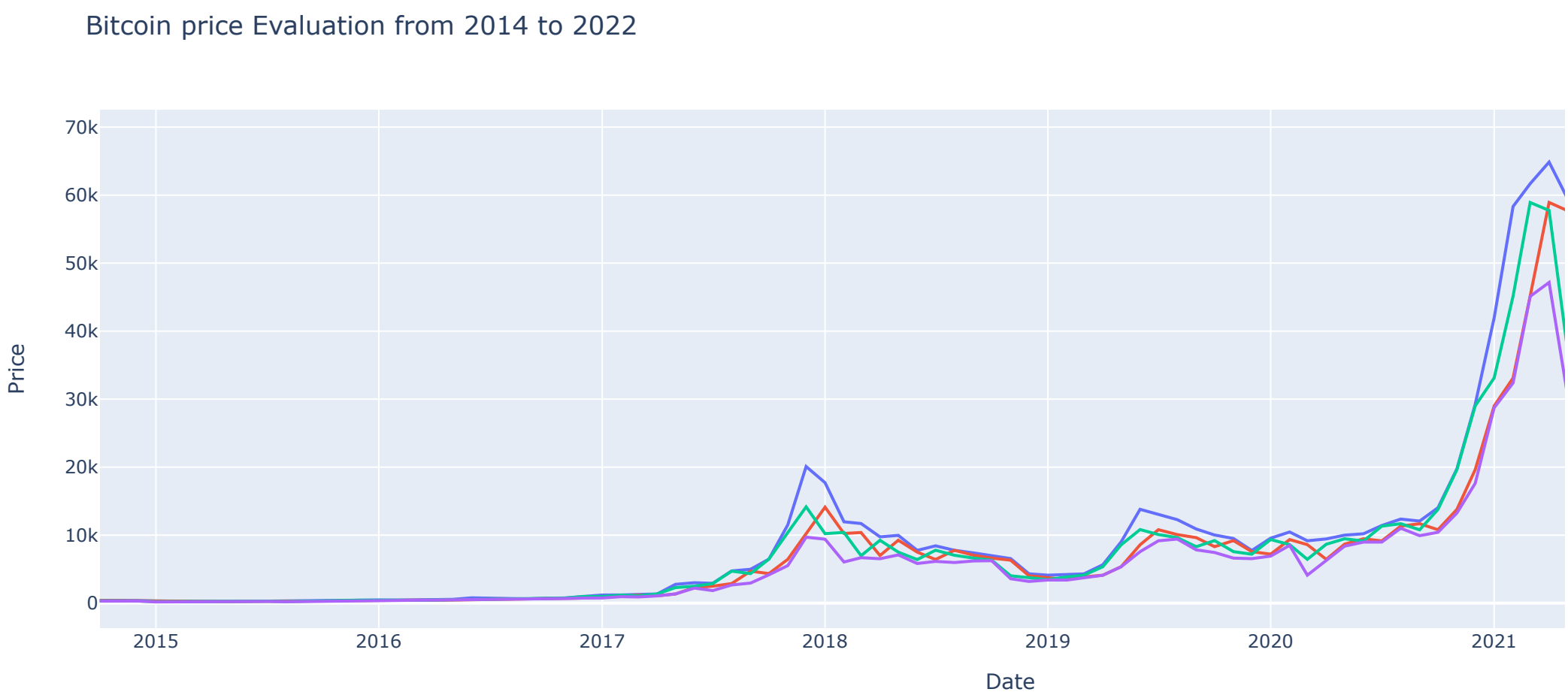
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

```
BTC_USD_data ='https://raw.githubusercontent.com/ukantjadia/30-days-of-Mahcine-Learning/Main/DAY-03/BTC-USD.csv'
df = pd.read_csv(BTC_USD_data)
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-10-01	387.427002	411.697998	289.295990	338.321014	338.321014	902994450
1	2014-11-01	338.649994	457.092987	320.626007	378.046997	378.046997	659733360
2	2014-12-01	378.248993	384.037994	304.231995	320.192993	320.192993	553102310
3	2015-01-01	320.434998	320.434998	171.509995	217.464005	217.464005	1098811912
4	2015-02-01	216.867004	265.610992	212.014999	254.263000	254.263000	711518700
...	...	...	...	...	...	...	...
94	2022-08-01	23336.718750	25135.589844	19600.785156	20049.763672	20049.763672	894192654543
