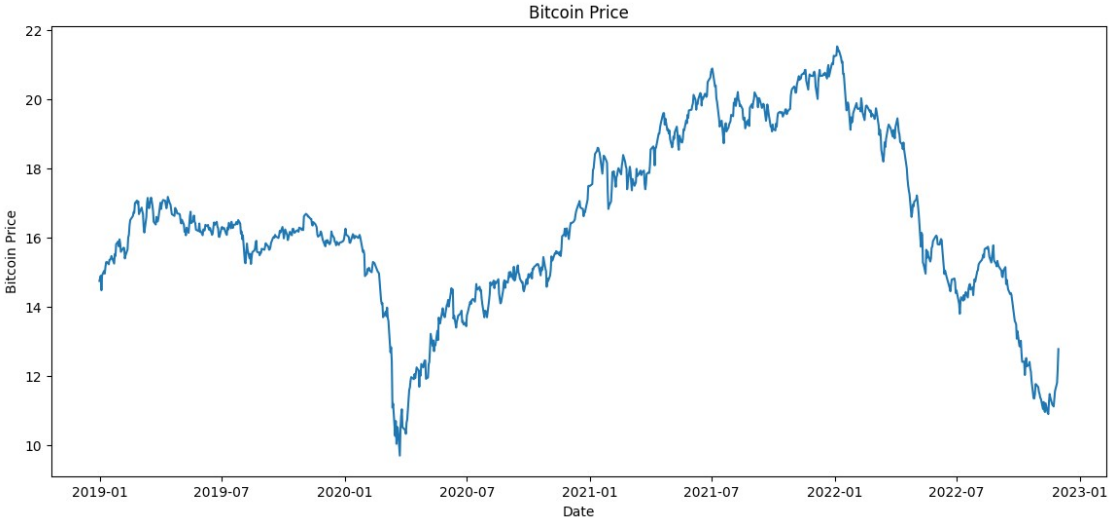
```
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt

df = pd.read_csv("VNM.csv", index_col="Date", parse_dates=["Date"])
df = df.dropna()

# use feature 'Date' & 'Close'
import matplotlib.dates as mdates
target_column = ["Close"]
dataset = pd.DataFrame(df[target_column])
print(' Count row of data: ',len(dataset))

fig = plt.figure(figsize=(14, 6))
plt.plot(dataset)
plt.xlabel('Date')
plt.ylabel('Bitcoin Price')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Bitcoin Price')
plt.show()
```

```
 Count row of data:  988
```




```
#Data normalize
from sklearn.preprocessing import MinMaxScaler
dataset_norm = dataset.copy()
dataset[['Close']]
scaler = MinMaxScaler()
dataset_norm['Close'] = scaler.fit_transform(dataset[['Close']])
dataset_norm
```

```
               Close
Date
2018-12-31  0.426881
2019-01-02  0.439560
2019-01-03  0.404057
2019-01-04  0.440406
2019-01-07  0.451395
...              ...
2022-11-23  0.120034
2022-11-25  0.158073
2022-11-28  0.177515
2022-11-29  0.211327
2022-11-30  0.260355

[988 rows x 1 columns]
```

```
#split data into train and test set
totaldata = dataset.values
totaldatatrain = int(len(totaldata)*0.75)
totaldatatest = int(len(totaldata)*0.25)

training_set = dataset_norm[0:totaldatatrain]
test_set = dataset_norm[totaldatatrain:]

#Sliding windows
lag = 2
```

```
# sliding windows function
def create_sliding_windows(data,len_data,lag):
    x=[]
    y=[]
    for i in range(lag,len_data):
        x.append(data[i-lag:i,0])
        y.append(data[i,0])
    return np.array(x),np.array(y)

# Formating data into array for create sliding windows
array_training_set = np.array(training_set)
array_test_set = np.array(test_set)

# Create sliding windows into training data
x_train, y_train =
create_sliding_windows(array_training_set,len(array_training_set),
lag)
# Create sliding windows into test data
x_test,y_test =
create_sliding_windows(array_test_set,len(array_test_set),lag)

#Apply train and test set into the model
model_rf = RandomForestRegressor(n_estimators=50)
model_rf.fit(x_train, y_train)
pred_train_rf= model_rf.predict(x_train)

pred_test_rf = model_rf.predict(x_test)

set_test = dataset["Close"]
set_test

Date
2018-12-31    14.75
2019-01-02    14.90
2019-01-03    14.48
2019-01-04    14.91
2019-01-07    15.04
              ...
2022-11-23    11.12
2022-11-25    11.57
2022-11-28    11.80
2022-11-29    12.20
2022-11-30    12.78
Name: Close, Length: 988, dtype: float64

#Inverse normalize
y_pred_invert_norm =
scaler.inverse_transform(pred_test_rf.reshape(245, 1))

#Compare table
datacompare = pd.DataFrame()
```

