Algorithmic Trading Bot

Arjun Khanijau

Abstract- We aim to dive into the field of financial machine learning by building an algorithmic trading bot. This would be used to make better investment decisions and for seeking out more profitable trades. A lot of time is spent by traders on continuous monitoring of transactions. These activities can instead be monitored by the trading bot, thereby saving cost and time for constant supervision. This would lead to a reduction in transaction costs. It would also greatly reduce the possibility of human errors while placing trades. The bot can be used for institutional investors and big brokerage houses for both stock market companies like Apple, Microsoft, and Tesla (AAPL, MSFT, TSLA) as well as cryptocurrency investments like Bitcoin and Ethereum (BTCUSD, ETHUSD). We have implemented two machine learning algorithms using the concepts of ensemble learning and support vector machine. A random forest regressor is implemented into the buy-and-hold trading strategy for long-term investments and a support vector regressor is implemented into the scalping trading strategy for short-term investments. We have also used backtesting for a successful deployment.

Index Terms— Ensemble learning, financial machine learning, hyperparameter optimization, random forests, regression, support vector machine, trading algorithms

1 INTRODUCTION

"ARTIFICIAL intelligence is to trade what fire was to the cavemen." This is how one industry player described the impact of algorithmic trading in the financial industry.

There is hardly any aspect of our lives that is untouched by artificial intelligence, and the financial sector is no exception. An estimated 70%-80% of overall trading volume is generated through algorithmic trading in many developed financial markets, including the US Stock Market. However, this figure is estimated to be around 40% for emerging economies like India, and this is where we want our project to create an impact. Our research aims to integrate engineering skills with the financial sector.

Algorithmic trading, or black-box trading, is used for speedily placing a trade to generate profits. This is done by using a set of automated and pre-programmed trading instructions. ML algorithms have proved to be highly useful to optimize the decision-making process of human traders as they manoeuvre data and predict the market picture with high accuracy.

The trading algorithms scan the markets for qualifying trade setups, and once the right setup is encountered, they execute trades and manage them through an entirely automated process!

ML algorithms are utilized to take in huge historical data and predict the future picture with accuracy. These help traders find time and space-limited localized patterns. These patterns are constantly changing, which would otherwise require a lot of time and energy to be spotted by human traders. The limitation is that patterns vanish immediately because of intense competition in the market. Due to this high competition, many traders in the same market use these machine learning algorithms for the same purpose. So, the patterns that are identified by one trader are also available for other traders in the market.

Our bot will attract investors of all types, young or old. It would also be beneficial for those retail traders who do intraday trading. According to some sources, 75% of retail traders lose money with intraday trading. This project would benefit them as it predicts data based on previous datasets. Automated trading would help corporate workers and other people busy with their work or daily schedules to focus on their work without thinking about their trades. It will also help beginners in the trading industry to understand and make trades quickly and conveniently. These trading bots make it very easy for both novice traders and seasoned ones to achieve successful results, and these results take minor work, time, and loss compared to humans. Incorporating economic and financial knowledge with AI and machine learning is very profitable for future trading and boosts both efficiency and revenue.

Compared to human traders, algorithmic trading leads to faster trading speeds and improved accuracy- With the advent of high-frequency trading (HFT), a large number of trading orders are executed within a fraction of seconds, adding liquidity to markets. Placing bids manually after reading market trends would take a significant amount of time whereas AI algorithms automate the process, recognize market movements with higher accuracy, enhance trading strategies and modify portfolios accordingly. Support vector machines (SVMs), utilized in high-frequency trading, create a line of separation in data and can identify the features which indicate an approaching variation in the bid and market pricing.

Algorithmic trading also eliminates human error. Human traders may get affected by intense market pressures, which may affect their judgement and lead to poor market decisions. Algorithmic trading aims to reduce such errors caused by psychological and emotional factors.

2 RELATED WORK

A. Stock market prediction using natural language processing (2019)

[1] In this patent they used a method of extracting information from online news feeds using natural language processing (NLP) techniques and then using that information to predict changes in stock prices or volatility. The algorithms are then used in a wide range of texts, including publications from online newspapers such as financial newsletters and the Wall Street Journal, as well as television transcripts, radio broadcasts, and annual reports. Their solution is made up of two parts: a text understanding component that fills in simple templates instantly and a statistical correlation component that analyses the relationship between this pattern and stock price gains or declines.

Similarities with our solution:
Both The solutions are related to financial trading algorithms, namely the assessment of rapidly changing sources of information combining natural language processing and user trading behaviour to predict fluctuations in stock price or volatility. Both These predictions can be utilised to develop successful trading strategies.

Differences between the patent and our solution:
In the patent, they used natural language processing in extracting information from news and parsing or pattern match on words to identify natural language text describing activities or announcements of a particular publicly-traded company to fetch the dataset. Where in our solution we used Zipline Data Portal Interface to Plot and Chart the Pricing Data using Matplotlib and analyse Candle Stick Charts

B. A kind of Stock Market Forecasting method merged based on sentiment analysis and HMM (2014)

[2] In this patent, they used a method of Sentiment analysis and HMM by Enhancing the accuracy of Stock Market Forecasting by employing the emotional tendency information in economic and financial news webpages and has great potential for application in domains like sentiment analysis, subject identification, Stock Market Forecasting, and Website content monitoring.

Similarities with our solution:
Both the solutions include gathering information, text extraction, information pre-processing, technically analysing the Stock Market, and improving the accuracy of Stock market prediction.

