## Optiver Realized Volatility Prediction

A Mathematical Programing Report

Submitted in the partial fulfillment of the requirements for the award of the degree of

# Bachelor of Technology in

Department of Computer Science and Engineering

By

Ankush kumar (2010030360)

Charan Reddy (2010030376)

Bharath Simha Reddy (2010030257)

Rithvik Reddy (2010030310)

under the supervision of

# **Dr. Anal Paul**

# Assistant professor



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

March, 2022

**Declaration**

The mathematical programming Report entitled “Optiver Realized Volatility Prediction” is a record of Bonafede work of Ankush kumar (2010030360) Charan Reddy (2010030376) Bharath Simha Reddy (2010030257) Rithvik Reddy (2010030310)

submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/ University/ Institute.

**Certificate**

This is to certify that the Mathematical programming Report entitled “Optiver Realized Volatility Prediction” is being submitted Ankush kumar (2010030360) Charan Reddy (2010030376) Bharath Simha Reddy (2010030257) Rithvik Reddy (2010030310)

submitted in partial fulfillment for the award of B.Tech in CSE to the K L University, Hyderabad is a record of Bonafede work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

## Signature of the Supervisor

Dr. Anal Paul

Assistant Professor

## Signature of the HOD Signature of the External Examiner

**ACKNOWLEDGEMENT**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this Social Internship. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for his exemplary guidance, monitoring, and constant encouragement throughout the course of the Social Internship. We thank Mrs. Anuradha of our department who has supported throughout this Social Internship holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this Social Internship a reality. We would like to extend our sincere thanks especially to our parent, our family members and friends who have supported us to make this Social Internship a grand success.

**ABSTRACT**

Volatility is one of the most prominent terms you’ll hear on any trading floor – and for good reason. In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated to periods of market turbulence and to large price swings, while low volatility describes more calm and quiet markets. For trading firms like Optiver, accurately predicting volatility is essential for the trading of options, whose price is directly related to the volatility of the underlying product.

As a leading global electronic market maker, Optiver is dedicated to continuously improving financial markets, creating better access and prices for options, ETFs, cash equities, bonds and foreign currencies on numerous exchanges around the world. Optiver’s teams have spent countless hours building sophisticated models that predict volatility and continuously generate fairer options prices for end investors

**TABLE OF CONTENT**

|  |  |
| --- | --- |
| S.No | Topic |
| 1 | Abstract |
| 2 | Introduction |
| 3 | Problem Statement |
| 4 | Objective |
| 5 | Methodology |
| 6 | Data set |
| 7 | Code |
| 8 | Software/Hardware requirements |
| 9 | Conclusion |
| 10 | Thanking |

**Introduction**

Forecasting volatility is crucial in risk management and asset pricing in general. The availability of high-frequency price data over the past two decades has spurred the field of modeling and forecasting realized variance, RV , estimated by summing squared intraday returns.1 Most of the existing RV forecasting models propose a handful of new predictors and then examine them one by one within the framework of classical statistical inference. In this paper, instead of arguing the dominance of a particular feature or algorithm, we have an ambitious objective: building an automatic forecasting system that: 1) reduces human intervention in choosing features and algorithms; 2) scales to fit many features while controlling for overfitting; 3) utilizes more flexible and state-of-the-art learning algorithms; and 4) achieves good and consistent out-of-sample performance. Our system has two main components: feature engineering and learning algorithm fitting. In the feature engineering step, we consider 118 features that might be useful in predicting future volatility, including 16 realized-variance-based (RV -based) features proposed by five popular RV forecasting models and 102 implied-variance-based (IV -based) features across all deltas and with maturity between one and three months. Our feature set is, to the best of our knowledge, the largest that has ever been examined in the volatility forecasting literature. In the learning step, we aim to learn the relation between volatility and features by five popular learning algorithms: LASSO, Principal Component Regression (PCR), Random Forecast (RF), Gradient Boosted Regression Trees (GBRT), and Neural Network (NN). These algorithms are more prediction-oriented and capable of capturing complicated relations than simple OLS. Rather than emphasizing the performance of a particular algorithm, we consider a simple combination of all machine learning algorithms.

