CAPSTONE PROJECT [Aug 2023]

PROJECT DESCRIPTION!!

In today's dynamic financial landscape, accurately predicting stock returns has become a challenging yet crucial task for investors and financial analysts. This project aims to harness the power of machine learning to forecast stock returns over a multi-day horizon using a novel approach that incorporates a diverse set of temporal data.

> The project's primary objective is to develop a predictive model that can anticipate the returns of a given stock over a 5-day window. The predictive process leverages a unique dataset structure: the model is provided with historical stock returns for the first day of the trading week, followed by the return of the subsequent day. On the third day ("D day"), the dataset includes minute-wise stock returns for the initial 120 minutes of trading. Based on this information, the model is expected to forecast the stock's return for the next 60 minutes and the returns for the subsequent two trading days

DATASET DESCRIPTION

We provide 5-day windows of time, days D-2, D-1, D, D+1, and D+2. You are given returns in days D-2, D-1, and part of day D, and you are asked to predict the returns in the rest of day D, and in days D+1 and D+2.

During day D, there is intraday return data, which are the returns at different points in the day. We provide 180 minutes of data, from t=1 to t=180. In the training set you are given the full 180 minutes, in the test set just the first 120 minutes are provided.

For each 5-day window, we also provide 25 features, Feature_1 to Feature_25. These may or may not be useful in your prediction.

Each row in the dataset is an arbitrary stock at an arbitrary 5-day time window.

How these returns are calculated is defined by Winton, and will not to be revealed to you in this competition. The data set is designed to be representative of real data and so should bring about several challenges.

File descriptions

- **csv** the training set, including the columns of:
  - Feature_1 - Feature_25
  - Ret_MinusTwo, Ret_MinusOne
  - Ret_2 - Ret_120
  - Ret_121 - Ret_180: **target variables**
  - Ret_PlusOne, Ret_PlusTwo: **target variables**
  - Weight_Intraday, Weight_Daily

Data fields

- **Feature_1 to Feature_25:**different features relevant to prediction
- **Ret_MinusTwo:** this is the return from the close of trading on day D-2 to the close of trading on day D-1 (i.e. 1 day)
- **Ret_MinusOne:**this is the return from the close of trading on day D-1 to the point at which the intraday returns start on day D (approximately 1/2 day)
- **Ret_2 to Ret_120**: these are returns over approximately one minute on day D. Ret_2 is the return between t=1 and t=2.
- **Ret_121 to Ret_180**: intraday returns over approximately one minute on day D. **These are the target variables you need to predict as {id}_{1-60}.**
- **Ret_PlusOne**: this is the return from the time Ret_180 is measured on day D to the close of trading on day D+1.

(approximately 1day). **This is a target variable you need to predict as {id}_61.**

- **Ret_PlusTwo**: this is the return from the close of trading on day D+1 to the close of trading on day D+2 (i.e. 1 day) **This is a target variable you need to predict as {id}_62.**
- **Weight_Intraday**: weight used to evaluate intraday return predictions Ret 121 to 180
- **Weight_Daily**: weight used to evaluate daily return predictions (Ret_PlusOne and Ret_PlusTwo).

