Long Short Portfolio Using Machine Learning

G. V. Divakar

February 27th, 2018

MLND Capstone Project Report

Definition

Project Overview

Trading is one of the most challenging professions, and it is a place where even a small human error can cost millions. This is also a field that attracts huge investments from leading banks across the world. In this project, I have attempted to implement a new version of one of the most extensively used strategies in the finance world - A Market Neutral Long-Short Portfolio Strategy.

This particular strategy is very hard to optimize, using a traditional quantitative approach. As the information factored in the market is hard to extract, and is very difficult to predict how the traders in general would react.

In this project, I have attempted a different approach to create the Long-Short portfolio. Here I have tried to predict if the market would beat the day before's Highs and Lows. I will use the machine learning classifier algorithms to accomplish this.

The motivation for picking up this particular challenge is that as a trader, with 10 years of successful experience, I have developed many algorithmic trading strategies, both for the firm and myself, and in due course of the time I have trained many traders. It has been my observation that the traders find it hard to trade the trend ignoring the noise. I believe using the same inputs as a human trader, a machine learning algorithm will successfully filter out the noise and classify the trend (Up/Down) correctly.

Literature Review:

This is a recent research paper that I have found interesting and coinciding with my field of interest. The accuracy of the model in this research paper is really good. Although this research paper does not include the application or profitability metrics, I have included a thorough backtesting model that can help anyone with a basic understanding of trading appreciate the results much better. In the research paper stated above, the algorithm has machine learning models based on ANN and SVM with different kernels to compare the results. In my project, I have conducted a preliminary selection of algorithm from a set of 6 different supervised machine learning classifier models.

In the first part of my project I imported the necessary libraries and the required data.
I imported some basic libraries used in data visualization and data preprocessing.
For this I imported pandas, numpy, datetime, and matplotlib to work with the data and to visualize it.

I also imported some trading specific libraries such as talib and nsepy.
Talib is a technical analysis library that I used to create a technical indicator.

To install talib,
please visit the following link and download the appropriate ".whl" file suitable to your computer

https://www.lfd.uci.edu/~gohlke/pythonlibs/

At the above specified location, you can find the suitable file under the name TA-Lib.
Once you have downloaded the correct file. Please copy it in your C drive or the root folder.
After you have completed this, you need to run the following command to install talib.

!pip install TA-lib

I have also used sklearn to import various Classifier algorithms and scorer to measure the performance of the classifiers. After installing TA-lib, install the other library: nsepy using the following command

!pip install nsepy

Problem Statement

In the Long Short strategy, the portfolio will consist of stocks in two categories:

1. Long - Those which I buy
2. Short - Those which I sell

I will allocate equal money to both the categories and then rebalance the portfolio at the end of each day. By having equal money to short and to long, the portfolio will be delta neutral, in other words, our portfolio will not be affected by any unforeseen (six sigma) events in the market. I will create my own indicators that represent how traders see the price action and how they consider price rejections to build their expectations for the following day. These indicators are fundamentally based on Price Auction Theory. The classifier that I will build will use these features to train on the past data to predict the latest trend (next day's market direction) in the market. In the end, I will use these predictions to complete the strategy and backtest it's performance. I intend to use the final returns of the strategy, after deducting the commissions,tax, and transactions costs, as the basis for measuring the algorithms performance. The results can be verified and authenticated as the data that will be used will be from public domain and fetched directly from the NSE (National Stock Exchange of India).

Given the simple daily data of any stock and its corresponding set of features, the classifier will be able to learn on the train data and predict the trend on the test dataset. The features created will be mathematical in nature and devoid of any forward looking bias. The input data will be universal in nature, meaning it should be able to accept the OHLCV data of any stock in the world and perform the predictions. Although the accuracy will vary depending on the

quality of the data provided (without null or random values) and the continuous nature (should contain data for all trading days without any gaps) of the data being provided.

Metrics

The performance of the classifier will be measured in terms of its accuracy on test data, the percentage profitability of the strategy, and the Sharpe ratio of the performance. The Sharpe ratio will give much better indication of the volatility in the predictions. The interest rate required for calculating the Sharpe ratio for the test period is taken as 6%. Please check the link to see the interest rates of RBI (this is not the interest rates for banks). As an individual consumer, the interest rate offered by banks would be typically one percent less than this RBI rate.

