Bajaj Auto Stock Market Machine Learning Data Analysis

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.offline import plot
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
```

In [2]:
```
# Load the dataset
bajaj_auto = pd.read_csv('C:\\Users\\windows 10 pro\\Documents\\archive (36)\\BAJAJ-AUTO
bajaj_auto.head()
```

Out[2]:

	Date	Symbol	Series	Prev_Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover
0	5/26/2008	BAJAJ-AUTO	EQ	2101.05	898.00	898.0	551.35	600.25	604.75	624.61	3972485	2.481240e+14
1	5/27/2008	BAJAJ-AUTO	EQ	604.75	624.70	639.0	580.30	595.50	593.15	606.43	1751063	1.061900e+14
2	5/28/2008	BAJAJ-AUTO	EQ	593.15	561.65	621.9	561.65	605.10	608.15	608.75	1652355	1.005880e+14
3	5/29/2008	BAJAJ-AUTO	EQ	608.15	619.40	619.4	576.00	600.00	599.45	600.98	669269	4.022170e+13
4	5/30/2008	BAJAJ-AUTO	EQ	599.45	605.40	607.0	538.00	576.25	571.70	565.55	1262117	7.137940e+13

In [3]:
```
bajaj_auto.describe()
```

Out[3]:

	Prev_Close	Open	High	Low	Last	Close	VWAP	Volum
count	3202.000000	3202.000000	3202.000000	3202.000000	3202.000000	3202.000000	3202.000000	3.202000e+0
mean	2189.871065	2191.537883	2219.934510	2162.215209	2190.307917	2190.412196	2191.294288	4.114639e+0
std	774.552766	776.148452	781.289529	770.192017	775.065896	775.095766	775.657353	3.911067e+0
min	301.900000	262.000000	307.050000	262.000000	300.000000	301.900000	301.980000	4.966000e+0
25%	1661.925000	1663.250000	1693.775000	1636.362500	1660.512500	1661.925000	1664.845000	2.054658e+0
50%	2269.950000	2270.000000	2305.575000	2240.250000	2270.750000	2270.225000	2269.900000	3.125365e+0
75%	2808.437500	2810.000000	2834.950000	2778.362500	2808.975000	2808.487500	2808.565000	5.000315e+0
max	4237.450000	4260.000000	4361.400000	4200.000000	4236.000000	4237.450000	4260.500000	8.537143e+0

In [4]:
```
bajaj_auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3202 entries, 0 to 3201
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  3202 non-null   object 
 1   Symbol                3202 non-null   object 
 2   Series                3202 non-null   object 
 3   Prev_Close            3202 non-null   float64
 4   Open                  3202 non-null   float64
 5   High                  3202 non-null   float64
 6   Low                   3202 non-null   float64
 7   Last                  3202 non-null   float64
 8   Close                 3202 non-null   float64
 9   VWAP                  3202 non-null   float64
 10  Volume                3202 non-null   int64  
 11  Turnover              3202 non-null   float64
 12  Trades                2456 non-null   float64
 13  Deliverable_Volume    3202 non-null   int64  
 14  Percentage_Deliverble  3202 non-null   float64
dtypes: float64(10), int64(2), object(3)
memory usage: 375.4+ KB
```

In [7]:
```
bajaj_auto.isna().any()
```

Out[7]:
```
Date                   False
Symbol                 False
Series                 False
Prev_Close             False
Open                   False
High                   False
Low                    False
Last                   False
Close                  False
VWAP                   False
Volume                 False
Turnover               False
Trades                  True
Deliverable_Volume     False
Percentage_Deliverble  False
dtype: bool
```

In [8]:
```
bajaj_auto['Date'] = pd.to_datetime(bajaj_auto['Date'])
```

In [9]:
```
print(f'Dataframe contains stock prices between {bajaj_auto.Date.min()} {bajaj_auto.Date
print(f'Total day = {(bajaj_auto.Date.max() - bajaj_auto.Date.min()).days} days')
```

