Tesla, Inc. is an American multinational automotive and clean energy company headquartered in Austin, Texas. Tesla designs and manufactures electric vehicles, stationary battery energy storage devices from home to grid-scale, solar panels and solar roof tiles, and related products and services.

The Tesla stock prediction for 2025 is currently $ 598.56, assuming that Tesla shares will continue growing at the average yearly rate as they did in the last 10 years. This would represent a 113.91% increase in the TSLA stock price.

Tesla has investments in various sectors. Here we are going to predict the daily stock closing value in correspondance to open,high and low values. For training we have took 2010-2020 stock price dataset.

Import Required Libraries and Dataset

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

```
df=pd.read_csv('/content/TSLA.csv')
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.000000	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.420000	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.920000	20.270000	21.959999	21.959999	8218800
3	02-07-2010	23.000000	23.100000	18.709999	19.200001	19.200001	5139800
4	06-07-2010	20.000000	20.000000	15.830000	16.110001	16.110001	6866900
...	...	...	...	...	...	...	...
2411	28-01-2020	568.489990	576.809998	558.080017	566.900024	566.900024	11788500
2412	29-01-2020	575.690002	589.799988	567.429993	580.989990	580.989990	17801500
2413	30-01-2020	632.419983	650.880005	618.000000	640.809998	640.809998	29005700
2414	31-01-2020	640.000000	653.000000	632.520020	650.570007	650.570007	15719300
2415	03-02-2020	673.690002	786.140015	673.520020	780.000000	780.000000	47065000

2416 rows × 7 columns

```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.92	20.270000	21.959999	21.959999	8218800

```
df.tail()
```

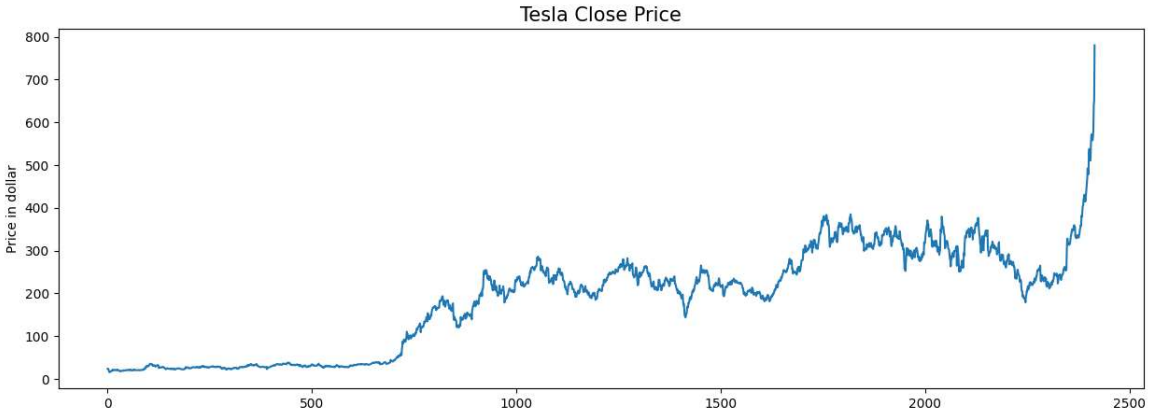
	Date	Open	High	Low	Close	Adj Close	Volume
2411	28-01-2020	568.489990	576.809998	558.080017	566.900024	566.900024	11788500
2412	29-01-2020	575.690002	589.799988	567.429993	580.989990	580.989990	17801500
2413	30-01-2020	632.419983	650.880005	618.000000	640.809998	640.809998	29005700
2414	31-01-2020	640.000000	653.000000	632.520020	650.570007	650.570007	15719300
2415	03-02-2020	673.690002	786.140015	673.520020	780.000000	780.000000	47065000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2416 entries, 0 to 2415
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2416 non-null   object 
 1   Open       2416 non-null   float64
 2   High       2416 non-null   float64
 3   Low        2416 non-null   float64
 4   Close      2416 non-null   float64
 5   Adj Close  2416 non-null   float64
 6   Volume     2416 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 132.2+ KB
```

Tesla Close price graphical representation