LIBRARIES USED FOR EXECUTION

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.multioutput import MultiOutputRegressor
from sklearn.base import RegressorMixin
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from xgboost import XGBRegressor
%matplotlib inline
```

EXPLORATORY DATA ANALYSIS

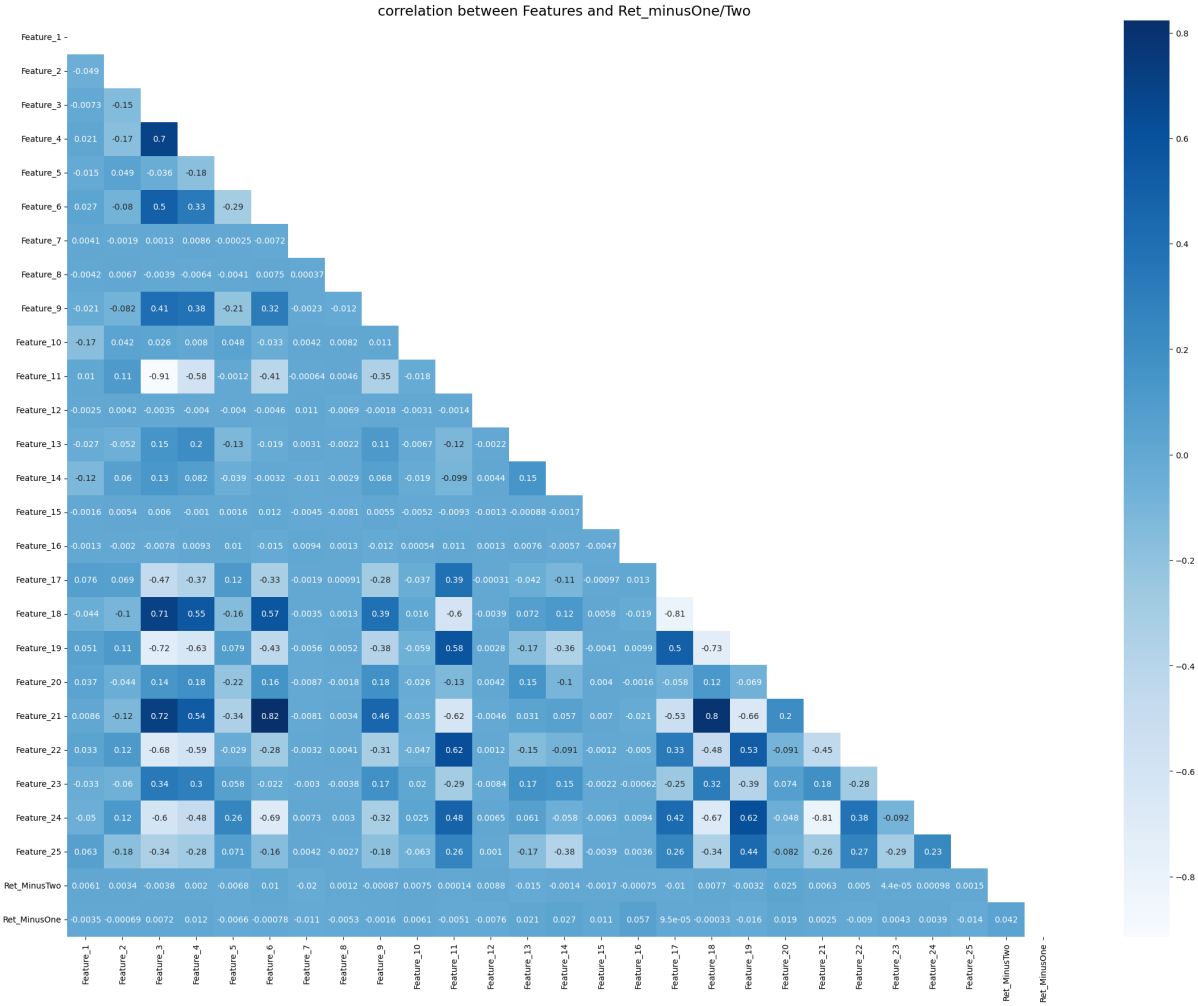
- The data was of moderate size with 211 features and 40,000 data entries.
- each row denoted an arrbitary stock, and column 1 to 12 was titled as Feature_1 to Feature_25, these featurse was so factors which was determines stock prices and returns, however these featurs were not revealed hence it was masked with a number in the surffix ranging from 1 to 25,

- these feature when correlated with each other didnt displayed much of correation, but some feature showed high correlation with each other below is the list of most correlated feature, along with an all feautre correlation map!

```
corr_df = data.iloc[:,1:28]
a = corr_df.corr()

mask = np.zeros_like(a)
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True