The sharpe ratio is defined as follows:

The Sharpe ratio is the average return earned in *excess* of the risk-free rate per unit of volatility or total risk.

Sharpe ratio = (Mean portfolio return − Risk-free rate)/Standard deviation of portfolio return

The standard deviation parameter in the denominator penalizes the returns for higher risk. Hence, Shape ratio gives a much better indication of the performance.

Analysis

Data Exploration

I have used two stocks to construct a sample Long Short portfolio. I have used the stock data of Punjab National Bank (PNB) and State Bank of India (SBI).

I created variables to specify the first company's name, the start date and the end date between which I will fetch the data, and then store this data in a dataframe with the name PNB.

Data before 2006 is not available in OHLCV format, you can get only Close prices, which will not be of much use for this project.

Later I printed the data to check for null values and the data format.

```
In [3]: PNB.tail()
```

Out[3]:

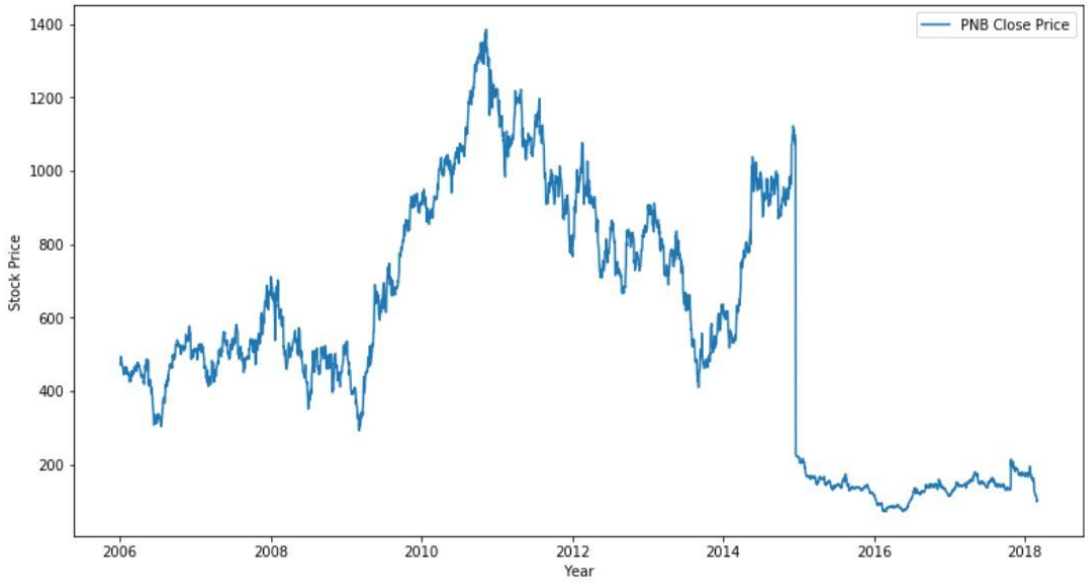
	Open	High	Low	Close	Volume
Date					
2018-02-20	320.50	324.20	316.30	317.25	16030558
2018-02-21	319.00	320.70	315.70	319.20	9974873
2018-02-22	317.00	319.30	313.95	318.35	25530698
2018-02-23	319.05	324.05	318.35	322.75	7949765
2018-02-26	325.00	327.80	323.15	327.05	7133735

