STOCK PRICE PREDICTION USING APPLIED DATA SCIENCE

INTRODUCTION:

The stock market is a dynamic and complex system influenced by a multitude of factors, including economic indicators, news events, investor sentiment, and market trends. Predicting its movements has been a subject of great interest for investors, traders, and researchers alike. Applied Data Science provides a powerful toolkit to analyze historical data, extract meaningful patterns, and build models to make informed predictions about future market behavior by preprocessing,model training and evaluation, and finally visualization of the dataset.

OBJECTIVES:

Project Objectives:

1.Stock Price Demographics Analysis:

-Identify and analyze key characteristics of stocks

2.Analyze Market trends:

-Analysis the market trends of each stock values

3.Optimize portfolio allocations:

-Investigate any correlations between stock trends

Analysis Approach:

1.Data Collection and Preprocessing:

-Acquire historical stock market data, including price, volume, and relevant financial indicators, from reputable sources.

-Clean and preprocess the data, handling missing values, and ensuring consistency.

2.Data Splitting:

- Split the dataset into training and testing sets.Typically 80% for training and 20% for testing

3.Insights Derivation:

- Derive actionable insights from the analyzed data.

4.Model Selection:

-Choose appropriate machine learning or statistical models based on the project's objective.

-Linear Regression.

-Logistic Regression

-Support vector machine.

-XGB Classifier.

5.Training and Evaluation:

-Split the data into training and testing sets.

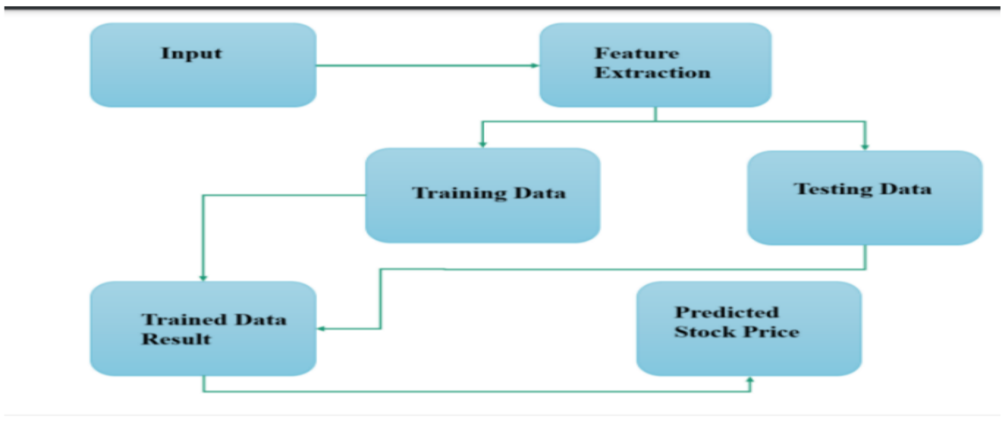
-Train the selected models and evaluate their performance using metrics for overfitting condition by using ROC-AUC curve

6.Model Interpretation:

-Understand how the model arrives at its predictions.

-Utilize techniques like feature importance analysis or SHAPE values for interpretability

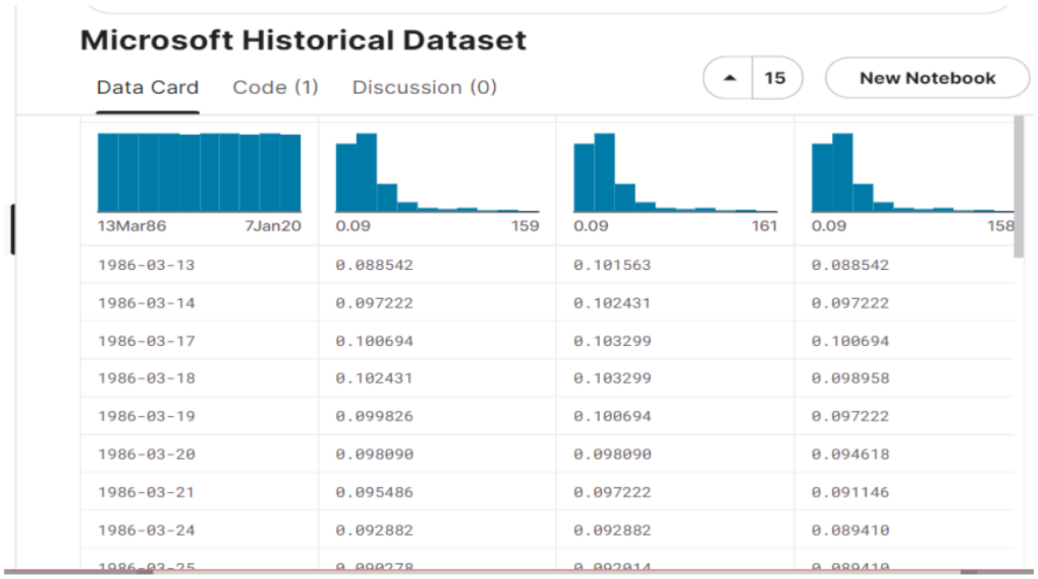
FLOWCHART:



DATA COLLECTION:

Collect data from the resource provided (https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-d ataset)

Here is a snippet of the provided dataset,



13Mar86 – 7Jan20	0.09 – 159	0.09 – 161	0.09 – 158
1986-03-13	0.088542	0.101563	0.088542
1986-03-14	0.097222	0.102431	0.097222
1986-03-17	0.100694	0.103299	0.100694
1986-03-18	0.102431	0.103299	0.098958
1986-03-19	0.099826	0.100694	0.097222
1986-03-20	0.098090	0.098090	0.094618
1986-03-21	0.095486	0.097222	0.091146
1986-03-24	0.092882	0.092882	0.089410
1986-03-25	0.090278	0.092014	0.089410

FEATURE SELECTION:

Utilize machine learning models such as decision trees or random forests to assess feature importance. Features that contribute the most to model accuracy are considered important.

CORRELATION ANALYSIS

Calculate correlations between each feature and the target variable (stock prices or returns). Features with high correlation are likely to be important. However, be cautious of multicollinearity, where two or more features are highly correlated with each other.

CODE:

The goals of this project have been outlined, a data analysis plan has been developed, useful visualization types have been identified, and Python and data visualization libraries are being used for the analysis in “Jupyter Notebook”.

VISUALIZATION:

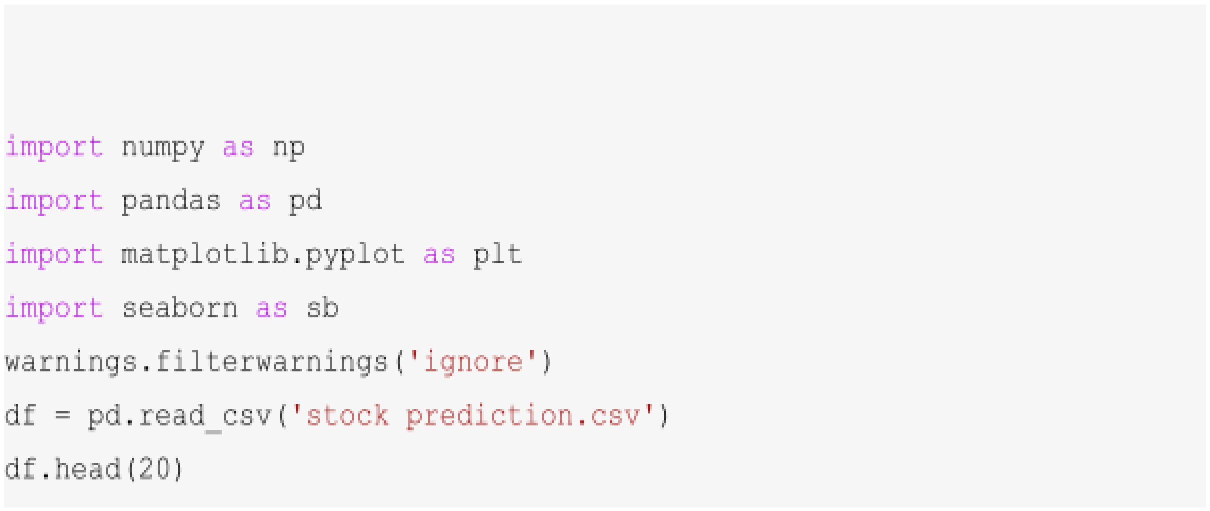
This stage focuses on prepping the provided CSV file in order to prepare it for other operations including analysis, exploratory data analysis, and dataset visualization. Using the "ANALYZE Tool" to visualize the data is part of phase 3 of the study "STOCK PRICE PREDICTION". This document includes a subplot, bar chart, line chart, and boxplot among other charts

