STOCK MARKET ANALYSIS

Chandana Priya Kasimkuppam
cxk34130@ucmo.edu
University of Central Missouri

Sravan Kumar Goud Kasula
sxk26860@ucmo.edu
University of Central Missouri

Sai Pratap Jungili
sxj65600@ucmo.edu
University of Central Missouri

Abstract*—There has been an enormous number of researches on applying machine learning to forecast risk factor in buying or selling the stocks in stock markets worldwide. Many of the proposed models also consider the effect of transaction costs, which is an important factor for intraday trading. Most of the models examine the forecasting of price or direction of the underlying instrument only in the next time unit. Considering stock market instrument's underlying values as time series data points, predicting the value or direction for only the immediate data point is not justified. There has been also a lack of studies inspecting the predictability of profit over transaction costs for certain time durations ahead. This experimental research tries to predict the profitability over and above the transaction cost within the window of next few time units for an equity instrument traded in the National Stock Exchange in India. The underlying machine learning approaches used to perform the experiment are non-linear supervised algorithms like Logistic Regression and Extreme Gradient Boosting (XGBoost). Extensive research has been made to derive the independent variables to perform the experiment from direct price data points of underlying equity instruments. The experimental research suggests that XGBoost algorithm outperforms the Logistic Regression method in terms of predicting the profitability from trading of the underlying instrument.

Keywords*—Logistic Regression, XGBoost, Hyperparameter optimization, Risk Factor, Machine Learning

1. INTRODUCTION

Intraday trading in various stock market instruments is a very popular method of trading in major stock exchanges around the world mostly because of few reasons such as profit within a short span of time, minimal effect of economic factors, possibility of both long and short positions etc. Since, speed is a challenging factor to decide the position to be taken, most of the intraday trades placed in the exchanges these days are machine trades i.e. computers decide the trade to be taken. Underlying algorithms to machine trades require you to be intelligent enough to make accumulated profits over the long run. Hence, being able to accurately forecast the trades is significant to researchers worldwide. The prediction of any tradable instrument is complex due to the inherent nature of financial time series consisting of noise and non-stationarity. Noise refers to the serially uncorrelated random variables with zero mean and finite variance. Thus, it is extremely difficult to establish a dependency relation between future data points with respect to the present data point. The nonstationary refers to the constant change in mean and variance of the time series data. Change in value of any tradable instrument occurs due to uncountable factors such as sentiment of traders, reaction of participating algorithms, economic or political change etc. Henceforth, predicting the profitability of any tradable instrument in stock exchange is extremely difficult. The main aim of this paper is to design an efficient model which will accurately predict the trend of stock market using extreme Gradient Boosting(XGBoost) which has proved to be an efficient algorithm with over 87% of accuracy for 60 day and 90 day periods and it has proved to be much better when compared to traditional non-ensemble learning techniques

Even though there exist many literatures to predict the price or direction of any tradable instrument, price or direction cannot be accurately predicted for the immediate next time period due to enormous number of factors involved in change of the price. Since the fact is that different traders and algorithms employ different strategies to trade any instrument, conducting an empirical study is important to analyze the behavior of price over next few time periods instead of restricting to next. Also, predicting the direction of the underlying instrument does not necessarily account for profitability, almost certainly not when the chosen forecasting period is restricted to the immediate next period. Thus, there seems to exist a gap in existing literatures to analyze the predictability of profit in intraday financial time series considering the transaction costs and forecasting period. The proposed experiment study realizes the rapid growth of algorithmic trades in Indian stock market and tries to accurately predict the profitability of intraday applying nonlinear classification techniques in intraday financial time series.

The prediction problem has been reconstructed as a classification problem and XGBoost turned out to be significantly better than the algorithms found in literature.

Although we have many models based on decision trees. All these models have some disadvantages of having biased overfitting, poor handling of missing values and more computational. As XGBoost overcame all the problems and showed better results in prediction we would like to use XGBoost algorithm to predict future stock prices.

