PROJECT REPORT

Stock Market Volatility Prediction

Sai Sahana Bhargavi Byrapu - G01330358
Maitreyi Shekhar Pitale - G01326362

Video Link:
https://youtu.be/2TZQGbcyPDc

Introduction :

Volatility is one of the most prominent terms used on any trading floor – and for good reason. In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated with periods of market turbulence and to large price swings, while low volatility describes more calm and quiet markets. For trading firms, accurately predicting volatility is essential for the trading of options, whose price is directly related to the volatility of the underlying product. Many market makers are dedicated to continuously improving financial markets creating better access and prices for options, exchange-traded funds(ETFs), cash equities, bonds and foreign currencies on numerous exchanges around the world.

Types of Volatility:

1. Realized Volatility: This measures the fluctuations in the security's prices in the past. It is used to predict the future movements of prices based on previous trends. However, it does not provide insights regarding the future trend or direction of the security's price.
2. Implied Volatility: This refers to the volatility of the underlying asset, which used to price options contracts will return the theoretical value of an option equal to the option's current market price. Implied volatility is a key parameter in option pricing. It provides a forward-looking aspect on possible future price fluctuations.

Main Focus :

We are focusing on calculating **Realized volatility** which is the representation of price movements, market's volatility and the trading risks.This is important because it helps to quantify the inherent price risk arising out of volume fluctuations and external factors of a stock based on its historical performance.

Dataset and Analysis :

https://www.kaggle.com/competitions/optiver-realized-volatility-prediction/data

The dataset is extracted from Kaggle competitions - Optiver realized volatility prediction. This collection comprises stock market data that is relevant to the practical execution of financial market trades. It covers, in particular, order book snapshots and performed deals. It gives a unique fine-grained view at the microstructure of current financial markets with a one-second resolution. To do the experiments, we have 429k records in the data-set and this is split into (70%) train and (30%) test data that can be used to construct features to predict target values. The main files in use are book.parquet and trade.parquet. The ground truth values for training are mentioned in the data-set.The train dataset consists of time_id, stock_id and target which is the realized volatility computed over the 10 minute window following the feature data under the same stock/time_id. There is no overlap between feature and target data.

Dataset Size: ~2 GB Type: parquet,csv

Book.parquet is a parquet file partitioned by stock id. Order book data on the most competitive purchase and sell orders put into the market is provided.

1.stock id - The stock's identifier. Not every stock ID is present in every time bucket. When this column is loaded, Parquet converts it to a categorical data type.
2. time id - the time bucket's ID code. Time IDs are not always consecutive, but they are constant across all stocks.
3.seconds_in_bucket -The number of seconds since the bucket's inception, always starting at 0.
4.bid price[1/2] - The most/second most competitive purchase level's normalized price.
5.ask price[1/2] - Price normalization of the most/second most competitive sell level.
6. bid size[1/2] - The number of shares available at the most/second most competitive buy level.
7. ask_size[1/2] - The number of shares on the most/second most competitive sell level.

Trade.parquet file is a parquet file that is partitioned by stock id. Contains information about deals that were really completed. Because there are more passive buy/sell intention updates (book updates) in the market than actual trades, this file may be more sparse than the order book.

1.stock id - the stock's identifier.
2.time id - the time bucket's ID code
3.seconds_in bucket-As mentioned above,since trade and book data are obtained from same time window and trade data is more scarce in general,this field does not always begin at 0.
4.price - The average price of deals completed in one second. The prices were standardized, and the average was weighted by the number of shares exchanged in each transaction.
5.size - The sum number of shares traded.
6.order_count - The number of unique trade orders taking place.

Data Pre-processing:
Book-preprocessor and trade-preprocessor functions process the both train and test data after reading from respective book and trade parquet files for corresponding stock-ids where realized volatility is calculated using weighted average prices and log returns and and the statistics for each group for different windows(seconds-in-bucket -500,400,300,200,100) and the features are merged into a dataframe for the set of records of each stock-id and further all of the stock-id data-frames are concatenated into single dataframe.

1. Weighted average price: The weighted average price is abbreviated as WAP. It is used to calculate instantaneous stock valuation and calculate realized volatility as our target. The formula for the WAP can be given as follows:

$$WAP = \frac{BidPrice_1 * AskSize_1 + AskPrice_1 * BidSize_1}{BidSize_1 + AskSize_1}$$

2. Log return: For mathematical modeling purposes,Stock returns are computed in the form of Log returns where St is the price of the stock at time t,at various time intervals say t1 and t2 It is given by the formula

$$r_{t_1,t_2} = \log\left(\frac{S_{t_2}}{S_{t_1}}\right)$$

3. Realized volatility: When we trade options, a valuable input to the models is the standard deviation of the stock log returns. The standard deviation will be different for log returns computed over longer or shorter intervals, for this reason it is usually normalized to a 1-year period and the annualized standard deviation is called volatility. log returns are computed over all consecutive book updates and realized volatility is defined as where, sigma is the square root of the sum of squared log returns.

$$\sigma = \sqrt{\sum_t r^2_{t-1,t}}$$

Evaluation metric:
Root Mean Square Error (RMSE): is the square root of Mean Squared error. It measures the standard deviation of residuals.

Mean absolute error(MAE) represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

R2 score The represents the proportion of the variance in the dependent variable which is explained by the regression model. It is a scale-free score i.e. irrespective of the values being small or large, the value of R square will be less than one.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}| \qquad RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \qquad R^2 = 1 - \frac{\Sigma(y_i - \hat{y})^2}{\Sigma(y_i - \bar{y})^2}$$

Where,
$\hat{y}$ – *predicted value of y*
$\bar{y}$ – *mean value of y*

The lower value of MAE, MSE, and RMSE implies higher accuracy of a regression model. However, a higher value of R square is considered desirable.

Approach:

Regression Models: for predicting Continuous output which means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values imported from `pyspark.ml.regression` package.

Cross Validation: K-fold cross validation performs model selection by splitting the dataset into a set of non-overlapping randomly partitioned folds which are used as separate training and test datasets.with k=5 folds, K-fold cross validation will generate 5 (training, test) dataset pairs, each of which uses 4/5 of the data for training and 1/5 for testing. Each fold is used as the test set exactly once. CrossValidator() performs cross validation for the given 'Regression' model using the corresponding parameters given in 'paramGrid' and evaluates using RegressionEvaluator() and further it fits the training data to the model and returns the model tuned with respective parameters..After the above cross-validation procedure, the tuned model is used to predict the target values for the test data and RMSE, MAE, R2 scores are computed.

Linear Regression:
Linear regression can be used to find a relationship between two or more variables of interest and allows us to make predictions once these relationships are found.Simple linear regression

will provide a line of best fit, or the regression line. This regression line can be written as the following formula: $y_i = \alpha + \beta x_i + \varepsilon_i$
Parameters: *regParam=0.0,maxIter: 15, standardization:True loss:squaredError*

Gradient Boosting Tree: This estimator allows for the optimization of any differentiable loss function and constructs an additive model in a forward stage-wise manner. In the Gradient boost below are the steps involved. In Gradient boosting weak learners are decision trees. Create a foundation tree that has just one root node. It represents the initial guess for all samples.Build a tree from errors of the previous tree.Modify the tree depending on learning rate. This learning rate determines the contribution of the tree in the prediction, Repeat step 2 until the maximum number of trees are used or the fit is not improved by adding the new tree to the previous trees in order to forecast the outcome. The final prediction of the model is the combination of all the trees.