Differences between the patent and our solution:
In the patent, they used a method of Sentiment analysis and HMM, whereas in our solution we have used Zipline Data Portal Interface to Plot and Chart the Pricing Data using Matplotlib and analyse Candle Stick Charts

C. Coordination of algorithms in algorithmic trading engine (2010)

[3] In this patent, they used a method for optimizing algorithmic trading by making the chores of initiating and running algorithms easier while also giving real-time feedback on the user's automated trade executions. Preferred implementations of the specific topic system overcome recognised algorithmic trading products' limitations by
(1) allowing financial markets to use a simplified, instinctive graphical interface to click and drag complicated, multi-algorithm investment strategies,
(2) allowing users to track informational market impact costs in real-time, and
(3) automating the classification, management, and cancellation of algorithms based on user input.

Similarities with our solution:
Both the Patent’s and our solution’s goal is to provide real-time feedback to traders on both their order implementations and the market impact they have. Both the solutions are trying to complement rather than replace the value a human brings towards the trading process, broadening rather than narrowing his point of view on the industry and also how his orders affect it all through direct visual evidence of changes

in the market, the strategies his algorithms employ, and the degree of market effect these strategy cause.

Differences between the patent and our solution:
In the patent they used a graphical interface to start a complex, multi-algorithm investment strategy by using the drag and drop method, in our solution we fetched the data using Zipline Data Portal Interface to Plot and Chart the Pricing Data using Matplotlib and analyse Candle Stick Charts

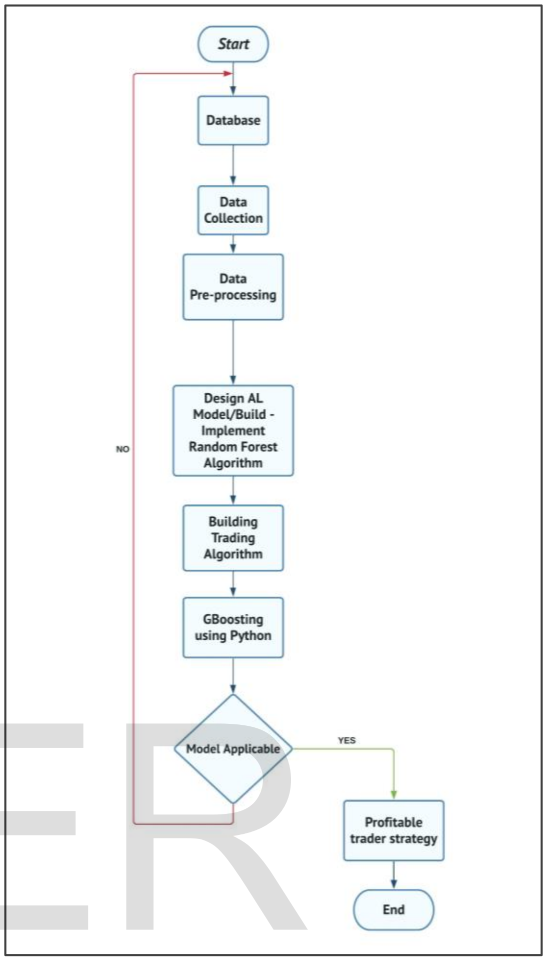
D. User-defined algorithm electronic trading (2021)

[4] In this patent, they tried to provide an algorithm Trading development tool that removes the problems associated with conventionally coded algorithms, like as syntax errors, ambiguous logic, and the necessity for a non-trader developer to construct the algorithm according to a trader's specifications. A market grid shows market data for a certain marketable item. A simulation indicative order input area generates feedback for assessing operational characteristics of an algorithm described as in the algorithm. along with an auto hedging option, a scratch quantity is employed. If a quantity in a market at a counter order's market price falls underneath the stated scratch amount, the counter order's price level decreases.

Similarities with our solution:
Both the patent and our solution refer to an automated trading platform Certain implementations related to a consumer online trading algorithm.

Differences between the patent and our solution:
In the patent, they are more focused on creating a competitive environment, including acquiring trading opportunities, such as computerised trading, where each second or fraction of a second count, aid market participants in competing effectively. Where in our solution we are a more focused growing number of market players to be engaged at any given moment and helping them in predicting fluctuations in stock price. Increasing the number of potential participants in the market

3 FLOW DIAGRAM


Start
Database
Data Collection
Data Pre-processing
Design AL Model/Build - Implement Random Forest Algorithm
Building Trading Algorithm
GBoosting using Python
Model Applicable
NO
YES
Profitable trader strategy
End

The flow diagram provided here is of our project algorithmic trading stating a closed form that is a solution to algorithmic trading strategies by building a financial machine learning model with the proper understanding of financial terminology and methodology. The flow diagram gives a conceptual guide about our project with the basic steps which we are going to apply.

Here we are going to ensemble our model, cross-validating in financial applications and backtesting. We are going to start our project with data collection by fetching the dataset for our design model. After the dataset collection, we will be building the conventional buy-and-hold strategy. Followed by data preprocessing, a machine learning model will be designed in which implementation of Random Forests Algorithm is built as well as going to plug-in RFI in our bot.

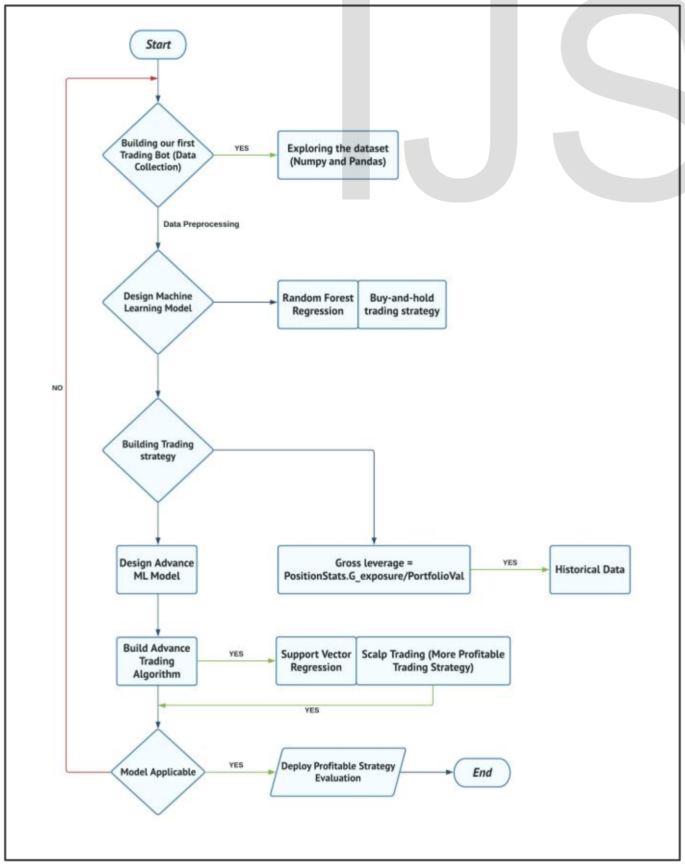
The random forests which we are using here are for a better result due to the bagging of many decisions trees which will sort the required features according to their importance. We will have a regression analysis for statistical analysis with the use of a regression tree called split criterion. After the complete evaluation of our RFI, we will be building a trading algorithm such as a One-pass algorithm, Linear algorithm, Scikit-learns, etc. followed by implementing exploit the correlation strategy.

We will then implement the GBoosting which is an ML technique for regression as well as classification problems.

We will build the model in a stage-wise way and will generate them by optimizing an arbitrary differentiable loss function using the Python language and then evaluating the model performance. As we reach this step, we going to introduce risk management which is very important as it gives a signal to the traders for the financial indicators for profitably driving the trades. In the advance trading algorithm, we are going to introduce a scalping strategy where it will be very helpful to the local investors due to which they will be less prone to risks and hence attractive. Many such advances are benefiting our project from settling targets to higher rise to the opportunity by proper observation of the trend. The assumed positions which we are keeping are the real-time trading which will ensure profitability. The process which we will be following for risk management will be defining goals, measuring risk followed by designing a system.

4 PROPOSED METHODOLOGY

A. Data extraction, preprocessing, and feature selection


Start
Building our first Trading Bot (Data Collection)
YES
Exploring the dataset (Numpy and Pandas)
Data Preprocessing
Design Machine Learning Model
Random Forest Regression
Buy-and-hold trading strategy
NO
Building Trading strategy
Design Advance ML Model
Gross leverage = PositionStats.G_exposure/PortfolioVal
YES
Historical Data
Build Advance Trading Algorithm
YES
Support Vector Regression
Scalp Trading (More Profitable Trading Strategy)
YES
Model Applicable
YES
Deploy Profitable Strategy Evaluation
End

We will be starting our project by building our Trading Bot by exploring our dataset using python libraries such as NumPy and pandas. This part will work on the services of statistics and probability and will introduce the financial terminology as well as financial data structures as these are essential in designing any sort of financial machine learning model and hence will build a bot skeleton. Data preprocessing is done through designing a machine learning model and diving deeper into ML and training and evaluating our model.
We will be fetching the data and the process used will be, installing the zipline into the packages of the python interpreter system. To retrieve publicly available data, set we are creating an account on quandl.com. To reduce the basic errors, we tried using Quantopian-quandl. We are going to take the help of getting a history window to get the data frame containing the history window as it is available as a member function of the zipline data portal interface used internally to answer the question about data. (df=data_port.get_history_window ()). We will be initializing the data portal interface which requires respected three mandatory parameters. The first parameter is asset unscored finder a method reference used internally to solve assets and here it's a member method of bundle objects which is initialized previously calling bundle. Methods and passing the name of quandl data sets. Our second parameter is the trading calendar used internally for minutes and session scheduling. The other two parameters in the data set are also defined as bundle objects.

To identify the combination of features and strategy parameters that can give the most accurate model we are using different segments of the dataset. To fetch the data, set we are going to use quandl and use for data preprocessing. We will also try using Quantopian-quandl for a more accurate result. For an accurate model and to evaluate the ultimate feature space the technique we are going to apply due to its iterative method is the Random Forest strategy in all the combinations of strategy data giving a statistical outcome of the most accurate model. In feature selection, we are using the filter method. This will pick up the properties which are intrinsic to the features. Since this method is faster and less expensive than wrapper techniques. So, it is cheaper to use the filter method. To measure the linear relationship of multiple variables and to predict the data we will be using a correlation coefficient. The logic we are generating behind using this for feature selection relation is the correlation with the target. We are keeping in mind the uncorrelation of the target variables between themselves. The information gain techniques are used to calculate the entropy further in transforming a dataset. We are also trying other techniques and will finalize based on the value which will be close to our profitable prediction values.

We are using the concepts of ensemble learning and support vector machine to implement the machine learning algorithms.

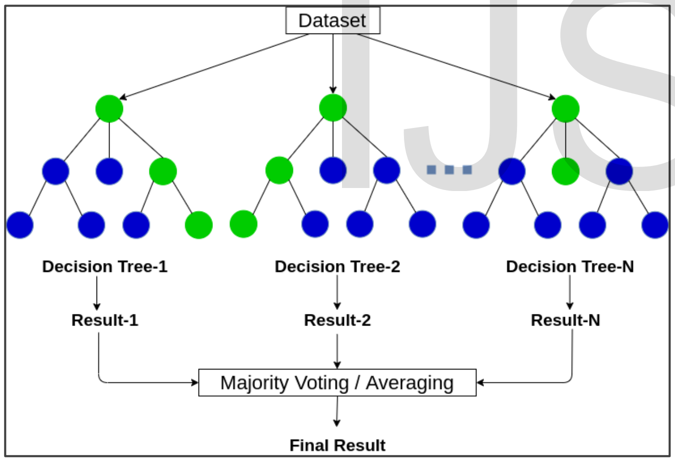
B. Ensemble learning

This works on the principle that multiple weak entities are more powerful than a single strong entity. If we use a single decision tree, the number of levels of the tree increases significantly with the increasing complexity of data. Increasing the number of levels of the decision tree makes it more prone to overfitting.

