GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI - 631209

Approved by AICTE, New Delhi Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT TITLE

STOCK PRICE PREDICTION

COLLEGE CODE:1103

ARJUN S

3rd year, 5th sem

Reg no.:110321104002

sankeramuniyamal@gmail.com

PROJECT TITLE: STOCK PRICE PREDICTION

BUILDING THE PPROJECT

Stock Price Prediction using machine learning is the process of predicting the future value of a stock traded on a stock exchange for reaping profits. With multiple factors involved in predicting stock prices, it is challenging to predict stock prices with high accuracy, and this is where machine learning plays a vital role.

Stock Price Prediction System is a data-driven project aimed at forecasting the future prices of publicly traded stocks. This project leverages advanced machine learning and data analysis techniques to provide insights into stock market behaviour and assist investors, traders, and financial decisions.

This system incorporates several critical components to achieve its objectives. It begins with data collection and preprocessing, where historical stock price data, trading volumes, and external factors such as economic indicators and news sentiment are meticulously curated and cleaned.

GIVEN DATASET:

LINK:

https//www.kaggle.com/datasets/prasoonkottrathil/microsoft-lifetime-stocks-datasets

	Date	Open	High	Low	Close	Adj Close	Volume
0	13-03-1986	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	14-03-1986	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	17-03-1986	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	18-03-1986	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	19-03-1986	0.099826	0.100694	0.097222	0.098090	0.063107	47894400
...	...	...	...	...	...	...	...
8520	31-12-2019	156.770004	157.770004	156.449997	157.699997	157.699997	18369400
8521	02-01-2020	158.779999	160.729996	158.330002	160.619995	160.619995	22622100
8522	03-01-2020	158.320007	159.949997	158.059998	158.619995	158.619995	21116200
8523	06-01-2020	157.080002	159.100006	156.509995	159.029999	159.029999	20813700
8524	07-01-2020	159.320007	159.669998	157.330002	157.580002	157.580002	18017762

8525 rows × 7 columns

PREPROCESSING THE DATA:

Preprocessing the dataset in stock price prediction involves cleaning, normalizing, and transforming the data for better analysis. It helps remove noise and inconsistencies to improve the accuracy of the prediction models.

- **Data Loading**: Begin by loading the dataset into your environment. Depending on the format of the data (e.g., CSV, Excel, SQL database), you'll use different libraries or methods to import the data into a data structure like a DataFrame.
- **Data Collection:** Gather historical stock prices, trading volumes, and relevant financial data. Sources could include financial databases, APIs, or web scraping tools.

- **Data Cleaning:**

  1.Handle Missing Values: Check for missing data and decide how to handle it. You can choose to remove rows with missing values, fill them with the mean, median, or a specific value, or use more advanced imputation methods

  2.Data Validation: Check for data inconsistencies or errors and correct them. This may involve handling outliers or anomalies in the data.

- **Data Pre-processing**: Clean the data, handle missing values, and perform feature engineering. This step might involve normalization, scaling, or transforming the data to make it suitable for the chosen model.Data preprocessing may involve more or fewer steps depending on your dataset and objectives.

- **Data Transformation**:

  1.Feature Selection: Choose relevant features that might affect stock prices, such as historical prices, trading volumes, news sentiment, economic indicators, and company-specific information

  2.Model Selection: Select an appropriate machine learning algorithm such as linear regression, decision trees, random forests, or deep learning models like recurrent neural networks (RNNs) or long short-term memory networks (LSTMs)

- **Training the Model:** Use a portion of the data to train the model, adjusting parameters and hyper parameters to optimize performance. Training a machine learning model involves selecting an appropriate algorithm, preparing your data, and using that data to train the model.

- **Model Evaluation:** Assess the model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on a validation dataset. Model evaluation is a critical step in the development and assessment of machine learning models. It involves assessing the model's performance to determine how well it generalizes to new, unseen data.