plt.figure(figsize = [27,20])
sns.heatmap(a, mask = mask, annot  = True, cmap = "Blues")
plt.title("correlation between Features and Ret_minusOne/Two", fontsize = 17)
plt.show()
```

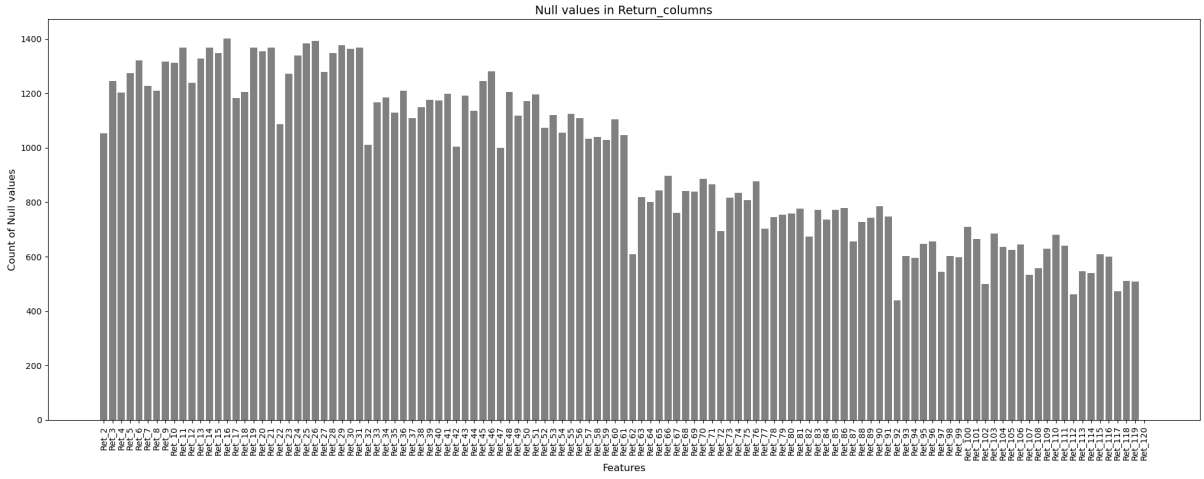


the visual representation of the correlation map!

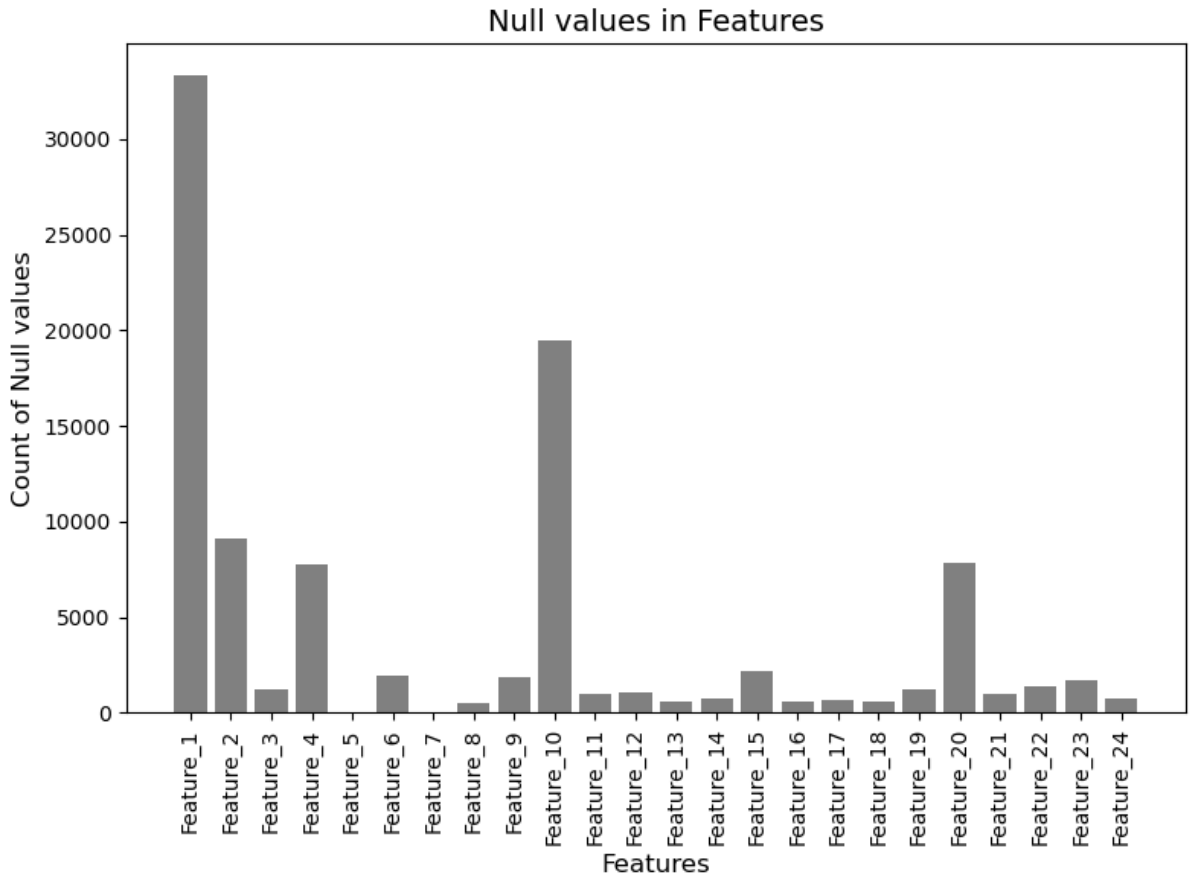
Here is the List of feature who have correlation index of more than 0.6

(Feature_6 ~ Feature_21)
(Feature_18 ~ Feature_21)

- Dealing with Null Values: when the data was seen, it had numerous null values, throughout the row and columns, some columns were completely updated with proper values and some not!
- identifying these columns with null values was done by the help of a python library **“missingno”**
- later on top features with null values was identified too and also other features was identified!
- charts below are self explanatory, here are the glimps



features vs sum of null values

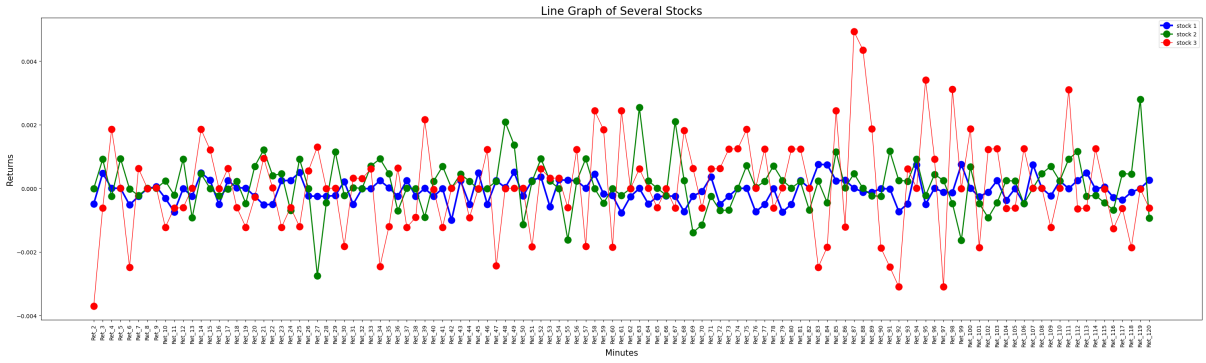


- after taking a look at null values it was a decision making time, of what to be done with it. i decided to fill those null values with mean of the data, rather than dropping it,