We can relate the use of diverse decision trees with the diversity of financial portfolios. Like it is always better to maintain a mixed portfolio across debt and equity funds to reduce risk, multiple and diverse decision trees lead to more efficient performance on unseen data. Ensemble learning aggregates multiple results leading to more stability and robustness, and a noise reduction. Another advantage of ensemble learning is that it can catch both linear and non-linear relationships of data by using different models and then forming their ensemble.

C. Bagging using random forest

Bagging is a combination of two terms- bootstrap and aggregating. Bootstrap refers to creating random samples with replacements from the training data and then building a decision tree for each of the samples. Aggregating refers to combining the results of multiple models using different techniques like average (for regression) and majority voting (for classification). We shall be using the average method since ours is a regression model for predicting stock prices.



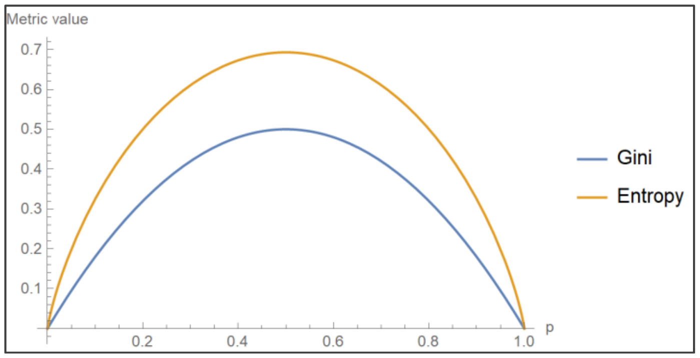
Bagging is a great way to decrease variance. The reason is that each model takes in a different subset of the training data due to randomness. This avoids overfitting the data.

Random forest uses the above concept. In addition, it also introduces randomness in selecting a different subset of features for every individual tree. This reduces the variance even further.

D. Missing values

Missing values can be handled by dropping the data points or by filling them with the median. We opted for dropping the data points with missing values using the dropna() function as this was giving better results. The reason is that the Zipline starting year of data availability is much before Tesla's foundation year, making the null values not useful.

E. Splitting criteria



Random forest needs splitting criteria which can be either the Gini index (used in the CART algorithm) or information entropy (used in ID3 and C4.5 algorithms).

Both Gini index and information entropy were giving similar results. But we shall be using the Gini index as it is faster. The reason is that entropy makes the use of logarithmic function, making it more computationally expensive. So, we shall use multiple CART trees in the random forest.

F. Calculation of time and space complexity
Let n= Number of nodes and d= Number of features

Time complexity for training runtime= O(dn2 log2n)
Reason- Training requires the values to be sorted at each node, which requires nlog2n time. This is multiplied by the number of nodes(n). This will be done for every feature so this will be multiplied by the number of features(d).

Time complexity for inference runtime= O(log2n)
Reason- One path of the tree needs to be traversed, which is logarithmic.

Space complexity= O(n)
Reason- As there are n nodes, linear space is required, which makes the model lightweight.

G. Parameters of Random Forest Regression

The first 32 columns of the data were used as features and the next 8 columns were set as the target. Due to the large size of the dataset, we used 40% of the data as testing data by setting the test_size parameter of the train_test_split() function to 0.4. We started with setting the number of trees to 50. This gave an R2
value of 0.75. This parameter was tweaked several times and the maximum value of R2 was recorded as 0.9937 using 100 trees. There was no significant improvement in the R2 value after increasing the number of trees beyond 100.

H. Integrating Random Forest Regressor to trading strategy

We integrate the above random forest model to the trading bot using Zipline simulation using a simple buy-and-hold strategy. In this strategy, stocks are bought and held for the long term, regardless of market fluctuations.

The trading strategy involves computing the maximum of the future predicted values to the historical mean of the stock. If this maximum value is greater than the historical mean, this indicates that the stock value is increasing and that it would be a good time to buy more shares of this stock. In this case, the bot would automatically order 1000 shares of the stock. On the other hand, if the maximum of the future predicted values is less than the historical mean of the stock, this indicates that it is a good time to exit our position by selling the stock shares as the stock value is reducing. In this case, the bot would automatically sell 1000 shares of the stock.

I. Support Vector Regression for implementing Scalpers Trading Strategy

Scalping is a more advanced trading strategy than the one we used before. Scalping is used to execute trades at very high speeds. This strategy works on the principle of opening and closing positions rapidly. This limits the exposure to the market. A strict exit strategy is needed so that one huge loss does not affect multiple small profits.