In recent times, a growing number of experiments have been performed considering the trend of instruments traded in stock markets e.g. O' Connor, Remus and Griggs, 1997[2]. These days, many foreign institutional investors are more attracted towards developing markets. According to Harvey, 1995[3] developing markets contain more regional information than developed markets; thus, predicting developing markets are comparatively less complex than developed markets.

An extremely new machine learning technique known as Extreme Gradient Boosting (Xgboost) is applied to solve multiple machine learning problems in diverse domains. The underlying method employs traditional Gradient Boosting machine learning techniques. It was initially started as a research project by Tianqi Chen and Carlos Guestrin[4] as part of Distributed (Deep) Machine Learning Community (DMLC) group, whereas the first version has been released on early 2014. The major benefit of Xgboost is that it supports distributed processing environments like Apach Spark and Apach Hadoop, which are being used widely in big data analytics research areas. Thus, the proposed experimental research considers Xgboost to model the underlying instrument's time series data. This research does not consider Decision tree and Gradient Boosting because Xgboost is based on the original model of Gradient Boosting.

2. METHODS

2.1 Workflow

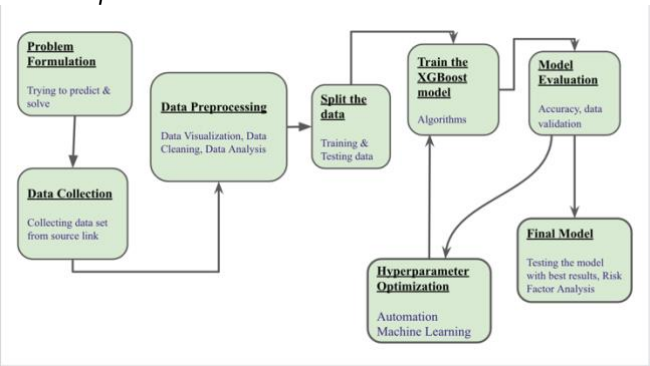

Fig. 1. Data Workflow Process

2.2 Extreme Gradient Boosting (XGBoost)

The underlying principle behind XGBoost is Gradient Boosting and Gradient Boosting itself relies heavily on Gradient Descent

2.2.1 Gradient Descent

Considering x being scalar, let f(x) be the function to be minimized. One way to iteratively minimize and find the corresponding x at the minima is to follow below update rule at the i th iteration:

$x(i) = x(i-1) - q.df(x(i-1))/dx$

Where q is a positive constant and x (0) can be any arbitrary value. In effect, the value of x found in the current iteration is its value in the previous iteration added to some fraction of the gradient (slope) at the previous value.
The iteration is stopped when x (i) = x (i-1)

In effect, every move is considered estimating an amount proportional to the gradient, because the gradient has to gradually become 0 near the minima, and the gradient is higher farther away from minima. That is why longer step iterations are taken when farther away from minima, whereas shorter steps are taken when nearer to minima. In similar fashion if x is a vector, the theory remains the same. Thus, for the i th iteration and the j th dimension, the update rule would be:

$x(i)\,j = x(i-1)\,j - q.df(x(i-1)\,j)/dx$

All dimensions are adjusted at every iteration i.e. the vector x itself is moved in a direction where each individual component minimizes f(x).

2.2.2 Gradient Boosting

Gradient Boosting incorporates the technique of Gradient Descent in supervised learning i.e. a function f(x) is minimized. A loss function L is incorporated whose value increases when the classifier performance degrades. For Gradient Boosting loss functions, must be differentiable e.g. the squared error between the actual and predicted value: $L = (yi - h(xi))2$ Hence, $f(x) = \sum N\ i=1\ L(yi\ ,\ h(xi))$ loss requires to be minimized for all points, where h(x) is the classifier and N is the number of points. Therefore, as like gradient Descent, minimization requires to happen with respect to classification function h(x), because a predictor requires to be established that minimizes total loss of f(x). The minimization is performed in multiple steps, where at every step a tree is added that emulates adding a gradient based correction as like in GD. The h(x) after the most minimized step becomes the ultimate result, where the classification function exists as a bunch of trees and each tree represents the update in some iteration.