```
mean = data.mean()
data = data.fillna(mean)
```

the above code helped me in filling all the null values with its respective column mean!, and after doing that my data was cleaned and was ready to be used for further analysis!

- curious of how my data looked after cleaning, i plotted some rows against their values in order to see how it looks



these points which are plotted represent minute wise returns of an stock, starting from 2nd miunte to 120th minute

PRE PROCESSING

- For any model training the must requirment is **“target Variable”** in my case the target was further classified into two classes a. the intraday target, b. the interday target.
- for Intraday target i isolated feature_1 to Feature_25 and then isolated the last 10 mins from intraday return columns and formed a blend of below mentioned independent variables

```
'Feature_1', 'Feature_2', 'Feature_3', 'Feature_4', 'Feature_5',
'Feature_6', 'Feature_7', 'Feature_8', 'Feature_9', 'Feature_10',
'Feature_11', 'Feature_12', 'Feature_13', 'Feature_14', 'Feature_15',
'Feature_16', 'Feature_17', 'Feature_18', 'Feature_19', 'Feature_20',
'Feature_21', 'Feature_22', 'Feature_23', 'Feature_24', 'Feature_25',
'Ret_111', 'Ret_112', 'Ret_113', 'Ret_114', 'Ret_115', 'Ret_116',
'Ret_117', 'Ret_118', 'Ret_119', 'Ret_120'
```

then defined my intraday target as **“target_for_intraday”**, the dependent variables were as mention below

```
'ret_121','ret_122','ret_123','ret_124','ret_125',.....................'ret_180'
```

- after dealing with intraday target, the same procedure was applied for interday, however the targets for interday was diffrent and so the independent variable

- the extration of interday independnt variable started from forming a dataframe consisting of all the feature and the colums Ret_MinusOne, Ret_MinusTwo.list below gives a clearer look at independent variables for interday

