


MINI PROJECT of Machine Learning

Use the following dataset to analyze ups and downs in the market and predict future stock price returns based on Indian Market data from 2000 to 2020. Dataset Link:

<https://www.kaggle.com/datasets/sagara9595/stock-data>

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Step 1: Load the Dataset
data = pd.read_csv("/content/20MICRONS.BO.csv")
# Display the first few rows of the dataset
data.head()
```



	Date	Open	High	Low	Close	Adj Close	Volume
0	2008-10-06	25.000	48.125	15.750	16.650	13.664763	25826400
1	2008-10-07	16.850	18.975	13.825	15.200	12.474738	8742400
2	2008-10-08	13.775	14.475	12.625	13.175	10.812808	2941200
3	2008-10-10	11.000	12.450	10.750	11.675	9.581747	1362600
4	2008-10-13	12.000	13.350	11.925	12.475	10.238314	878000

```
data.describe()
```




	Open	High	Low	Close	Adj Close	Volume
count	2856.000000	2856.000000	2856.000000	2856.000000	2856.000000	2.856000e+03
mean	33.226975	33.976495	32.326380	32.972470	30.807002	1.210241e+05
std	12.330555	12.647315	12.007803	12.254755	12.145972	5.842101e+05
min	6.625000	7.275000	6.625000	7.025000	5.765463	0.000000e+00
25%	27.000000	27.500000	26.000000	26.750000	24.357159	1.040375e+04
50%	32.000000	32.599998	31.150000	31.750000	30.158554	3.121850e+04
75%	39.000000	39.700001	37.950001	38.549999	36.843975	9.288100e+04
max	82.474008	86.100007	79.250000	81.074008	76.841072	2.582640e+07



```
data.info()
```

```
data.columns
```



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

```
# Step 2: Preprocess the Data
# Convert Date column to datetime format
data['Date'] = pd.to_datetime(data['Date'])

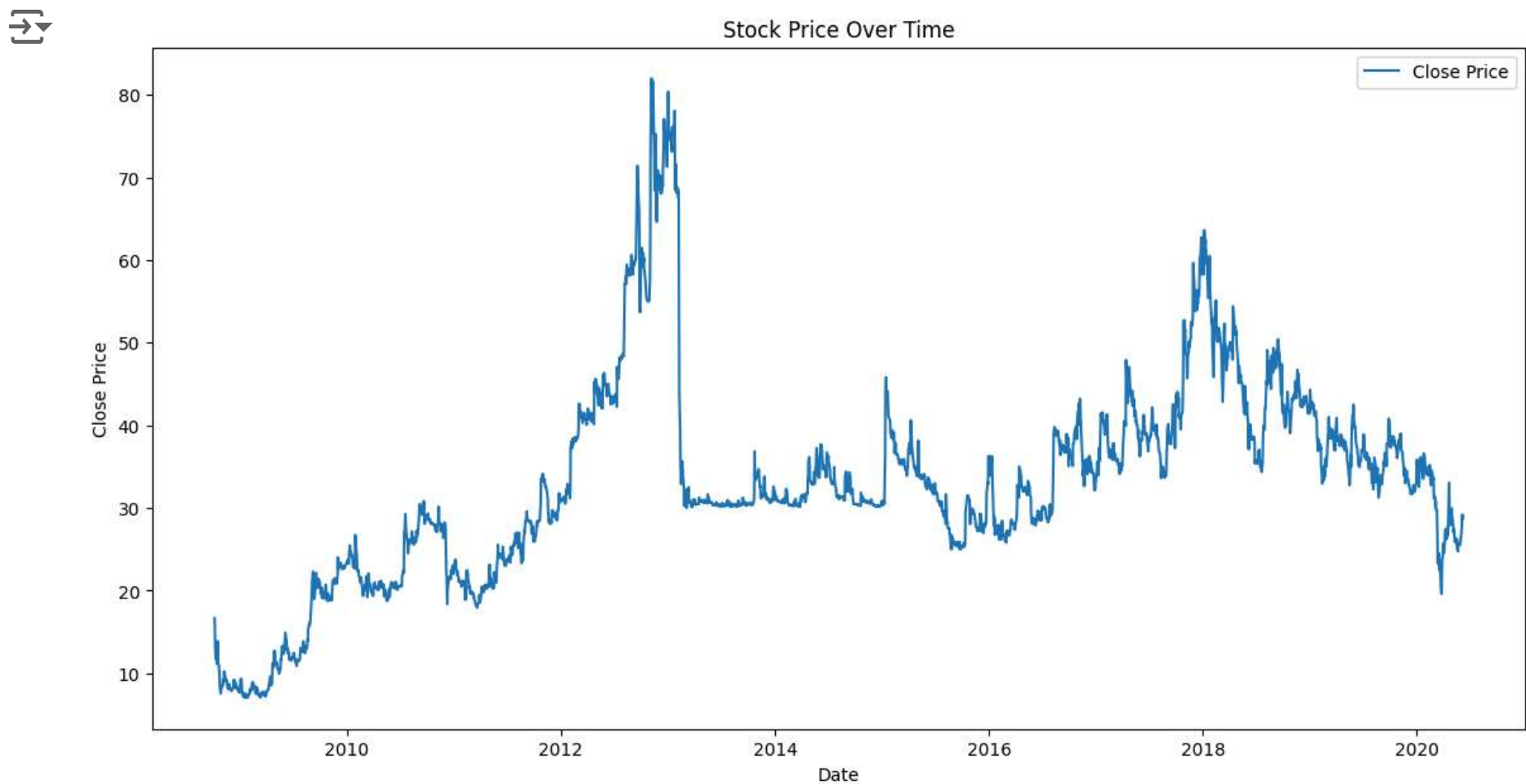
# Sort the data by Date
data.sort_values('Date', inplace=True)