2.2.3 XGBoost

XGBoost follows the same principle of gradient boosting but includes regression penalties in the boosting equation. XGBoost uses a more regularized model formalization to control over-fitting but it also leverages the structure of the underlying hardware to speed up computing times and facilitates memory usage, which are very important resources to consider while performing computation of boosted tree algorithms. Thus, XGBoost provides a better real time computational performance.

Fig. 2. XGBoost in Machine Learning

2.3 Hyperparameter Optimization

Manual tuning takes time away from important steps of the machine learning pipeline like feature engineering and interpreting results but requires long run times because they waste time evaluating unpromising areas of the search space.

Increasingly, hyperparameter tuning is done by automated methods that aim to find optimal hyperparameters in less time using an informed search with no manual effort necessary beyond the initial set-up.

There are four parts to a Bayesian Optimization problem:

i. *Objective Function:* what we want to minimize, in this case the validation error of a machine learning model with respect to the hyperparameters.

ii. *Domain Space:* hyperparameter values to search over.

iii. *Optimization algorithm:* method for constructing the surrogate model and choosing the next hyperparameter values to evaluate.

iv. *Result history*: stored outcomes from evaluations of the objective function consisting of the hyperparameters and validation loss.

With those four pieces, we can optimize (find the minimum) of any function that returns a real value. This is a powerful abstraction that lets us solve many problems in addition to tuning machine learning hyperparameters.

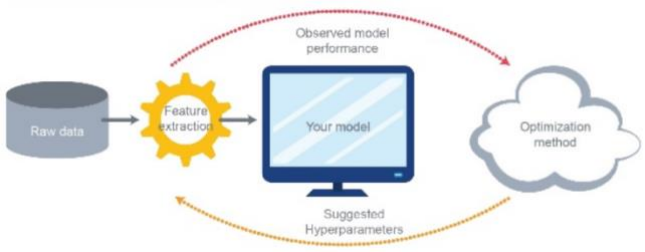

Fig. 3. Hyperparameter Optimization (Automated Machine Learning)

2.4 Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for classification problems, it is a predictive analysis algorithm and based on the concept of probability.

Types of Logistic Regression

a. Binary Logistic Regression

The categorical response has only two 2 possible outcomes. Example: Spam or Not

b. Multinomial Logistic Regression

Three or more categories without ordering.

Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

c. Ordinal Logistic Regression

Three or more categories with ordering. Example: Movie rating from 1 to 5

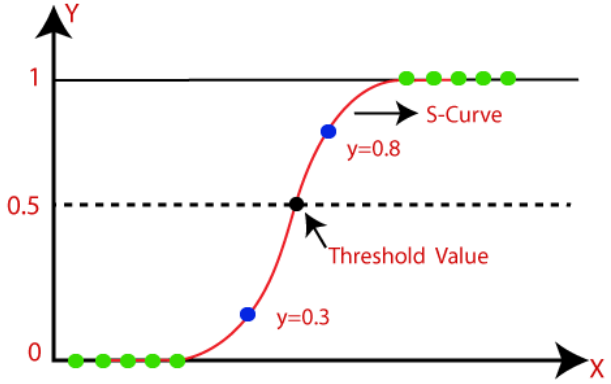

Fig. 4. Logistic Regression in Machine Learning

3. EXPERIMENTAL RESULTS

3.1 Data Collection

The data considered for this study was collected from Kaggle. The raw data format is shown in Table 1 with a sample from the complete dataset.