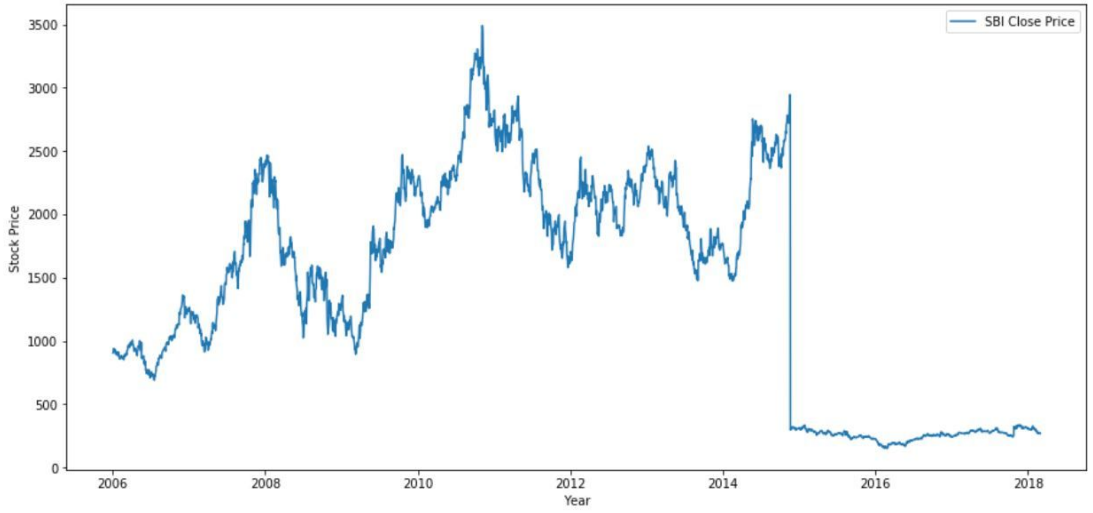
I have checked the data for null values, and even though they were not present in the data, I wrote code to handle such a scenario. I have also checked for any infinity values in the data.

The procedure I adopted to handle the hypothetical NaNs is to replace null values with the values immediately preceding it.

Then I printed the mean and median values to show that replacing NaNs with the past value was the best possible option.

Exploratory Visualization





Later, I plotted the close prices to get a much better understanding of the price movement. I also noted the reason for a big drop in price is that there was a stock split. Since the indicators that I will be creating will be on a day basis and scaled to the range between the Open and Close prices, this sudden change in the price will have no effect in the learning of the algorithm.

```
In [13]: PNB.describe()
```

Out[13]:

	Open	High	Low	Close	Volume
count	3018.000000	3018.000000	3018.000000	3018.000000	3.018000e+03
mean	752.225994	764.350944	738.686034	751.432190	7.345766e+06
std	361.973051	366.812500	357.089907	361.942403	8.076670e+06
min	184.300000	185.950000	180.750000	183.000000	4.348700e+04
25%	335.350000	342.125000	327.287500	334.662500	2.569472e+06
50%	838.750000	852.425000	821.075000	835.650000	4.518010e+06
75%	1020.000000	1037.162500	1004.275000	1020.825000	9.770782e+06
max	1767.050000	1798.150000	1760.150000	1794.100000	1.236154e+08

Next, I converted all raw data to numeric form, just to avoid passing any strings if present in the data. If there were any strings then I would caught them now.

Then I modified the data to handle the market specification for no volume and no trades.

Then I printed the data to check that the input values are in a wide range, so I converted them to logarithmic values to get them into a similar range. This will help when trading stocks with large variation in their price values prices.

Algorithms and Techniques

I have created the target variables for predicting if the High and Low values will be breached the next day. I have not used a single target vector to refer the High with 1, Low with -1 and rest other cases with 0, as this results in loss of some information.

Let us discuss the four possible scenarios with the two Up and Down signal columns.

CASE 1: Up--1, Down--0 : Bullish (Sentiment is to the upside) <br\>
CASE 2: Up--1, Down--1 : Highly Volatility (Market can go up and down.It can break the previous days range)
CASE 3: Up--0, Down--0 : Low Volatility (Market is not expected to break the previous day's range, a day to avoid trading)
CASE 4: Up--0, Down--1 : Bearish (Sentiment is to the downside)<br\>

Had we used the 3 signals case, combining all the four into just there outcomes, then we would have ended up combining CASE 2 and CASE 3, both of which require a different strategy to trade. When we create a portfolio with a big size, it is better to predict the High Volatile days to adjust your risk accordingly.

Hedging for the short stock positions in the Long-Short portfolio is one of the biggest risks of this strategy, by using case 3 condition we can buy the options on such days and protect the capital erosion.