```
Dataframe contains stock prices between 2008-05-26 00:00:00 2021-04-30 00:00:00
Total day = 4722 days
```

In [10]:
```
closing_trending = bajaj_auto.groupby('Date')[['Close']].sum()
closing_trending
```

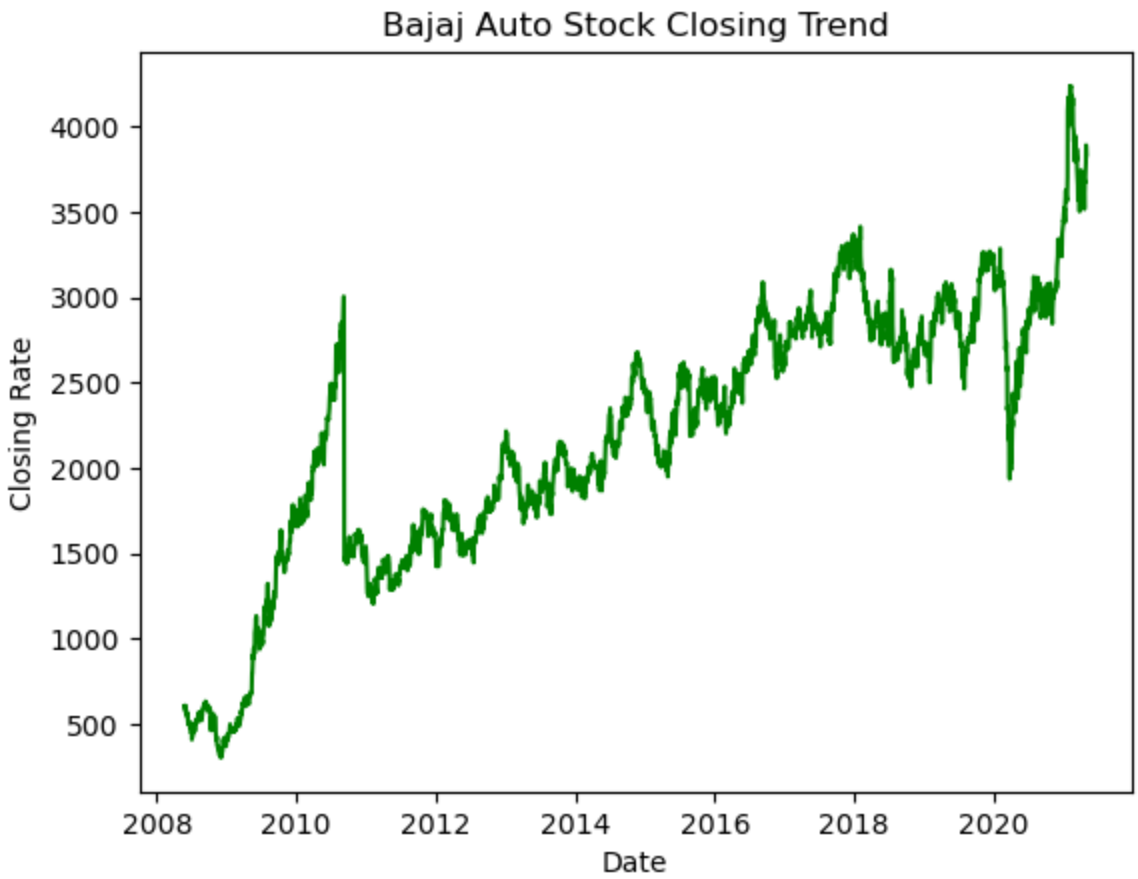
Out[10]:

Date	Close
2008-05-26	604.75
2008-05-27	593.15
2008-05-28	608.15
2008-05-29	599.45
2008-05-30	571.70
...	...
2021-04-26	3738.85
2021-04-27	3785.50
2021-04-28	3889.75
2021-04-29	3836.45
2021-04-30	3833.75

3202 rows × 1 columns

In [11]:
```
plt.plot(closing_trending.index, closing_trending.Close, color='green')
plt.xlabel('Date')
plt.ylabel('Closing Rate')
plt.title('Bajaj Auto Stock Closing Trend')
```

Out[11]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Closing Trend')
```



In [13]:
```
volume_trending = bajaj_auto.groupby('Date')[['Volume']].sum()
volume_trending
```

Out[13]:

Date	Volume
2008-05-26	3972485
2008-05-27	1751063
2008-05-28	1652355
2008-05-29	669269
2008-05-30	1262117
...	...
2021-04-26	480016
2021-04-27	518487
2021-04-28	1252958
2021-04-29	1335444
2021-04-30	1340273

