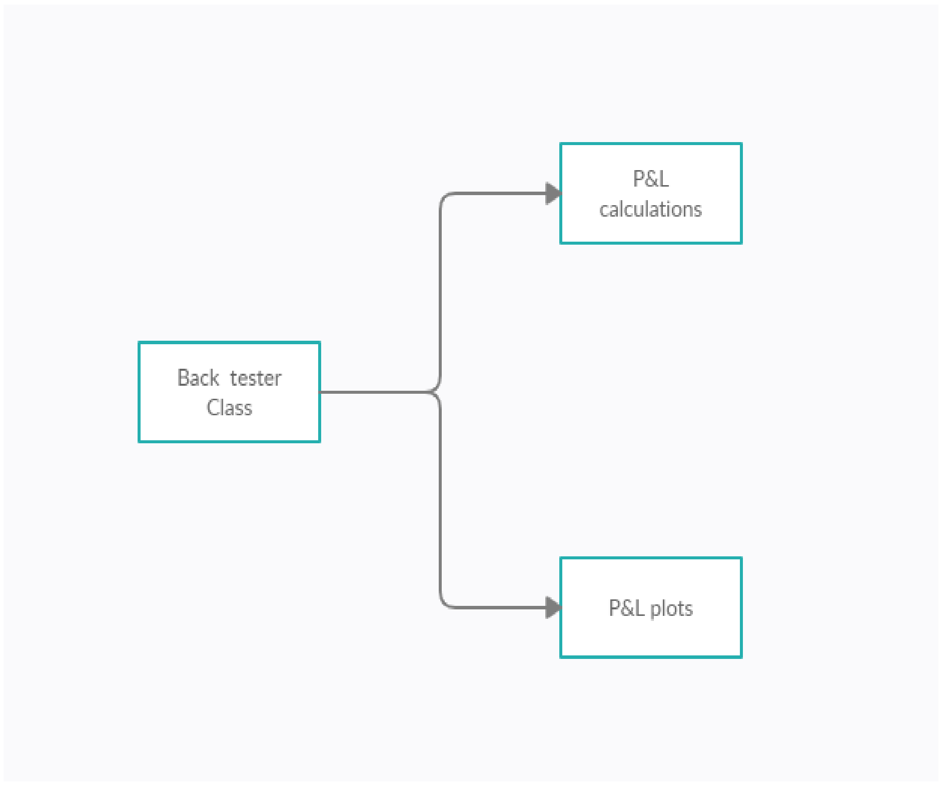
If signal is -1 and if the (Open price - 0.0010) is greater than the predicted target then we consider it as a sell signal.

example: Consider signal is 1 from past data and our open price is 1.1145 and predicted is 1.1160 and if (1.1145+0.0010)is less than 1.1160, which implies that we have atleast space to cover for 10 pips (0.0110) because predicted is 1.1160, which has 15 pip(0.0015) difference from open price, which will act as capacity, higher the capacity is we have higher probability to capture small pips.

Step 15: We pass the predicted signal that is 1 and -1 for backtesting, where backtester assumes all the positions are closed at the end of the day.

Appendix D

In this appendix we show the framework for the back tester,



The back tester has two components:

- P&L calculations
- P&L plots

P&L calculations:

In P&L calculation we pass the strategy signals as a data frame and we calculate the profit and loss at the end of the day close price.

For example: If our signal is 1 that is Buy, and Open price is 1.1145 and today end of the day close price is 1.1155, then we make 10 pips(0.0010) as profit, (Close price - Open price), however, here we are using 1 lots therefore we make profit of $10 USD (Initially our investment was in USD), so if our initial deposit is $10000, then we make $10010 that is $10 gain and we do the same for all the signals based on the type of strategy.

The code used for this is given here:

```
elif (self.strategyType == 'Long-Short'):
  AI_pnl = []
  for price, open, signal in zip(self.data['Close'], self.data['Open'], self.data['AI_signal']):
    if (signal == 1):
      AI_pnl.append((price-open)*10000)
    elif (signal == -1):
      AI_pnl.append((open-price)*10000)
    else:
        AI_pnl.append(0)
  self.data['AI_P&L'] = AI_pnl
```

Here we consider our strategy as long-short, we select the signal data that is passed from the strategy which has Close, Open and AI Signal, therefore we use for loop to check if the signal is 1 we take the difference of price-open to get the pips and we multiply with lots that is 1 lots = 10000 to the difference, and we later append the value in pnl list, if the signal is Not 1 and -1 we append it as 0, after all calculation we put the list value into the data frame to get that as output.

<u>P&L plots:</u>

In the previous explanation I told that it will use the initial money for recurrent calculation, but actually it uses that step in this function and we do that for each type of strategy and we pass the data as output for dashboard to plot the P&L plots.

Code used for recurrent calculation is shown below.

