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


df = pd.read_csv("/content/NFLX1.csv")
df1 = df.copy()
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-	267.079987	267.619995	250.000000	250.100006	250.100006	9206700

Next steps: Generate code with df | View recommended plots

```
df.shape
```

```
(1009, 7)
```

```
df.dtypes
```

```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

```
df.duplicated().sum()
```

```
0
```

```
df.isna().sum()
```

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

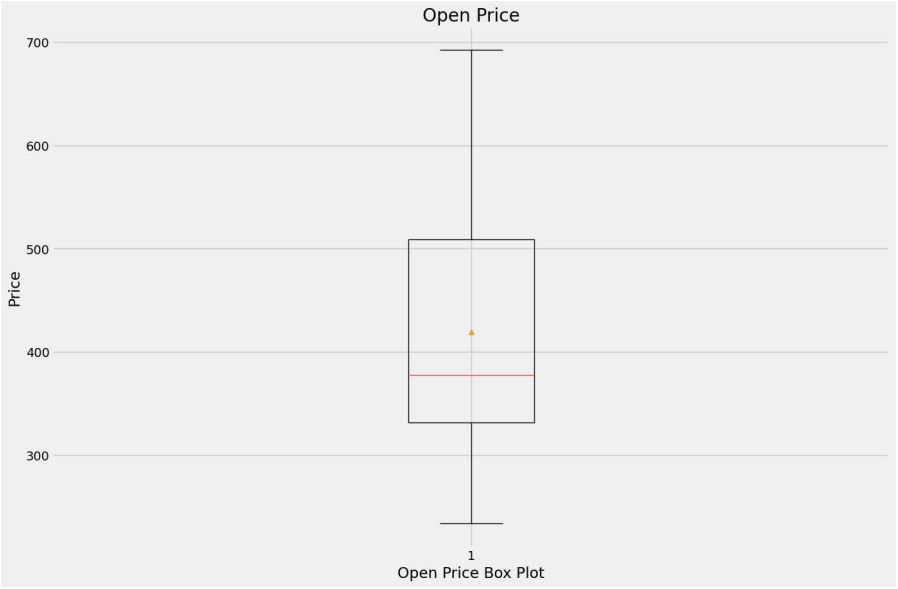
```
df.nunique()
```

```
Date         1009
Open          976
High          983
Low           989
Close         988
Adj Close     988
Volume       1005
dtype: int64
```

```
plt.style.use('fivethirtyeight')
```

```
plt.subplots(figsize=(15, 10))
plt.title("Open Price")
plt.boxplot(df['Open'], showmeans=True)
plt.xlabel("Open Price Box Plot")
```

```
plt.ylabel("Price")
plt.show()
```




```
print("Mean price is :", statistics.mean(df['Open']))
print("Median price is :", statistics.median(df['Open']))
```

```
Mean price is : 419.05967286223984
Median price is : 377.769989
```

```
plt.subplots(figsize=(25, 8))
plt.title("Open Price vs Close Price")
plt.plot(df['Open'], color='red', linestyle='solid',  label = 'Open Price')
plt.plot(df['Close'], color='green', linestyle='dashed',  label = 'Close Price')
plt.xlabel("Date")
plt.ylabel("Open vs Close Price")
plt.legend(loc="upper left")
plt.show()
```


```
from sklearn.preprocessing import StandardScaler


df['Date']=pd.to_datetime(df['Date'],format='%d-%m-%Y')

# set date to index
df = df.set_index('Date')



train = df.loc['2018-02-05':'2018-12-31']
test = df.loc['2021-01-01':'2021-01-31']



X_train = train.drop(columns = ['Open'])
y_train = train['Open']
# split testing data
X_test = test.drop(columns = ['Open'])
y_test = test['Open']



from sklearn.ensemble import RandomForestRegressor



rf = RandomForestRegressor(max_depth=20, random_state = 42,  n_estimators=150)
rf.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=20, n_estimators=150, random_state=42)
```

```
rf_train_score = rf.score(X_train, y_train)
rf_test_score = rf.score(X_test, y_test)
print(rf_train_score)
print(rf_test_score)
```