Next, I fixed the amount of past data that needs to be used for generating the features.Then I created more past input data from the initial raw data. I did this by shifting the HIgh, Low, Open and Volume columns by the past time variable that I had decided.

Next, I saved the column names of the data frame so that I can drop them later while creating the feature dataset. Then I created various day trading features and their first and second order differentials. I have explained each of the variable and their purpose.

I also created many mean reverting and trending features.

I wrote a function that takes the input data, along with the target column, the past time period for which the data was generated (t), the variance threshold for selecting the number of decomposed columns after applying PCA to the raw input features, and finally, the amount of data that we will be using as the test data.

I set the default variance threshold to 99% and the default test data size to be 200 rows. These values can be changed while computing, and I took 25% of the input size to be the test size.

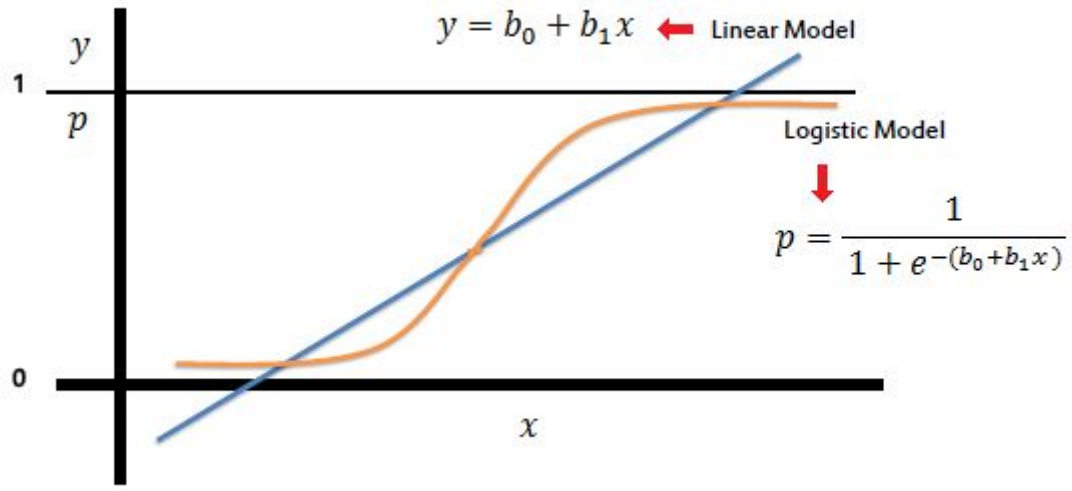
Logistic Regression

In this section I will briefly discuss the logistic regression. In the project, while developing intuition about the best machine learning model it was observed that the logistic regression outperformed the others. I have given a brief understanding of the same here below. (Source:http://www.saedsayad.com/logistic_regression.htm)

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

A linear regression will predict values outside the acceptable range (e.g. predicting probabilities
outside the range 0 to 1)
Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp(b_0 + b_1 x)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b1) is the amount the logit (log-odds) changes with a one unit change in x.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

There are several analogies between linear regression and logistic regression. Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta^1 = \beta^0 + [X^T W X]^{-1} . X^T (y - \mu)}$$

$\boldsymbol{\beta}$ *is a vector of the logistic regression coefficients.*

$\boldsymbol{W}$ *is a square matrix of order N with elements* $n_i \pi_i (1 - \pi_i)$ *on the diagonal and zeros everywhere else.*

$\boldsymbol{\mu}$ *is a vector of length N with elements* $\mu_i = n_i \pi_i$.

A pseudo R2 value is also available to indicate the adequacy of the regression model. Likelihood ratio test is a test of the significance of the difference between the likelihood ratio for the baseline model minus the likelihood ratio for a reduced model. This difference is called "model chi-square“. Wald test is used to test the statistical significance of each coefficient (b) in the model (i.e., predictors contribution).

Gradient Boosting

Source:https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function used depends on the type of problem being solved.
It must be differentiable, but many standard loss functions are available such as ‘deviance’, ‘exponential’. Also, for example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner
Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Larger trees can be used generally with 4-to-8 levels. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3. Additive Model
Trees are added one at a time, and existing trees in the model are not changed.
A gradient descent procedure is used to minimize the loss when adding trees.
Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.
Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss.
Generally this approach is called functional gradient descent or gradient descent with functions.
The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.
A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Benchmark

I will compare the performance of the strategy with a simple buy and hold strategy. The buy and hold strategy will buy one and sell the other stock, and it will hold these positions for the entire period of testing time without daily rebalancing. It is similar to the base case classifier, used in classification project, where I assumed that all the predictions on the test data are the same. In this case, I assume that the prediction is a ‘Buy’ for one stock for the entire length of the testing time and ‘Sell’ for the other stock for the entire length of the testing time. Once I make a comparison in terms of a base case classifier, I will compare the strategy with other leading industry hedge funds’ performance to get a better understanding of the trend in the markets.

Methodology

Data Preprocessing

I adopted the expanding min max scaling technique for the dataset, this will help in avoiding any forward looking bias, as only the past data will be used to scale the input values on any given day. Apart from this, I have also commented out the expanding Z-scores for scaling the data.

After scaling, I omitted the first row of the data as it will contain NaN due to the expanding standard deviation for the first value being undefined. After scaling the entire data, I split the data into train and test data based on the split criterion.

Since the number of features is very high compared to the train data, I used the Principal Component Analysis (PCA) to reduce the dimensions. I fitted the train data to a PCA instantiation and trained it. Then I transformed the entire data using this PCA object. To find the number of features matching the variance threshold, I have written a small for-loop that would keep adding the explained variance component of the columns.

Finally, I rescaled the data using the expanding min max scaler, and returned the complete feature set and the train and test sets. The initial target variable was the Down column values, which I have shifted by -1 to predict the future values.

I rechecked the data for null values and the shapes of the train and test data and took the corresponding values in X (feature) and y(target) data sets.

Implementation

I have used a set of 6 different classifiers to get an intuition about their performance on the train data.

I used a function to fit the train data with different sizes, to the classifiers to get an intuition on how the size of the data is affecting the prediction accuracy. I split the train data into two different parts with the latest 200 data points reflecting the validation data and those older in the train data.

Next, I selected the classifier (in this case it was the Logistic Regression function) that gives the best accuracy and F-score on the validation set to perform a Randomized search, which is followed by a thorough grid search in the vicinity of the selected random parameters of the classifier.

To give a better intuition of this process, you can think of an example where a person from outer space is trying to land on the highest building of the most populated city in U.S.A. To do this, he has to first find the most populated city, or in this case the best performing

classifier, on the validation set. Then he will find the area that is likely to have tallest buildings, say Manhattan. In our project, this would be equivalent to performing a randomised search which is closer to the actual location. Then in Manhattan, he would perform a thorough search of the heights of every building and determine the highest one and then land on it. This would be the GridSearch algorithm that looks in the vicinity of the selected random search parameters.

The initial train and validation accuracy values for the best classifiers on PNB

```
'LogisticRegression': {0: {'acc_train': 0.56999999999999995,
  'acc_val': 0.54000000000000004,
  'f_train': 0.56000000000000005,
  'f_val': 0.45180722891566266},
 1: {'acc_train': 0.76333333333333331,
  'acc_val': 0.61499999999999999,
  'f_train': 0.7621082621082621,
  'f_val': 0.56818181818181812},
 2: {'acc_train': 0.72999999999999998,
  'acc_val': 0.70499999999999996,
  'f_train': 0.71705426356589141,
  'f_val': 0.67164179104477617}},
```

And SBI

```
'GradientBoostingClassifier': {0: {'acc_train': 0.56999999999999995,
  'acc_val': 0.52500000000000002,
  'f_train': 0.41071428571428564,
  'f_val': 0.12931034482758622},
 1: {'acc_train': 0.84333333333333338,
  'acc_val': 0.53000000000000003,
  'f_train': 0.86993243243243246,
  'f_val': 0.39583333333333331},
 2: {'acc_train': 0.93000000000000005,
  'acc_val': 0.70499999999999996,
  'f_train': 0.92672413793103459,
  'f_val': 0.68609022556390986}},
```

Refinement

Once I selected the best classifer, I trained it using the Randomsearch cross validation technique. After this I created a grid search around the randomised search parameter values.

