**PROJECT COMPLETION REPORT– VLG OPEN PROJECT**

# A close up of a white background Description automatically generated

**OBJECTIVE:**

Develop a ML model to predict US stock closing price for NASDAQ listed stocks using the order book data available on Kaggle.

**PROCESSES IN SEQUENTIAL ORDER:**

(Use each list member as a link)

1. [List of libraries](#_1._List_of)
2. [Data Pre-Processing](#_Data_Pre-Processing)
   1. [Data Cleaning](#_Data_Cleaning:)
   2. [Feature Engineering](#_Feature_Engineering:)
   3. [Splitting Data](#_Splitting_Data:)
   4. [Data Transformation](#_Data_Transformation:)
3. [Fitting Data into different ML models](#_Fitting_Data_into)
4. [Checking accuracy using Cross Validation Method](#_Checking_accuracy_using)
5. [Selection of Model](#_Selection_of_Model)
6. Prediction of target value for test set

## List of libraries



## 

















# Data Pre-Processing

## Data Cleaning:

Initially the data looks like this:

A white background with black text

Description automatically generated

1. As the far\_price and near\_price are sparse columns and there is no possible way to replace NAN values we drop them.



1. Then we drop the NAN values of other columns. ( Note : We can also replace these values with the mean using an imputer but these values are very less as compared to the data-set that they won’t make much of a difference.



1. The resulting data looks like this:

A screen shot of a computer code

Description automatically generated

## Feature Engineering:

We create a new feature which is the difference between the bidding column and the ask column and dividing by its sum. The intuition behind this is that the price of a stock changes based on the supply and demand of that stock. The demand is proportional to the price. We divide by its sum just to scale the values.



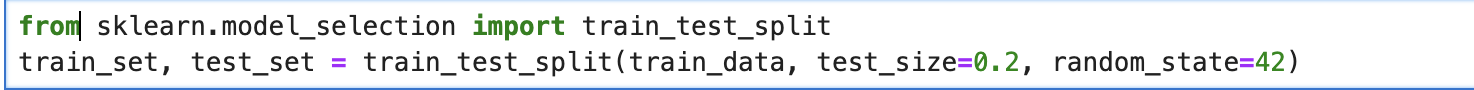
The Increase in the correlation between the target and demand size to 11.45% compared to bid size and ask size confirms our intuition.

A table of numbers with a flag

Description automatically generated

## Splitting Data:

The Data is split into two parts of 20% and 80% and the random\_state is set to 42 to generate the same data split even if the program is run again.



## Data Transformation:

Step 1: Scaling of data

The numerical data is scaled down using StandardScaler because ML algorithms work better with data that lies in a small range.

Step 2: The categorical data field imbalance\_buy\_sell\_flag is passed through a one hot encoder. This transforms this column into different columns for each unique value of the data.

Step 3: Both the steps are combined through a column transformer. Other columns such as row\_id, stock\_id etc are dropped because they are not relevant features for prediction.

A close-up of a white background

Description automatically generatedA close-up of a computer screen

Description automatically generated

Then the training set is transformed by passing the data through this pipeline.

# 

# Fitting Data into different ML models

We need to test different models and choose whichever suits us best. The models we would be testing are:

1. Linear Regressor Model
2. XGBoost Regressor Model
3. CatBoost Regressor Model

A close-up of a computer code

Description automatically generated

# 

# Checking accuracy using Cross Validation Method

Cross validation method splits the data into a train dataset and a validation dataset and after training the train dataset with the selected model it predicts the results for the validation set and check the error. It repeats the process n (parameter for algorithm) times where n is the number of folds.

1. Linear Regressor Model

A white background with black text

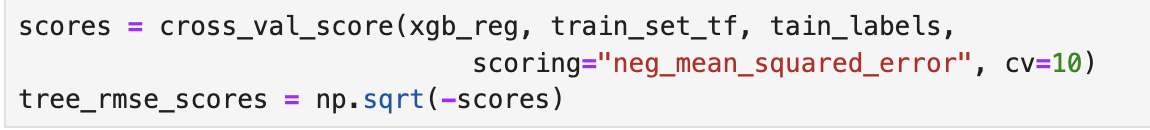
Description automatically generated

The result gives the root mean squared error for each process. Here n (cv) is set to 10.

A screenshot of a computer

Description automatically generated

1. XGBoost Regressor Model

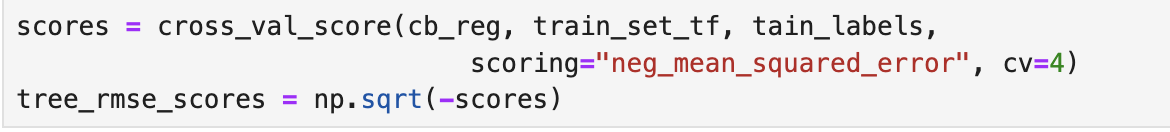


The result gives the root mean squared error for each process. Here n (cv) is set to 10.

A close-up of numbers

Description automatically generated

1. CatBoost Regressor Model



Here n is set to be 4 as the catboost model uses a lot of RAM which my device was not able to handle. Still n=4 would work to compare results.

A close up of numbers

Description automatically generated

# Selection of Model

We can choose the model based on the accuracy and how fast we can get the results. Although linear regression is fast but XGBoost and CatBoost are more accurate. The RMSE for XGBoost and CatBoost is similar but CatBoost is much faster. Even though the error is similar for both algorithms CatBoost is much faster. But if we want to use a model which we completely understand Linear Regression would be the best choice as CatBoost is much more complex to understand. The benefit with an algorithm we understand completely is that we know the use of parameters to tweak to optimize our results.

# Prediction of target value for test set

We apply similar transformation to the test set data and pass through the trained model to predict output.

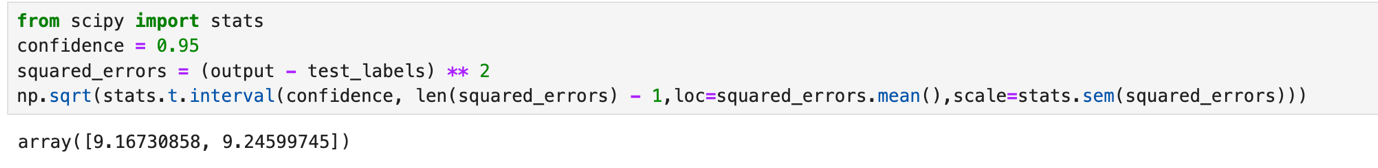
A screenshot of a computer

Description automatically generated

OUTPUT DRIVE LINK:

<https://drive.google.com/file/d/1dtjwXzmWZMF1j9oDViImOx7efh13ls5l/view?usp=sharing>

To have an idea of how precise this estimate is, we can compute a 95% confidence intervalfor the generalization error.



Finally save the output:

