LOYOLA ACADEMY DEGREE & PG COLLEGE

(An Autonomous Degree College affiliated to Osmania University)

Alwal, Secunderabad -500010

Accredited with “A” grade by NAAC

A “College with Potential for Excellence” by UGC

2023-2025

MINI PROJECT REPORT

ON

“NETFLIX STOCK PRICE PREDICTION”

BY

ROOHI AFSHA 111723720003

Department of MSc. Data Science

LOYOLA ACADEMY DEGREE & PG COLLEGE

(An autonomous Degree College affiliated to Osmania University)
Alwal, Secunderabad-500010

Accredited with "A" grade by NAAC
A COLLEGE WITH POTENTIAL FOR EXCELLENCE BY UGC

Project Report

This is to certify that this is the bonafide record of the mini project titled

"Netflix Stock Price Prediction"

Done during second year first semester for the completion of

MSc. Data Science Degree by

ROOHI AFSHA UID:111723720003

Signature of Internal **Signature of HoD**

Signature of External **Signature of Principal**

ACKNOWLEDGEMENT

"Task successful" makes everyone happy. But the happiness will gold without glitter if we didn't state the persons who have supported us to make a success.

Success will be crowned to people who made it reality but people who constant guidance and encouragement made it possible crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped me for the completion of my project work.

I consider myself lucky enough to get such a project. This mini project would as an asset to my academic profile.

I express my thanks to our beloved principal **Rev Fr Dr N. B. Babu SJ**, who always inspire and motivated us to perform in a better and efficient manner.

I would like to express our thankfulness to my project guide **Mr. N Ashfaq, Head of Department of MSc. Data Science** for his constant motivation and valuable help through the project work, and I express my gratitude to, for his supervision, guidance and cooperation through the project.

ROOHI AFSHA (111723720003)

DECLARATION

This is to inform that I **ROOHI AFSHA** the student of **MSc. Data Science** had completed the mini project work on “Netflix stock price prediction” in the year 2024-25.

I have completed my mini project under the guidance of **Mr. N Ashfaq (HOD),** department of MSc Data Science.

I hereby declare that this mini project report submitted by me is the original work done by a part of my academic course and has not been submitted to any university or institution for the award of any degree or diploma.

ROOHI AFSHA (UID: 111723720003)

ABSTRACT

Stock price prediction is a critical area of research in financial analytics, aiding investors in making precise and informed decisions. Accurate forecasting of stock prices is essential for minimizing risks and maximizing returns. With advancements in machine learning, stock prediction models are now more reliable and efficient. In this project, we propose a Netflix stock price prediction model utilizing data mining and machine learning techniques. We apply four regression-based approaches: Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Support Vector Regressor (SVR). The model is trained using Python and analyzed with a real dataset collected from Kaggle, which includes historical stock data such as Open Price, High Price, Low Price, Volume, and Close Price. Furthermore, the performance of the proposed models is evaluated using metrics like Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R^2 Score. Among the tested algorithms, Linear Regression achieved the highest performance with an R^2 Score of 99%. The developed framework also includes a user-friendly graphical user interface (GUI) designed with Tkinter, allowing users to input stock attributes and receive predictions for the closing price instantly. This project demonstrates the potential of machine learning techniques in stock price prediction, offering a practical solution for forecasting market trends. Future improvements could include the integration of external factors such as market sentiment, news analysis, and global economic indicators to further enhance predictive accuracy.

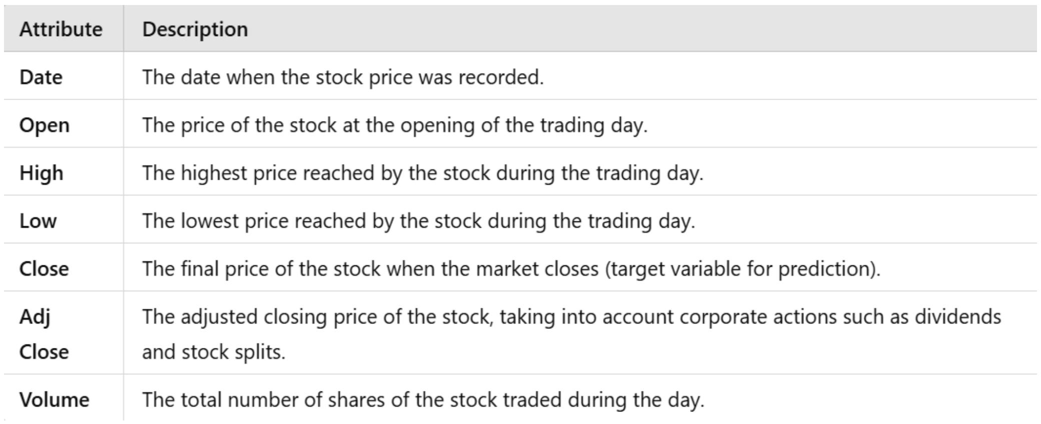
CONTENTS

I. INTRODUCTION

Netflix's stock is a strong candidate for prediction due to its unique market position and high volatility, driven by trends in consumer behavior and competition in the streaming space. Predicting its stock price can provide insights into market behavior and assist investors in understanding patterns influenced by industry-specific events. Netflix has become one of the most prominent players in the entertainment industry, with its stock reflecting the dynamic growth and competition within this sector. The rise of streaming services, increasing global subscribers, and frequent shifts in market sentiment make Netflix stock a fascinating and challenging case for prediction. Predicting stock prices is challenging due to the dynamic nature of the market and external factors like global events.

1.1. Data Description:

The dataset for this project includes data from Kaggle about Netflix stock prices. So, totally I have 7 features with rows.



Attribute	Description
Date	The date when the stock price was recorded.
Open	The price of the stock at the opening of the trading day.
High	The highest price reached by the stock during the trading day.
Low	The lowest price reached by the stock during the trading day.
Close	The final price of the stock when the market closes (target variable for prediction).
Adj Close	The adjusted closing price of the stock, taking into account corporate actions such as dividends and stock splits.
Volume	The total number of shares of the stock traded during the day.

1.2. Important Libraries:

In this project, I have used various libraries to perform the tasks. Few libraries are Numpy, Pandas, Pickle, Sklearn, Matplotlib etc.

1.3. Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

1.4. Data Preprocessing:

Preprocessing refers to the transformations applied to our data before feeding it to the algorithm.

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. In this project that I have performed only one preprocessing task i.e. feature selection.

1.5. Algorithm Application:

The algorithm that I have applied for my project is Random Forest. Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

1.6. Pickling the File:

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can convert the byte stream (generated through pickling) back into python objects by a process called as unpickling. In real world scenario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

1.7. Deployment:

Machine learning deployment is the process of deploying a machine learning model in a live environment. The model can be deployed across a range of different environments and will often be integrated with apps through an API. Deployment is a key step in an organization gaining operational value from machine learning. For this project I deployed the model using tkinter package in python. Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

The dataset for this problem includes data from kaggle about Netflix. So,
the dataset consists of 7 features which are described below.

II. IMPORTANT LIBRARIES

In this project, I have used various libraries to perform the tasks. Following are the libraries that I have used in this project.