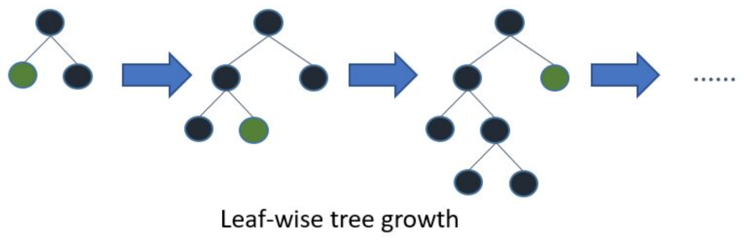
Parameters: *maxDepth:5, maxBins:32 , maxIter: 15, stepSize:0.1, lossType:squared*

Decision Tree: Decision tree is a is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility and its regression model observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.
A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.
Parameters: *maxDepth:5, maxBins: int = 32,minInfoGain: float = 0.0,impurity: 'variance'*

Random Forest Tree:.Random forests include another type of bagging scheme along with original bagging algorithm for trees: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.

Parameters:maxDepth: int = 5,numTrees: int = 20, subsamplingRate: float = 1.0

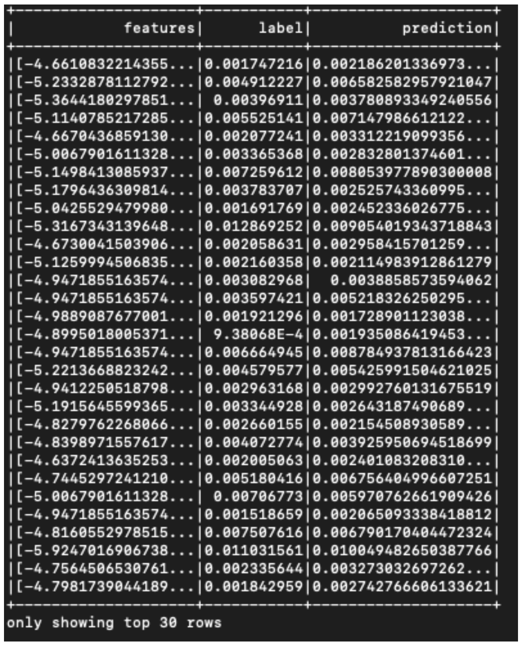
LightGBM: LightGBM is a tree-based gradient boosting algorithm that uses leaf-wise tree growth and not depth-wise growth. Besides, LightGBM does not use the widely-used sorted-based decision tree learning algorithm, which searches the best split point on sorted feature values.Instead, LightGBM implements a highly optimized histogram-based decision tree learning algorithm, which yields great advantages on both efficiency and memory consumption.



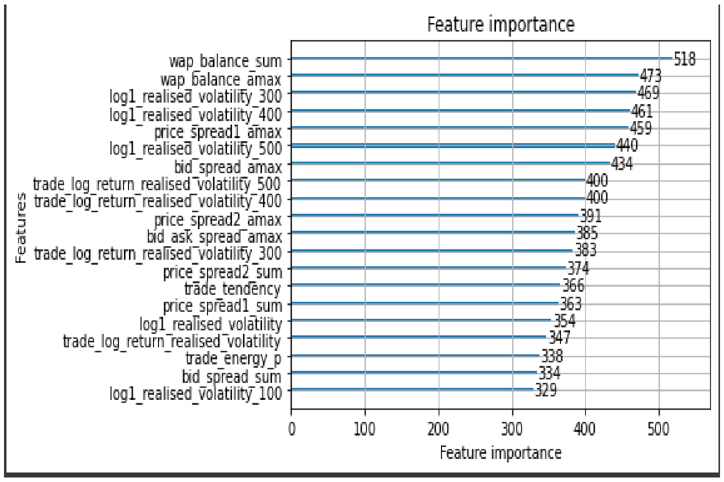
params = { 'learning_rate': 0.25, 'lambda_l1': 2, 'lambda_l2': 7, 'num_leaves': 800, 'num_iterations':5000, 'min_sum_hessian_in_leaf': 20, 'feature_fraction': 0.8, 'feature_fraction_bynode': 0.8, 'bagging_fraction': 0.9, 'bagging_freq': 42, 'min_data_in_leaf': 700, 'max_depth': 4, 'categorical_column':[0], 'seed': seed1, 'feature_fraction_seed': seed1, 'bagging_seed': seed1, 'drop_seed': seed1, 'data_random_seed': seed1, 'objective': 'rmse', 'boosting': 'gbdt', 'verbosity': -1, 'n_jobs':-1, }