Then I predicted the test data values and append the new column containing the prediction, Pred_Down_Signal, to the list of columns to be dropped while creating the feature set for the next target variable.

I have repeated the same steps as above to train a classifier and save it's results for the Up direction in a column named Pred_Up_Signal.

Once I have generated both the signals, I created a sample trading strategy to check for the performance of the algorithm. This is not the Long Short portfolio, but a comparison of the buy and hold base case strategy with the classifier predictions.

The refined accuracy values on the test data after the GridSearch, the test performance is slightly worse compared to the validation accuracy

The test data accuracy for PNB and SBI respectively,

```
LogisticRegression model
------
Accuracy score on testing data: 0.7118
F-score on testing data: 0.7118
```

```
GradientBoostingClassifier model
------
Accuracy score on testing data: 0.6547
F-score on testing data: 0.6547
```

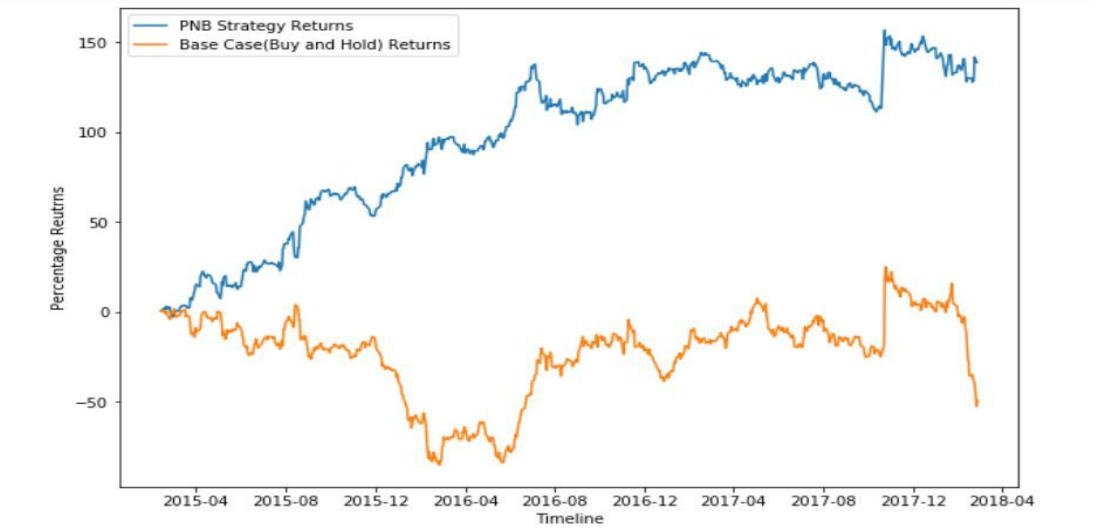
The Challenges:

The quality and quantity of data has been a challenge. Getting data that is authentic and freely available is a challenge. Earlier, Yahoo and Google provided reliable daily data for various stocks, but now their data sharing and usage policies have changed. For example, now you can only fetch the past one year data using Google, and for yahoo the data contains many NaN values and the continuity of the time series is lost. I found NSEpy to be very reliable and authentic, but the quantity of data is very limited. One can clearly see in the initial algorithm intuition part of the code that the accuracy of the algorithm keeps increasing as the data size increases. If more data can be fetched from a reliable data source then the algorithm will give a better result.

Results

Model Evaluation and Validation

The model is now tested on the latest 25% or the unseen data. I have plotted the performance of the sample strategies for the individual stocks and the sharpe ratio for the final Long Short portfolio.



Then I repeated the above procedure for the next stock,SBI, its prices are plotted below

The calculated Sharpe ratio for the Long short portfolio

```
In [207]:
1 Interest_Rate=6
2 Long_Short_Sharpe=(Net_Long_Short_Returns.sum()-Interest_Rate/100)/Net_Long_Short_Returns.std()

In [208]:
1 Long_Short_Sharpe

Out[208]: 18.6797330800003
```

Justification

In the final phase of the project, after performing the predictions on the second stock, I took the current existing market transaction cost. Then I created the final trading signals for the two stocks.