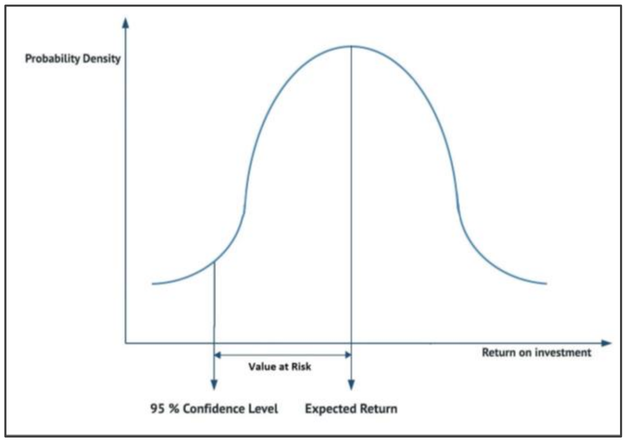
Support vector regression is a supervised model that uses the same concept as support vector machines to plot the best fit line. This best fit line is the hyperplane containing the maximum number of data points.

The advantage of using support vector regression is that it is very robust to outliers. Other regression models try to minimize the error between the ground truth and predicted values whereas SVR finds the best fit within a threshold value (distance between boundary line and hyperplane). The generalization capability of this model is superior to others, and it also gives a high prediction accuracy.

J. Parameters of Support Vector Regression

We extract 20 timesteps from the data out of which 15 are kept as lag timesteps which will be used to predict 5 future timesteps. We are passing 3 values to the 'estimator__kernel' parameter- linear, polynomial, and radial basis function (RBF) kernels. In this way, we will be able to find out which kernel gives the best performance. We used the 'best_estimator_' parameter of the GridSearchCV library to find out that the linear kernel was giving the best performance as compared to the other two kernels.

K. Risk management using CVaR (Conditional Value at Risk)



CVaR is used to quantify how much tail risk a portfolio contains. This is used for optimizing the portfolio for efficient management of risk. CVaR is calculated using the mean of the extreme losses which are present at the left tail of the distribution after concatenating the historical returns with the future returns. In our case, we choose this lowest percentile to be 5%.

This risk metric will trigger an action that would be executed whenever the returns for a particular trading minute fall below the value at risk. This action would trigger an exit signal to exit our trading position.

5 RESULT AND IMPLEMENTATION

A. Implementation details

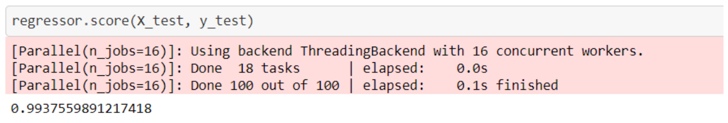
The machine learning models were implemented on the Jupyter notebook and then combined on distributed version control using Git. Our system is using the latest version of Mac OS Monterey having 16GB RAM and 14-core GPU. The 16-core neural engine helped in the efficient running of the model along with fine-tuning the hyperparameters.

B. Hyperparameter analysis

Since our project is based on regression, we have used a function named random forest regressor which we imported from sklearn.ensemble. We tried our model with min_impurity_decrease as 1 which was giving r score of 0.88, then after testing with many values finally we set its value to 0 which gave R2 value of 0.9937.

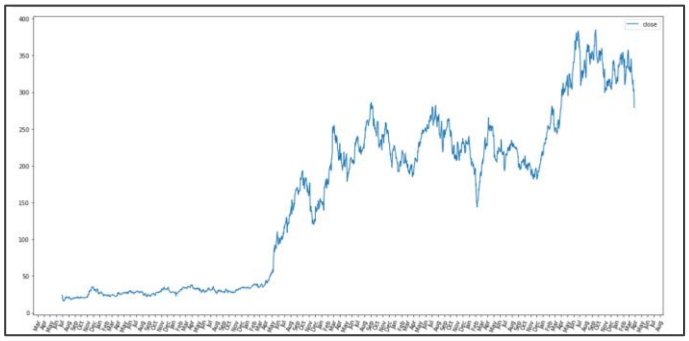
SECTION 1 – MACHINE LEARNING MODEL

In the image below we can see that the regression score which we are getting is 0.9937 which is a good score.

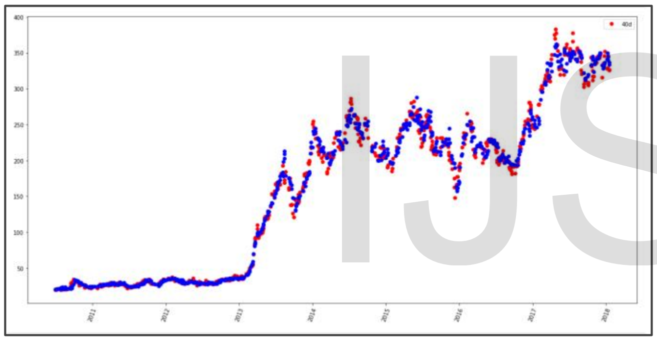


```
regressor.score(X_test, y_test)
[Parallel(n_jobs=16)]: Using backend ThreadingBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done  18 tasks      | elapsed:    0.0s
[Parallel(n_jobs=16)]: Done 100 out of 100 | elapsed:    0.1s finished
0.9937559891217418
```

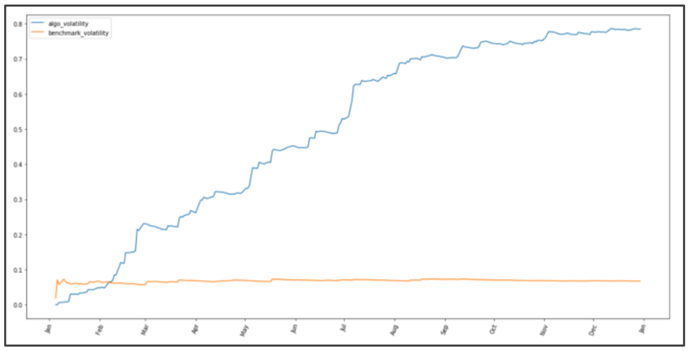
The graph below shows the price of Tesla stock over time.