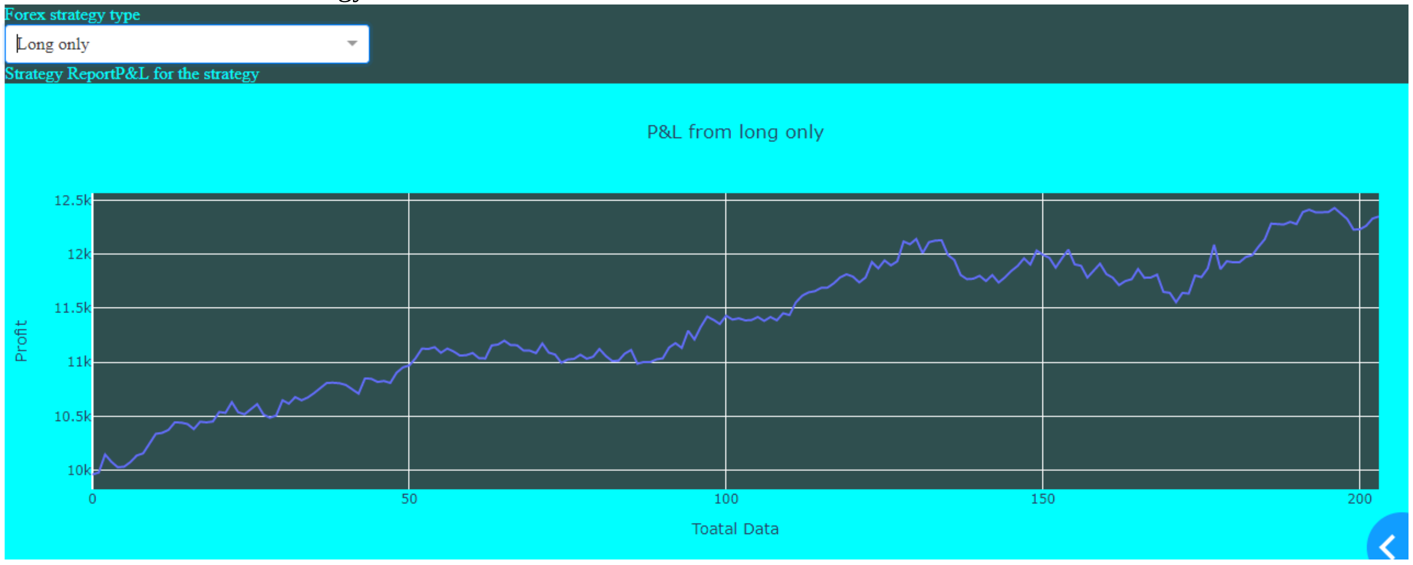
```
PL = self.PNL_Calculation()
#long-short
AI_pnl = []
initial_deposit = self.initial_Amount
for profit in PL['AI_P&L']:
  initial_deposit = initial_deposit + profit*self.lot
  AI_pnl.append(initial_deposit)
PnL_Both = pd.DataFrame(AI_pnl, columns = ["P&L"])
```

In this code we get output of P&L calculation and we use those pips for calculating the recurrent profit and loss, and later we pass the data for output.

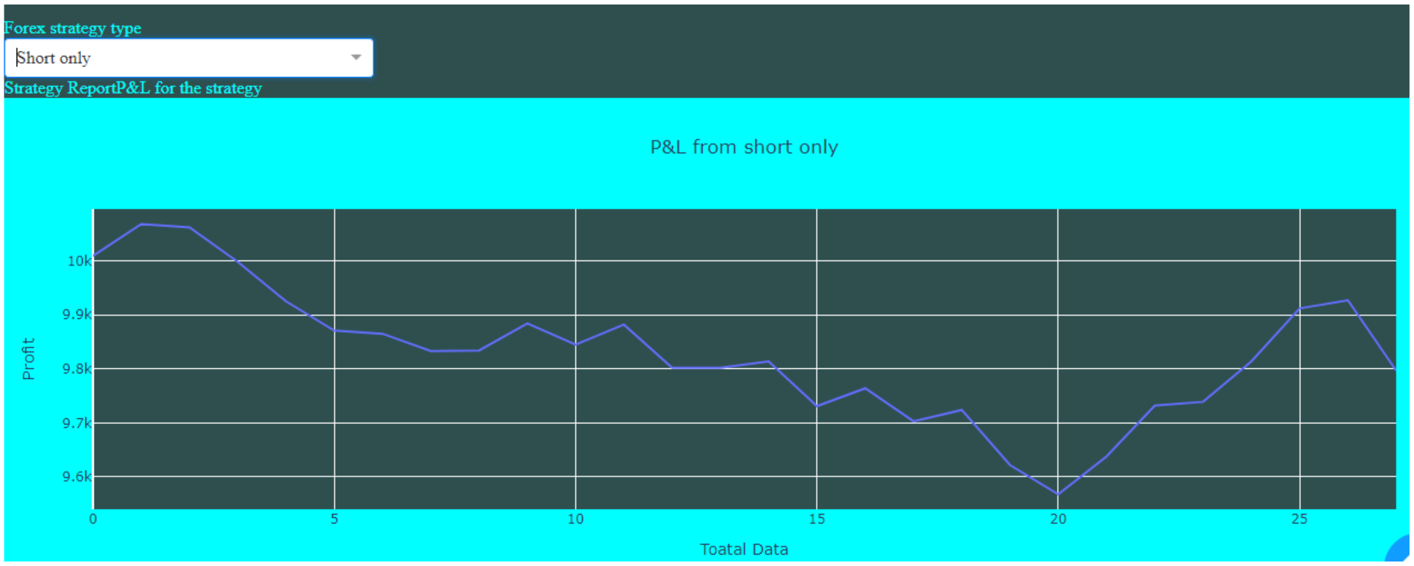
Appendix E

Here we will show the strategy and model performance for model without PCA.

