In [1]:
```
import pandas as pd
import numpy as np
import math
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_csv("tesla_stock_data.csv")
df.head()
```

Out[2]:

	Date	Year	Open	High	Low	Close	Volume	Adj Close
0	2020-01-02	2020	28.299999	28.713333	28.114000	28.684000	142981500	28.684000
1	2020-01-03	2020	29.366667	30.266666	29.128000	29.534000	266677500	29.534000
2	2020-01-06	2020	29.364668	30.104000	29.333332	30.102667	151995000	30.102667
3	2020-01-07	2020	30.760000	31.441999	30.224001	31.270666	268231500	31.270666
4	2020-01-08	2020	31.580000	33.232666	31.215334	32.809334	467164500	32.809334

In [3]:
```
x= df['Date']
y = df['Close']
```

In [4]:
```
y
```

Out[4]:
```
0       28.684000
1       29.534000
2       30.102667
3       31.270666
4       32.809334
          ...
991    252.539993
992    256.609985
993    261.440002
994    253.179993
995    248.479996
Name: Close, Length: 996, dtype: float64
```

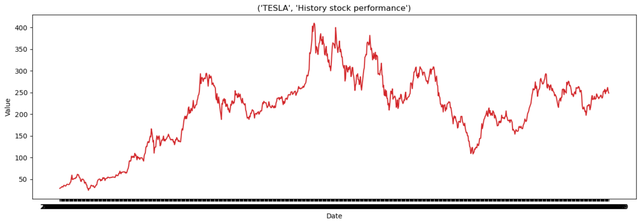
In [5]:
```
x
```

Out[5]:
```
0      2020-01-02
1      2020-01-03
2      2020-01-06
3      2020-01-07
4      2020-01-08
          ...
991    2023-12-22
992    2023-12-26
993    2023-12-27
994    2023-12-28
995    2023-12-29
Name: Date, Length: 996, dtype: object
```

In [6]:
```
def df_plot(data, x, y, title="", xlabel='Date', ylabel='Value', dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
```

In [7]:
```
stock_name = "TESLA"
title = (stock_name,"History stock performance")
df_plot(df , x , y , title=title,xlabel='Date',ylabel='Value',dpi=100)
```

Out[7]:



In [8]:
```
df.reset_index(inplace=True)
```

In [9]:
```
df.head(5)
```

Out[9]:

	index	Date	Year	Open	High	Low	Close	Volume	Adj Close
0	0	2020-01-02	2020	28.299999	28.713333	28.114000	28.684000	142981500	28.684000
1	1	2020-01-03	2020	29.366667	30.266666	29.128000	29.534000	266677500	29.534000
2	2	2020-01-06	2020	29.364668	30.104000	29.333332	30.102667	151995000	30.102667
3	3	2020-01-07	2020	30.760000	31.441999	30.224001	31.270666	268231500	31.270666
4	4	2020-01-08	2020	31.580000	33.232666	31.215334	32.809334	467164500	32.809334

In [17]:
```
df.drop(columns='Adj Close').head(2)
```

Out[17]:

	index	Date	Year	Open	High	Low	Close	Volume
0	0	2020-01-02	2020	28.299999	28.713333	28.114	28.684	142981500
1	1	2020-01-03	2020	29.366667	30.266666	29.128	29.534	266677500

In [20]:
```
df['Date'] = pd.to_datetime(df.Date)
```

In [21]:
```
df.describe()
```

Out[21]:

	index	Year	Open	High	Low	Close	Volume	Adj Close
count	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	9.960000e+02	996.000000
mean	497.500000	2021.493976	209.203720	213.956547	204.062843	209.121335	1.335149e+08	209.121335
std	287.664735	1.118579	85.884404	87.588506	83.894602	85.708640	8.888939e+07	85.708640
min	0.000000	2020.000000	24.980000	26.990667	23.367332	24.081333	2.940180e+07	24.081333
25%	248.750000	2020.000000	159.780838	162.617496	154.399996	160.250004	7.718922e+07	160.250004
50%	497.500000	2021.000000	223.989998	229.250000	218.266663	223.651665	1.074158e+08	223.651665
75%	746.250000	2022.000000	263.002510	268.010002	258.120010	262.667496	1.578296e+08	262.667496
max	995.000000	2023.000000	411.470001	414.496674	405.666656	409.970001	9.140820e+08	409.970001

