Import Data

In [20]:

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

In [21]:

```
df=pd.read_csv("C:/Users/Dell/Downloads/NETflix.csv")
df1 = df.copy()
df.head()
```

Out[21]:

	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	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [22]:

```
df.shape
```

Out[22]:

```
(1009, 7)
```

In [23]:

```
df.dtypes
```

Out[23]:

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

In [24]:

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

Out[24]:

```
0
```

In [46]:

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

Out[46]:

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

In [45]:

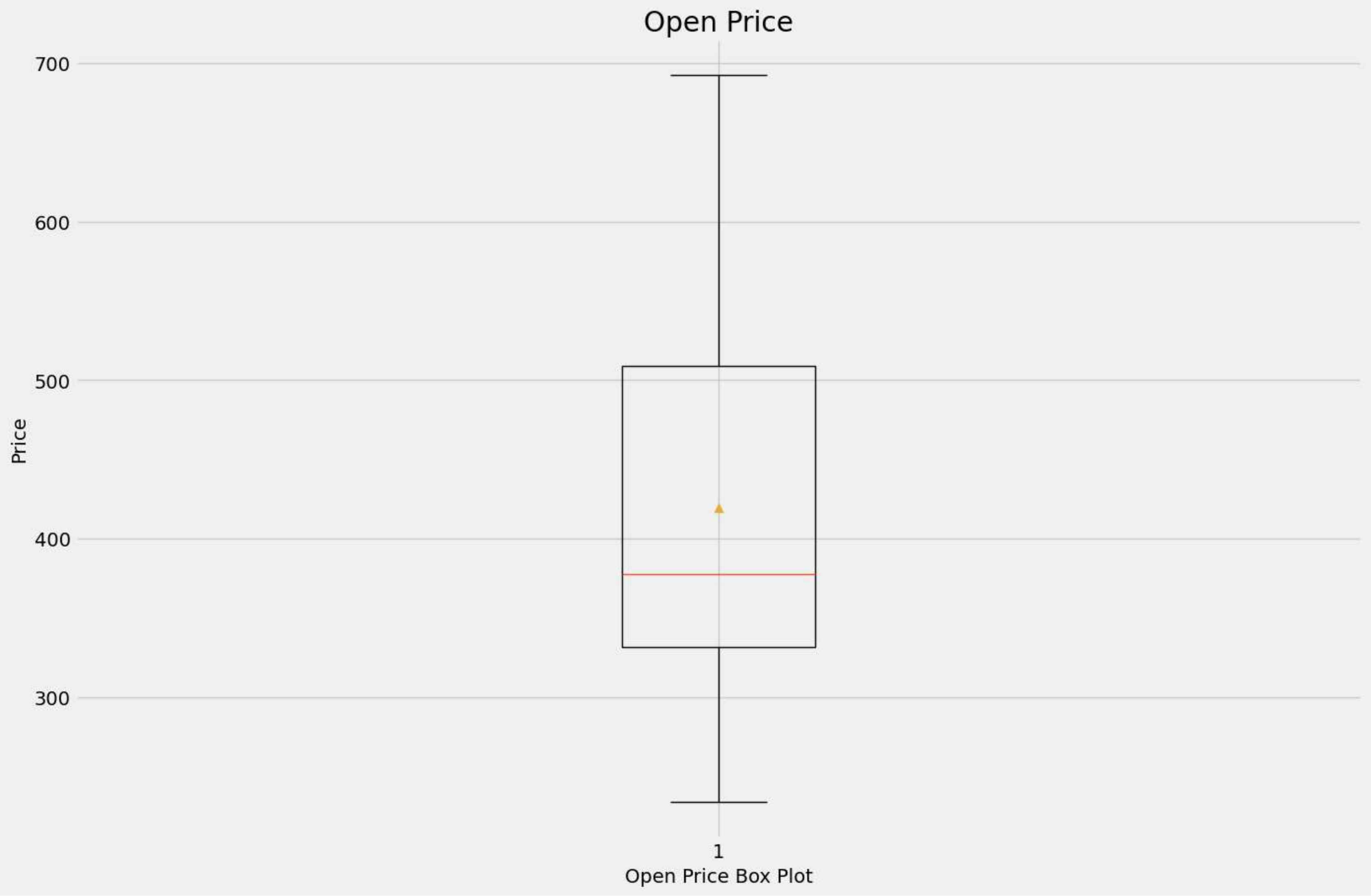
```
df.nunique()
```

Out[45]:

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

```
# EDA
```

In [27]:
```
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()
```



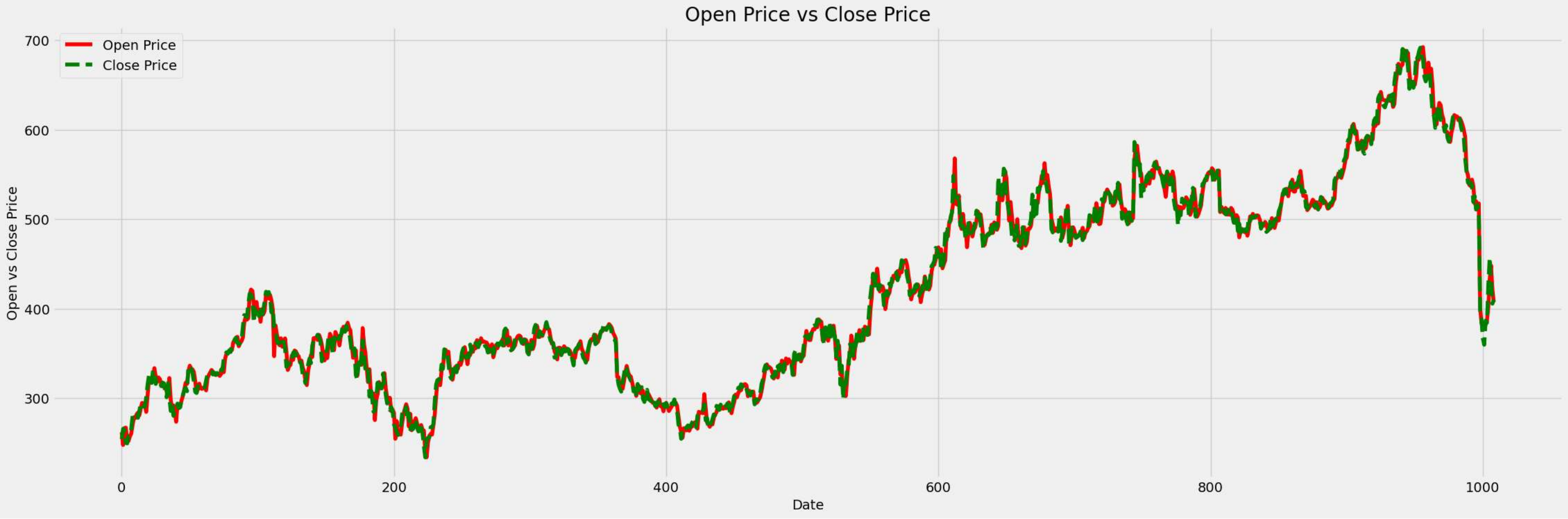
In [28]:

```python
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
```