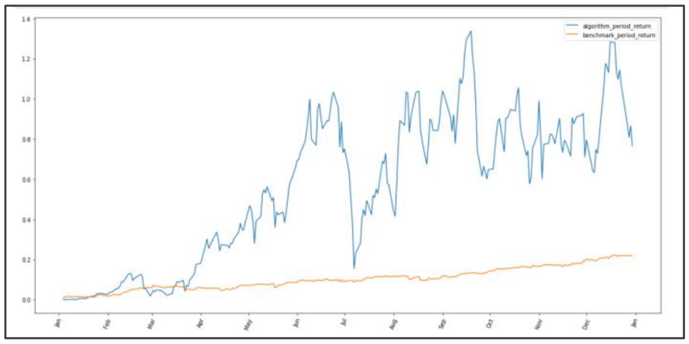
Here we can see there are two colours – red and blue. The red colour shows the ground truth(the data in the dataset) whereas the blue colour shows the data that our model predicted.



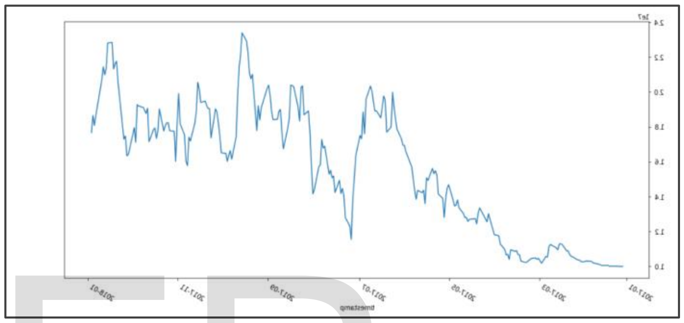
The graph below shows the algorithm volatility vs benchmark volatility. Here, our volatility is high, but since we know high risk equals high profit, it is required.



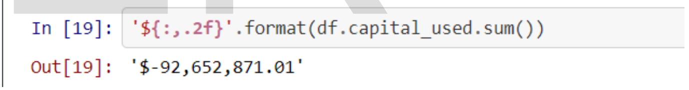
In the figure below the orange line indicates the benchmark return whereas the blue line represents the return that we are getting from our model which in comparison is much higher.



The graph below depicts the final portfolio value of the tesla stock and as we can see the final portfolio was somewhere around 24 million dollars

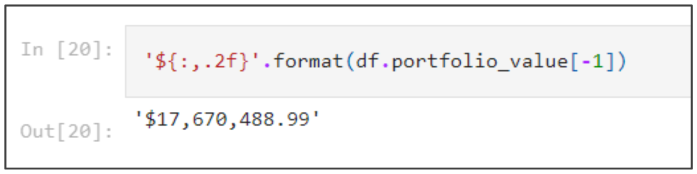


Here the output is negative which means that we have sold the stock for more value than we have purchased them indicating that we have done a profit.



```
In [19]: '${:,.2f}'.format(df.capital_used.sum())
Out[19]: '$-92,652,871.01'
```

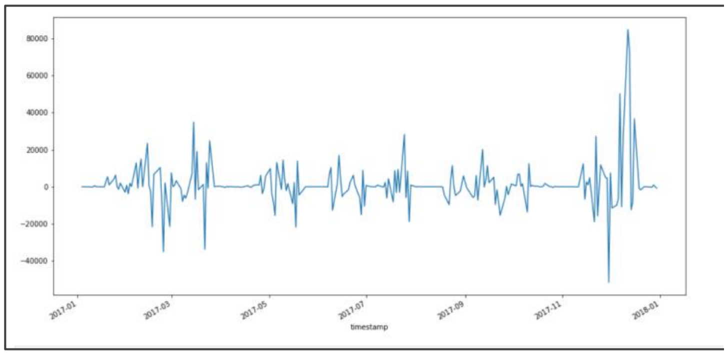
And this is the final portfolio value for a year which is around 17 million dollars



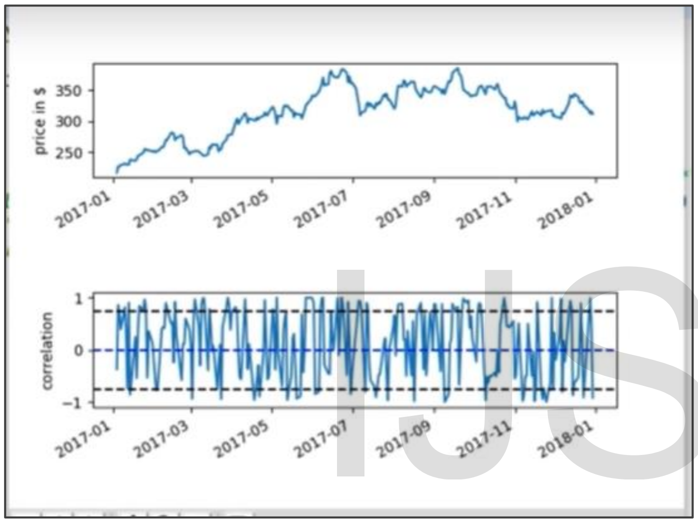
```
In [20]: '${:,.2f}'.format(df.portfolio_value[-1])
Out[20]: '$17,670,488.99'
```

SECTION 2 – TRADING ALGORITHM

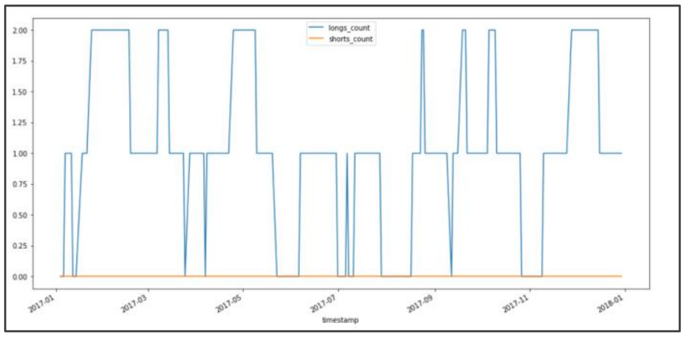
This graph depicts the profit or loss we made through our model in which when the bar is above 0 we experienced profit whereas when it is below 0 we experienced loss.