Table 1. Raw Data Format

Date	Open	High	Low	Close	Volume	OpenInt
1999-11-	30.71	33.75	27.00	29.70	66277506	0
1999-11-19	28.98	29.0	26.87	27.25	16142920	0
1999-11-22	27.88	29.70	27.04	29.70	6970266	0
1999-11-23	28.68	29.44	27.00	27.00	6332082	0

3.2 Data Preprocessing

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or encoded, to bring it to such a state that the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

3.2.1 Data Cleaning

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

Table 2. Processed Data Table

Date	Open	High	Low	Close	Volume	Label
1999-11-18	30.71	33.75	27.00	29.70	66277506	a
1999-11-19	28.98	29.0	26.87	27.25	16142920	a
1999-11-22	27.88	29.70	27.04	29.70	6970266	a
1999-11-23	28.68	29.44	27.00	27.00	6332082	a

3.2.2 Data Visualization

It helps in getting a high-level statistical overview on how the data is and some of its attributes like the underlying distribution, presence of outliers, and several more useful features.

Visualized the data with the graphical representation and produced some images that communicate relationship among represented data.

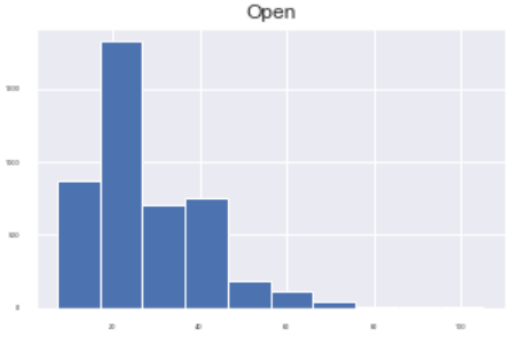

Fig. 5. Histogram for Open Price

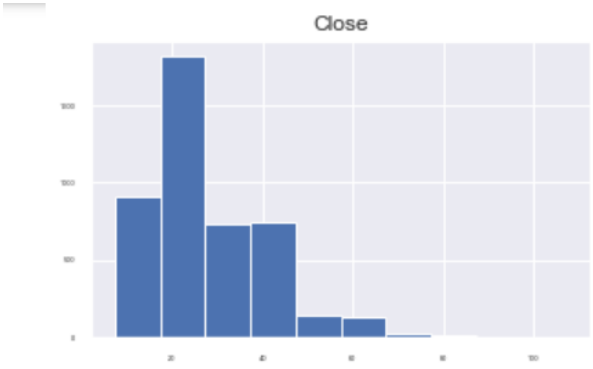

Fig. 6. Histogram for Close Price

Fig. 7. Graph between Year and Close Price

3.2.3 Data Analysis

Data analysis is a process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, informing conclusion and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains. In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively

Table 3. Processed Data Format

S_10_Close	S_10_High	S_10_Low	Corr_Close	Corr_High
30.07	30.68	29.65	-0.33	-0.38
30.28	30.83	29.57	-0.13	-0.18
30.38	30.93	29.64	-0.12	-0.05
30.46	30.96	29.70	0.10	0.12
30.55	31.01	29.82	0.27	0.23

Table 3. Processed Data Format

Corr_Low	RSI_Close	RSI_High	RSI_Low	Open_Close
-0.32	39.21	26.66	40.56	-0.67
-0.57	58.78	50.41	48.19	0.20
-0.49	54.21	49.94	65.27	-0.58
-0.25	56.88	49.23	67.62	0.20
-0.15	55.87	48.16	66.73	-0.12

Table 3. Processed Data Format

Close_Close	Open_Open	Volume_Diff	High_Diff	Low_Diff
-0.88	-2.28	-623964.0	2.07	0.71
3.75	0.00	353757.0	-3.33	-0.67
-0.88	2.95	-1191192	0.08	-23.24
0.63	-0.08	2661030	0.12	-04.60
-0.17	0.29	-1126007	0.17	0.08

3.3 Split the Data

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset in order to test our model's prediction. In our case 70% of data is used for the training set and 20% data has been used for the test set.

3.4. Select and Train the Model

Train the algorithm with a training dataset, then make predictions on the test set and evaluate the predictions against the actual results with the help of confusion matrix and determine the accuracy in prediction results.