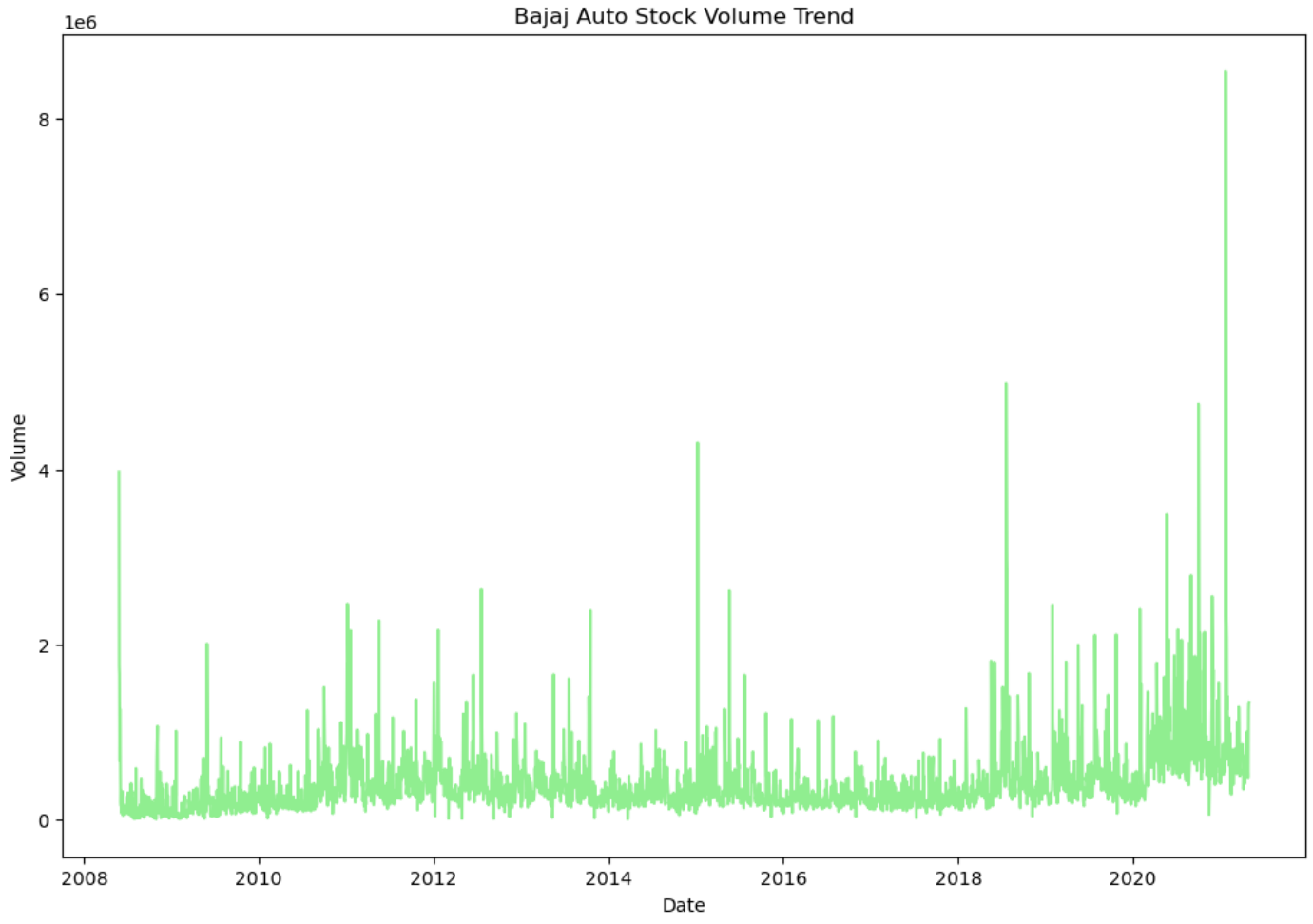
3202 rows × 1 columns

In [118…

```
plt.plot(volume_trending.index, volume_trending.Volume, color='lightgreen')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.title('Bajaj Auto Stock Volume Trend')
```

Out[118]:

```
Text(0.5, 1.0, 'Bajaj Auto Stock Volume Trend')
```



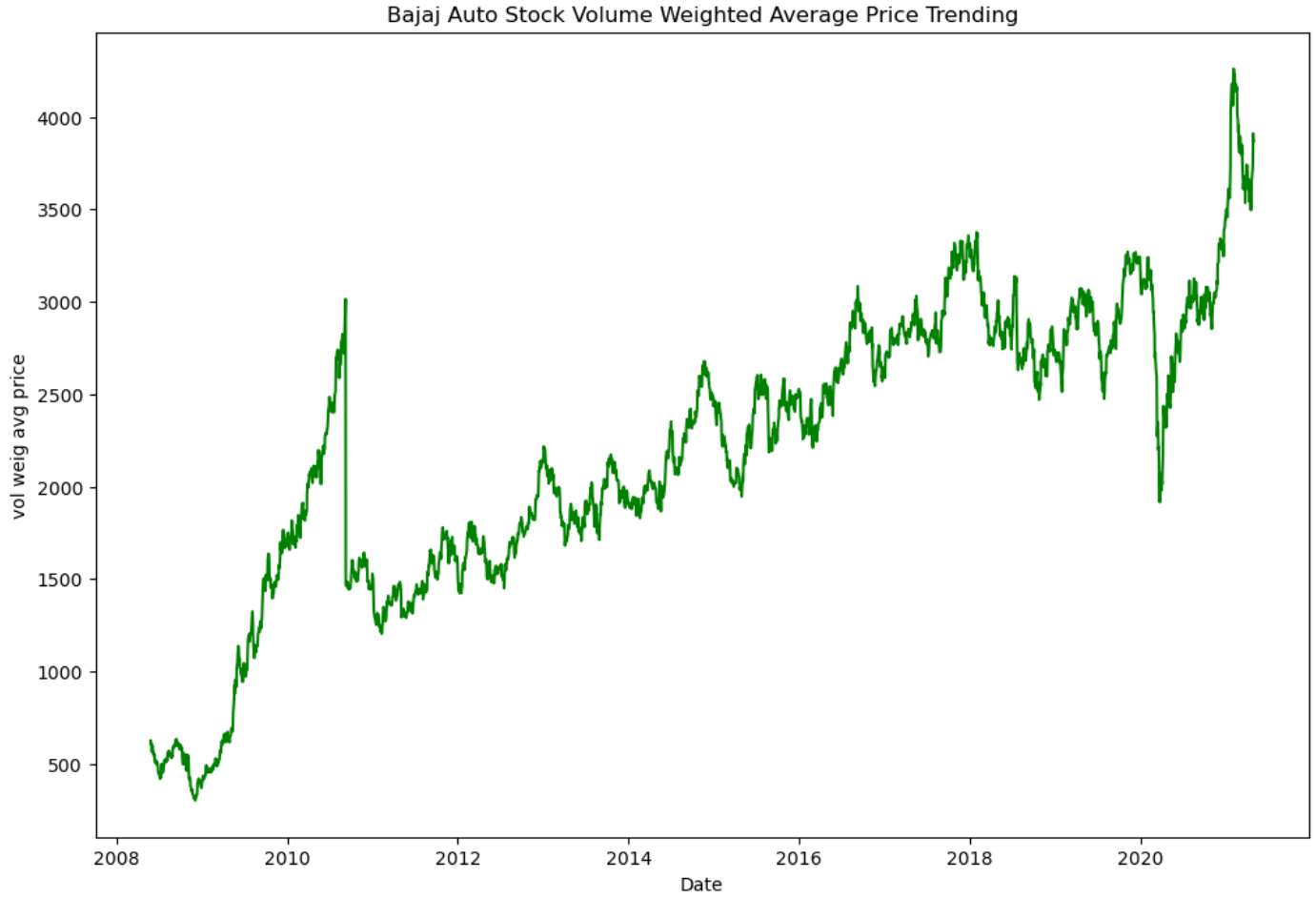
In [40]:

```
volume_weighted_avg_price_trending = bajaj_auto.groupby('Date')[['VWAP']].sum()
ding = bajaj_auto.groupby('Date')[['Turnover']].sum()
```

```
Trades_trending = bajaj_auto.groupby('Date')[['Trades']].sum()
deliverable_Volume_trending = bajaj_auto.groupby('Date')[['Deliverable_Volume']].sum()
percentage_deliverable_trending = bajaj_auto.groupby('Date')[['Percentage_Deliverble']].
```