Results:

Regression Models	RMSE	R2-score	MAE
Linear Regression	0.0012098	0.8316	0.0007125
Gradient Boosting Tree	0.0012343	0.8247	0.0007236
Decision Tree	0.0013495	0.7905	0.0007830
Random Forest Tree	0.0012968	0.8065	0.0007526
LightGBM	0.0013025	0.8123	0.0007612



```
+--------------------+-----------+--------------------+
|            features|      label|          prediction|
+--------------------+-----------+--------------------+
|[-4.6610832214355...|0.001747216|0.002186201336973...|
|[-5.2332878112792...|0.004912227|0.006582582957921047|
|[-5.3644180297851...| 0.00396911|0.003780893349240556|
|[-5.1140785217285...|0.005525141|0.007147986612122...|
|[-4.6670436859130...|0.002077241|0.003312219099356...|
|[-5.0067901611328...|0.003365368|0.002832801374601...|
|[-5.1498413085937...|0.007259612|0.008053977890300008|
|[-5.1796436309814...|0.003783707|0.002525743360995...|
|[-5.0425529479980...|0.001691769|0.002452336026775...|
|[-5.3167343139648...|0.012869252|0.009054019343718843|
|[-4.6730041503906...|0.002058631|0.002958415701259...|
|[-5.1259994506835...|0.002160358|0.002114983912861279|
|[-4.9471855163574...|0.003082968|  0.0038858573594062|
|[-4.9471855163574...|0.003597421|0.005218326250295...|
|[-4.9889087677001...|0.001921296|0.001728901123038...|
|[-4.8995018005371...| 9.38068E-4|0.001935086419453...|
|[-4.9471855163574...|0.006664945|0.008784937813166423|
|[-5.2213668823242...|0.004579577|0.005425991504621025|
|[-4.9412250518798...|0.002963168|0.002992760131675519|
|[-5.1915645599365...|0.003344928|0.002643187490689...|
|[-4.8279762268066...|0.002660155|0.002154508930589...|
|[-4.8398971557617...|0.004072774|0.003925950694518699|
|[-4.6372413635253...|0.002005063|0.002401083208310...|
|[-4.7445297241210...|0.005180416|0.006756404996607251|
|[-5.0067901611328...| 0.00706773|0.005970762661909426|
|[-4.9471855163574...|0.001518659|0.002065093338418812|
|[-4.8160552978515...|0.007507616|0.006790170404472324|
|[-5.9247016906738...|0.011031561|0.010049482650387766|
|[-4.7564506530761...|0.002335644|0.003273032697262...|
|[-4.7981739044189...|0.001842959|0.002742766606133621|
+--------------------+-----------+--------------------+
only showing top 30 rows
```



Conclusion & Future work:
Linear Regression model has better RMSE, R2 score, MAE values surprisingly, when compared to the Boosting algorithms GBT, Decision Tree, Random Forest and LightGBM whereas there is slight performance issue with Decision Tree regressor model.

Data-preprocessing has been a challenge as the stock data is huge and understanding and implementing mathematical operations like calculating weighted average price, stock returns in form of log, median, means,interquartile range and varied statistical measures in `pyspark` over the set of records in each of the 126 stock-ids in book parquet and trade parquet files and combine them with train and test dataframes.
Further,we got insight into the stock market domain and how the trade orders are practically executed in the financial market.In future, we are looking forward to extending the scope of the project by implementing other volatility type, implied volatility prediction and also check the performance of the models on different stock market datasets.