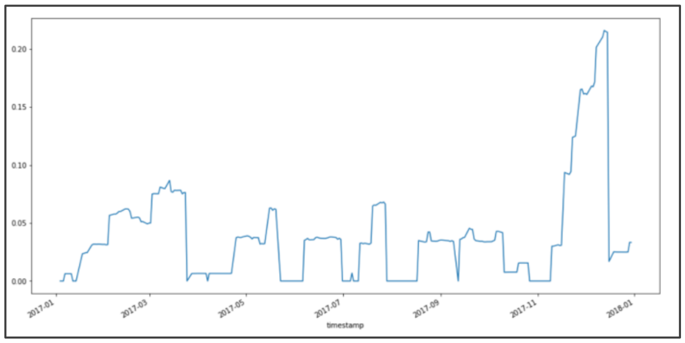
The graph below has two plots – one for price and the other one for correlation



The graph below shows that we have only entered the trades but not exiting them which is a sign of error in the long run.

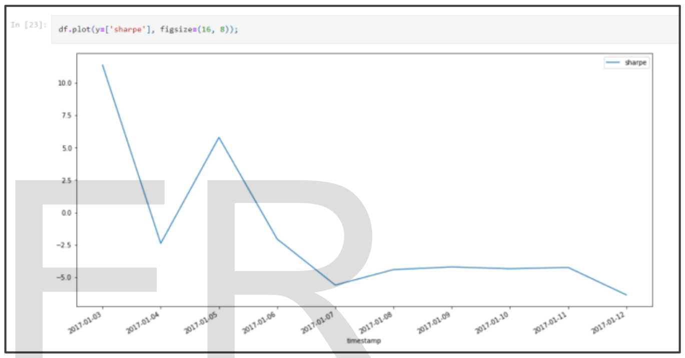


The graph below shows gross leverage



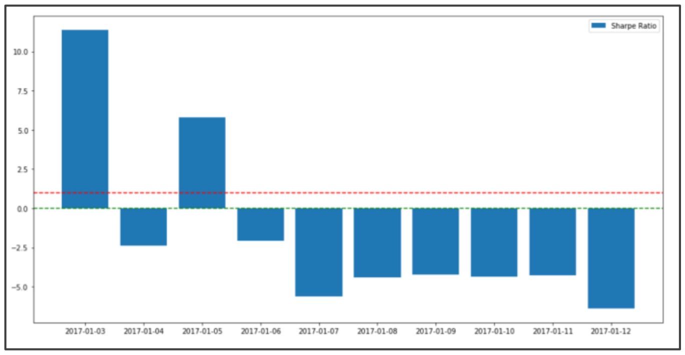
SECTION 3 – ADVANCED TRADING ALGORITHM

The graph we see below is the Sharpe ratio evaluation metric which is a financial metric often used by investors.



```
In [23]: df.plot(y=['sharpe'], figsize=(16, 8));
```

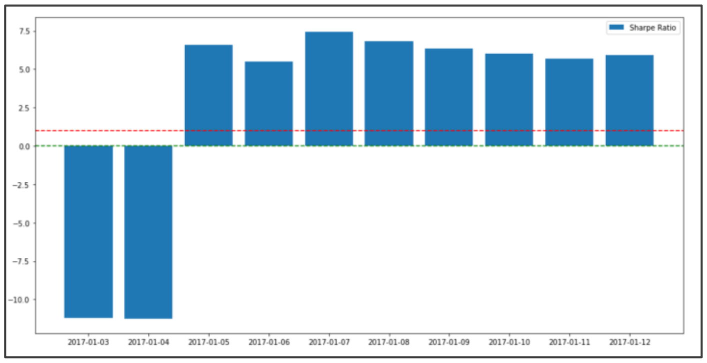
The graph below is the bar chart for sharpe ratios and here most of our values are on the negative side.



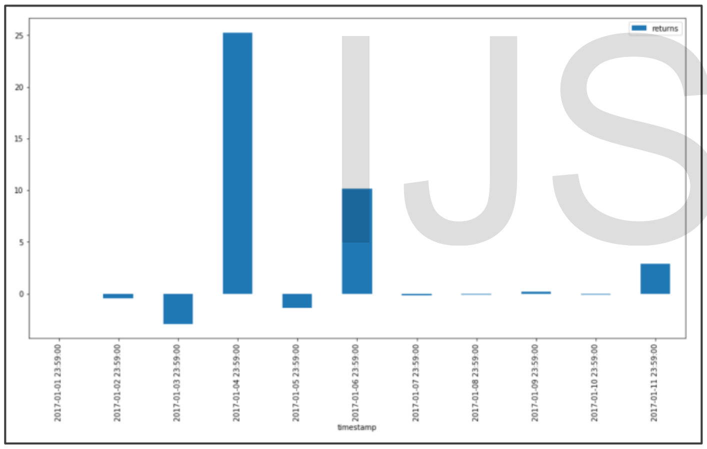
SECTION 4 – MODEL AND STRATEGY EVALUATION

Now for backtesting, we are using conditional VaR and implementing that using SVM. For that, we have imported SVR from sklearn.svm. As a parameter here we have used 'estimator_C' which will take a range of values between 1 and 10 where these two are also included. Then after grid searching, using 'clf.best_params_' we got our best value for estimator_c as '1' which gave us a good result.