**Problem Statement**

I formulated a classification problem with the goal of predicting, for each point in time (e.g. each trading day), whether or not a crash will occur within the next 1, 3, or 6 months.

If past price patterns are indicative of future price events, the relevant information to make a prediction on a certain day, is contained in the daily price changes of all days prior to that day. Thus, to predict a crash on day t, the daily price changes from each day prior to t could be used as a feature. However, because models presented with too many features do get slower and less accurate (“curse of dimensionality”), it makes sense to extract a few features that capture the essence of past price movements at any point in time. I therefore defined 8 different time windows that measure mean price changes over the past year (252 trading days) for each day. I used increasing window sizes from 5 days (leading up to day t) to 126 days (for t-₁₂₆ to t-₂₅₂) to get a higher resolution of price changes in more recent times. Because price volatility is not captured when averaging price changes over multiple days, I added 8 features for the mean price volatilities over the same time windows. For each data set I normalized the mean price changes and volatilities.

**Objective**

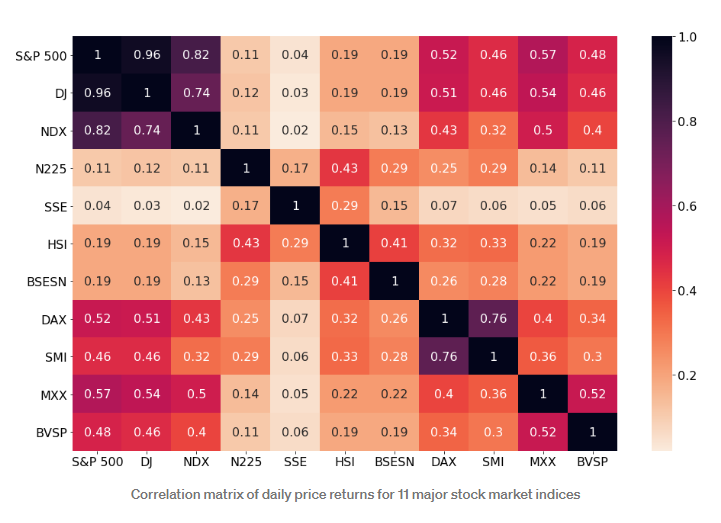
A stock market crash is a sharp and quick drop in total value of a market with prices typically declining more than 10% within a few days. Famous examples of major stock market crashes are the Black Monday in 1987 and the real estate bubble in 2008. A crash is usually attributable to the burst of a price bubble and is due to a massive sell-off that occurs when a majority of market participants try to sell their assets at the same time.

**Methodology**

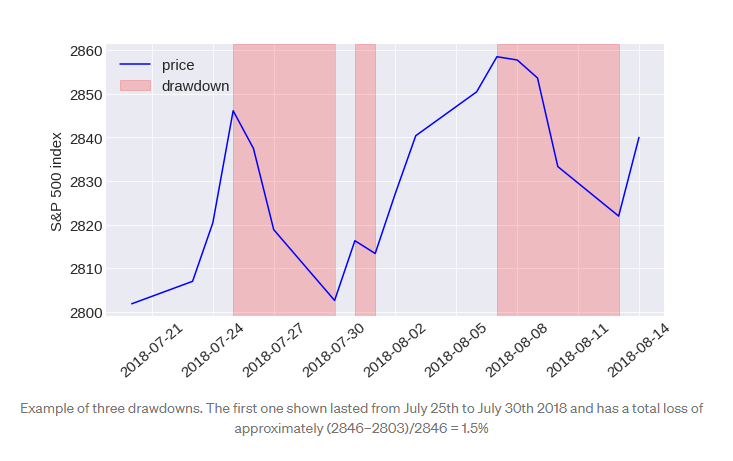
This section reviews the five machine learning algorithms we investigated in this paper. The first two are linear: Least Absolute Shrinkage and Selection Operator (LASSO) and Principal Component Regression (PCR). The next three are nonlinear: Random Forest (RF), Gradient Boosted Regression Trees (GBRT) and Neural Network (NN)