```
['Feature_1', 'Feature_2', 'Feature_3', 'Feature_4', 'Feature_5',
'Feature_6', 'Feature_7', 'Feature_8', 'Feature_9', 'Feature_10',
'Feature_11', 'Feature_12', 'Feature_13', 'Feature_14', 'Feature_15',
'Feature_16', 'Feature_17', 'Feature_18', 'Feature_19', 'Feature_20',
'Feature_21', 'Feature_22', 'Feature_23', 'Feature_24', 'Feature_25',
'Ret_MinusTwo', 'Ret_MinusOne']
```

- the target was in the variable names as **“target_for_interday”** and it contained two columns namely **Ret_PlusOne** and **Ret_plusTwo**
- after slicing the main dataframe into convnient dataframes, it was expected to split the data into Train and Test , since data was of 40,000 row i used Train_test_split method to split data in proporton of 80-20%, this split ratio is considerd to be the most optimum. hence the 80-20 split

MODEL EXPLAINATION

for the sake of understanding and to get a quick grip of structure, while training each model, i divided training into two parts as follows

1. intraday
2. interday

so every machine learing will have two outcome intraday and interday, this will make understanding metric scores and other component better

Here below are the explaninations of the ML model

Linear Regression:

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes that this relationship can be approximated by a linear equation, where the independent variables are multiplied by certain coefficients and summed up to predict the dependent variable.

The goal of linear regression is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the difference between the predicted values and the actual values of the dependent variable. This is typically achieved by minimizing the sum of squared differences between the observed and predicted values.

In essence, linear regression seeks to quantify the linear association between variables and enables us to make predictions or understand the impact of changes in the independent variables on the dependent variable. It's widely used in various fields for tasks like forecasting, understanding correlations, and identifying trends in data.

A.Intraday:

- After splitting the data into two parts—one for training and another for testing—I used a technique called linear regression. Think of it as a way to find patterns in the data. I treated one set of numbers as the main influencers (let's call them 'X'), and another set as the results we wanted to predict (we'll call them 'y').
- By having the computer learn from the training data, it became pretty smart at guessing the results. When I tested it with new data, it gave its best guesses for the outcomes.
- To tidy things up and make them easier to understand, I came up with a clever trick. I created a special rule where I added "pred" to the end of every column name in the results. This way, it was crystal clear which columns showed the computer's predictions.
- This whole process was like solving a puzzle: dividing the data, teaching the computer, and then making its

answers neat and tidy

B.Interday

- Next, I took a step further and used linear regression for something called "interday" analysis. This just means I was looking at how things change from one day to the next. To do this, I used a special table I made earlier, where each column had important information for this study (let's call it 'interday_columns'). The main thing I wanted to figure out was how one particular number (called 'target_for_interday') changes.
- After the computer learned from the data, it started making predictions. The predictions it gave me were for two timeframes: the day right after (Ret_plusone) and the one after that (Ret_PlusTwo). It's like predicting what might happen one day and two days ahead based on what we know.
- So, in short, I used linear regression to understand how things change between days. The computer learned from the data and told me what it thought would happen in the next two days.

The Tabular comparison of Metrics

Model = Linear Regression

	LINEAR REGRESSION	
	INTRADAY	INTERDAY
MSE	1.3734484222998877e-06	0.000596
MAE	0.0006523970622858782	0.015628
RMSE	0.00117194215825692	0.024414
R^2	-0.0020036250409579386	-0.001687

Extreme Gradient Boost(XGBoost)

Extreme Gradient Boosting (XGBoost) regressor is an advanced machine learning technique that belongs to the ensemble learning family. It's specifically designed for regression

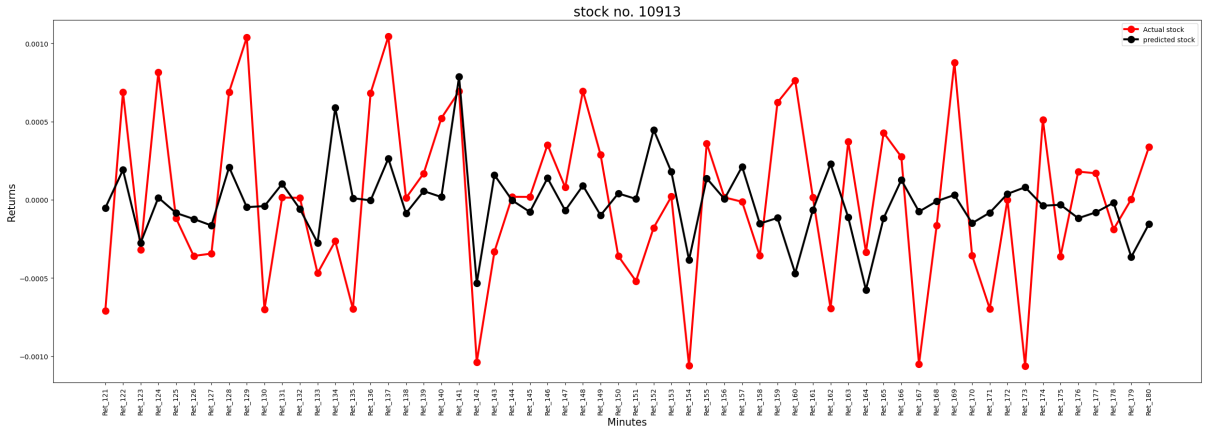