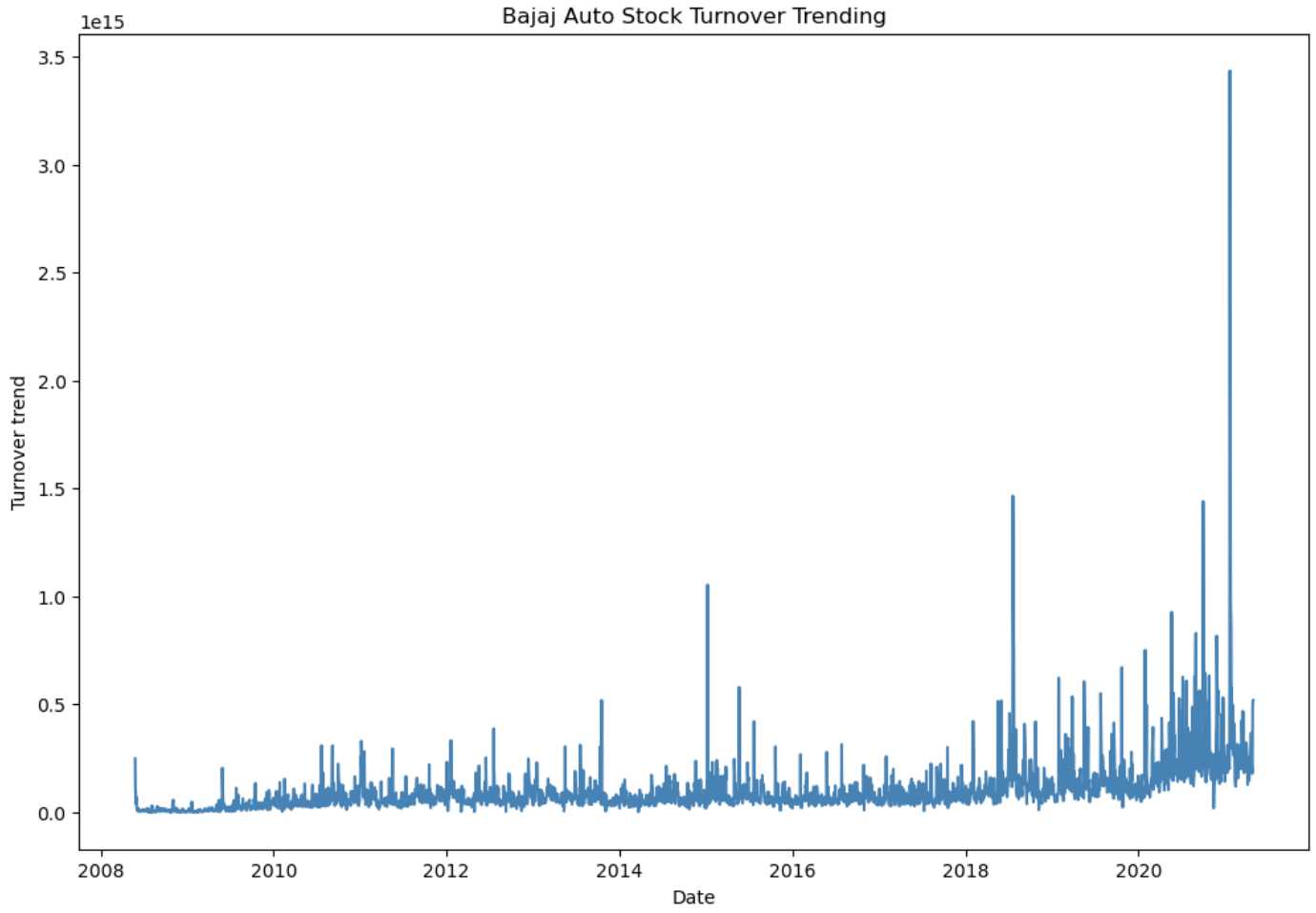
In [113…
```
plt.plot(volume_weighted_avg_price_trending.index, volume_weighted_avg_price_trending.VW
plt.xlabel('Date')
plt.ylabel('vol weig avg price')
plt.title('Bajaj Auto Stock Volume Weighted Average Price Trending')
```