**Data**

The first step was to collect financial data and identify crashes. I was looking for daily price information from low correlated major stock markets. Low cross-correlation is important for valid cross validation and testing of the model. The matrix below shows the cross-correlation of daily returns from 11 major stock markets.

****

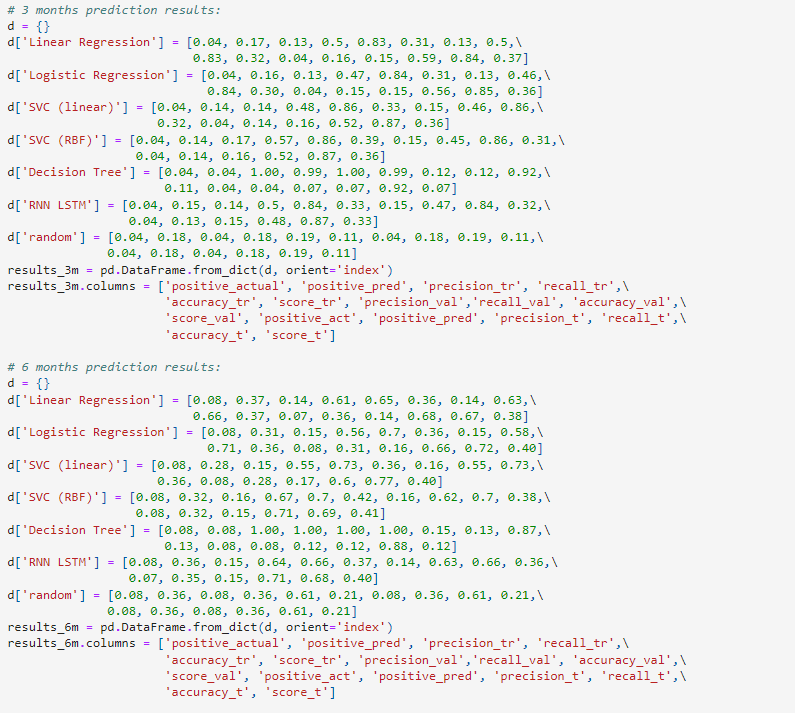
To avoid having any two data sets with a cross-correlation greater than 0.5 in my collection, I proceeded with only data from the S&P 500 (USA), Nikkei, (Japan), HSI (Hong Kong), SSE (Shanghai), BSESN (India), SMI (Switzerland) and BVSP (Brazil).

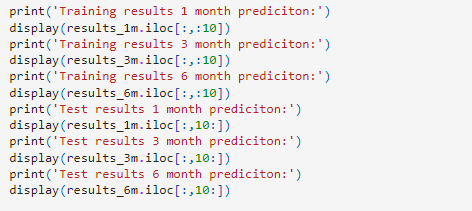


I considered two different methodologies to identify crashes. The first one follows a suggestion by Emilie Jacobsson [2] who defines crashes in each market as drawdowns in the 99.5% quantile . With this methodology I found drawdown thresholds that classify a crash ranging from around 10% for less volatile markets like the S&P 500 to more than 20% for volatile markets such as the Brazilian one. The second methodology follows the suggestion from Johansen and Sornette [3] who identify crashes as outliers, that is drawdowns that lie far from the fitted Weibul distribution when the logarithm of the rank of drawdowns in a data set is plotted vs the drawdown magnitude.