In [29]:

```python
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()
```



PREPARE DATA

In [30]:

```python
from sklearn.preprocessing import StandardScaler
```

In [34]:

```
# change object to datetime
import pandas as pd
df=pd.read_csv("C:/Users/Dell/Downloads/NETflix.csv")
df['Date']=pd.to_datetime(df['Date'],format='%Y-%m-%d')

# set date to index
df.set_index('Date',inplace=True)
print(df)
```

```
                  Open        High         Low       Close   Adj Close  \
Date
2018-02-05  262.000000  267.899994  250.029999  254.259995  254.259995
2018-02-06  247.699997  266.700012  245.000000  265.720001  265.720001
2018-02-07  266.579987  272.450012  264.329987  264.559998  264.559998
2018-02-08  267.079987  267.619995  250.000000  250.100006  250.100006
2018-02-09  253.850006  255.800003  236.110001  249.470001  249.470001
...                ...         ...         ...         ...         ...
2022-01-31  401.970001  427.700012  398.200012  427.140015  427.140015
2022-02-01  432.959991  458.480011  425.540009  457.130005  457.130005
2022-02-02  448.250000  451.980011  426.480011  429.480011  429.480011
2022-02-03  421.440002  429.260010  404.279999  405.600006  405.600006
2022-02-04  407.309998  412.769989  396.640015  410.170013  410.170013

              Volume
Date
2018-02-05  11896100
2018-02-06  12595800
2018-02-07   8981500
2018-02-08   9306700
2018-02-09  16906900
...              ...
2022-01-31  20047500
2022-02-01  22542300
2022-02-02  14346000
2022-02-03   9905200
2022-02-04   7782400

[1009 rows x 6 columns]
```

In [35]:

```
train = df.loc['2018-02-05':'2021-12-31']
test = df.loc['2022-01-01':'2022-01-31']
```

In [36]:

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

Random Forest Model

In [37]:

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

Out[37]:

```
▾                     RandomForestRegressor

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

In [38]:

```
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)
```

```
0.9996830741916087
0.9918135946459509
```

In [39]:

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

In [40]:

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

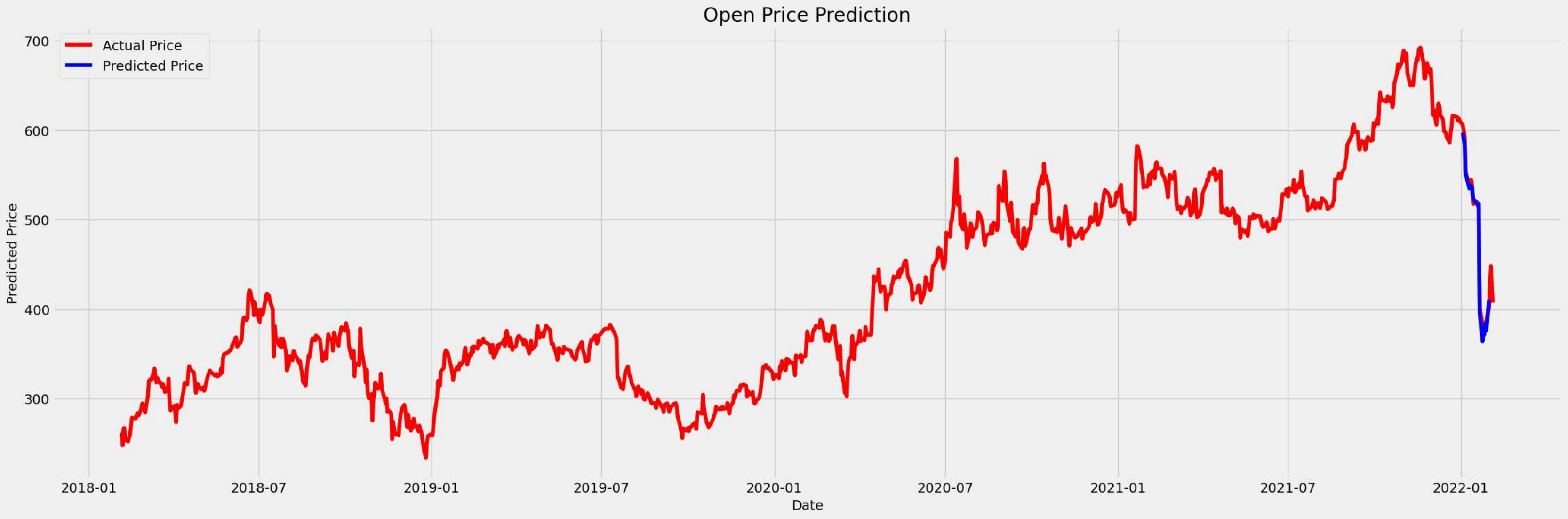
Out[40]:

	High	Low	Close	Adj Close	Volume	Open	Predicted Price
Date							
2022-01-03	609.989990	590.559998	597.369995	597.369995	3067500	605.609985	597.302736
2022-01-04	600.409973	581.599976	591.150024	591.150024	4393100	599.909973	589.919729
2022-01-05	592.840027	566.880005	567.520020	567.520020	4148700	592.000000	582.541523
2022-01-06	563.359985	542.010010	553.289978	553.289978	5711800	554.340027	550.311327
2022-01-07	553.429993	538.219971	541.059998	541.059998	3381700	549.460022	547.067863

Results

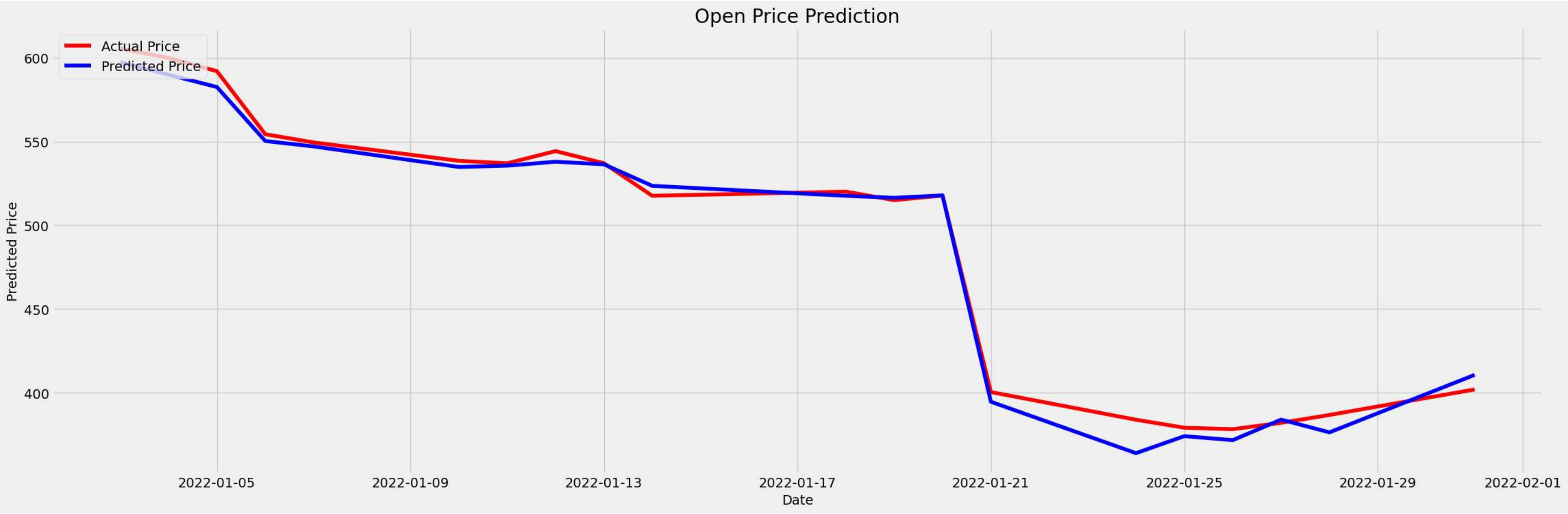
In [41]:

```
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()
```



In [42]:

```
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()
```



Model Evaluation

In [43]:

```
from sklearn import metrics
```

In [44]:

```
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
errors = abs(pred - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Mean Absolute Error: 5.725
Mean Squared Error: 53.3474
Root Mean Squared Error: 7.3039
(R^2) Score: 0.9918
Train Score : 99.97% and Test Score : 99.18% using Random Tree.
Accuracy: 98.75 %.
```