- **Testing the Model:** Apply the trained model to a separate test dataset to evaluate its predictive power. Test models are simplified representations of real-world systems used for experimentation and analysis.They help assess the behavior and performance of systems under various conditions, providing insights and aiding decision-making.

- **Iterate and Refine:** Fine-tune the model by iterating on the pre-processing steps, feature selection, and model selection to improve predictive accuracy. The process begins with an initial version, idea, or solution. Instead of seeking perfection from the start, you make incremental improvements through successive cycles or iterations. Each iteration allows for feedback, learning, and adaptation.

  It's important to note that stock market prediction is inherently complex and subject to a variety of external factors, including market sentiment, geopolitical events, and economic indicators, which might not be fully captured by historical data alone. Therefore, it's essential to exercise caution and understand the limitations of any predictive mode

IMPORTANCE OF LOADING AND PREPROCESSING THE DATASET:

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately

NECESSARY STEPS TO FOLLOW:

1.Import Libraries:

Start by importing the necessary libraries:

Code:

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

2. Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically find stock price datasets in CSV format, but you can adapt this code to other formats as needed.

Code:

```
df = pd.read_csv("D:downloads/MFST.csv")
```

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: stock=pd.read_csv("Downloads/MSFT.csv")
        stock
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	13-03-1986	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	14-03-1986	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	17-03-1986	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	18-03-1986	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	19-03-1986	0.099826	0.100694	0.097222	0.098090	0.063107	47894400
...	...	...	...	...	...	...	...
8520	31-12-2019	156.770004	157.770004	156.449997	157.699997	157.699997	18369400
8521	02-01-2020	158.779999	160.729996	158.330002	160.619995	160.619995	22622100
8522	03-01-2020	158.320007	159.949997	158.059998	158.619995	158.619995	21116200
8523	06-01-2020	157.080002	159.100006	156.509995	159.029999	159.029999	20813700
8524	07-01-2020	159.320007	159.669998	157.330002	157.580002	157.580002	18017762

8525 rows × 7 columns

3. Exploratory Data Analysis (EDA): Perform EDA to understand your data better. This includes checking for missing values, exploring the data ' s statistics, and visualizing it to identify patterns.

Code:

```
# Check for missing values
  print(df.isnull().sum())
# Explore statistics
  print(df.describe())
# Visualize the data (e.g., histograms, scatter plots, etc.)
```

4. Split the Data: Split your dataset into training and testing sets. This helps you evaluate your model' s performance later. Thus it can be implemented by using the dataset and extracted for splitting the dataset.

Code:

X = df.drop(' low ' , axis=1) # Features

y = df[' price '] # Target variable

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

5.DISPLAYING THE PARTICULAR ROW:

It can be implemented by performing the particular rows in the dataset can be displayed.

Code:

stock.head()

stock.tail()

stock.info()

```
In [3]: stock.head()
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	13-03-1986	0.088542	0.101563	0.088542	0.097222	0.062549	1031788800
1	14-03-1986	0.097222	0.102431	0.097222	0.100694	0.064783	308160000
2	17-03-1986	0.100694	0.103299	0.100694	0.102431	0.065899	133171200
3	18-03-1986	0.102431	0.103299	0.098958	0.099826	0.064224	67766400
4	19-03-1986	0.099826	0.100694	0.097222	0.098090	0.063107	47894400

```
In [4]: stock.tail()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
8520	31-12-2019	156.770004	157.770004	156.449997	157.699997	157.699997	18369400
8521	02-01-2020	158.779999	160.729996	158.330002	160.619995	160.619995	22622100
8522	03-01-2020	158.320007	159.949997	158.059998	158.619995	158.619995	21116200
8523	06-01-2020	157.080002	159.100006	156.509995	159.029999	159.029999	20813700
8524	07-01-2020	159.320007	159.669998	157.330002	157.580002	157.580002	18017762

```
In [5]: stock.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       8525 non-null   object
 1   Open       8525 non-null   float64
 2   High       8525 non-null   float64
 3   Low        8525 non-null   float64
 4   Close      8525 non-null   float64
 5   Adj Close  8525 non-null   float64
 6   Volume     8525 non-null   int64
dtypes: float64(5), int64(1), object(1)
```

6.TO VIEW SHAPE,CORR:

```
In [6]: stock.shape
Out[6]: (8525, 7)
```

```
In [7]: stock.describe()
Out[7]:
```

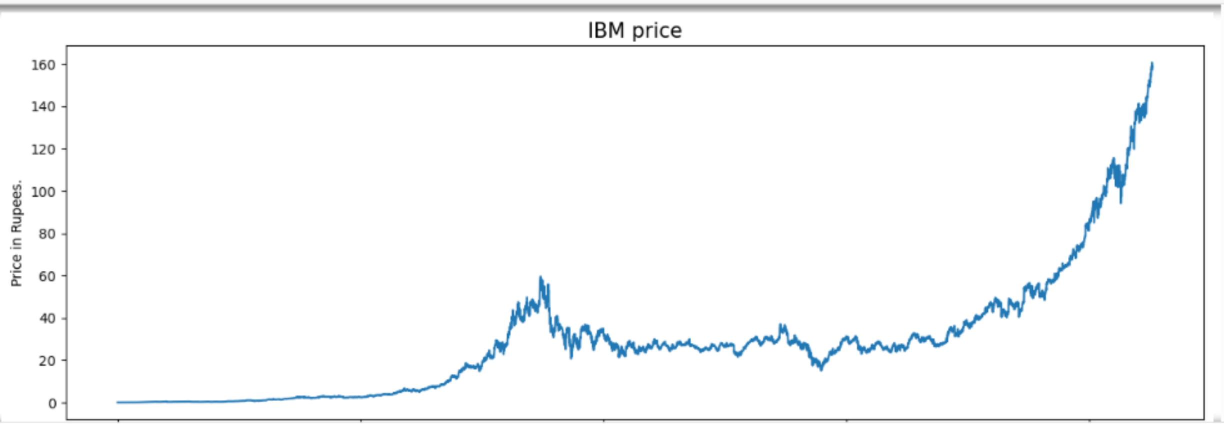
	Open	High	Low	Close	Adj Close	Volume
count	8525.000000	8525.000000	8525.000000	8525.000000	8525.000000	8.525000e+03
mean	28.220247	28.514473	27.918967	28.224480	23.417934	6.045692e+07
std	28.626752	28.848988	28.370344	28.626571	28.195330	3.891225e+07
min	0.088542	0.092014	0.088542	0.090278	0.058081	2.304000e+06
25%	3.414063	3.460938	3.382813	3.414063	2.196463	3.667960e+07
50%	26.174999	26.500000	25.889999	26.160000	18.441576	5.370240e+07
75%	34.230000	34.669998	33.750000	34.230000	25.392508	7.412350e+07
max	159.449997	160.729996	158.330002	160.619995	160.619995	1.031789e+09

```
In [8]: stock.corr()
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\2678749480.py:1: FutureWarning: The default value of num
me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify
_only to silence this warning.
  stock.corr()
Out[8]:
```

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.999921	0.999902	0.999825	0.989637	-0.319446
High	0.999921	1.000000	0.999868	0.999908	0.989255	-0.317238
Low	0.999902	0.999868	1.000000	0.999920	0.990123	-0.321940
Close	0.999825	0.999908	0.999920	1.000000	0.989804	-0.319720
Adj Close	0.989637	0.989255	0.990123	0.989804	1.000000	-0.333682
Volume	-0.319446	-0.317238	-0.321940	-0.319720	-0.333682	1.000000

7. CALCUALTING THE PRICE AND SUM:

```
In [12]: plt.figure(figsize=(15,5))
         plt.plot(stock['Close'])
         plt.title('IBM price', fontsize=15)
         plt.ylabel('Price in Rupees.')
         plt.show()
```




```
In [13]: stock.isnull().sum()
Out[13]: Date         0
         Open         0
         High         0
         Low          0
         Close        0
         Adj Close    0
         Volume       0
         dtype: int64
```

8. REGRESSION AND ACCURACY:

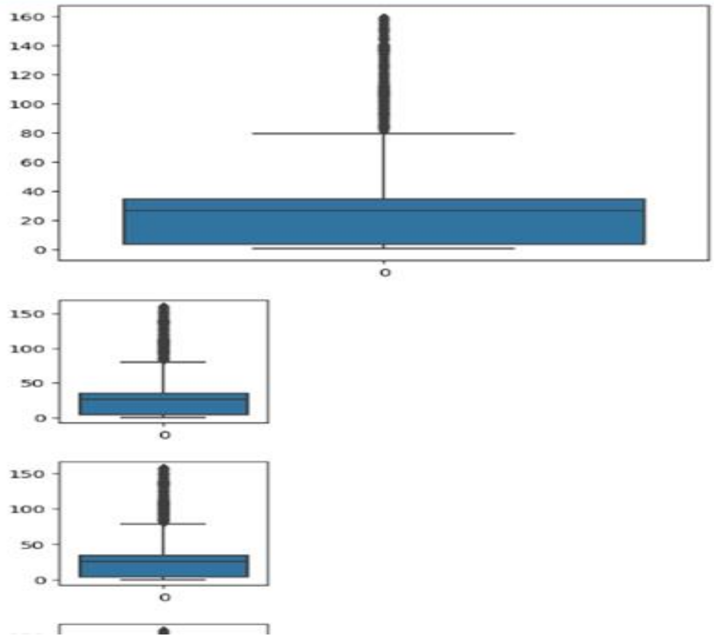
In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np
np.random.seed(0)
data = {
    'Exam1': np.random.rand(100) * 100,
    'Exam2': np.random.rand(100) * 100,
    'Admitted': np.random.randint(2, size=100)
}
df = pd.DataFrame(data)
print(df)
X = df[['Exam1', 'Exam2']]
y = df['Admitted']
print(X)
print(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("-----------------")
print(y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
        Exam1      Exam2  Admitted
0   54.881350  67.781654         0
1   71.518937  27.000797         1
2   60.276338  73.519402         0
3   54.488318  96.218855         0
4   42.365480  24.875314         1
..        ...        ...       ...
95  18.319136  49.045881         0
96  58.651293  22.741463         1
97   2.010755  25.435648         0
98  82.894003   5.802916         1
99   0.469548  43.441663         0

[100 rows x 3 columns]
        Exam1      Exam2
0   54.881350  67.781654
1   71.518937  27.000797
2   60.276338  73.519402
3   54.488318  96.218855
4   42.365480  24.875314
..        ...        ...
95  18.319136  49.045881
96  58.651293  22.741463
97   2.010755  25.435648
98  82.894003   5.802916
99   0.469548  43.441663

[100 rows x 2 columns]
0     0
1     1
2     0
3     0
4     1
     ..
95    0
96    1
97    0
98    1
99    0
Name: Admitted, Length: 100, dtype: int32
-----------------
[1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1]
Accuracy: 0.45
              precision    recall  f1-score   support

           0       0.27      0.50      0.35         6
           1       0.67      0.43      0.52        14

    accuracy                           0.45        20
   macro avg       0.47      0.46      0.44        20
weighted avg       0.55      0.45      0.47        20

[[3 3]
 [8 6]]
```

9.VIEW AS CHART:

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

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\1085974214.py:3: MatplotlibDeprecationWarning: Auto-removal of overlapping a
xes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(2,3,i+1)
```



In [26]:
```
plt.figure(figsize=(10, 10))
sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
plt.show()
```

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\4185554148.py:2: FutureWarning: The default value of numeric_only in DataFra
me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
```