tasks, where the goal is to predict a continuous numerical output based on input features.

At its core, XGBoost is a collection of decision trees, which are fundamental units of decision-making in machine learning. However, it takes decision trees a step further by training them sequentially and refining their predictions with each iteration. This process is known as boosting

Overall, the Extreme Gradient Boosting regressor, or XGBoost regressor, stands out for its remarkable accuracy, efficient computation, and robustness against common challenges in machine learning. It's a powerful tool that's widely used in predictive modeling, ranking, and recommendation systems, among other applications

A.Intraday

- the initial approch started with setting up Dependent and Target Variables
- the dependent variable as we mentioned in our pre-processing, was the features along with 10 mins of returns, considering that as X and target_for_intraday as y.
- import XGBRegressor Class for our help, passing some paramaters and giving X and y to it, the model was trained
- plotting the trained outcome with the actual one was really a reality tester. Outcomes were bit favoruable and here is the plotting



- the Metric score was really mind blowing, which is mention in tabular form at the end of the explaination

B.Interday

- same process was followed with predicting interday, just the independent and target variable was changed

The Tabular comparison of Metrics

Model = XG Boost Regressor

	XG BOOSTING	
	INTRADAY	INTERDAY
MSE	1.39082866108186e-06	0.0005283921891333426
MAE	0.0006339275303183812	0.014628009713045289
RMSE	0.0011793339904716815	0.02298678292265672
R^2	-0.019573739312693277	0.1114314558981484

Last Observation Carried Forward(LOCF)

Certainly, here's a concise explanation of the LOCF (Last Observation Carried Forward) model:

The LOCF model, an abbreviation for Last Observation Carried Forward, is a straightforward data imputation technique. It's employed to fill missing values by using the most recent available observation. This method assumes that the most recent data point is a reasonable estimate of the missing value. While simple and quick to implement, LOCF might not capture underlying trends or variations in the data accurately and can potentially introduce bias if the assumption of consistency doesn't hold true

	LOCF	
	INTRADAY	INTERDAY
MSE	2.599705560723548e-06	0.0014552718244046979
MAE	0.0009217528408451569	0.023633166010756976

	LOCF	
RMSE	0.0016123602453309085	0.03814802517044228
R^2	-0.9195864943126764	-1.4457593643780904

CONCLUSIONS

Linear Regression:
Utilizing Linear Regression, we sought to capture the intricate relationships between variables and their impact on stock price predictions. The results indicate a commendably low Mean Squared Error (MSE) of 1.373e-06, suggesting that the model's predictions are closely aligned with the actual values. However, the negative R^2 score of -0.002 implies that the model might not fully capture the variance in the data, indicating areas where further refinement could enhance its predictive power.

XGBoost:
Harnessing the prowess of XGBoost, our endeavor was to unlock the hidden dynamics of stock price movements. The achieved metric scores, including an MSE of 1.391e-06 and an impressive Mean Absolute Error (MAE) of 0.000634, underscore the model's accuracy in prediction. Yet, the negative R^2 of -0.020 signals that while the model performs well in terms of error metrics, it may benefit from capturing additional complexity within the data.

LOCF:
The utilization of LOCF, while practical for imputing missing values, didn't extend to predictive capabilities in the same vein as the other models. The model metrics, encompassing a relatively higher RMSE of 0.001612 and a significantly negative R^2 of -0.920, indicate limitations in capturing

the intricate patterns necessary for accurate stock price forecasts.

In summary, while each model showcased distinct performance traits, the journey towards predictive prowess continues. To elevate our models, addressing the nuances of the R^2 scores and fine-tuning methodologies will be crucial. Amidst these insights, we stride forward, poised to navigate the complexities of financial prediction with a steadfast commitment to precision and innovation.

> 💡 Please find .ipynb file attach as “Capstone_project_soham”

THANK YOU

-Soham chandekar

/