Out[113]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Volume Weighted Average Price Trending')
```



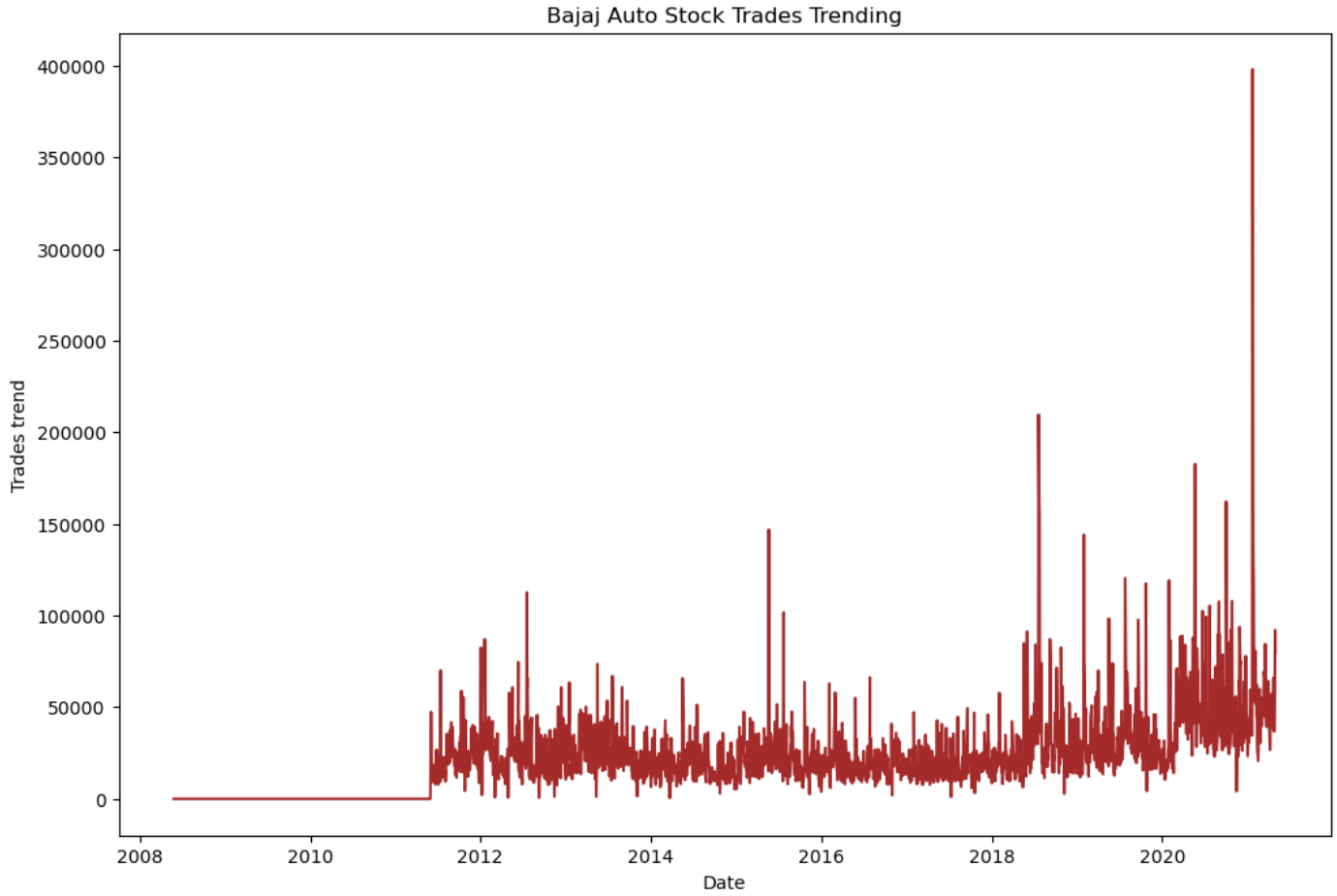
In [34]:
```
plt.plot(turnover_trending.index, turnover_trending.Turnover, color='steelblue')
plt.xlabel('Date')
plt.ylabel('Turnover trend')
plt.title('Bajaj Auto Stock Turnover Trending')
```

Out[34]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Turnover Trending')
```



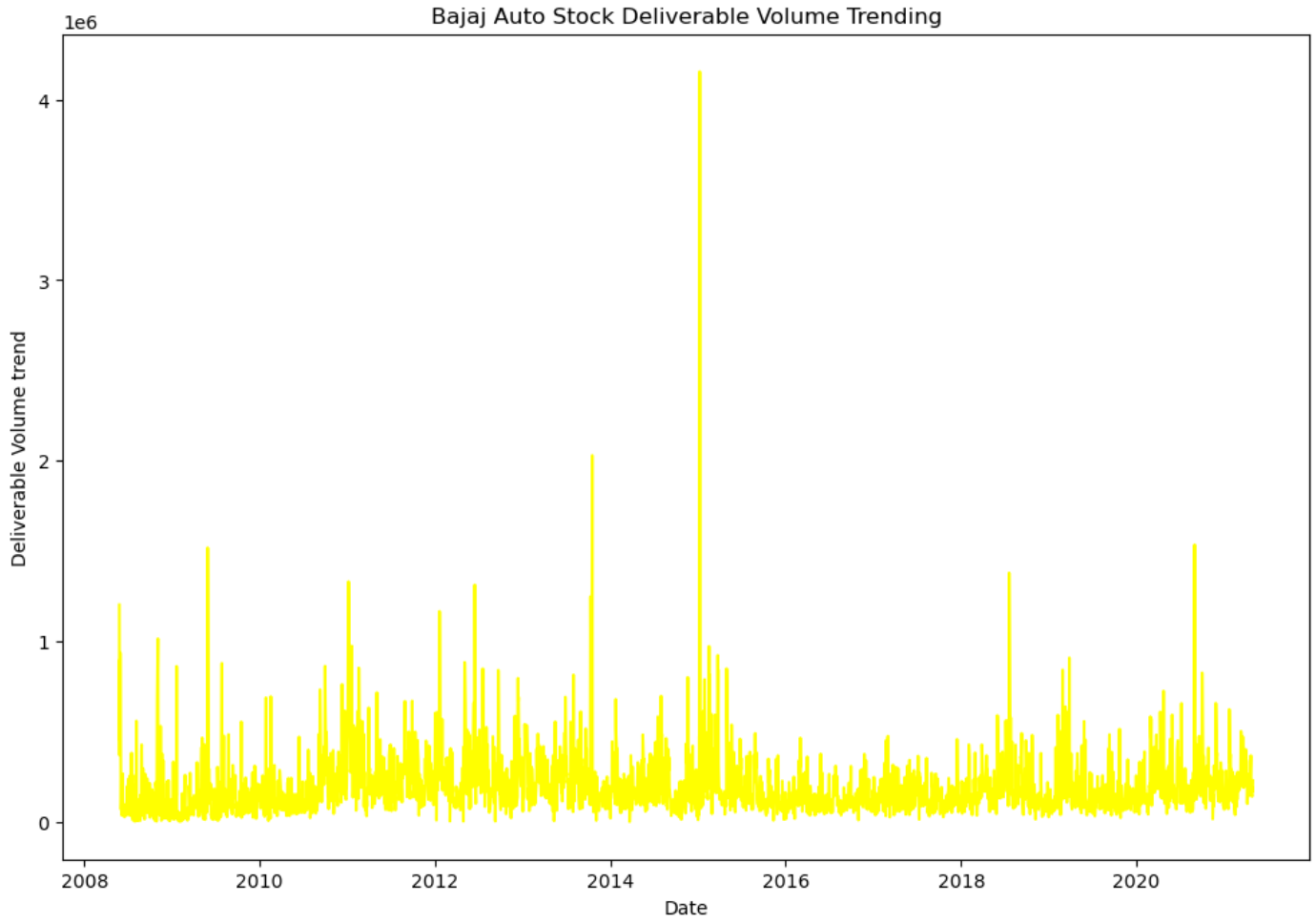
In [37]:
```
plt.plot(Trades_trending.index, Trades_trending.Trades, color='brown')
plt.xlabel('Date')
plt.ylabel('Trades trend')
plt.title('Bajaj Auto Stock Trades Trending')
```