```
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close Price',fontsize=15)
plt.ylabel('Price in dollar')
plt.show()
```



Drop unwanted columns

```
df=df.drop(['Adj Close','Date','Volume'],axis=1)
df
```

	Open	High	Low	Close
0	19.000000	25.000000	17.540001	23.889999
1	25.790001	30.420000	23.299999	23.830000
2	25.000000	25.920000	20.270000	21.959999
3	23.000000	23.100000	18.709999	19.200001
4	20.000000	20.000000	15.830000	16.110001
...	...	...	...	...
2411	568.489990	576.809998	558.080017	566.900024
2412	575.690002	589.799988	567.429993	580.989990
2413	632.419983	650.880005	618.000000	640.809998
2414	640.000000	653.000000	632.520020	650.570007

Find missing values

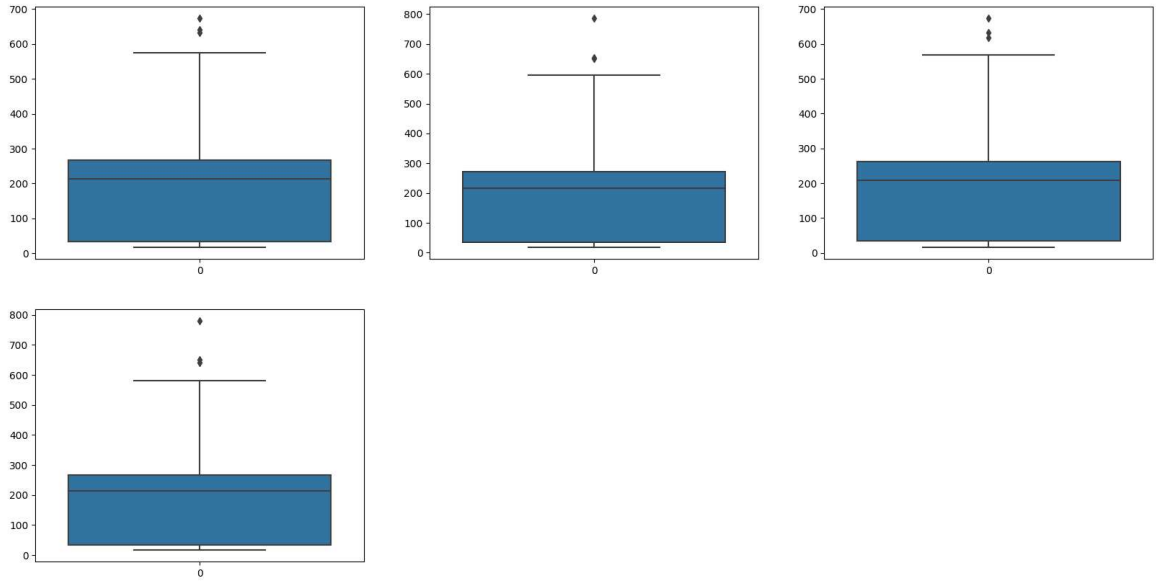
```
df.isna().sum()  #there is no missing values
```

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

Comparison of Open,High,Low,Close values

```
features = ['Open', 'High', 'Low', 'Close']
plt.subplots(figsize=(20,10))
for i,col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.boxplot(df[col])
plt.show()
```

```
<ipython-input-92-06da07af79a3>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated s
  plt.subplot(2,3,i+1)
```



Split into x and y components

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
x
y
```

```
0        23.889999
1        23.830000
2        21.959999
3        19.200001
4        16.110001
           ...
2411    566.900024
2412    580.989990
2413    640.809998
2414    650.570007
2415    780.000000
Name: Close, Length: 2416, dtype: float64
```