Dataset

https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

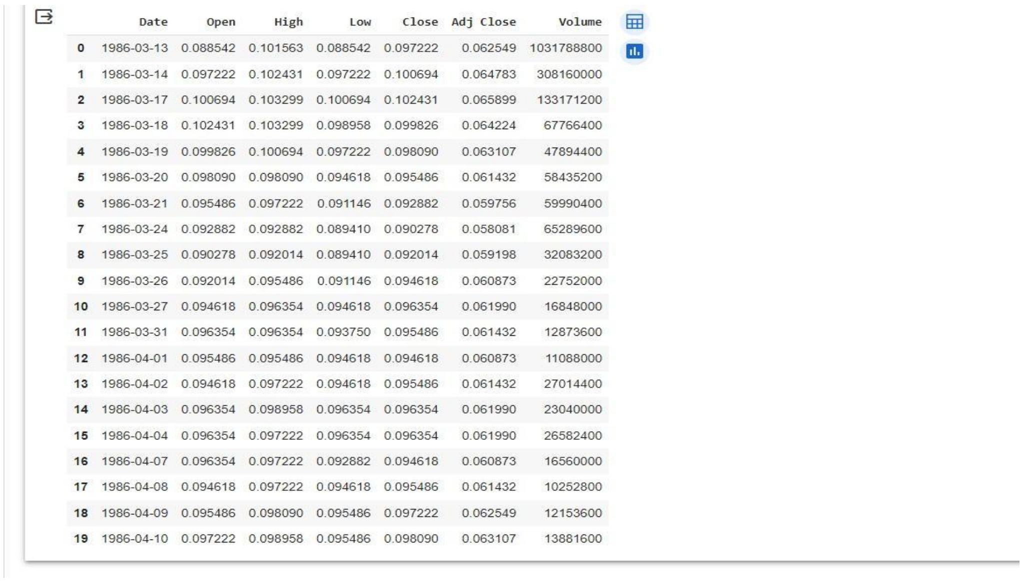
Load the dataset

Load the dataset a csv file from kaggle and using Pandas package load them



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
warnings.filterwarnings('ignore')
df = pd.read_csv('stock prediction.csv')
df.head(20)
```

OUTPUT



	Date	Open	High	Low	Close	Adj Close	Volume
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.063107	47894400
5	1986-03-20	0.098090	0.098090	0.094618	0.095486	0.061432	58435200
6	1986-03-21	0.095486	0.097222	0.091146	0.092882	0.059756	59990400
7	1986-03-24	0.092882	0.092882	0.089410	0.090278	0.058081	65289600
8	1986-03-25	0.090278	0.092014	0.089410	0.092014	0.059198	32083200
9	1986-03-26	0.092014	0.095486	0.091146	0.094618	0.060873	22752000
10	1986-03-27	0.094618	0.096354	0.094618	0.096354	0.061990	16848000
11	1986-03-31	0.096354	0.096354	0.093750	0.095486	0.061432	12873600
12	1986-04-01	0.095486	0.095486	0.094618	0.094618	0.060873	11088000
13	1986-04-02	0.094618	0.097222	0.094618	0.095486	0.061432	27014400
14	1986-04-03	0.096354	0.098958	0.096354	0.096354	0.061990	23040000
15	1986-04-04	0.096354	0.097222	0.096354	0.096354	0.061990	26582400
16	1986-04-07	0.096354	0.097222	0.092882	0.094618	0.060873	16560000
17	1986-04-08	0.094618	0.097222	0.094618	0.095486	0.061432	10252800
18	1986-04-09	0.095486	0.098090	0.095486	0.097222	0.062549	12153600
19	1986-04-10	0.097222	0.098958	0.095486	0.098090	0.063107	13881600

DATA PREPROCESSING:

Check the data reductance and null or missing value in the given dataset and preprocess them and check the null value and remove them if any.

```
df.isnull().sum()
```

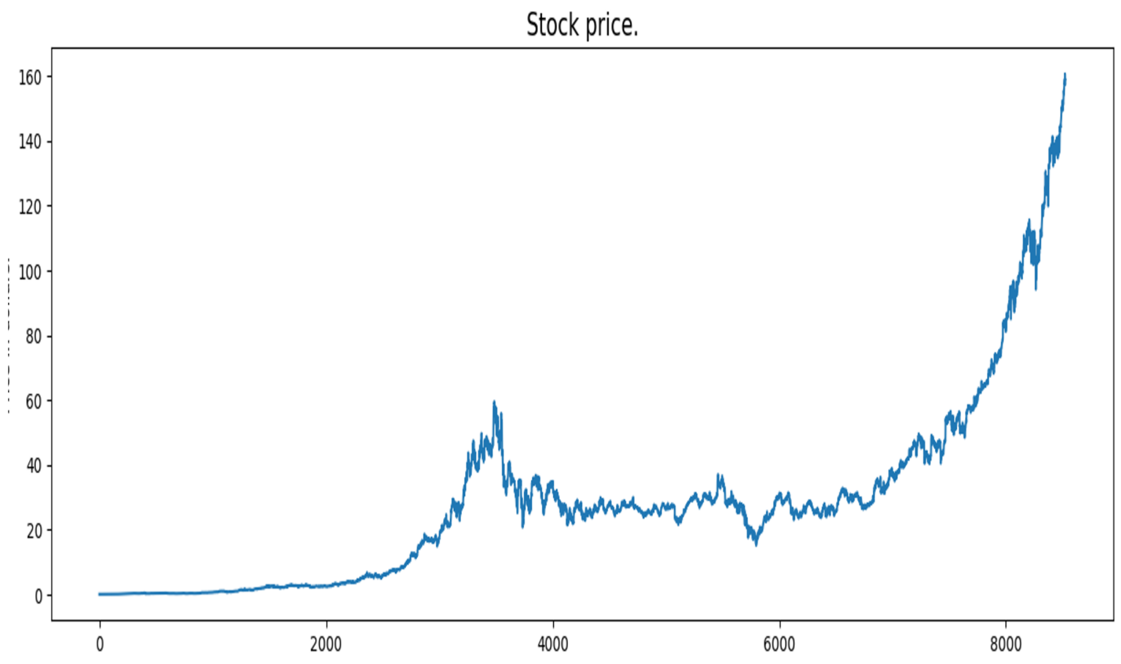
```
OUTPUT
Date        0
Open         0
High        0
Low         0
Close        0
Adj Close     0
Volume       0
dtype: int64
```

ANALYSIS

1. **PLOT FUNCTION -** For plotting the stock price in dollar which are close in the given dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

df = pd.read_csv('stock prediction.csv')
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Stock price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
```