Out[37]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Trades Trending')
```



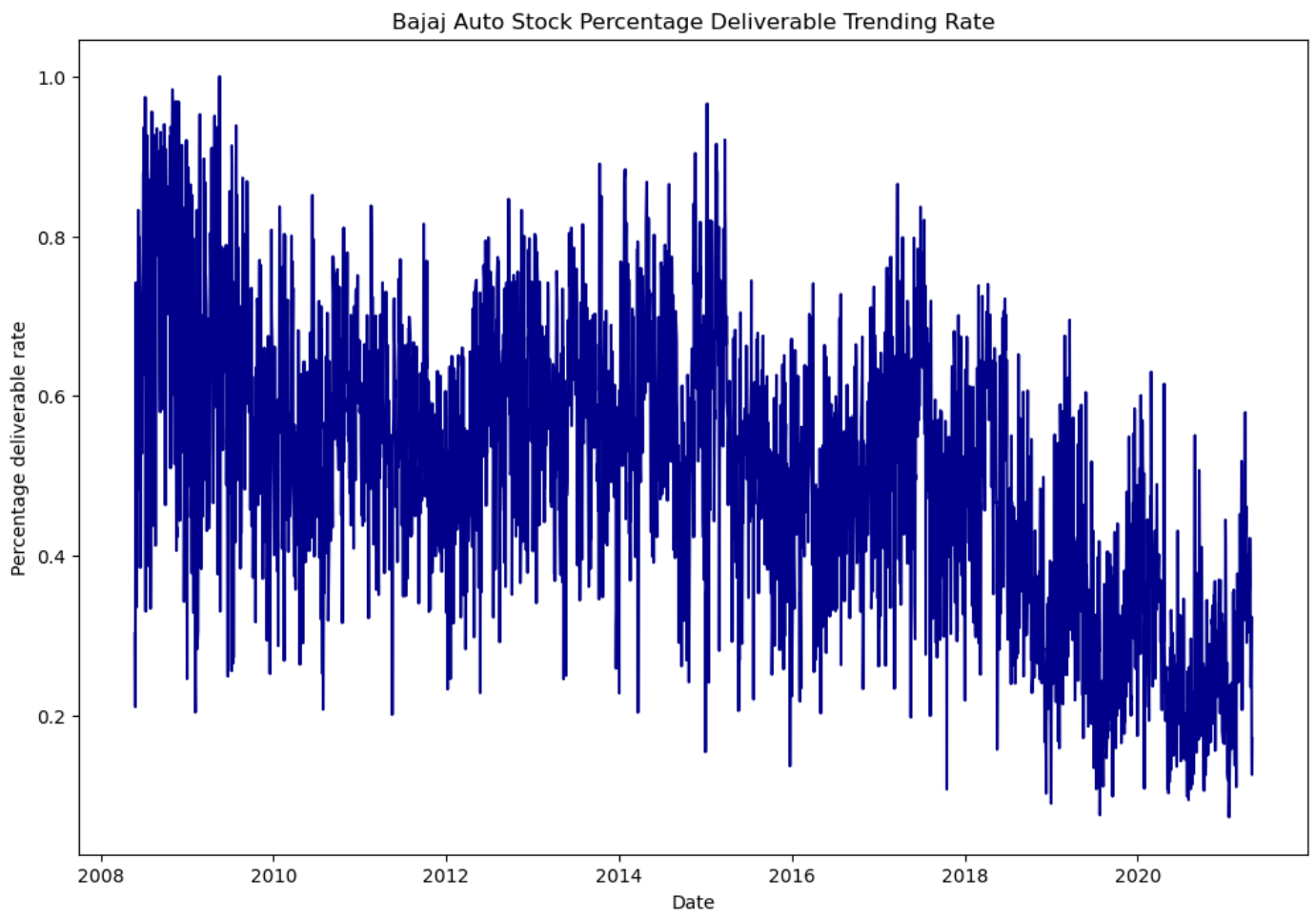
In [42]:
```
plt.plot(deliverable_Volume_trending.index, deliverable_Volume_trending.Deliverable_Volu
plt.xlabel('Date')
plt.ylabel('Deliverable Volume trend')
plt.title('Bajaj Auto Stock Deliverable Volume Trending')
```

Out[42]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Deliverable Volume Trending')
```