95	2022-09-01	20050.498047	22673.820313	18290.314453	19431.789063	19431.789063	1123272250385
96	2022-10-01	19431.105469	20988.394531	18319.822266	20495.773438	20495.773438	957903424925
97	2022-11-01	20494.898438	21446.886719	15599.046875	17168.566406	17168.566406	1224531549126
98	2022-12-01	17168.001953	18318.531250	16398.136719	16602.585938	16602.585938	530117529578

99 rows × 7 columns

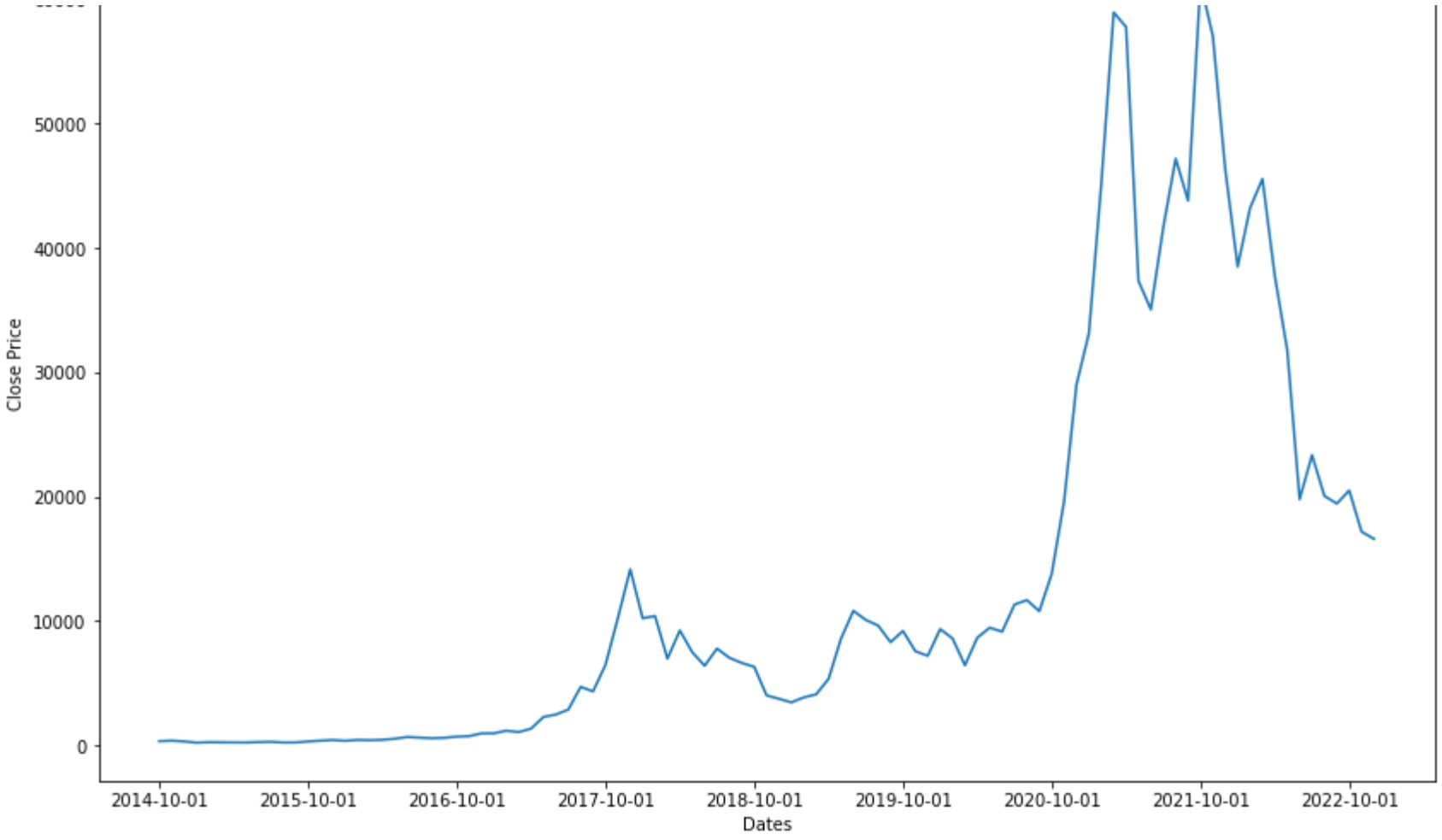
```
# Like Matplotlib, Plotly also provides various low-level, high-level, helpers interfaces to create, manipulate and render g
import plotly.express as px
fig = px.line(df,x="Date",y=["High","Open","Close","Low"],title="Bitcoin price Evaluation from 2014 to 2022")
fig.update_yaxes(title_text="Price")
fig.show()
```