Allocate to test and train

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train
x_test
```

	Open	High	Low
410	31.549999	32.060001	30.900000
199	25.080000	25.209999	24.299999
1670	280.000000	282.239990	276.440002
1934	333.750000	336.369995	327.029999
1036	255.479996	263.739990	255.000000
...	...	...	...
270	29.010000	29.250000	28.440001
1041	263.250000	267.260010	259.750000
1047	264.980011	265.500000	261.660004
1292	255.559998	256.589996	250.509995
2405	507.609985	515.669983	503.160004

725 rows × 3 columns

Model Creation

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
645.07619325, 327.87112623, 202.66554395,  37.57096189,
122.32416348,  41.79572077, 183.19781999, 283.60016004,
 21.73902183, 240.50192378,  32.36812295,  30.3580994 ,
216.04761501,  26.30931517, 353.14640446, 250.79628353,
 33.64755236, 218.77263536,  90.41148213,  28.3821549 ,
207.62703202, 227.43629581, 336.60885812,  46.87212307,
164.19307399,  27.86362079, 255.37090619, 202.45930819,
214.67223405, 229.45985667, 195.94455594, 260.71793701,
 17.26799156,  25.24291905,  23.10398871,  33.29780011,
 35.21825731, 245.77825413,  31.75040034, 178.9907422 ,
177.50233064, 347.45387205, 200.73690912,  32.04265436,
142.92216608, 199.70356345, 216.12490323, 206.51264546,
202.75081377, 344.6574381 , 544.48934246,  29.23132929,
124.80008904, 184.62608367, 345.02041715, 156.66235527,
 28.20660962,  22.11066103, 223.86753601, 243.31301671,
229.82493868, 253.48212578, 162.23787359, 240.32206791,
 29.8392651 , 193.58646663, 329.13938195,  30.37241625,
201.35242458, 240.25528412, 165.93618052, 313.39164986,
 20.64082185,  21.1510197 , 122.38656354, 286.36941015,
245.17065538, 269.25228577, 300.23307581,  32.36414966,
128.17541934, 234.05507809,  27.66500516, 227.53959318,
 25.66057075, 252.85237343, 354.48002214,  28.34206269,
 27.54748219,  29.7800058 , 168.6169339 , 347.31328187,
 25.34747208,  30.90082312, 244.02324024,  27.94328338,
 32.49662247,  25.4252777 , 348.58614653, 306.82457528,
 90.30874298,  20.05802443, 256.39886369, 278.00476772,
 26.08915901, 309.63582263, 250.8187841 , 344.12927243,
 24.50475268, 188.23597929, 225.99049628, 116.5350393 ,
 29.14176824,  31.35242391, 350.93360435, 351.43803538,
349.74298828, 101.68418784, 234.07508262,  20.75167665,
305.57022034, 207.32827491,  28.65596876, 179.85898774,
 29.13868813, 306.07832123, 348.82641353,  21.46005998,
233.45023775, 240.36142563, 204.43442166, 364.09933831,
 28.70393428, 263.79074101, 262.52222119, 252.27852035,
510.71600817])
```

```
df1=pd.DataFrame({'Actual_value':y_test,'Pred_value':y_pred,'Error':y_test-y_pred})
df1
```

	Actual_value	Pred_value	Error
410	31.490000	31.422550	0.067450
199	24.650000	24.523803	0.126197
1670	279.760010	278.883579	0.876431
1934	328.200012	330.530668	-2.330656
1036	259.320007	262.105086	-2.785079
...	...	...	...
270	28.490000	28.703934	-0.213934
1041	259.940002	263.790741	-3.850739
1047	261.739990	262.522221	-0.782231
1292	254.990005	252.278520	2.711485
2405	510.500000	510.716008	-0.216008

725 rows × 3 columns

Performance Evaluation

```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error,mean_squared_error,r2_score,accuracy_score
print('Mean absolute Error is',mean_absolute_error(y_test,y_pred))
print('Error percentage is',mean_absolute_percentage_error(y_test,y_pred))
print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
root=mean_squared_error(y_test,y_pred)
print('Root mean squared error is',np.sqrt(root))
print('r2 score is',r2_score(y_test,y_pred))
```

```
Mean absolute Error is 1.3324973992978846
Error percentage is 0.007928958024365522
Mean Squared Error is 4.1582362981384895
Root mean squared error is 2.03917539660974
r2 score is 0.9997266059258266
```

Conclusion

We can observe that the accuracy attained is better than simply predicting. Using this machine learning algorithm we can predict the next possible closing share of Tesla. This is not accurate however, the closing number will be somehow arround the prediction value.