```
0.9979090389971715
-15.961320806912266
```

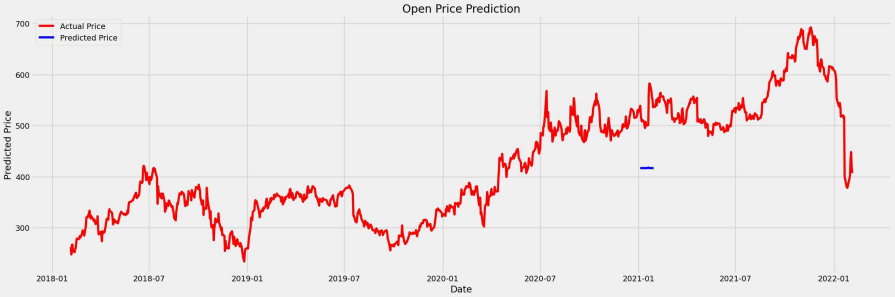
```
pred = rf.predict(X_test)
train_pred = rf.predict(X_train)


prediction_df = X_test.copy()
prediction_df['Open'] = y_test
prediction_df['Predicted Price'] = pred
prediction_df.head()
```

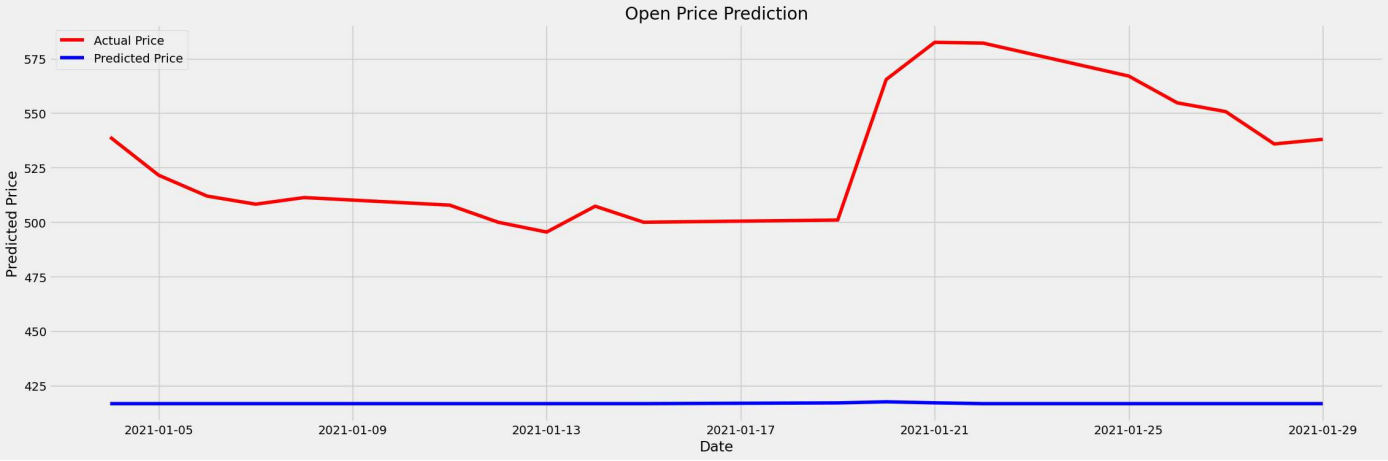
Date	High	Low	Close	Adj Close	Volume	Open	Predicted Price
2021-01-04	540.799988	515.090027	522.859985	522.859985	4444400	539.000000	416.850669
2021-01-05	526.780029	515.890015	520.799988	520.799988	3133900	521.549988	416.850669
2021-01-06	513.099976	499.500000	500.489990	500.489990	5346200	511.970001	416.850669

Next steps: Generate code with prediction_df | View recommended plots

```
plt.subplots(figsize=(25, 8))
plt.title("Open Price Prediction")
#plt.plot(prediction_df['Open'], color='red', linestyle='solid')
plt.plot(df['Open'], color='red', linestyle='solid', label = 'Actual Price')
plt.plot(prediction_df['Predicted Price'], color='blue', linestyle='solid', label = 'Predicted Price')
plt.xlabel("Date")
plt.ylabel("Predicted Price")
plt.legend(loc="upper left")
plt.show()
```




```
plt.subplots(figsize=(25, 8))
plt.title("Open Price Prediction")
plt.plot(prediction_df['Open'], color='red', linestyle='solid',  label = 'Actual Price')
plt.plot(prediction_df['Predicted Price'], color='blue', linestyle='solid', label = 'Predicted Price')
plt.xlabel("Date")
plt.ylabel("Predicted Price")
plt.legend(loc="upper left")
plt.show()
```




```
from sklearn import metrics
```

```
from sklearn import metrics

print("Mean Absolute Error:",round(metrics.mean_absolute_error(y_test, pred), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, pred), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test, pred)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, pred), 4))
print(f'Train Score : {rf.score(X_train, y_train) * 100:.2f}% and Test Score : {rf.score(X_test, y_test) * 100:.2f}% using Random Tree.')
errors = abs(pred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 113.5985
Mean Squared Error: 13708.832
Root Mean Squared Error: 117.0847
(R^2) Score: -15.9613
Train Score : 99.79% and Test Score : -1596.13% using Random Tree.
Accuracy: 78.81 %.
```