In [22]:
```
x = df[['Open', 'High','Low', 'Volume']]
y = df['Close']
```

In [23]:
```
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.15 , shuffle=False,random_state = 0)
```

In [25]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score
regression = LinearRegression()
regression.fit(train_x, train_y)
print("regression coefficient",regression.coef_)
print("regression intercept",regression.intercept_)
```

```
regression coefficient [-6.76270564e-01  8.61728779e-01  8.13913493e-01  9.45162856e-10]
regression intercept 0.020748991544706996
```

In [26]:
```
regression_confidence = regression.score(test_x, test_y)
print("linear regression confidence: ", regression_confidence)
```

```
linear regression confidence:  0.9856630951374947
```

In [27]:
```
#As the coefficient of determination is 98%, our model is a linear model
```

```
The history saving thread hit an unexpected error (OperationalError('database or disk is full')).History will not be written to the database.
```

In [28]:
```
predicted=regression.predict(test_x)
print(test_x.head())
```

```
          Open        High         Low      Volume
846  186.539993  186.779999  180.580002   96870700
847  184.619995  190.600006  184.529999  162061500
848  200.100006  204.479996  197.529999  128818700
849  199.779999  203.949997  195.119995  150711700
850  202.589996  209.800003  199.369995  148029900
```

In [29]:
```
predicted.shape
```

Out[29]:
```
(150,)
```

In [30]:
```
dfr=pd.DataFrame({'Actual_Price':test_y, 'Predicted_Price':predicted})
dfr.head(10)
```

Out[30]:

	Actual_Price	Predicted_Price
846	184.470001	181.891001
847	193.169998	196.651652
848	201.160004	201.799388
849	203.929993	199.618242
850	207.520004	206.215640
851	213.970001	215.986555
852	217.610001	218.164555
853	221.309998	218.197755
854	224.570007	226.564932
855	234.860001	232.758132

In [31]:
```
#we can see that the predicted prices are very close to the actuals
```

In [32]:
```
#we will evaluate the model with the root mean squared error, the mean squared error
print('Mean Squared Error (MSE) :', metrics.mean_squared_error(test_y, predicted))
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean_squared_error(test_y, predicted)))
```

```
Mean Squared Error (MSE) : 6.288024085326087
Root Mean Squared Error (RMSE): 2.5075932854683765
```

In [33]:
```
#let's use mean absolute error
print('Mean Absolute Error (MAE):', metrics.mean_absolute_error(test_y, predicted))
```

```
Mean Absolute Error (MAE): 2.0157606423213767
```

In [34]:
```
#those metrics are quite low, indicated a good prediction of prices
```

In [35]:
```
dfr.describe()
```

Out[35]:

	Actual_Price	Predicted_Price
count	150.000000	150.000000
mean	246.035333	246.097121
std	21.012702	21.147234
min	184.470001	181.891001
25%	235.482498	234.310411
50%	249.099998	248.650634
75%	259.932495	259.627858
max	293.339996	293.503193

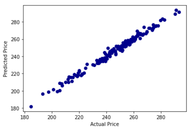
In [36]:
```
x2 = dfr.Actual_Price.mean()
y2 = dfr.Predicted_Price.mean()
Accuracy1 = x2/y2*100
print("The accuracy of the model is " , Accuracy1)
```

```
The accuracy of the model is  99.97489266684406
```

In [37]:
```
#good acuracy
```

In [38]:
```
#scatter plot of actual vs predicted price
plt.scatter(dfr.Actual_Price, dfr.Predicted_Price,  color='Darkblue')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.show()
```

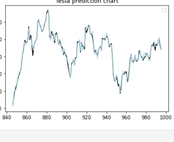
Out[38]:



In [41]:
```
#prediction chart
plt.plot(dfr.Actual_Price, color='black')
plt.plot(dfr.Predicted_Price, color='lightblue')
plt.title("Tesla prediction chart")
plt.legend()
```

```
No handles with labels found to put in legend.
<matplotlib.legend.Legend at 0x21297fbd9a0>
```

Out[41]:



In [ ]:
```
```