# Check for missing values
print(data.isnull().sum())
```

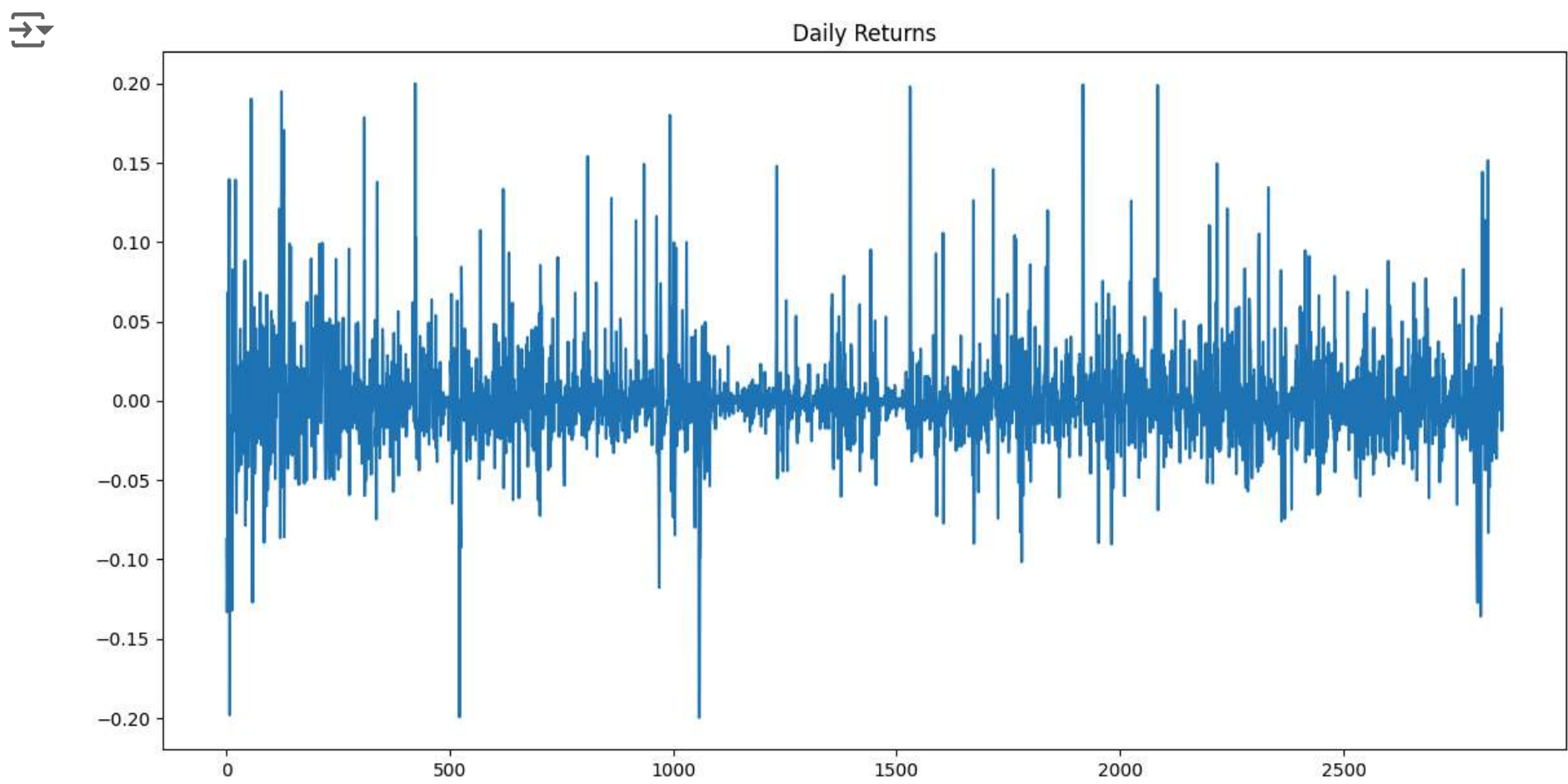
⇒

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

```
# Step 3: Exploratory Data Analysis (EDA)
plt.figure(figsize=(14, 7))
plt.plot(data['Date'], data['Close'], label='Close Price')
plt.title('Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



```
# Calculate daily returns
data['Return'] = data['Close'].pct_change()
data['Return'].dropna().plot(figsize=(14, 7), title='Daily Returns')
plt.show()
```



```
# Step 4: Feature Engineering
# Create lagged features
data['Lag1'] = data['Return'].shift(1)
data['Lag2'] = data['Return'].shift(2)

# Drop NaN values created by lagging
data.dropna(inplace=True)

# Step 5: Split the Data
X = data[['Lag1', 'Lag2']]
y = data['Return']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Step 6: Train a Predictive Model
model = LinearRegression()
model.fit(X_train, y_train)
```

LinearRegression ⓘ ?

LinearRegression()

```
# Step 7: Evaluate the Model
y_pred = model.predict(X_test)

# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.4f}')
print(f'R² Score: {r2:.4f}')
```

⇒ Mean Squared Error: 0.0010
R² Score: -0.0151

```
# Plot predictions vs actual returns
plt.figure(figsize=(14, 7))
plt.plot(y_test.index, y_test, label='Actual Returns', color='blue')
plt.plot(y_test.index, y_pred, label='Predicted Returns', color='red')
plt.title('Actual vs Predicted Returns')
plt.xlabel('Index')
plt.ylabel('Returns')
plt.legend()
plt.show()
```