OUTPUT:



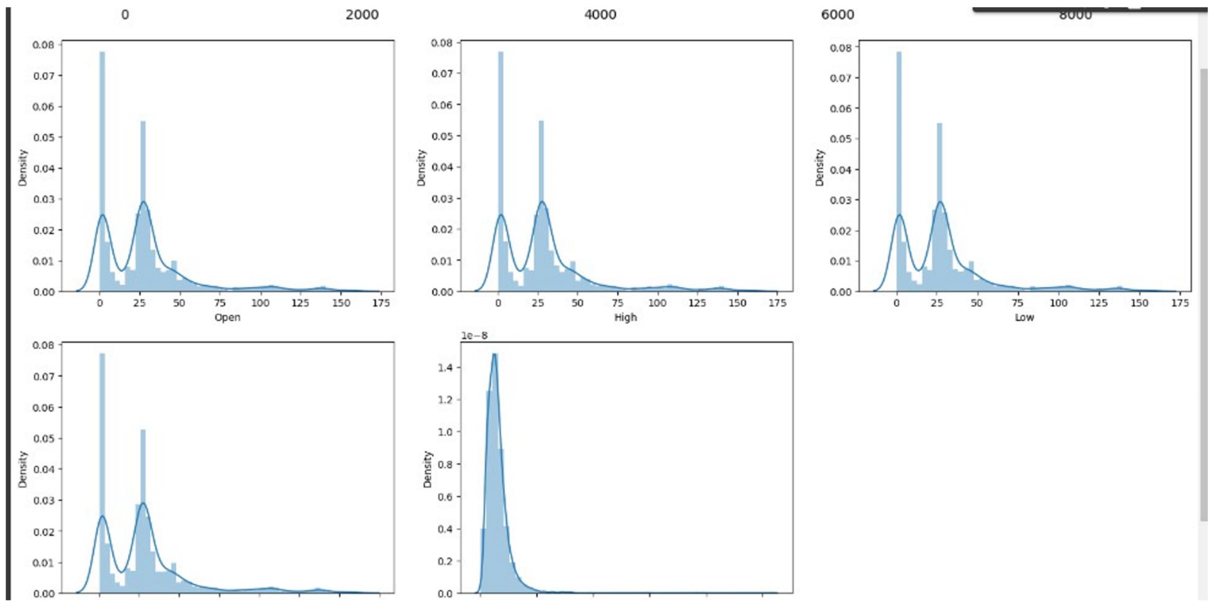
2. Distplot & Subplot:

A "distplot" stands for a distribution plot. It is a type of plot used to visualize the distribution of a univariate dataset, which means it's typically used for examining the distribution of a single variable. It is often used to understand the underlying shape and characteristics of the data, such as whether it follows a normal distribution, is skewed, or has multiple peaks.

A "subplot" refers to a way of creating multiple smaller plots or charts within a single figure or canvas. Subplots allow you to display and compare different aspects of your data simultaneously. They are commonly used when you want to show several visualizations side by side or in a grid.

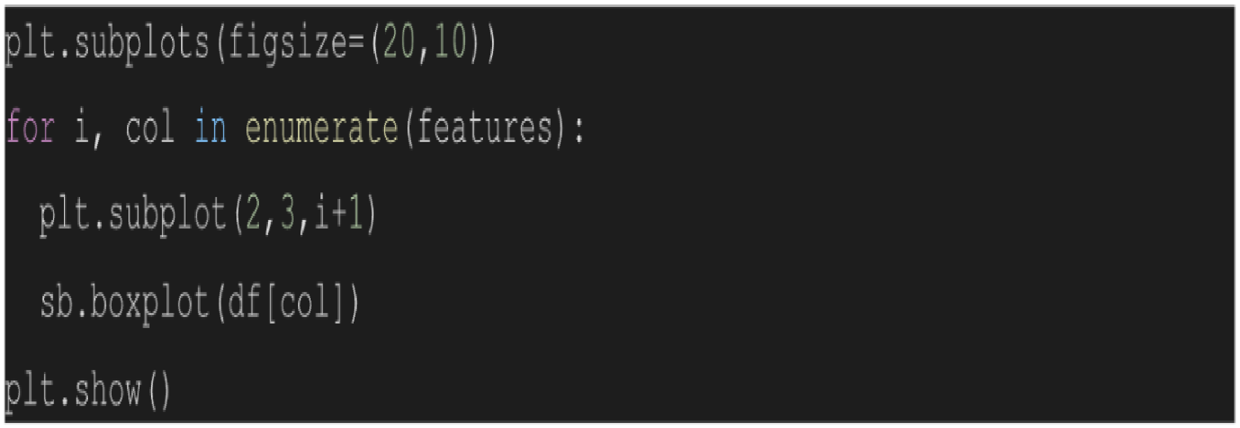
```
plt.subplots(figsize=(20,10))

features = ['Open', 'High', 'Low', 'Close', 'Volume']
for i,col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.distplot(df[col])
plt.show()
```

OUTPUT:



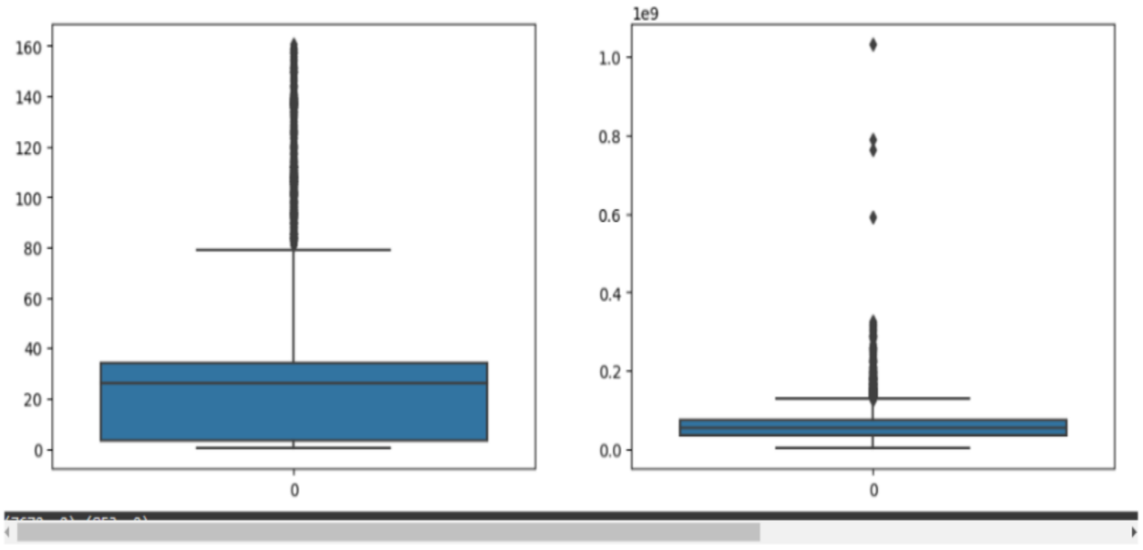
3.BOXPLOT

A boxplot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset.They can help identify skewness, the presence of outliers, and the general shape of the data distribution.



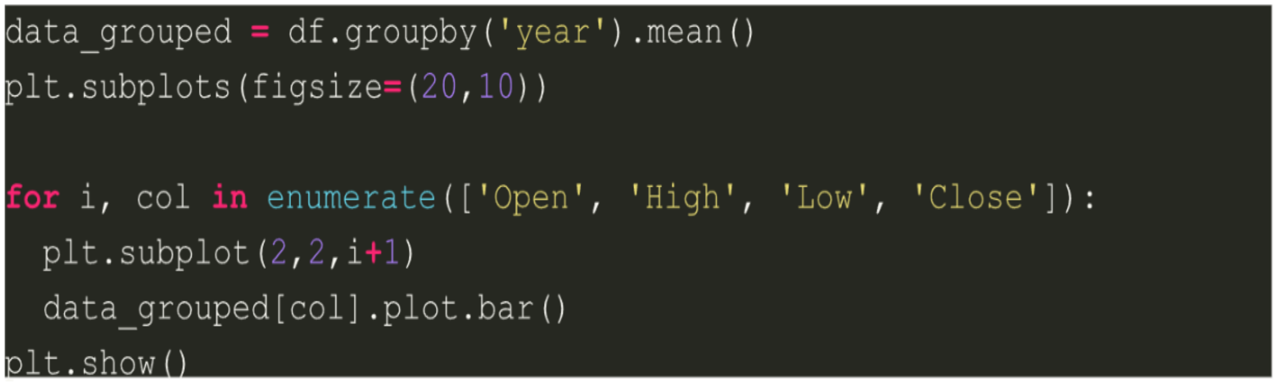
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.boxplot(df[col])
plt.show()
```

OUTPUT



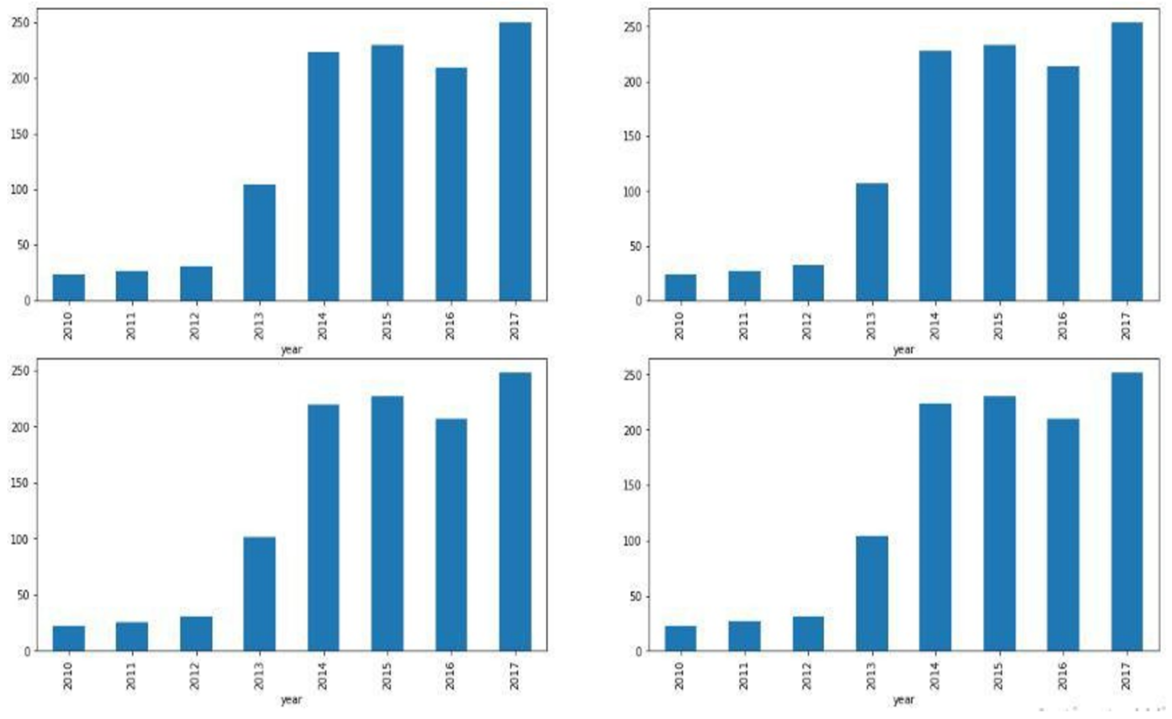
4.BAR CHART

A bar chart is a graphical representation of data that uses rectangular bars or columns to represent the values of different categories, items, or groups.



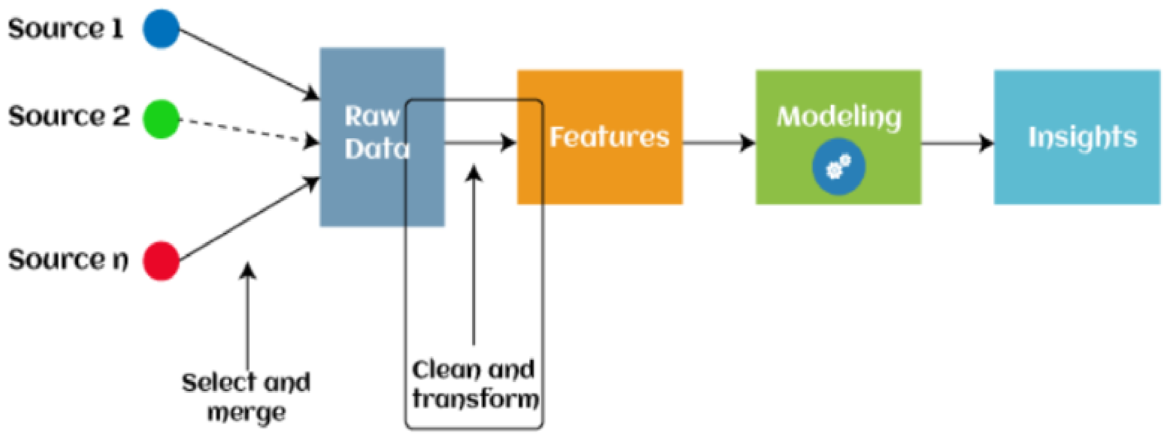
```
data_grouped = df.groupby('year').mean()
plt.subplots(figsize=(20,10))