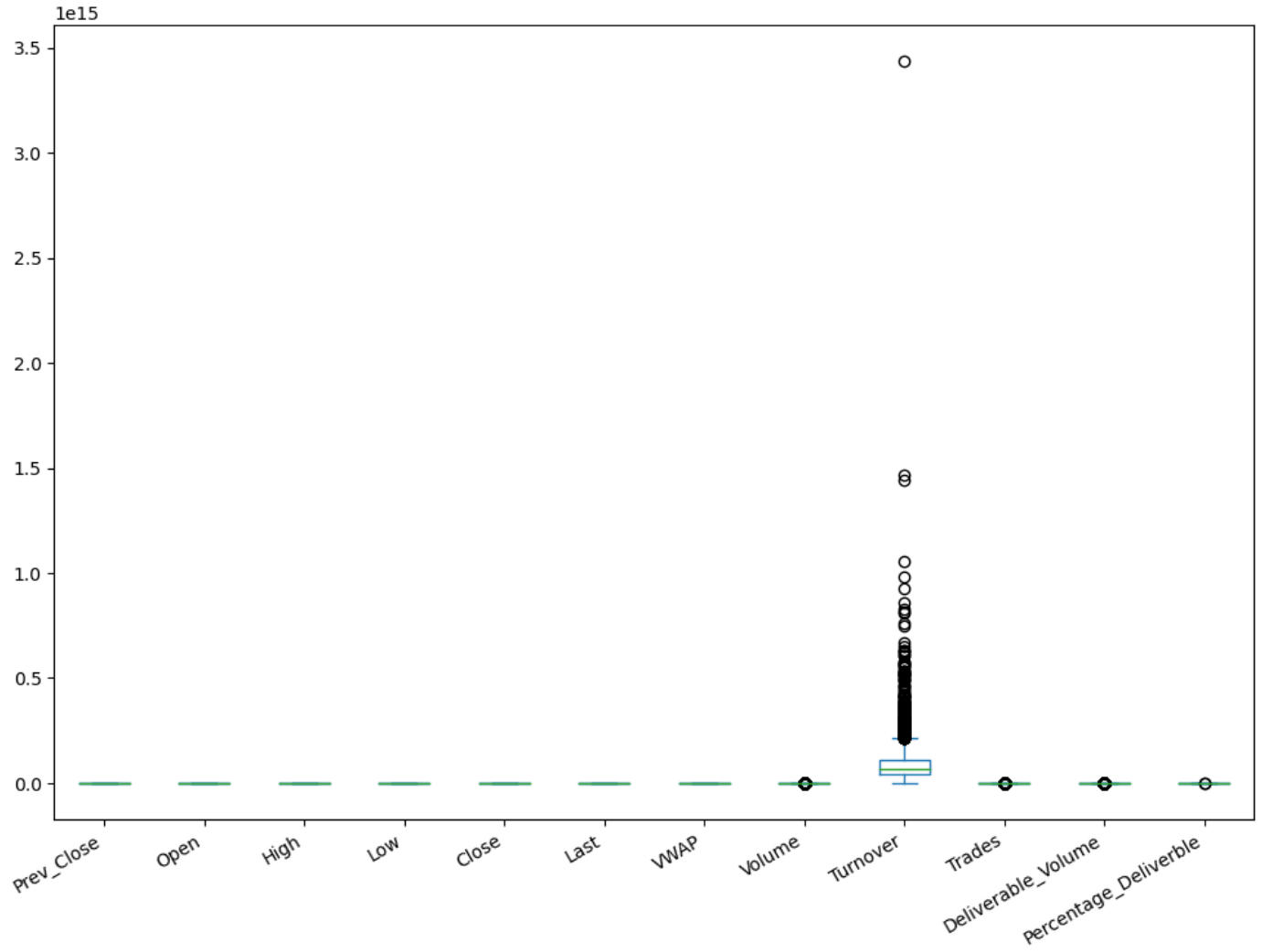
In [117…
```
plt.plot(percentage_deliverable_trending.index, percentage_deliverable_trending.Percenta
plt.xlabel('Date')
plt.ylabel('Percentage deliverable rate')
plt.title('Bajaj Auto Stock Percentage Deliverable Trending Rate')
```

Out[117]:
```
Text(0.5, 1.0, 'Bajaj Auto Stock Percentage Deliverable Trending Rate')
```



In [44]:
```
bajaj_auto[['Prev_Close','Open', 'High', 'Low', 'Close', 'Last', 'VWAP', 'Volume', 'Turn
plt.xticks(rotation = 30, ha = 'right')
```

Out[44]:
```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 [Text(1, 0, 'Prev_Close'),
  Text(2, 0, 'Open'),
  Text(3, 0, 'High'),
  Text(4, 0, 'Low'),
  Text(5, 0, 'Close'),
  Text(6, 0, 'Last'),
  Text(7, 0, 'VWAP'),
  Text(8, 0, 'Volume'),
  Text(9, 0, 'Turnover'),
  Text(10, 0, 'Trades'),
  Text(11, 0, 'Deliverable_Volume'),
  Text(12, 0, 'Percentage_Deliverble')])
```