```
time_stamp = ['2014-10-01','2015-10-01','2016-10-01','2017-10-01','2018-10-01','2019-10-01','2020-10-01','2021-10-01','2022-
fig, ax = plt.subplots(figsize=(14,9))
ax.plot(df.Date,df.Close)

ax.set_xticks(time_stamp)
ax.set_xlabel("Dates")
ax.set_ylabel("Close Price")

plt.show()
```

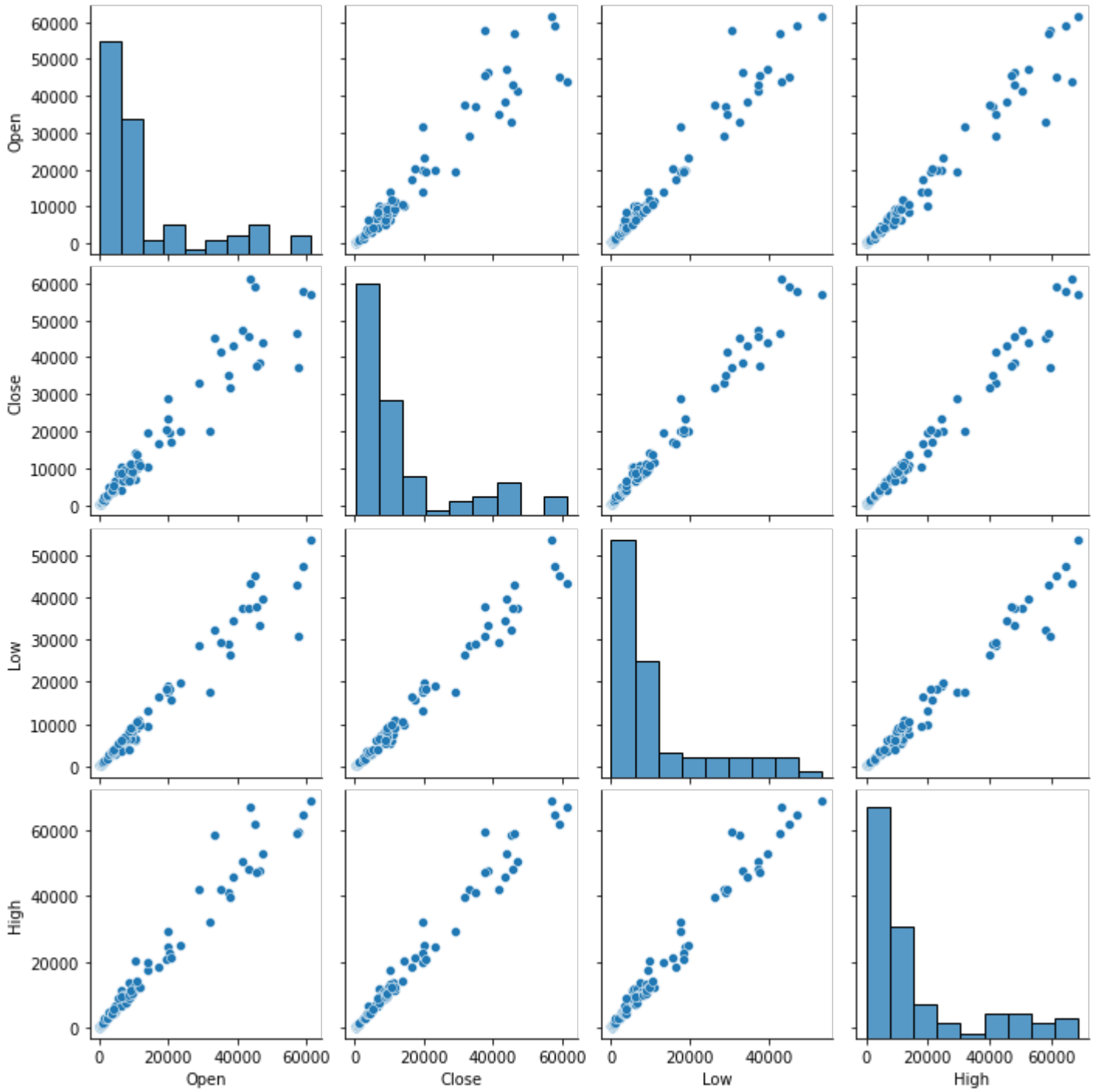



```
# Checking for the Nan values
for i in df.columns:
  print(i,"\t\t",df[i].isna().mean()*100)
```

```
Date            0.0
Open            0.0
High            0.0
Low             0.0
Close           0.0
Adj Close               0.0
Volume          0.0
```

```
df1 = df[['Open','Close','Low','High']]
sns.pairplot(df1)
# plt.tight_layout()
```