3.5. Model Evaluation

Each model's performance is evaluated based on the accuracy derived from the confidence matrix of the prediction output. A confusion matrix is a summary of prediction results on a classification problem, where correct and incorrect predictions are summarized with count values and broken down by each class. Table 3 gives an overview of confusion matrix for two categories of output classes, where rows refer to class outputs in actual test data and columns refer to class outputs from prediction. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Table 4. Confusion Matrix

	True	False
True	Ctt	Ctf
False	Cft	Cff

It is obvious from Table 4 that total occurrences of correct predictions amount to (Ctt + Cff) and total occurrences of incorrect predictions amount to (Ctt + Ctf + Cft + Cff).

Thus, accuracy for two class outputs can be derived from the formula:

a = (Ctt + Cff) / (Ctt + Ctf + Cft + Cff) * 100

Table 5. Models Accuracy

Model	Confusion Matrix	Accuracy
Logistic Regression	[[0 368] [0 983]]	72.76%
XGBoost	[[221 147] 199 784]]	74.39%

3.5.1 K- Fold Cross Validation

Cross validation is an approach that you can use to estimate the performance of a machine learning algorithm with less variance than a single train-test set split.

It works by splitting the dataset into k-parts (e.g. k=5 or k=10). Each split of the data is called a fold. The algorithm is trained on k-1 folds with one held back and tested on the held back fold. This is repeated so that each fold of the dataset is given a chance to be the held back test set.

After running cross validation, you end up with k different performance scores that you can summarize using a mean and a standard deviation.

Table 6. Models Accuracy

Predictions Score	Logistic Regression	XGBoost
Precision	72.76%	84.21%
Recall	100.00%	79.76%
Area Under ROC Curve (AUC)	50.00%	69.91%
Cross Validation (10-fold)	71.64%	74.06%

3.5.2 Classification Report

The f1-score tells you the accuracy of the classifier in classifying the data points in that class compared to all other class. It is calculated by taking the harmonic mean of precision and recall. The support is the number of samples of the true response that lies in that class. This Report is used to measure the quality of predictions from a classification algorithm.

Table 7. Classification Report of Logistic Regression

	Precision	Recall	F1-score	Support
0	0.00	0.00	0.00	368
1	0.73	1.00	0.84	983
Accuracy	NA	NA	0.73	1351
Macro Avg	0.36	0.50	0.42	1351
Weighted Avg	0.53	0.73	0.61	1351

Table 8. Classification Report of XGBoost

	Precision	Recall	F1-score	Support
0	0.53	0.60	0.56	368
1	0.84	0.80	0.82	983
Accuracy	NA	NA	0.74	1351
Macro Avg	0.68	0.70	0.69	1351
Weighted Avg	0.76	0.74	0.75	1351

3.6 Hyperparameter Optimization

Below are the results for the Automated Machine Learning with XGBoost Algorithm.

Table 9. Parameter Tuning Details

Iteration	1	2	3	4	5
n_estimators	143	506	758	416	915
max_depth	5	4	6	3	3
gamma	0.08	0.36	0.37	0.12	0.46
colsample_bvtree	0.54	0.45	0.80	0.59	0.97
learning_rate	0.89	0.16	0.86	0.78	0.44
subsample	0.95	0.75	0.65	0.95	0.80
min_child_weight	500	100	800	800	200
reg_alpha	0.23	0.37	0.89	0.97	0.03
reg_lambda	0.67	0.78	0.47	0.09	0.03
Accuracy	73.1%	72.98%	72.83%	72.32%	73.72%

Table 10. Best Parameter Details

colsample_bytree	0.5210349943192396
gamma	0.43575482460187526
learning_rate	0.8414364163095022
max_depth	4.0
min_child_weight	600.0
n_estimators	941.0
reg_alpha	0.7790374624589286
reg_lambda	0.5553060531376773

subsample	0.9

3.7 Risk Factor Analysis

By analyzing the underlying exposures of stocks, funds and strategies, investors can identify which factors are providing the best risk-adjusted returns. This process is called factor analysis and allows investors to target the inherent risks which they believe will yield the best returns. There are many ways we can quantify risk, one of the efficient ways is by using the information we have predicted i.e. comparing the daily prices (open and close) mean with every year.

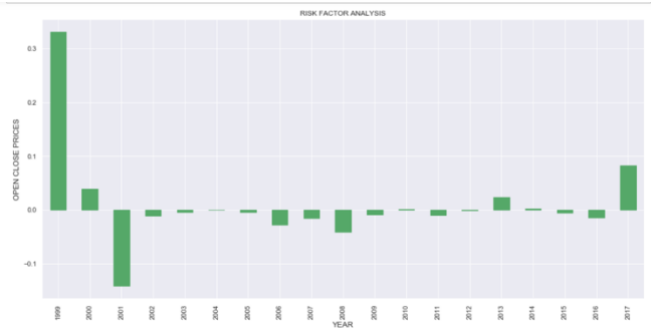

Fig. 8. Histogram for Risk Factor Analysis

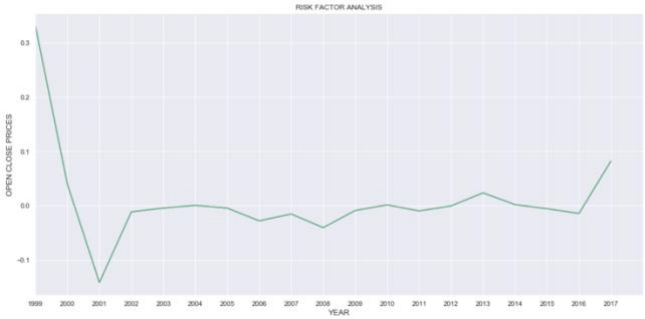

Fig. 9. Graph for Risk Factor Analysis

4. CONCLUSION AND FUTURE WORK

This experimental study used Logistic Regression and XGBoost to predict the risk factor in buying or selling the stocks in the stock market. The experimental results showed that XGBoost outperformed Logistic Regression. The reason behind the better performance of XGBoost models over the other model is due to the reason that XGBoost tries to fall into local minima using Gradient Boosting considering multiple trees.

Possible application of this study is to prepare a complete strategy considering other measures of the stock market. Since, forecasting risk factors in buying or selling the stocks is impressing, a complete intraday strategy can be implemented and back tested.

A further study can be performed by experimenting the independent variables used for modelling the classification in this study.

5. ACKNOWLEDGEMENT

We thank Zhiguo Zhou, Ph.D. Assistant Professor of Computer Science, University of Central Missouri for assistance with Extreme Gradient Boosting and Automated Machine Learning methods to predict the risk factor in buying/selling stocks.

6. REFERENCES

[1] Leung, M. T., Daouk, H., Chen, A. S., ―Forecasting stock indices: a comparison of classification and level estimation models‖, International Journal of Forecasting, 16, 2000, 173–190.

[2] O' Connor, M., Remus, W., & Griggs, K., Going up-going down: ―How good are people at forecasting trends and changes in trends?‖, Journal of Forecasting, 16, 1997, 165–176.

[3] Ferson W. E., Harvey C. R.., ―The risk and predictability of international equity returns‖, Review of Financial Studies, 6, 1993, 527–66

[4] Tianqi Chen, Carlos Guestrin, "A Scalable Tree Boosting System", 2016.

[5] Takashi, K., Kazuo, A., ―Stock Market Prediction System with Modular Neural Network‖, International Joint Conference on Neural Networks, 1, 1990, 1-6.

[6] Qin Qin, Qing-Guo Wang, Jin Li, Shuzhi Sam Ge, "Linear and Nonlinear Trading Models with Gradient Boosted Random Forests and Application to Singapore Stock Market", Journal of Intelligent Learning Systems and Applications, 2013, 5, 1-10.