2.1. Pandas:

pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

2.2. Numpy:

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project

2.3. Scikit learn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPyand Matplotlib.

2.4. Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

2.5. Pickle:

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

2.6. Tkinter:

Python offers tkinter module to create graphic programs. The tkinter represents 'toolkit interface' for GUI. This is an interface for Python programmers that enable them to use the classes of TK module of TCL/TK language. Let's see what this TCL/TK is. The TCL (Tool Command Language) is a powerful dynamic programming language, suitable for web and desktop applications, networking, administration, testing and many more. It is open source and hence can be used by any one freely. TCL language uses TK (Tool Kit) language to generate graphics. TK provides standard GUI not only for TCL but also for many other dynamic programming languages like Python. Hence, this TK is used by Python programmers in developing GUI applications through Python's tkinter module.

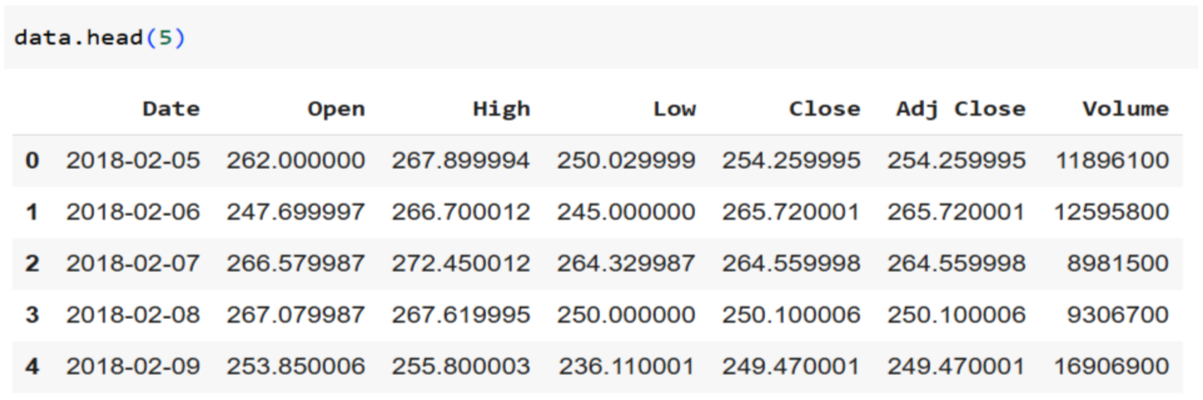
2.7. Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representationsfor same variables for better understanding of dataset.

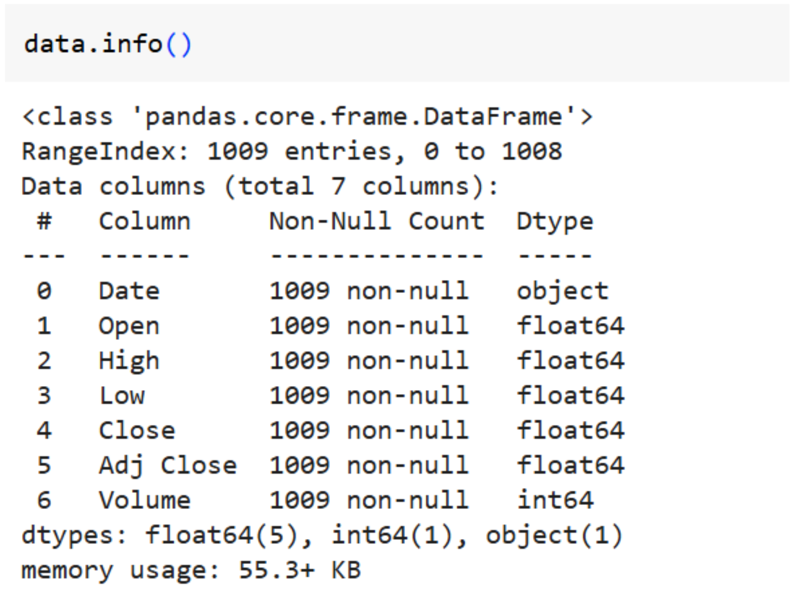
III. DATASET DESCRIPTION

This dataset is taken from the Kaggle website. The objective of the dataset is to predict the stock price of Netflix, based on various historical stock prices and trading volumes included in the dataset. Several factors were considered when selecting the data, including daily stock prices, volume of shares traded, and market conditions. The dataset contains multiple independent variables (such as Open, High, Low, Volume) and one dependent variable (Close Price), which is the target variable used for predictionIn the below pictures you can see the snippet of the dataset and its description



```
data.head(5)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1009 non-null   object 
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

IV. EXPLORATORY DATA ANALYSIS

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

4.1. Why is exploratory data analysis important in data science?

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modelling, including machine learning.

4.2. Data Visualization:

Data visualization is defined as a graphical representation that contains the information and the data. By using visual elements like charts, graphs, and maps, data visualization techniques provide an accessible way to see and understand trends, outliers, and patterns in data.

In modern days we have a lot of data in our hands i.e., in the world of Big Data, data visualization tools, and technologies are crucial to analyze massive amounts of information and make data-driven decisions.

It is used in many areas such as:

- To model complex events.
- Visualize phenomenon’s that cannot be observed directly, such as weather patterns, medical conditions, or mathematical relationships.

4.3. Bar plots:

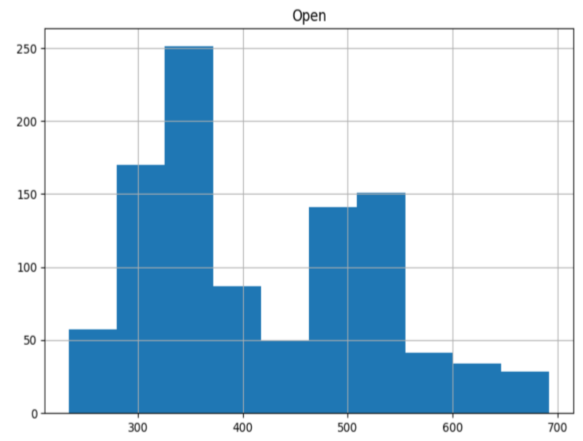
A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

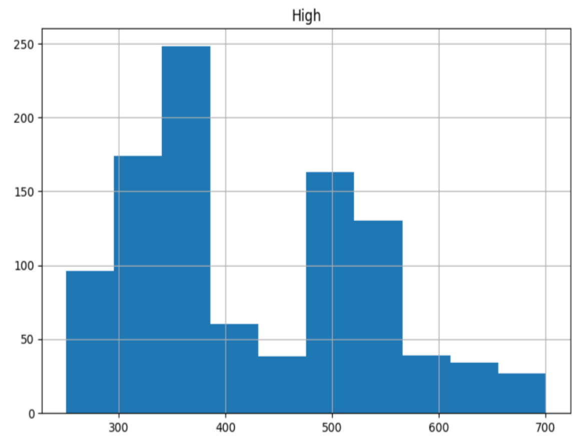
4.4. Bar plots Representation

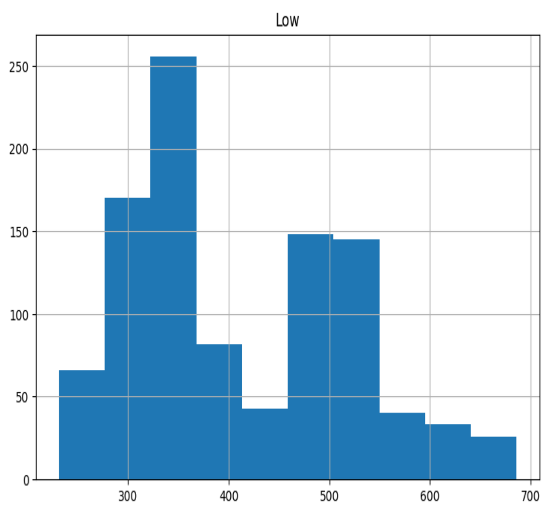
Bar graphs are the pictorial representation of data (generally grouped), in the form of vertical or horizontal rectangular bars, where the length of bars are proportional to the measure of data. They are also known as bar charts. Bar graphs are one of the means of data handling in statistics.
Usually, Histograms represent organized data into groups.

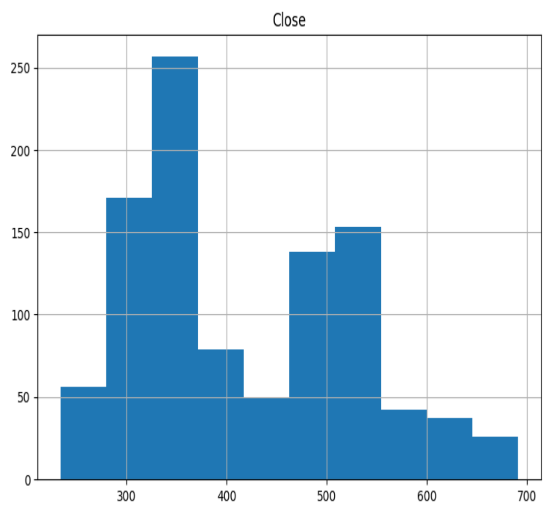
- Firstly, the X-axis represents bin ranges
- Secondly, the Y-axis represents frequency.

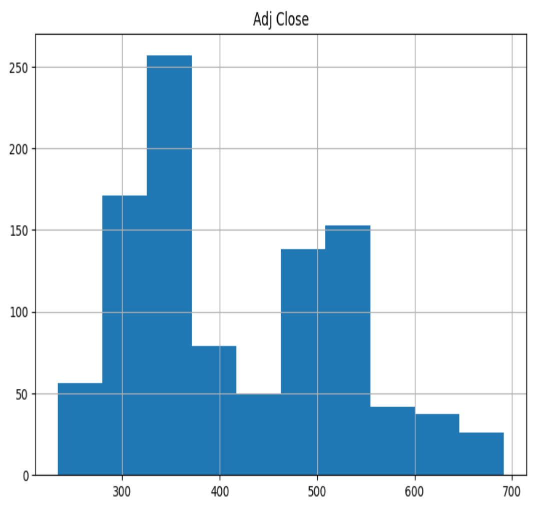
4.5. Visualization using Bar plots:

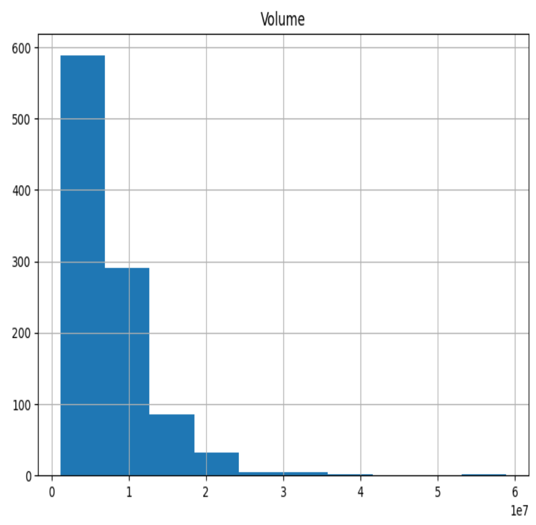





Low
0
50
100
150
200
250
300
400
500
600
700


Close
0
50
100
150
200
250
300
400
500
600
700


Adj Close
0
50
100
150
200
250
300
400
500
600
700


Volume
0
100
200
300
400
500
600
0
1
2
3
4
5
6
1e7

V. DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data preprocessing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

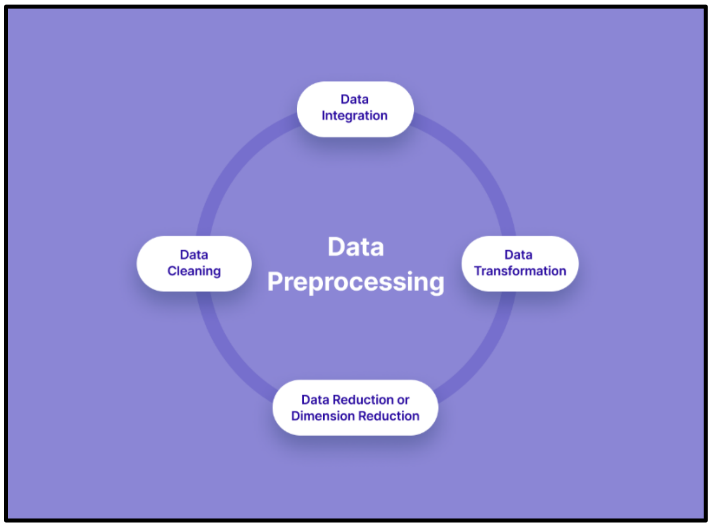
The majority of the real-world datasets are highly susceptible to missing, inconsistent, and noisy data due to their heterogeneous origin.

Applying data mining algorithms on this noisy data would not give quality results as they would fail to identify patterns effectively. Data Processing is, therefore, important to improve the overall data quality.

Duplicate or missing values may give an incorrect view of the overall statistics of data.

Outliers and inconsistent data points often tend to disturb the model’s overall learning, leading to false predictions.

Quality decisions must be based on quality data. Data Preprocessing is important to get this quality data, without which it would just be a Garbage In, Garbage Out scenario.



5.1. Feature Selection:

A feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

So, we can define feature Selection as, "It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

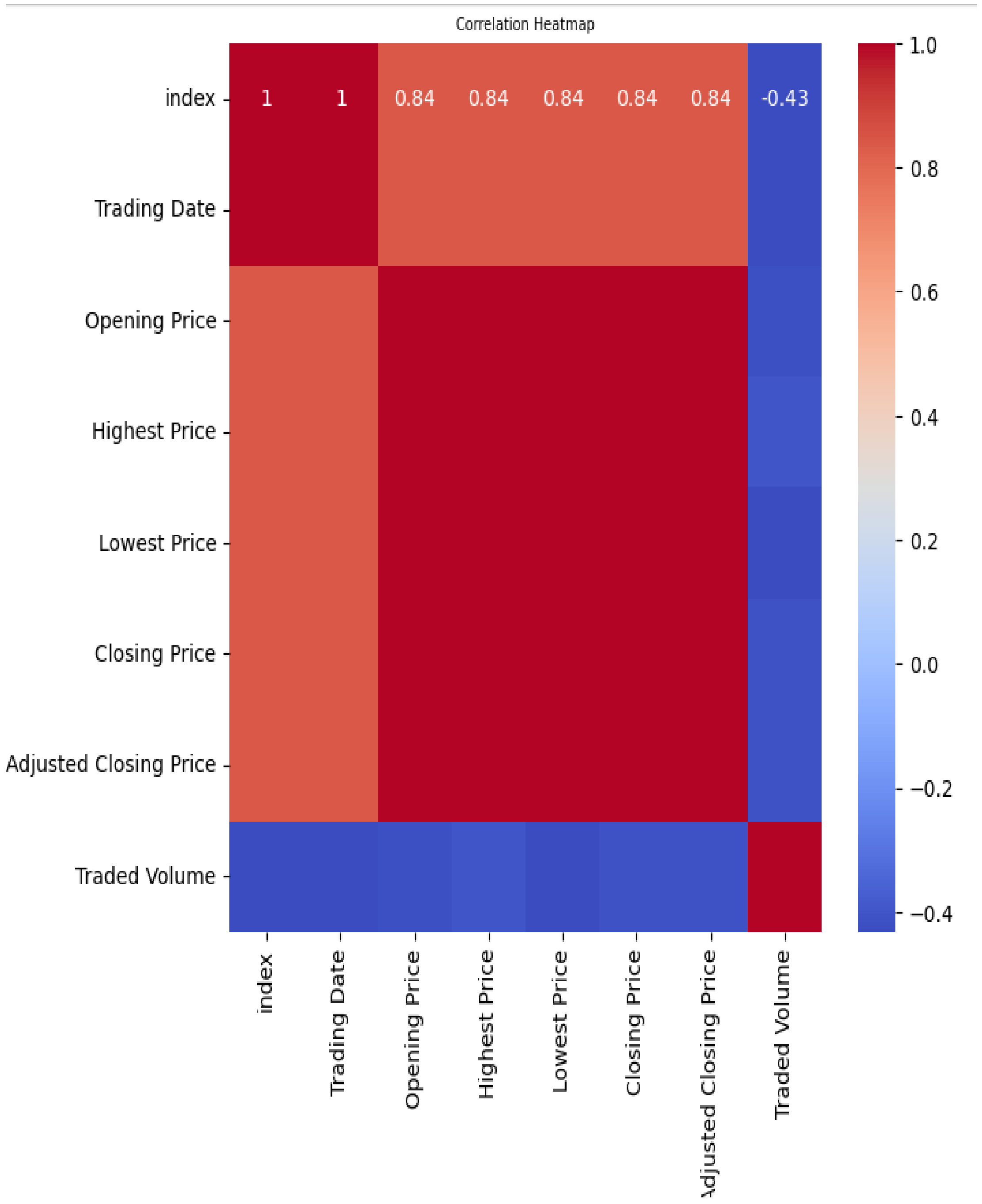
Among Different methods to apply feature selection, we have selected the below method are discussed below

5.2. Filter methods

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

Methods that come under this are

- Information Gain
- Chi-square Test
- Fisher’s Score
- Correlation Coefficient
- Variance Threshold
- Mean Absolute Difference (MAD)
- Dispersion ratio


Correlation Heatmap
index
Trading Date
Opening Price
Highest Price
Lowest Price
Closing Price
Adjusted Closing Price
Traded Volume
1
1
0.84
0.84
0.84
0.84
0.84
-0.43
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

VI. ALGORITHM APPLICATION

Algorithm application is an important part for any machine learning projects. Choosing the right algorithm based on the problem statement and the data gathered is one of the major steps to be taken. As a beginner we might not know the right algorithm at once. So, we need to try different algorithms on our data. The algorithm which performs good and produces the good accuracy score can be chosen.

So, in my project the algorithm that I have chosen is Random Forest, which is a supervised learning algorithm and can be applied for both regression and classification type.

Before we apply any algorithm to our data, we need to split the data into training and testing part. It is an essential step for all supervised algorithms.

Let us get to know what does actually training and testing data means.

6.1. Train-Test Split Evaluation:

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The train-test procedure is appropriate when there is a sufficiently large dataset available.

6.2. How to Configure the Train-Test Split

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or 62 | Pag e test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project's objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Now let us get into the algorithm part, which is the Random Forest.

6.3. Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

6.4. How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new datapoints to the category that wins the majority votes.

6.5. Applications of Random Forest

There are mainly four sectors where Random Forest mostly used:

1. Banking: Banking sector mostly uses this algorithm for the identification of loan risk.

2. Medicine: With the help of this algorithm, disease trends and risks of the disease can be identified.

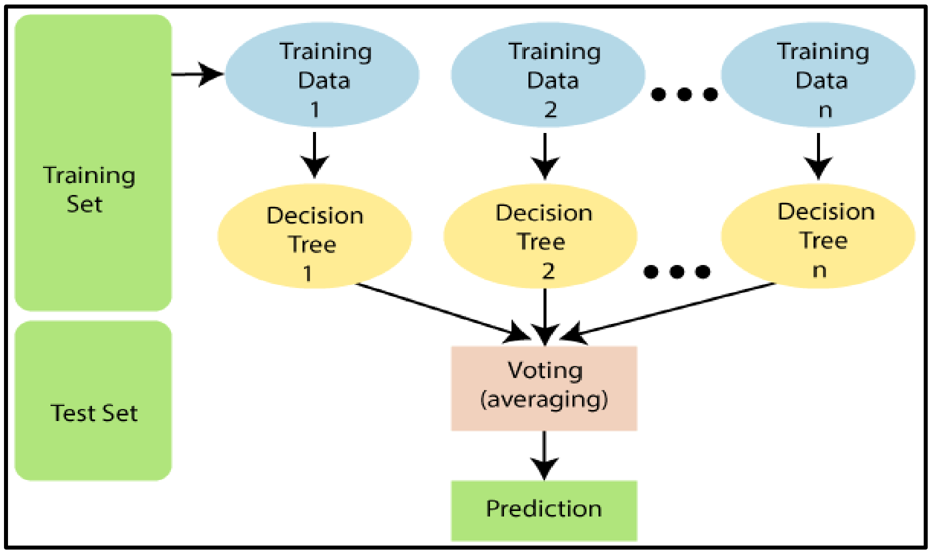
3. Land Use: We can identify the areas of similar land use by this algorithm.

4. Marketing: Marketing trends can be identified using this algorithm.

6.6. Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations



VII. PICKLING AND UNPICKLING THE FILE

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can convert the byte stream (generated through pickling) back into python objects by a process called as unpickling.

Why Pickle?

In real world scenario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

Precaution: It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data.

Only after importing pickle module, we can do pickling and unpickling. Importing pickle can be done using the following command –

import pickle

To pickle a file, you can use the following code

```
with open('file.pkl','wb') as fh:
    pickle.dump(model,fh)
```

To unpickle the file you can use the following code

```
with open('file.pkl','rb') as fh:

    ver=pickle.load(fh)
```

VIII. DEPLOYMENT

Machine learning deployment is the process of deploying a machine learning model in a live environment. The model can be deployed across a range of different environments and will often be integrated with apps through an API. Deployment is a key step in an organization gaining operational value from machine learning.

Machine learning models will usually be developed in an offline or local environment, so will need to be deployed to be used with live data. A data scientist may create many different models, some of which never make it to the deployment stage. Developing these models can be very resource intensive. Deployment is the final step for an organization to start generating a return on investment for the organization.

However, deployment from a local environment to a real-world application can be complex. Models may need specific infrastructure and will need to be closely monitored to ensure ongoing effectiveness. For this reason, machine learning deployment must be properly managed so it’s efficient and streamlined.

8.1 How to deploy machine learning models

Machine learning deployment can be a complex task and will differ depending on the system environment and type of machine learning model. Each organization will likely have existing DevOps processes that may need to be adapted for machine learning deployment. However, the general deployment process for machine learning models deployed to a containerized environment will consist of four broad steps.

The four steps to machine learning deployment include:

- Develop and create a model in a training environment.
- Test and clean the code ready for deployment.
- Prepare for container deployment.
- Plan for continuous monitoring and maintenance after machine learning deployment.

For this project I deployed the model using tkinter package in python.

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

8.2 To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

IX. CODE FOR IMPLEMENYING THE PROJECT

#Importing Packages

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

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.metrics import r2_score,mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
```

#Importing the dataset

```
data=pd.read_csv("NFLX.csv")
print(data.shape)
```

Output:

```
(1009, 7)
```

#Data Exploration

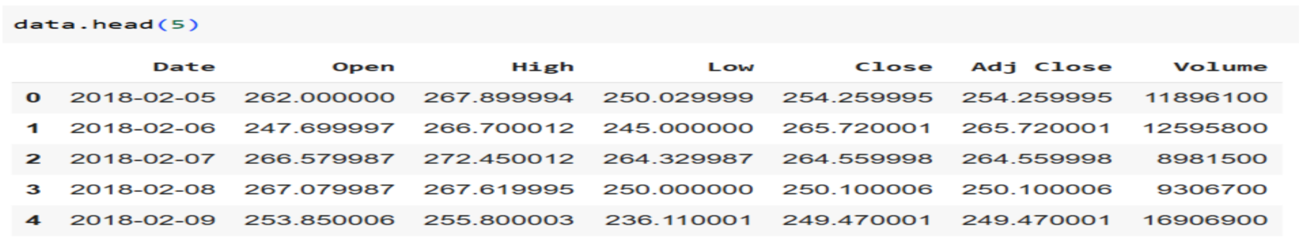
```
data.columns
```

Output:

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
 dtype='object')
```

data.head()

Output:

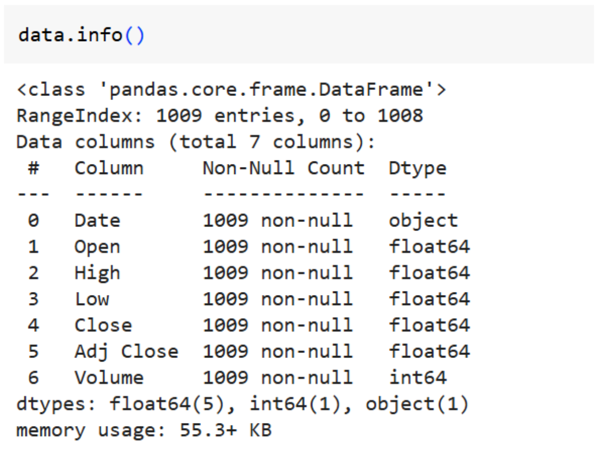


```
data.head(5)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

data.info()

Output:

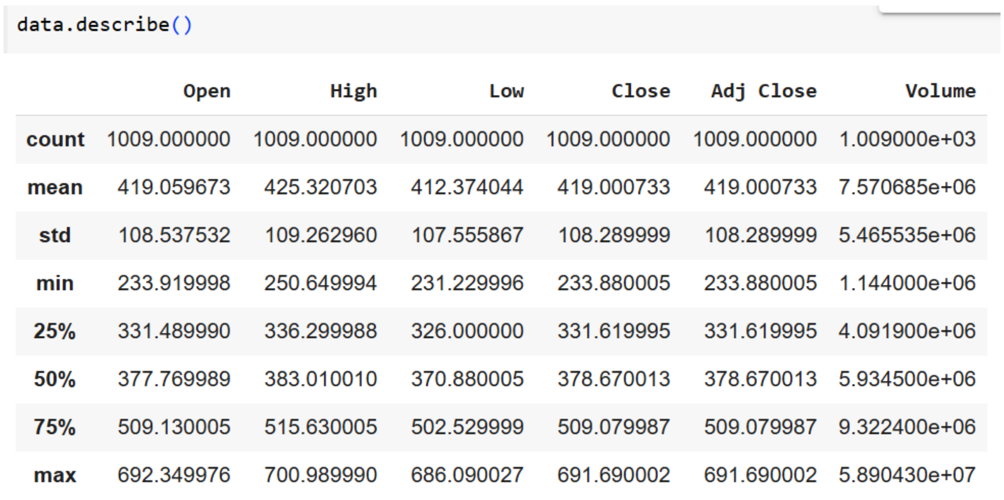


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1009 non-null   object 
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

data.describe()

Output:



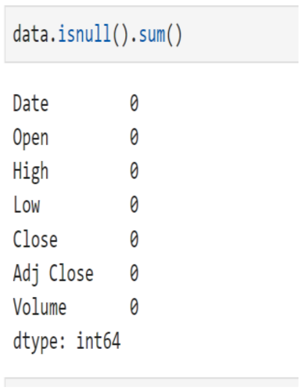
```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

- Now let's check that if our dataset have null values or not

data.isnull().sum()

Output:



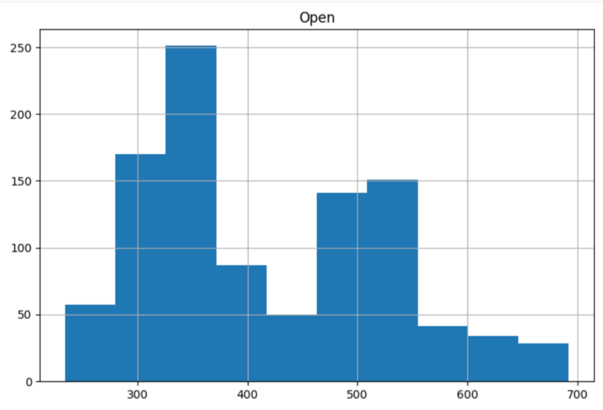
```
data.isnull().sum()

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

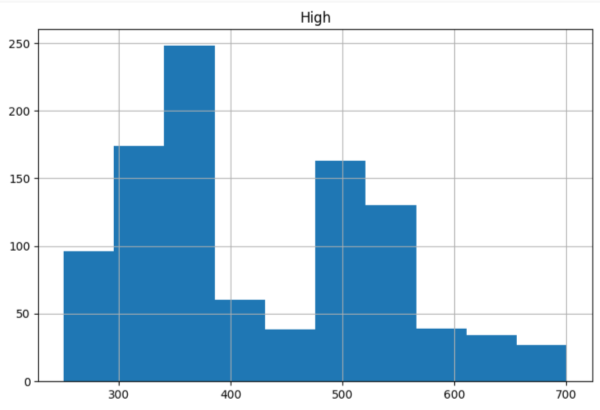
Data Visualization

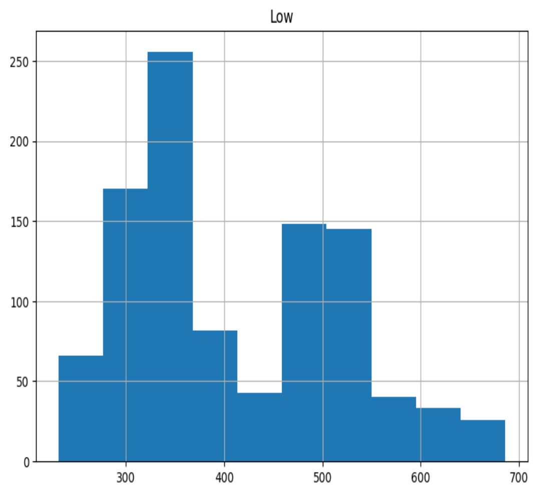
- Plotting the data distribution plots before removing null values

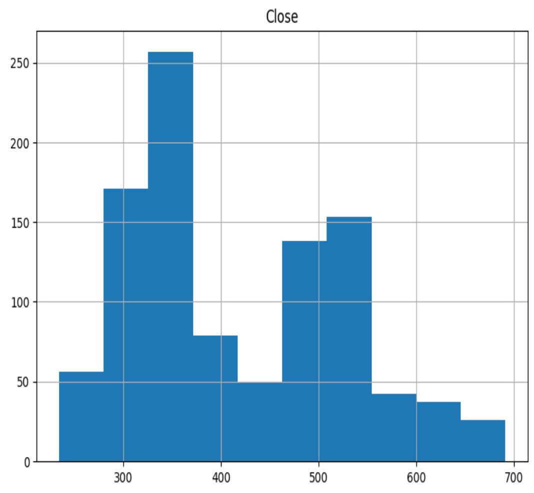
data.hist(figsize = (20,20))

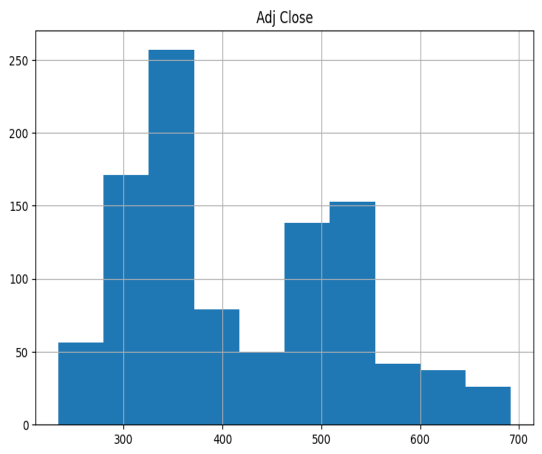
Output:

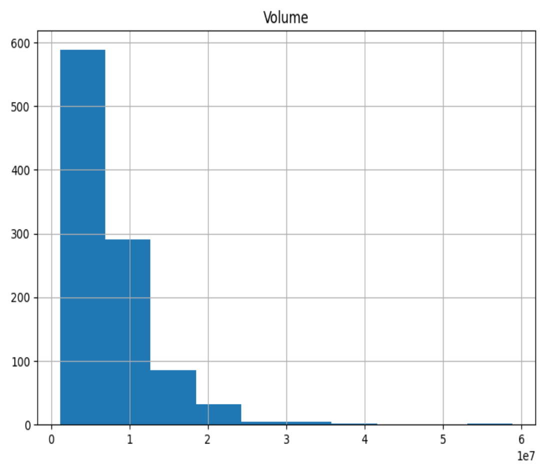





Low
0
50
100
150
200
250
300
400
500
600
700


Close
0
50
100
150
200
250
300
400
500
600
700

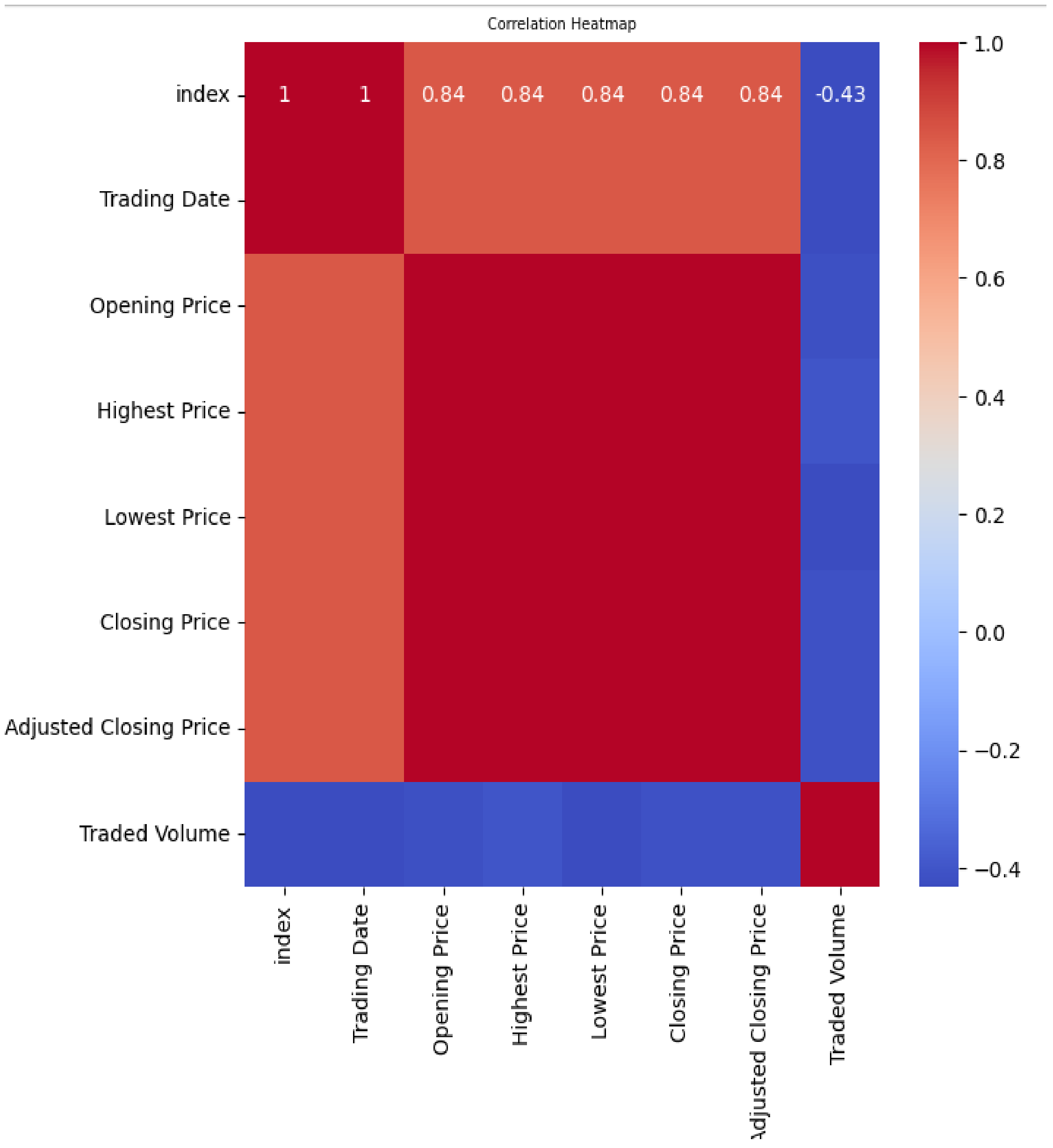

Adj Close
0
50
100
150
200
250
300
400
500
600
700


Volume
0
100
200
300
400
500
600
0
1
2
3
4
5
6
1e7

#Correlation between all the features

Correlation between all the features before cleaning
plt.figure(figsize=(7,7))
sns.heatmap(data.corr(),annot=True,
cbar=True,cmap='coolwarm')
plt.title('Correlation Heatmap', fontdict={'fontsize':
7})
plt.show()

Output:



#Model Building

splitting the dataset

```
X = data[['Opening Price', 'Highest Price', 'Lowest Price', 'Traded Volume']]
 y = data['Closing Price']
```

- Now we will split the data into training and testing data using the train_test_split function

```
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

  print("Training set shape:", X_train.shape)
  print("Testing set shape:", X_test.shape)
```

Output:

```
Training set shape: (807, 4)

Testing set shape: (202, 4)
```

#Scaling the data

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled.std()
```

Output:

```
1.0
```

```
X_test_scaled.std()
```

Output:

```
0.971267285558647
```

#Algorithm Application

```
Model1=RandomForestClassifier()
y_predict=model.predict(x_test)
accuracy_score(y_predict,y_test)*100
```

Output:

```
99.76232864811463
```

GRAPHICAL USER INTERFACE

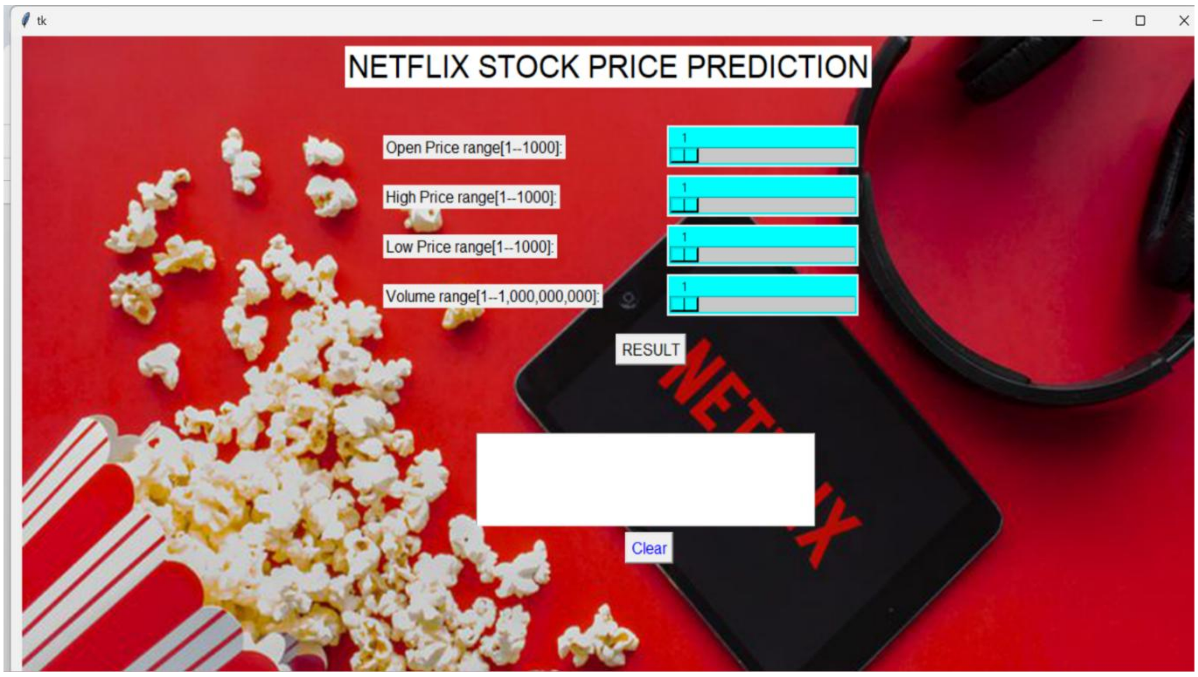
#Pickel file

```
import tkinter as tk
from tkinter import messagebox
import numpy as np
import pickle
from PIL import Image, ImageTk

filename="Untitled"
pickle.dump(model,open(filename,'wb'))
model22=pickle.load(open(filename,'rb'))

with open('model22.pkl','wb') as files:
    pickle.dump(model22,files)

with open('model22.pkl','rb') as file:
    Result=pickle.load(file)


def resize_image(image, max_width, max_height):
    original_width, original_height = image.size
    ratio_width = (max_width * 0.7) / original_width  # Reduce max width by 30%
    ratio_height = (max_height * 0.7) / original_height  # Reduce max height by 30%
    ratio = min(ratio_width, ratio_height)
    new_width = int(original_width * ratio)
    new_height = int(original_height * ratio)
    resized_image = image.resize((new_width, new_height), Image.LANCZOS)
    return resized_image
```

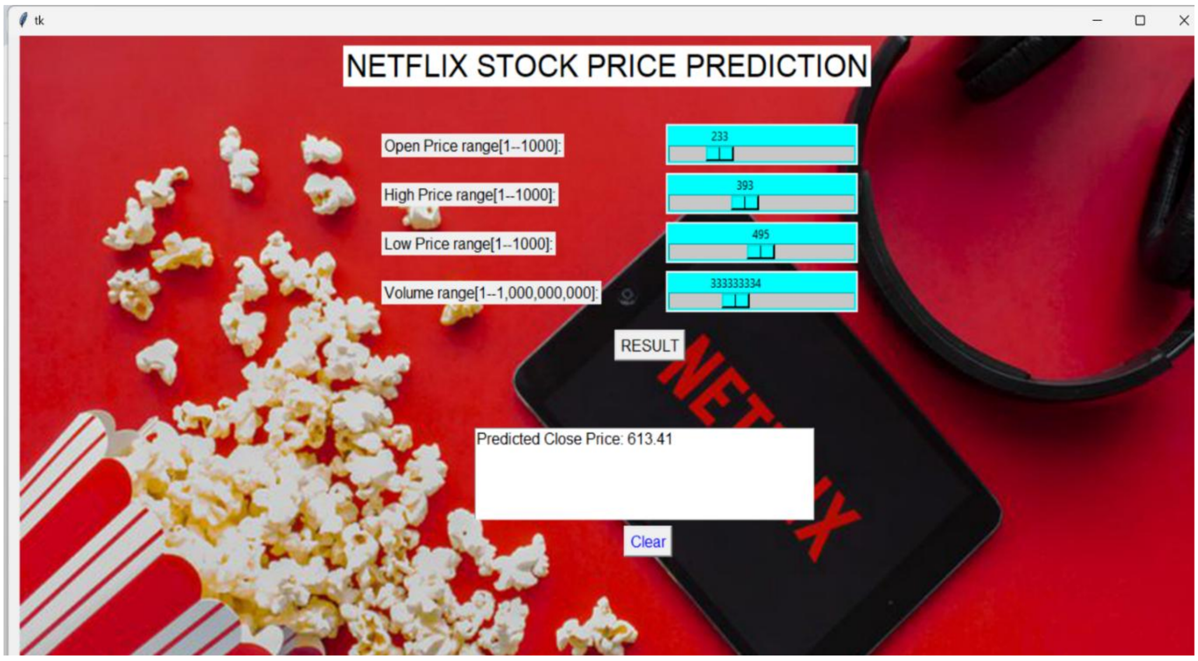
```
root = tk.Tk()
root.geometry("1920x1080")

image_path = "C:/Users/Roohi Afsha/Documents/Downloads/IMG_1376.JPG"
image = Image.open(image_path)
image = resize_image(image, 1920, 1080)
bg_image = ImageTk.PhotoImage(image)

bg_label = tk.Label(root, image=bg_image)
bg_label.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

label1 = tk.Label(root, text="NETFLIX STOCK PRICE PREDICTION", fg="black", bg="white",
font=("Arial", 24))
label1.pack(pady=10)

# Centralizing widgets - adjust x values
x_pos = 400  # Further adjusted value for more central placement

# Feature 1: Open Price
label2 = tk.Label(root, text="Open Price range[0--1000]:", fg="black", font=("Arial", 12))
label2.place(x=x_pos, y=100)
scale1 = tk.Scale(root, from_=0, to=1000, length=200, orient=tk.HORIZONTAL, bg="cyan",
fg="black")
scale1.place(x=x_pos + 305, y=90)
scale1.set(0)

# Feature 2: High Price
label3 = tk.Label(root, text="High Price range[0--1000]:", fg="black", font=("Arial", 12))
label3.place(x=x_pos, y=150)
scale2 = tk.Scale(root, from_=0, to=1000, length=200, orient=tk.HORIZONTAL, bg="cyan",
fg="black")
scale2.place(x=x_pos + 305, y=140)
scale2.set(0)

# Feature 3: Low Price
label4 = tk.Label(root, text="Low Price range[0--1000]:", fg="black", font=("Arial", 12))
label4.place(x=x_pos, y=200)
scale3 = tk.Scale(root, from_=0, to=1000, length=200, orient=tk.HORIZONTAL, bg="cyan",
fg="black")
scale3.place(x=x_pos + 305, y=190)
scale3.set(0)

# Feature 4: Volume
label5 = tk.Label(root, text="Volume range[0--1,000,000,000]:", fg="black", font=("Arial", 12))
label5.place(x=x_pos, y=250)
scale4 = tk.Scale(root, from_=0, to=1000000000, length=200, orient=tk.HORIZONTAL,
bg="cyan", fg="black")
```

```
scale4.place(x=x_pos + 305, y=240)
scale4.set(0)

# Predict button
button1 = tk.Button(root, text='RESULT', fg="black", command=lambda: predict(), font=("Arial",
12))
button1.place(x=500 + 150, y=300)
# Text box for displaying results
t = tk.Text(root, highlightbackground='green', fg="black", bg="white", height=5, width=40,
font=("Arial", 12))
t.place(x=500, y=400)
# Reset button
button2 = tk.Button(root, text='Clear', fg="blue", command=lambda: reset(), font=("Arial", 12))
button2.place(x=500 + 160, y=500)
# Prediction function
def predict():
    try:
        open_price = float(scale1.get())
        high_price = float(scale2.get())
        low_price = float(scale3.get())
        volume = int(scale4.get())
        if not (0 <= open_price <= 1000):
            raise ValueError("'Open Price' should be between 0 and 1000")
        if not (0 <= high_price <= 1000):
            raise ValueError("'High Price' should be between 0 and 1000")
        if not (0 <= low_price <= 1000):
            raise ValueError("'Low Price' should be between 0 and 1000")
        if not (0 <= volume <= 1000000000):
            raise ValueError("'Volume' should be between 0 and 1,000,000,000")
        inp = np.array([open_price, high_price, low_price, volume])
        prediction = Result.predict(inp.reshape(1, -1))

        t.delete('1.0', 'end')
        text = f"Predicted Close Price: {prediction[0]:.2f}"
        t.insert('1.0', text)

    except ValueError as ve:
        messagebox.showerror('Error', str(ve))
    # Reset function to clear all inputs
    def reset():
         scale1.set(0)
         scale2.set(0)
         scale3.set(0)
         scale4.set(0)
        t.delete('1.0', 'end')

root.mainloop()
```

X. SCREENSHOT'S OF GUI



Open Price range[1--1000]: 1
High Price range[1--1000]: 1
Low Price range[1--1000]: 1
Volume range[1--1,000,000,000]: 1



Open Price range[1--1000]: 233
High Price range[1--1000]: 393
Low Price range[1--1000]: 495
Volume range[1--1,000,000,000]: 333333334

Predicted Close Price: 613.41

XI. CONCLUSION

The Netflix stock price prediction project successfully demonstrates the potential of machine learning in analysing financial data and forecasting stock prices. By utilizing a regression model, specifically Random Forest, we were able to predict the closing price of Netflix stock with high accuracy. The analysis revealed key factors influencing stock price movements, such as opening price, high, low, and trading volume, enabling us to understand the dynamics of the stock market better.

Through data visualization, trends and patterns were identified, which offer valuable insights for investors seeking to make data-driven decisions. The project emphasizes the importance of historical data in predicting future stock performance and shows how machine learning can be a reliable tool for financial analysis. It also highlights the role of feature selection and preprocessing in building efficient predictive models.

This project not only provides a foundation for stock price forecasting but also demonstrates its potential to support traders and investors in making informed decisions. By bridging the gap between raw data and actionable insights, this project showcases how machine learning can be used to solve real-world problems effectively in the financial domain. It paves the way for more sophisticated systems that can integrate multiple data sources and deliver real-time predictions, ultimately contributing to better investment strategies and financial planning.

XII. FUTURE SCOPE

The project has immense potential for further development. Incorporating advanced machine learning models such as LSTM and deep neural networks can enhance the model's ability to capture complex and non-linear relationships in stock prices. Adding external features like global market trends, economic indicators, market sentiment analysis, and news-based events can significantly improve prediction accuracy. Real-time prediction systems that update stock price forecasts dynamically based on live data and news streams can make the tool more practical and actionable.

Furthermore, integrating risk assessment and volatility analysis into the model can provide users with a more comprehensive understanding of potential risks and returns, helping them make better investment decisions. Expanding the analysis to include predictions for multiple stocks or an entire portfolio will enable a broader financial outlook and portfolio optimization strategies. Implementing automated trading algorithms based on the model’s predictions can provide investors with a competitive edge by executing timely and data-driven trades.

Finally, creating an intuitive, user-friendly application or dashboard can make the tool accessible to a larger audience, ranging from professional traders to novice investors. With these enhancements, the project can evolve into a robust, all-in-one platform for stock market analysis, predictions, and decision-making, significantly contributing to the financial technology domain.

XIII. REFERENCES

- https://towardsdatascience.com/exploratory-data-analysis-in-python-a-step-by-step-process-d0dfa6bf94ee
- https://www.javatpoint.com/python-tutorial
- https://www.geeksforgeeks.org/introduction-machine-learning/
- https://www.geeksforgeeks.org/pandas-tutorial/
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- https://www.w3schools.com/python/numpy/numpy_intro.asp
- https://www.geeksforgeeks.org/what-is-python-scikit-library/
- https://pythonguides.com/matplotlib-in-python/
- https://www.geeksforgeeks.org/python-seaborn-tutorial/
- https://stackoverflow.com/questions/7501947/understanding-pickling-in-python
- https://www.digitalocean.com/community/tutorials/python-pickle-example