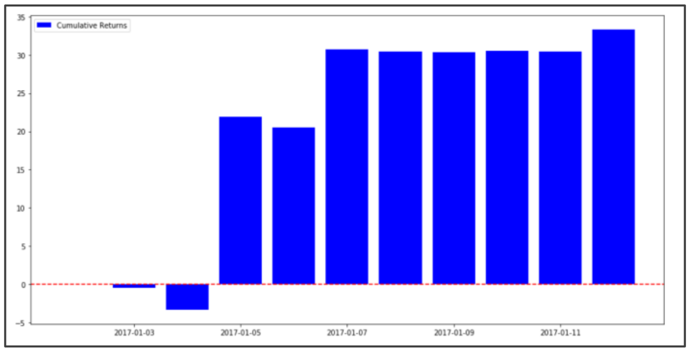
The below graph also depicts the Sharpe ratio, but this was at the time of backtesting. As we can see most of the time Sharpe ratio was in a positive direction. It was only in the beginning that it was negative



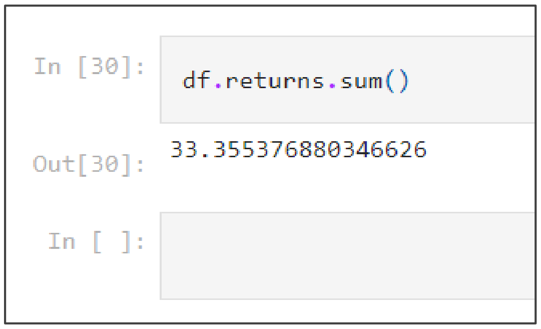
The graph below shows the returns which we have got and as we can see they are more in the positive direction.



The graph below depicts cumulative returns(which is defined as the sum of historical returns) and as we can see the return is very high.



Finally, the image below shows the percentage of returns i.e., we made around 33 per cent profit.



```
In [30]:  df.returns.sum()

Out[30]:  33.355376880346626

In [ ]:
```

B. Comparison

I.

A. Stock market prediction using natural language processing (2019)

[1] In this patent they used a method of extracting information from online news feeds using Neural Networks, a natural language processing technique and then using that information to predict changes in stock prices or volatility. While in our model we used Random forests and Support Vector Machines for the same. Random forests and SVM are comparatively inexpensive and don't require the use of a graphics processor to complete training. A random forest can provide a better understanding of a decision tree with better performance. Whereas for Neural Networks To be effective, they will require far more data than the average person has on hand. Support Vector Machines and Random Forest, on the other hand, require far fewer data as input. For the sake of performance, the neural network will just destroy the interpretability of the data to the level of making it just meaningless. Therefore, using Random forests and Support vector Machines over a Neural Networks is the best pick.

B. A kind of Stock Market Forecasting method merged based on sentiment analysis and HMM (2014)

[2] In this patent, they used Naïve Bayes, a method of Sentiment analysis and HMM to predict Stock Market Forecasting. All attributes are assumed to be conditionally independent in a Naïve Bayes. Since stock market prediction depends on various interdependent factors, the prediction, in this case, will be inadequate. We used Support Vector Machine to implement Scalper's trading strategy in our solution. Because it is not prone to mishaps, SVM can correlate with other parts within the data, allowing it to grasp

the dense characteristics in NLP, and thus results in sentimental analysis and machine translation, whereas

Naive Bayes' results are inconsistent. You don't anticipate the inputs to be substantially connected, thus Naïve Bayes is more of a generic approach that only works when we want to categorise a tiny corpus of data with a relatively limited number of input attributes.

C. Squareoff - Algo Trading firm

[5] Squareoff is a website that provides automated Trading bots based on quantitative trading strategies which automatically place trades in a Trading account. They used K-nearest neighbours (KNN) to achieve the same. When using KNN if the sample size is huge, there will be a high cost of computation during runtime. And since KNN is a lazy model, we must load all the training data and calculate distances to all training samples. The value of parameter K (number of nearest neighbours) and the type of distance to be utilized must also be determined. As we must compute the distance between each query instance and all training samples, the computation time is likewise very long. Therefore, the selection of K, as well as the metric (distance) to use in KNN, must be carefully calibrated. We used SVM in our model because Outliers are handled better by SVM than by KNN. Since there are many characteristics and little training data, SVM outperforms KNN, making or solution more efficient.

II. Comparison Table concerning Algorithms used

S. no	Evaluation Metris	A	B	C	Algorithmic Trading Bot
1	Algorithm	Neural Networks	Naieve Bayes	KNN	SVM and Random Forests
2	R2 Value	0.2567	0.36	0.9979	0.9937
3	Support Vector Regression profit %	12%	42%	27%	33%

6 CONCLUSION

We have successfully integrated the random forest regressor into the buy-and-hold strategy which would help investors make long-term investments without worrying about market fluctuations. The people who prefer a short-term investment can make use of our scalping trading strategy built using the support vector regressor. Upon backtesting, a high return value of 33% was obtained, making our models fit for deployment. This was possible by implementing a strict exit policy for short-term investments wherein the bot would make multiple small profits and exit the position before a huge loss can be inflicted.

7 REFERENCES

[1] https://patents.google.com/patent/US8285619B2/en

[2] https://patents.google.com/patent/CN103778215B/en?q=stock+market&oq=stock+market

[3] https://patents.google.com/patent/US8095455B2/en?q=algorithm+trading&oq=algorithm+trading

[4] https://patents.google.com/patent/JP2017117473A/en?q=algorithm+trading&oq=algorithm+trading

[5] https://squareoff.in/