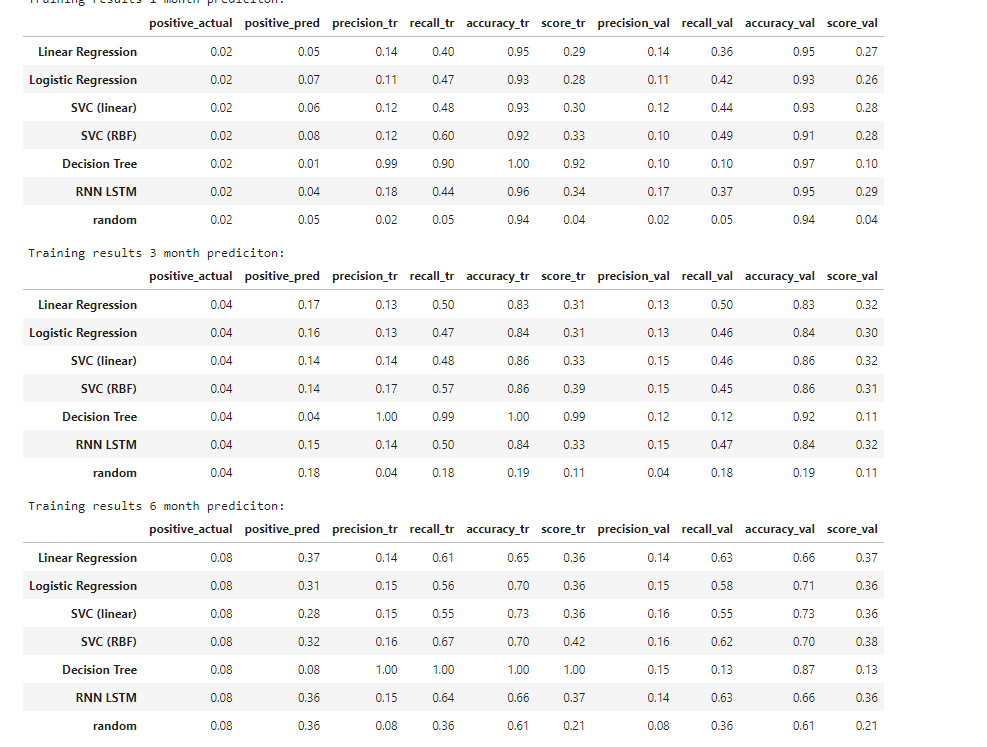
**Code**

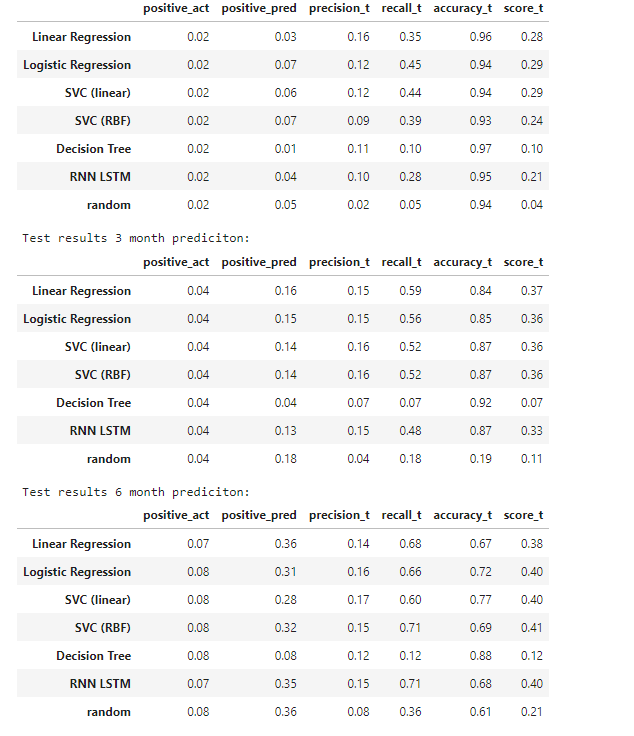
****

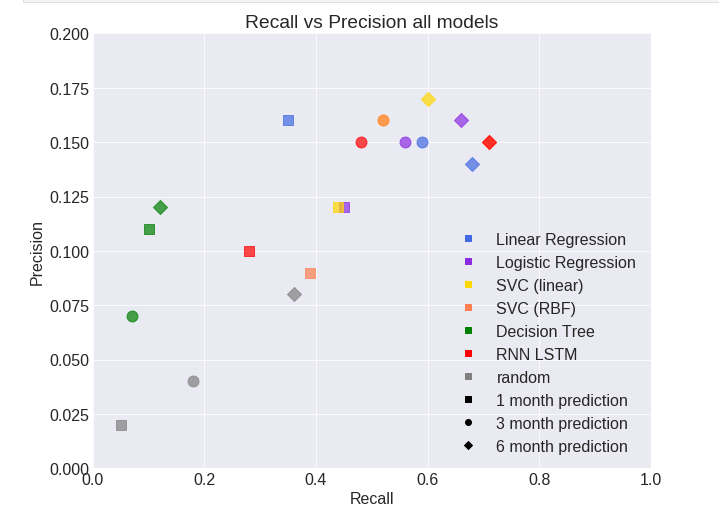
****

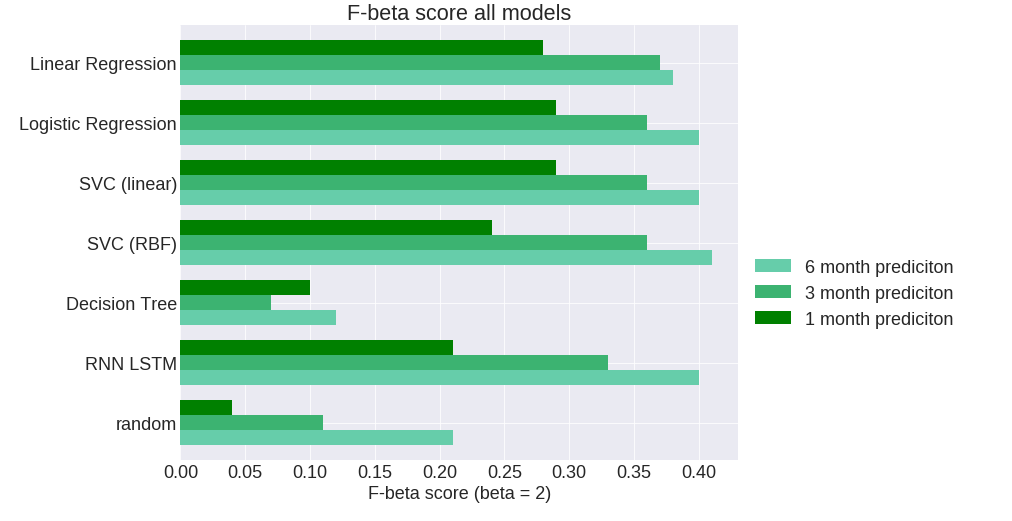
****

**Result**

****

****

****

****

**Software/Hardware Requirements**

Processor: Minimum 1 GHz; Recommended 2GHz or more.

Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

Hard Drive: Minimum 32 GB; Recommended 64 GB or more.

Memory (RAM): Minimum 1 GB; Recommended 4 GB or above.

**Conclusion**

First for each individual model in the library I need to well tune parameters and try to extract the best result and the best version from each.

Second I need to increase the variations of my models library by implementing either more of set of parameters for some models that potentially can perform well.

Also in term of library variation, I can apply a feature sampling by replicate each model with randomly sampled set of features, for example by sampling 60% of features for 5 times, that’s could provide a 40 model instead of 8 they all contribute for getting a better result.