```
<seaborn.axisgrid.PairGrid at 0x7f9cf2d08a00>
```

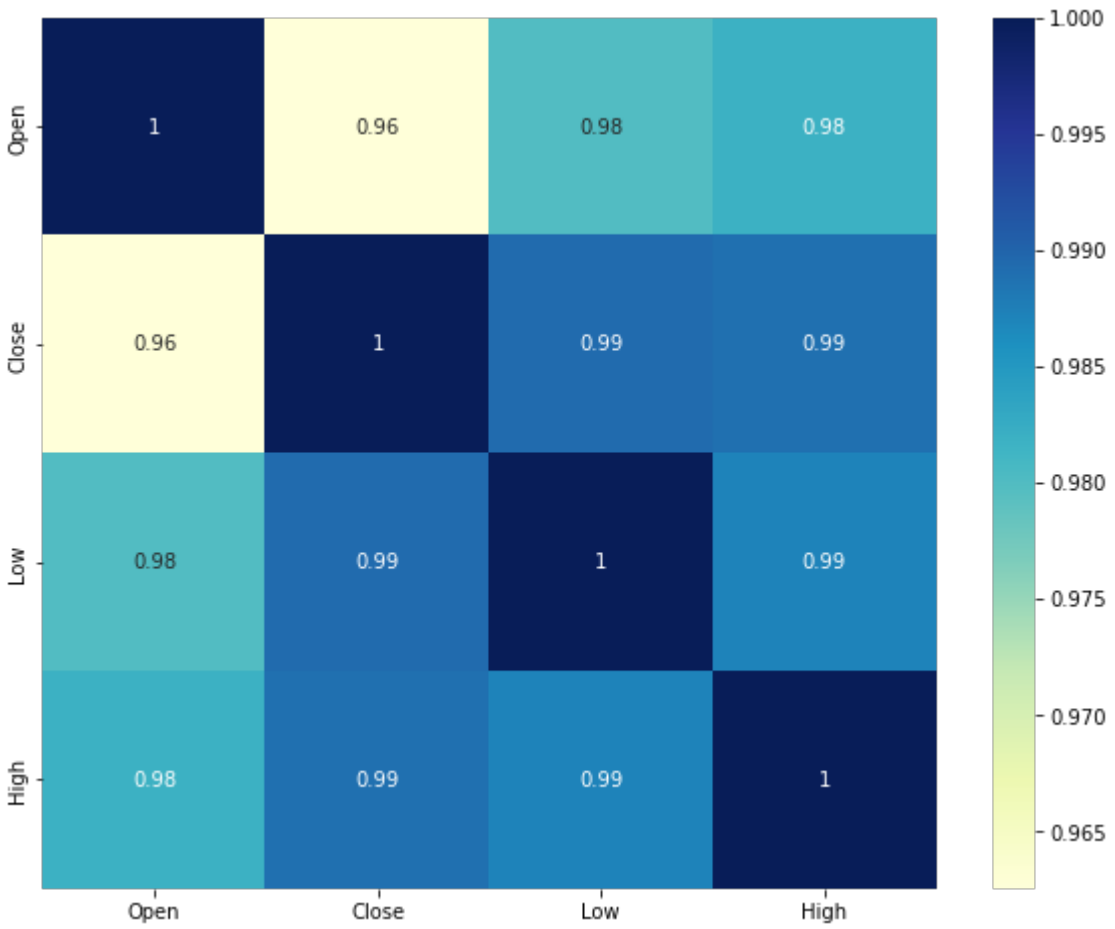


Since other parameters have linear relationship with Close, we are using some linear model for prediction

```
X = df1.drop(['Close'],axis=1)
y = df1['Close']
y.head()
```

```
0    338.321014
1    378.046997
2    320.192993
3    217.464005
4    254.263000
Name: Close, dtype: float64
```

```
plt.figure(figsize = (10, 8))
sns.heatmap(df1.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



Since range of data in different columns veries significantly we have to sacle independent variable X. For this we use Min-Max Scaling

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X = pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
X.head()
```

	Open	Low	High
0	0.002791	0.002206	0.002391
1	0.001993	0.002793	0.003053
2	0.002641	0.002486	0.001988
3	0.001695	0.000000	0.001060
4	0.000000	0.000759	0.000260

Prediction Model

```
# Split data into test train pairs
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, shuffle=False)
Acc = []
```

1. Linear Regression

```
from sklearn.linear_model import LinearRegression

model_LR = LinearRegression()
model_LR.fit(X_train,y_train)
```

```
LinearRegression()
```

```
y_pred_LR = model_LR.predict(X_test)
LR_pred_df = pd.DataFrame({'Actual':y_test,'Predicted':y_pred_LR})
# LR_pred_df.head()
```

```
from sklearn.metrics import r2_score
print("Accuracy Score of the Prediction: {0}".format(round(r2_score(y_test,y_pred_LR)*100,2)))
Acc.append(r2_score(y_test,y_pred_LR))
```

```
Accuracy Score of the Prediction: 94.11
```

```
plt.figure(figsize=(10,8))
plt.ylabel('Close Price',fontsize=16)
plt.plot(LR_pred_df)
plt.legend(['Actual Value','Prediction'])
plt.show()
```