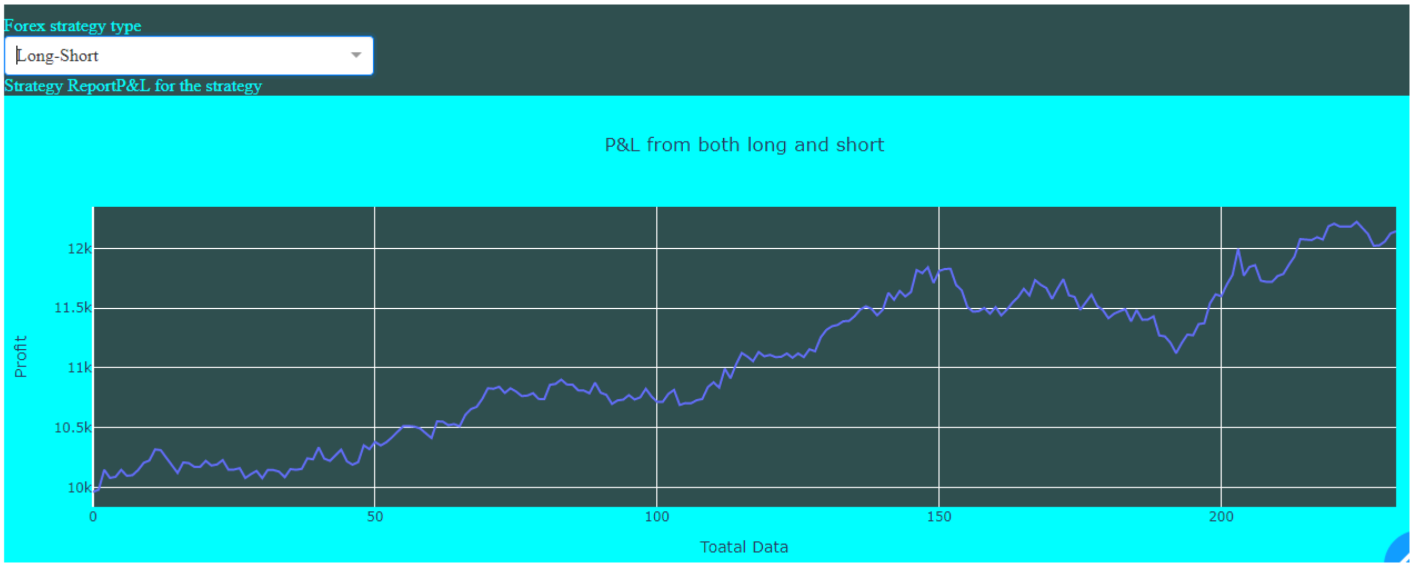
Profit and Loss for the strategy is:



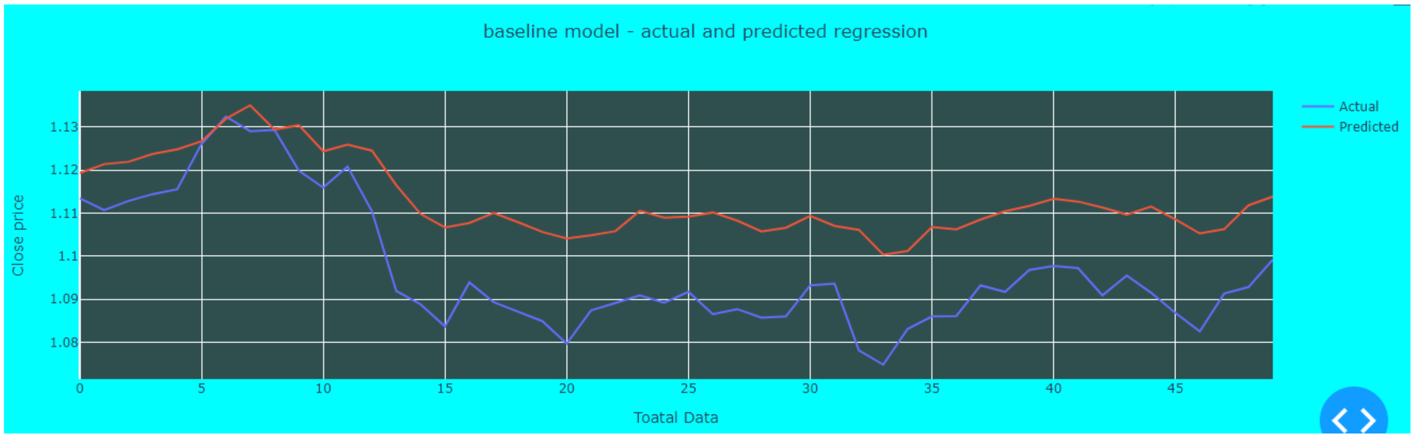
Profit and loss for Short only strategy is:



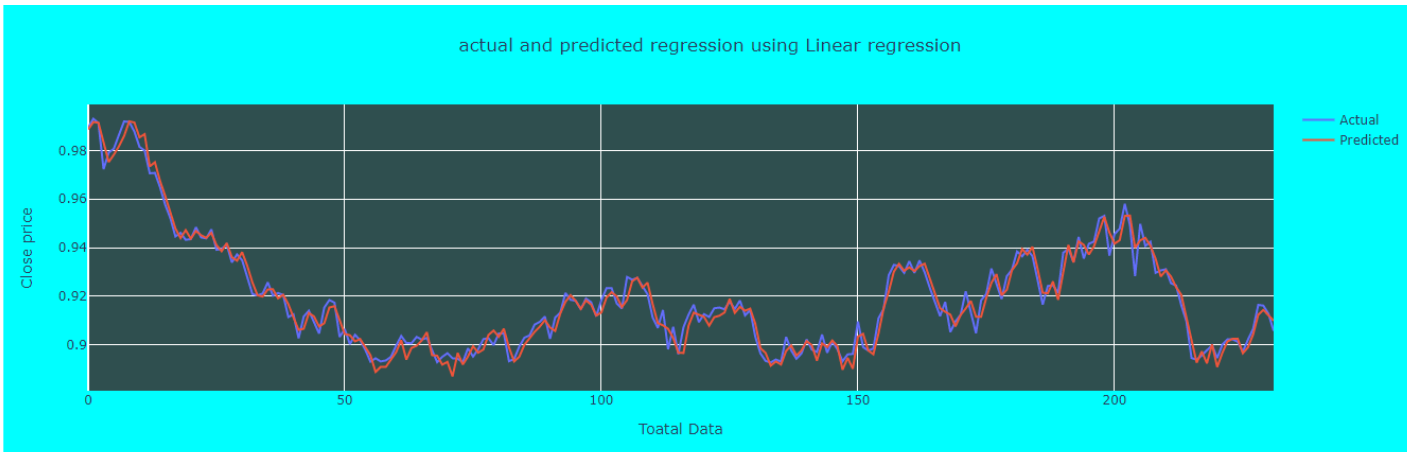
Profit and loss for long-Short strategy is:



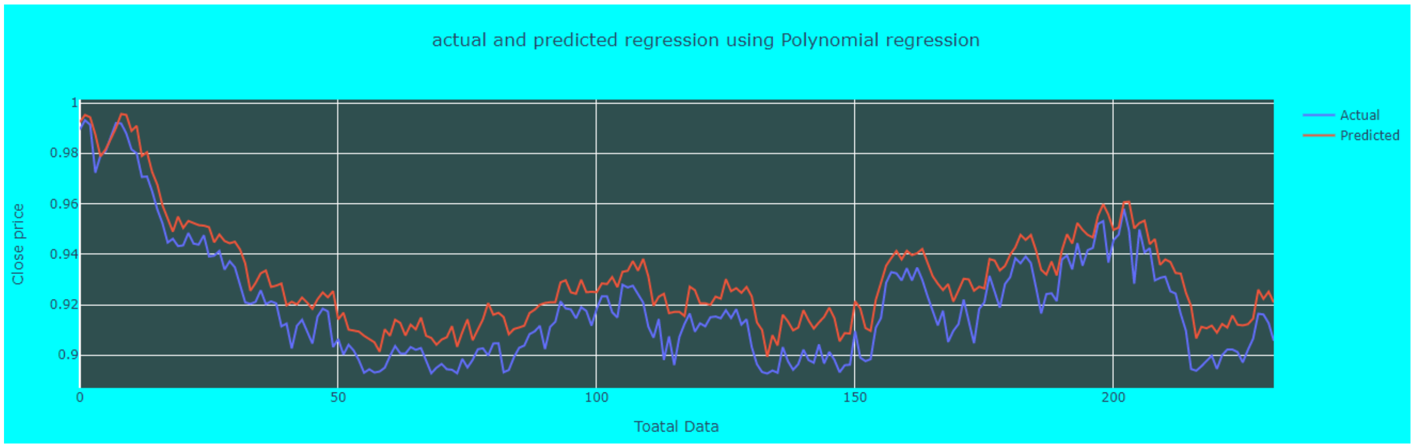
Baseline model prediction is shown below:



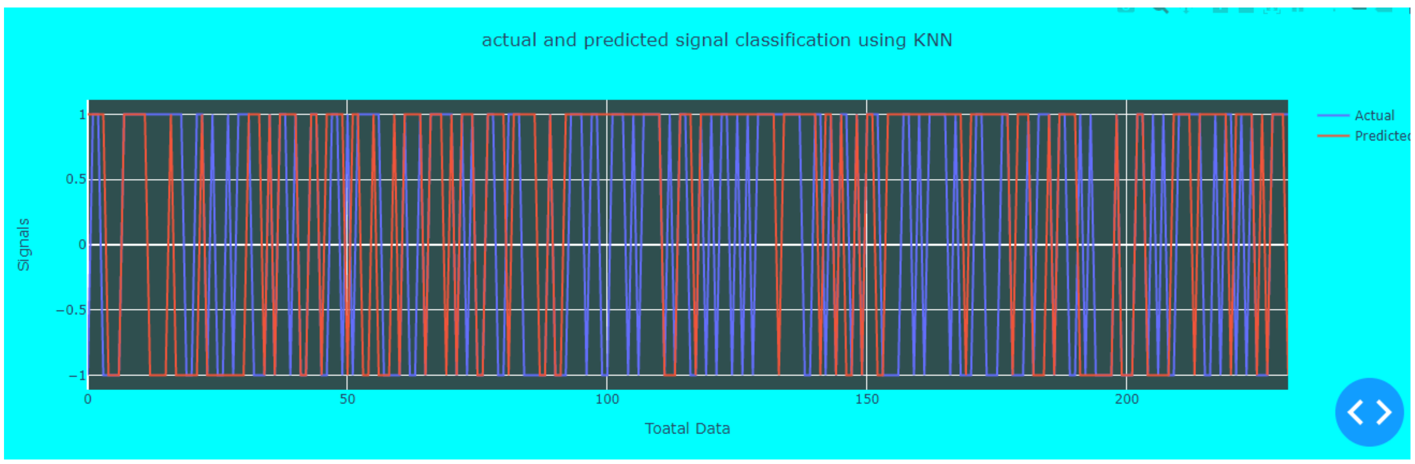
Prediction model – Linear regression



Prediction model – Polynomial Regression



Signal prediction from prediction model:



Results of the prediction model I shown below:

<u>Regression model:</u>

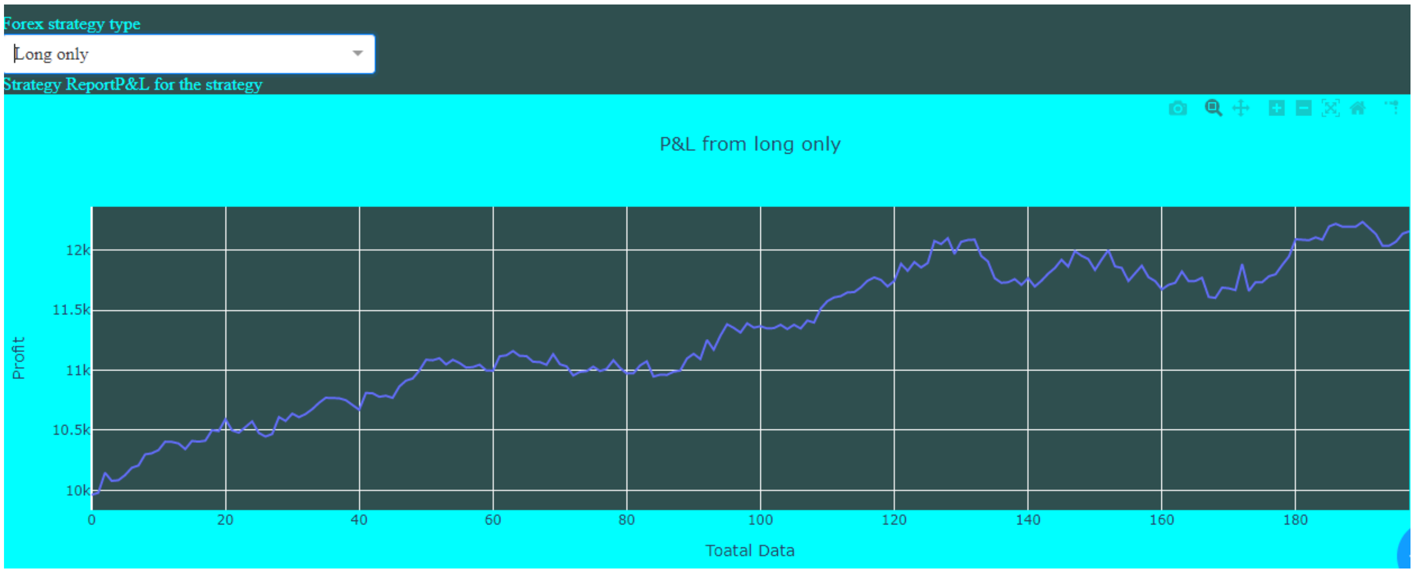
	MSE	RMSE	MAE	R2_score
Linear	0.000014	0.003775	0.003036	0.973238
Polynomial	0.000135	0.011620	0.010591	0.746392
SVR	0.046428	0.215472	0.214313	-86.209658
Random Forest	0.024909	0.157827	0.156131	-45.789243
Knn	0.015362	0.123944	0.121776	-27.855632

<u>Classification Model:</u>

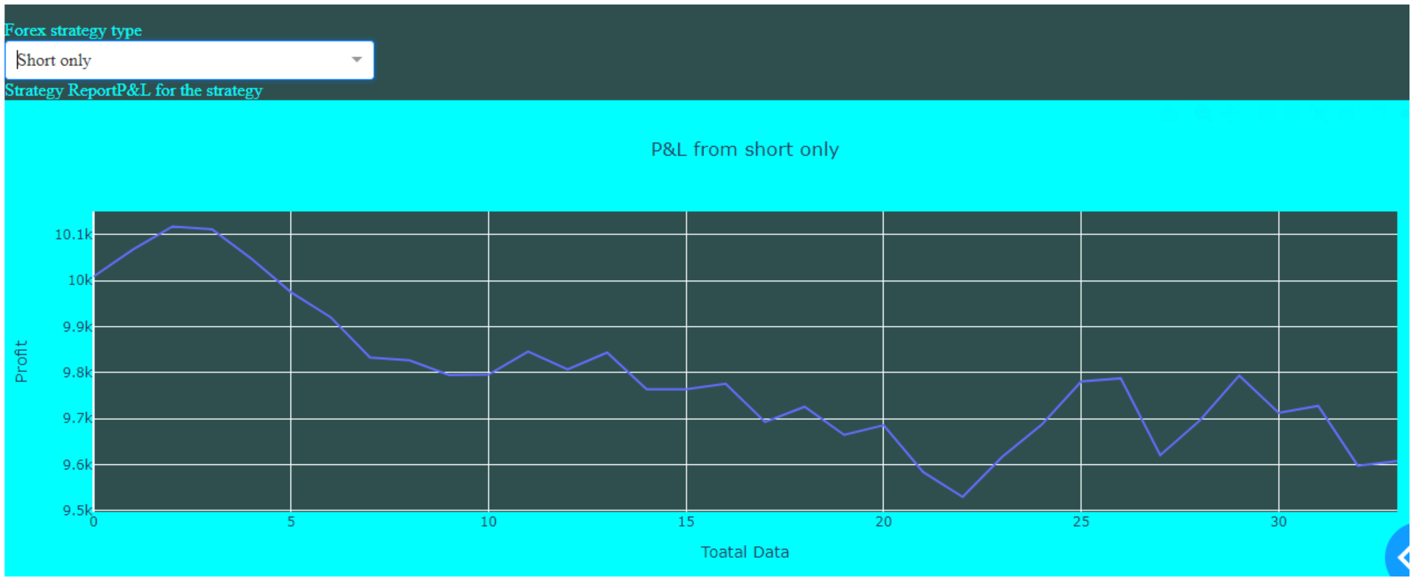
	Accuarcy
SVM	0.556034
Random Forest	0.538793
Logistic Regression	0.556034
MLP	0.443966
KNN	0.633621

Here, we will show the strategy and model performance for model with PCA.

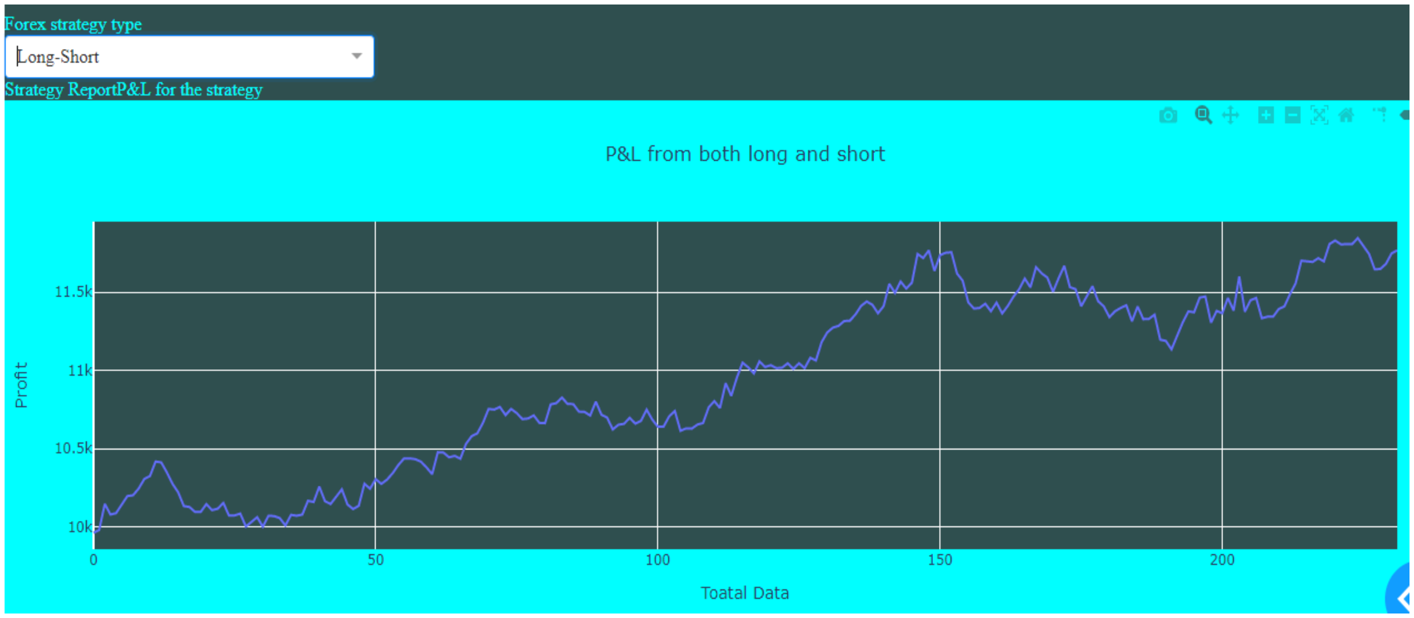
Profit and loss for long only strategy is:



Profit and loss for Short only strategy is:



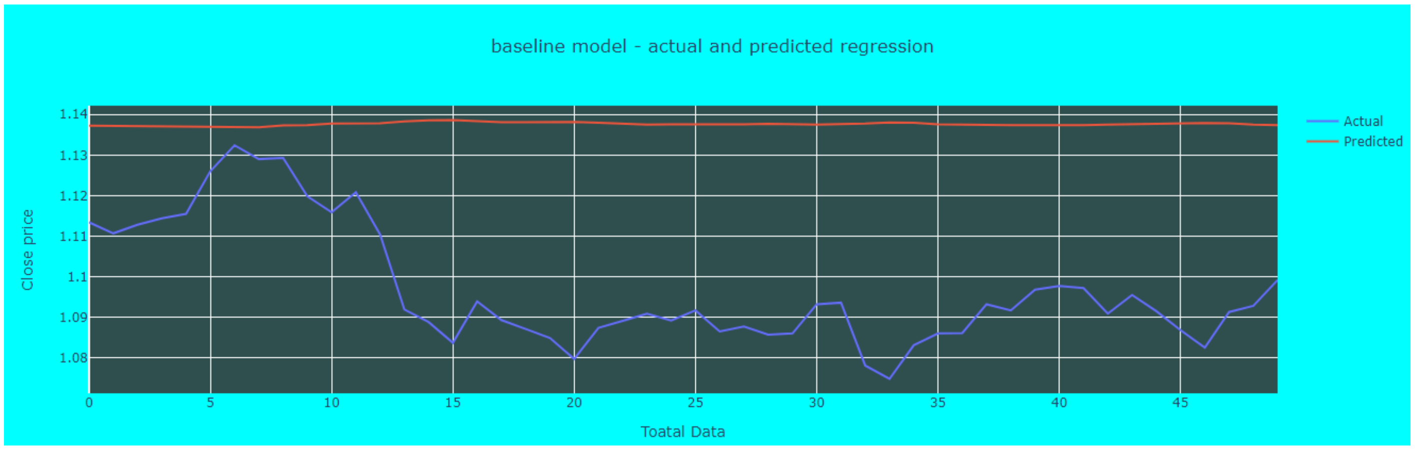
Profit and loss for Long- Short strategy is:


Forex strategy type

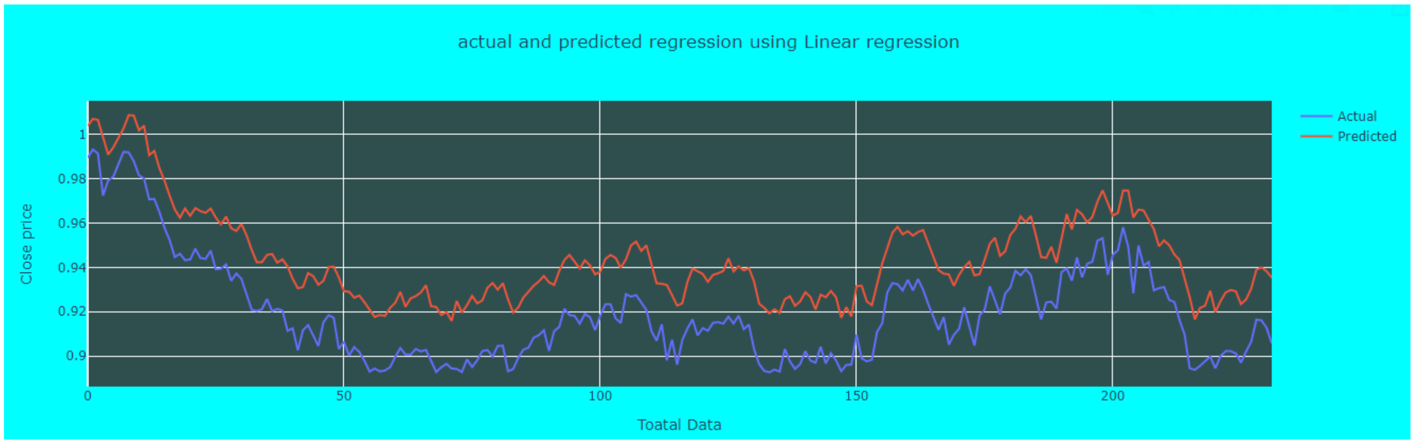
Long-Short

Strategy ReportP&L for the strategy

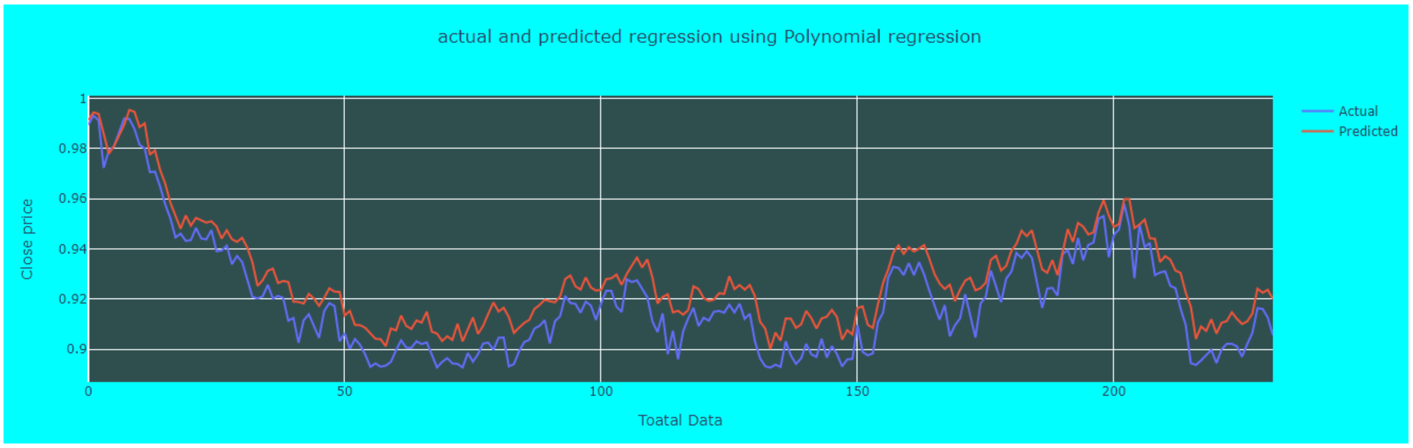
Baseline model prediction is shown below:



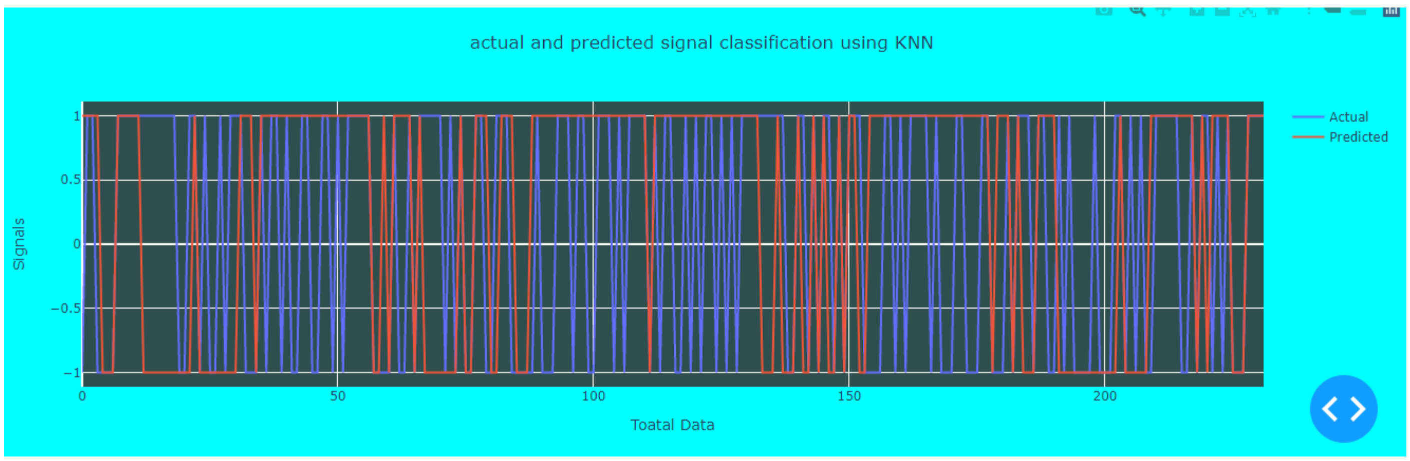
Prediction model – Linear regression



Prediction model – Polynomial Regression

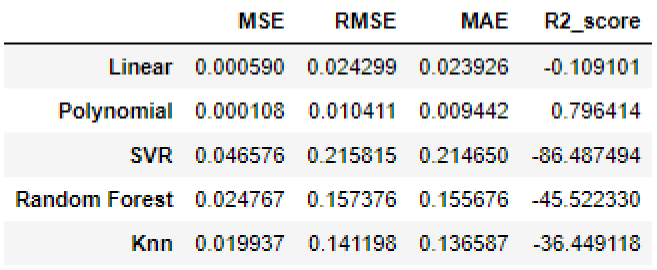


Signal prediction from prediction model:



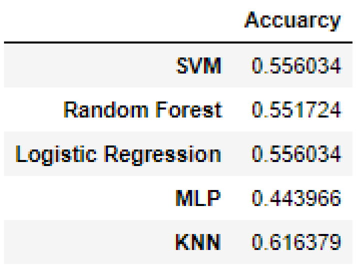
Results of the prediction model I shown below:

Regression model:



	MSE	RMSE	MAE	R2_score
Linear	0.000590	0.024299	0.023926	-0.109101
Polynomial	0.000108	0.010411	0.009442	0.796414
SVR	0.046576	0.215815	0.214650	-86.487494
Random Forest	0.024767	0.157376	0.155676	-45.522330
Knn	0.019937	0.141198	0.136587	-36.449118

Classification model:



	Accuarcy
SVM	0.556034
Random Forest	0.551724
Logistic Regression	0.556034
MLP	0.443966
KNN	0.616379