Next, I calculated the returns based on the predicted signals. Then i combined the positions of the two stocks on those days where they had opposite signals (+1 and -1)

From the final returns I subtracted the transaction cost to calculate the net returns after tax and brokerage.

Next, I plotted the Long Short portfolio returns along with the base case strategy returns to visualize their performances.

Next, I calculated the sharpe ratio on the net returns after assuming an average interest rate of 6% over the test data duration. This interest rate is the average interest rate during the test data time.

```
In [212]:  1 Percentage_Profitability

Out[212]: 63.157894736842103
```

I have calculated the model accuracy of various banking stocks along with their train data size, using an unoptimized logistic regression model. I have done this to check the robustness of the model on different stocks.

```
In [263]:  1 algo_results

Out[263]: {'ANDHRABANK': (0.70954907161803715, 2252),
           'AXISBANK': (0.64721485411140589, 2252),
           'HDFCBANK': (0.67241379310344829, 2252),
           'ICICIBANK': (0.66843501326259946, 2252),
           'IDBI': (0.6618037135278515, 2252),
           'INDUSINDBK': (0.67241379310344829, 2252),
           'PNB': (0.70291777188328908, 2252),
           'SBIN': (0.69230769230769229, 2251)}
```

The algorithm is very consistent in terms of accuracy and robust in performance,as long as the stocks of similar sector are used. This is a necessary prerequisite for a long short portfolio to remove any sector specific and market risks. The features that we have used will not work on all the stocks equally, so picking the stocks with high validation accuracy into the portfolio will help the performance further.

Conclusion

Free-Form Visualization

In this plot, I have projected the performance of the strategy and that of the base case portfolio. From this graph, I can clearly state two important things compared to the base case

1. The drawdown or the loss incurred by the classifier from it's past highs is very less at any point of time.
2. The performance of the strategy is clearly trending ot the upside with higher highs and lower lows.

Reflection:

I have summarized my entire project in a stepwise manner:

Step 1: Import Libraries and Data
Step 2: Check data visually and mathematically for consistency
Step 3: Data Preprocessing
Step 4: Creating Functions to generate Signals(target variables) and Indicators (Features)
Step 5: Standardize and Scale the data and Justify the approach adopted
Step 6: Train different Classifier algorithms from to choose the best performer from the validation set

Step 7: Train the classifier using Randomized search to get an intuition about the optimization parameters
Step 8: Use GridSearch to perform a thorough search in the vicinity of the best results obtained in step 7
Step 9: Print the accuracy score to verify if the results are consistent in train, test and validation data
Step 10: Repeat Process 6-9 for predicting a different target variable on same stock data
Step 11: Perform Backtesting and plot the results to see the performance of the algorithm
Step 12. Perform Step 1 to Step 11 for another stock in the portfolio
Step 13: Combine all the predictions and create the Long Short portfolio
Step 14: Plot the portfolio performance with respect to the base case portfolio
Step 15: Justifying the results and explaining the choices made for the stocks
Step 16: Detail the future course of action and any further improvements

One interesting thing that I have learnt is that even though the prediction accuracy is not as high as normal machine learning problems, a seventy percent accuracy, across various stocks that i tried out, seems to be the cut off or breakeven point for many of them. Stocks with more than 70% accuracy have done reasonably well, as it can be seen in the case of PNB. The accuracy is not a direct indication of profitability, as the returns of profitable days if lower than loss making days then the strategy will end up losing.

Improvement

This strategy can be used to further build an intra day trading strategy with the daily signals generated as inputs along with the above features.

While constructing the portfolio, apart from the sector consideration, it is better to consider the mean accuracy of the classifier on the validation set as a qualifying parameter for selecting a stock to the portfolio. From my experience, one can hedge risk best by taking the best performing (in terms of accuracy) stocks.

We can also use the feature set for every single day and with predicted signals for the past consecutive days as the states and their corresponding returns as the rewards, then use it to build a reinforcement learning model. This will optimize the algorithm further.

I have been working on an LSTM model to optimize this strategy further, hopefully I will be able to finish it soon.

Thank you.