for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
  plt.subplot(2,2,i+1)
  data_grouped[col].plot.bar()
plt.show()
```

OUTPUT



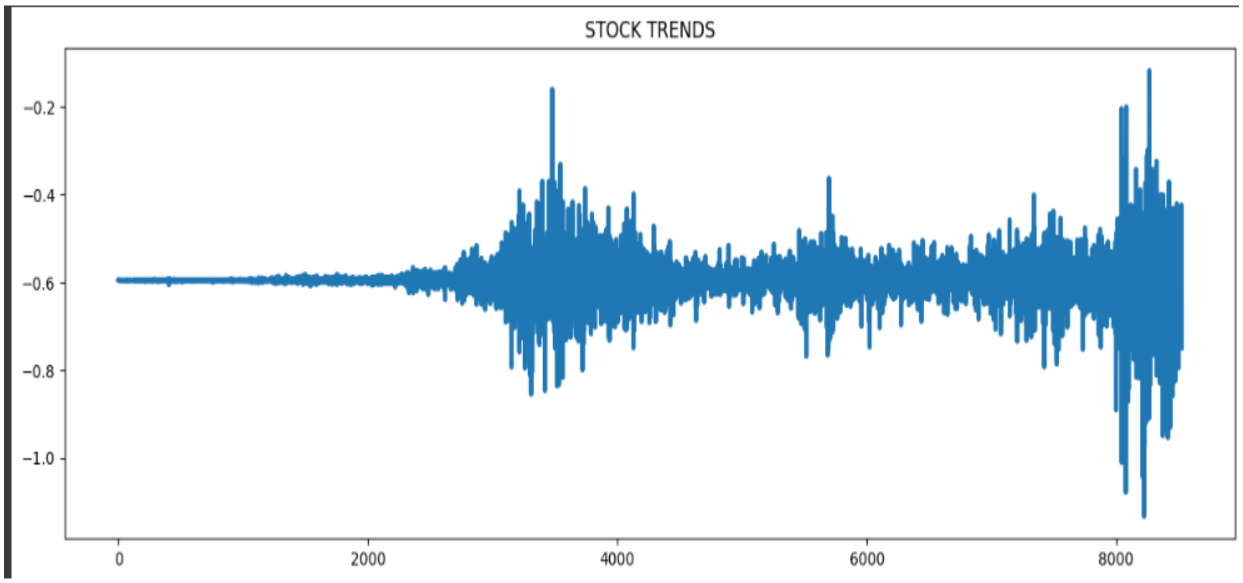
FEATURED ENGINEERING

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.
Here in this stock prediction model Open,Close,High,Low column are transformed as features.Using Linear Regression and Series for visualization.



```
df['open-close'] = df['Open'] - df['Close']
df['low-high']   = df['Low'] -
df['High'] # Training data
X = df.loc[:, ['open-close']]      #
features y = df.loc[:, 'low-high'] #
target

# Train the model
model =
LinearRegression()
model.fit(X, y)

# Store the fitted values as a time series with the same time index
as # the training data
y_pred = pd.Series(model.predict(X),
index=X.index) ax= y_pred.plot( linewidth=3)
ax.set_title('STOCK TRENDS')
```

OUTPUT



MODEL TRAINING:

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability of an instance of belonging to a given class or not. It is a kind of statistical algorithm, which analyzes the relationship between a set of independent variables and the dependent binary variables.

Here the independent and dependent variables are Open-Close and Low-High respectively by using XGBClassifier.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics
```

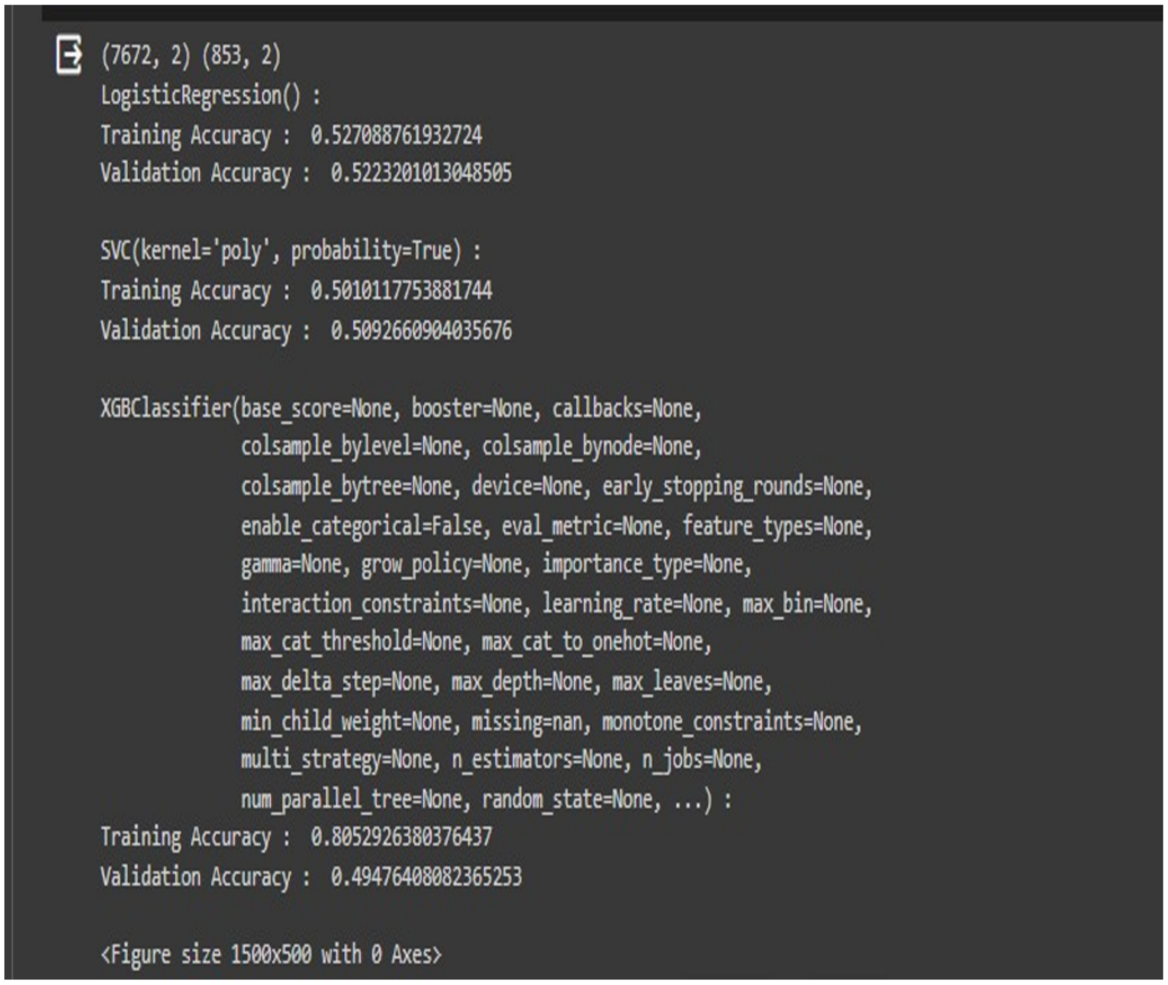
```python
df['open-close']  = df['Open'] - df['Close']
df['low-high']  = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)

features = df[['open-close', 'low-high']]
target = df['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(
    features, target, test_size=0.1, random_state=2022)

print(X_train.shape, X_valid.shape)

models = [LogisticRegression(), SVC(
  kernel='poly', probability=True), XGBClassifier()]
```

EVALUATION:

In stock prediction the model is evaluated based on its accuracy in prediction of the stocks future trends and also evaluated based on validation accuracy.If the accuracy is towards close to 1 then the model is more accurate and predictions are more true and accurate.For the evaluation metric, we will use the ROC-AUC curve but why this is because instead of predicting the hard probability that is 0 or 1 we would like it to predict soft probabilities that are continuous values between 0 to 1. And with soft probabilities, the ROC-AUC curve is generally used to measure the accuracy of the predictions.

```
for i in range(3):
  models[i].fit(X_train, Y_train)

  print(f'{models[i]} : ')

  print('Training Accuracy : ', metrics.roc_auc_score(
    Y_train, models[i].predict_proba(X_train)[:,1]))

  print('Validation Accuracy : ', metrics.roc_auc_score(
    Y_valid, models[i].predict_proba(X_valid)[:,1]))
  print()
```

OUTPUT

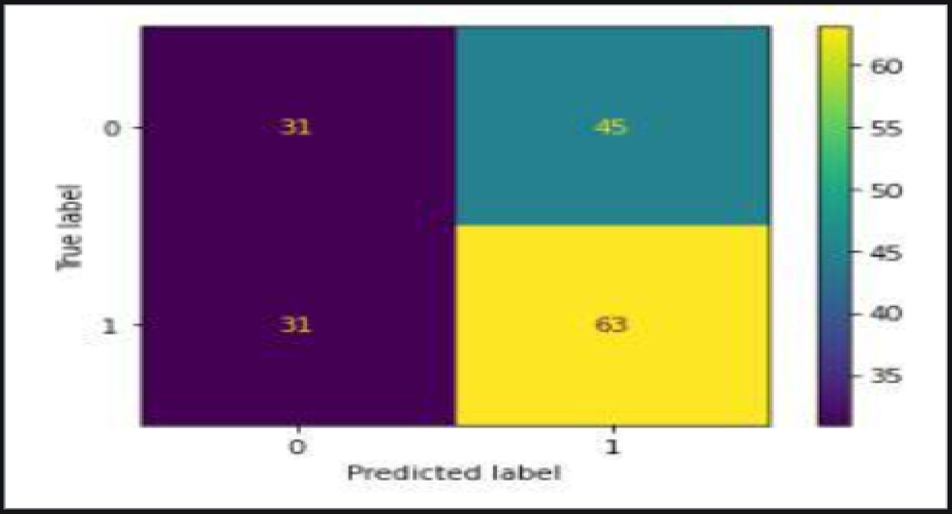


```
(7672, 2) (853, 2)
LogisticRegression() :
Training Accuracy :  0.527088761932724
Validation Accuracy :  0.5223201013048505

SVC(kernel='poly', probability=True) :
Training Accuracy :  0.5010117753881744
Validation Accuracy :  0.5092660904035676

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  0.8052926380376437
Validation Accuracy :  0.49476408082365253

<Figure size 1500x500 with 0 Axes>
```

VALIDATION

Among the three models, the XGBClassifier has the highest performance but it is pruned to overfitting as the difference between the training and the validation accuracy is too high. But in the case of the Logistic Regression, this is not the case

```
metrics.plot_confusion_matrix(models[0], X_valid, Y_valid)
plt.show()
```



Confusion matrix for the validation data

CONCLUSION:

This project on stock prediction using applied data science underscores the significance of responsible data handling, transparency, and fairness in modeling, particularly when dealing with financial data that impacts investors' decisions and market stability.We can observe that the accuracy achieved by the state-of-the-art ML model is no better than simply guessing with a probability of 50%. Instead of simply guessing by using supervised and reinforcement machine learning algorithms from analyzing the past performance of stock trends in market stock the prediction is done through training the model to predict the future stock trends with more accuracy.