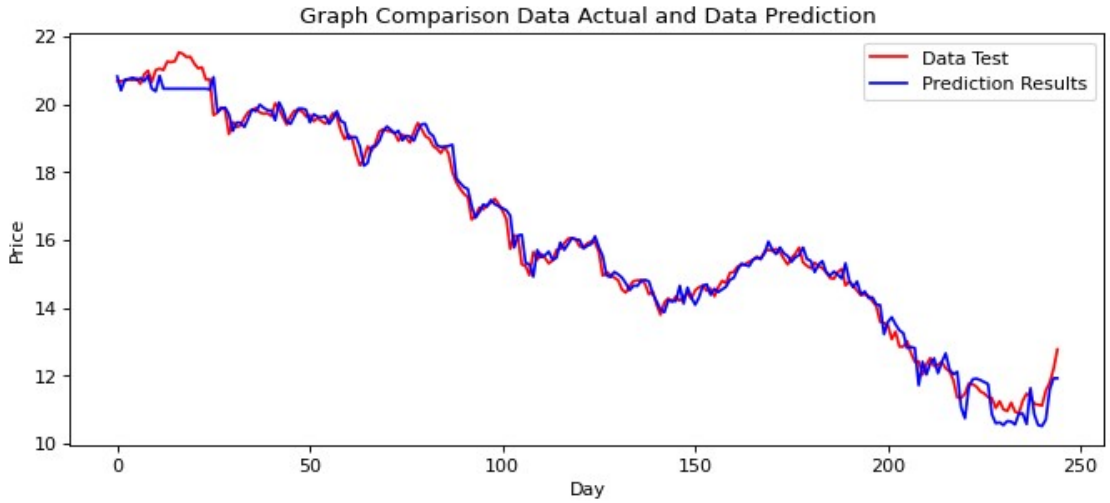
```
datatest=np.array(set_test[totaldatatrain+lag:])
datapred= y_pred_invert_norm

datacompare['Data Test'] = datatest
datacompare['Prediction Results'] = datapred
datacompare
```

	Data Test	Prediction Results
0	20.680000	20.821000
1	20.680000	20.409601
2	20.690001	20.728800
3	20.740000	20.738800
4	20.709999	20.788600
..	...	...
240	11.120000	10.515600
241	11.570000	10.713800
242	11.800000	11.598800
243	12.200000	11.927400
244	12.780000	11.927800

[245 rows x 2 columns]

```
plt.figure(num=None, figsize=(10, 4), dpi=80,facecolor='w',
edgecolor='k')
plt.title('Graph Comparison Data Actual and Data Prediction')
plt.plot(datacompare['Data Test'], color='red',label='Data Test')
plt.plot(datacompare['Prediction Results'],
color='blue',label='Prediction Results')
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend()
plt.show()
```




```
def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
```

```
    return mape
MAPE(datatest, datapred)
```

```
22.134349648258887
```

```
from sklearn.metrics import mean_squared_error
import math
MSE = mean_squared_error(datatest, datapred)
RMSE = math.sqrt(MSE)
print(RMSE)
```

```
0.3460340936012874
```

```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(
    y_true=datatest,
    y_pred=datapred
)
```

```
0.2556773949659871
```