In [5]:
```
# extracting colume for the model
stock_df = bajaj_auto.iloc[:, [0,3,4,5,6,7,8,9,10]]
X = np.array(stock_df.index).reshape(-1,1)
y = stock_df['Close']

# Splitting the dataset into the Traning set Test set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=10
```

In [6]:
```
# Feature scaling
scaler = StandardScaler().fit(X_train)
```

In [7]:
```
# creating a linear model
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[7]:
```
▼ LinearRegression

LinearRegression()
```

In [8]:
```
# Make prediction on the test data
y_pred = lm.predict(X_test)
```

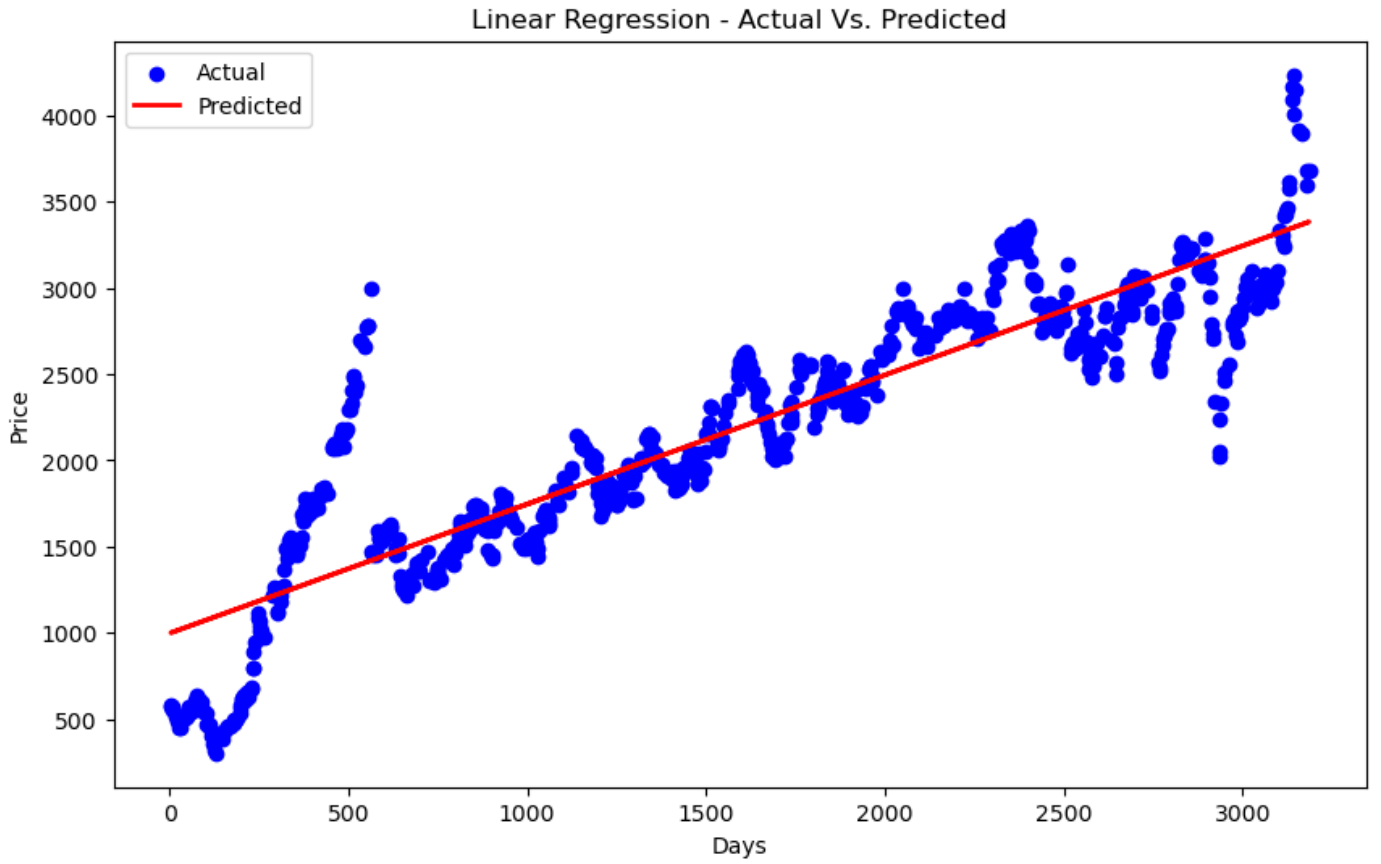
In [9]:
```
# Calculate score for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(y_train, lm.predict(X_train))}\t{r2_score(y_test, lm.pre
{'Mse' .ljust(10)}{mse(y_train, lm.predict(X_train))}\t{mse(y_test, lm.predict(X_test))}
```

```
'''
print(scores)
```

```
Metric          Train               Test
r2_score  0.798916093187329     0.8080277725978103
Mse       120379.8639164733     116181.66157047136
```

Plot Linear Regression Actual Vs Predicted values for train dataset

In [19]:
```
plt.figure(figsize=(10,6))
plt.scatter(X_test, y_test, color='b', label='Actual')
plt.plot(X_test, y_pred, color='r', linewidth=2, label='Predicted')
plt.title('Linear Regression - Actual Vs. Predicted')
plt.xlabel('Days')
plt.ylabel('Price')
plt.legend()
plt.show()
```



In [ ]: