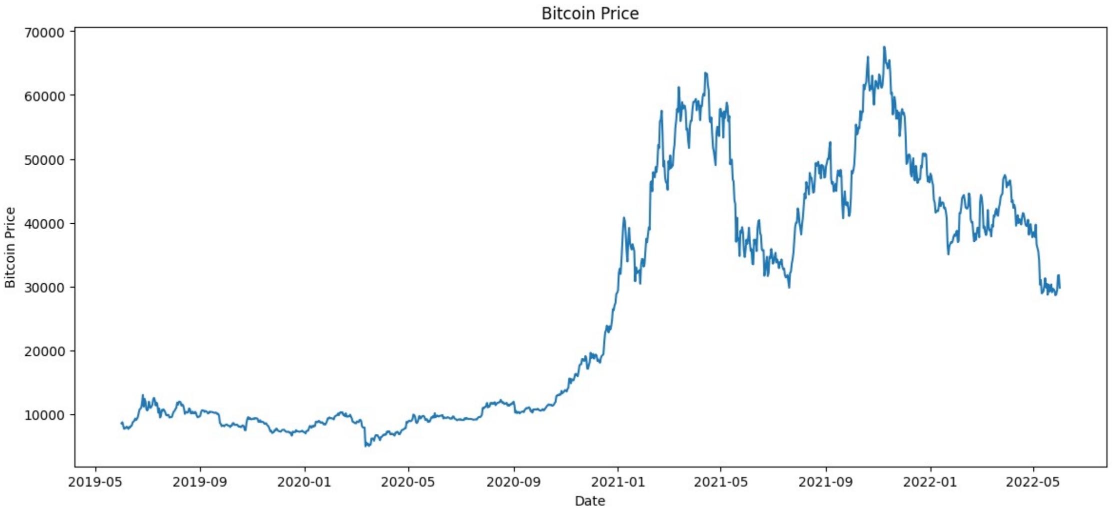
```
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
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("BTC.csv", index_col="Date", parse_dates=["Date"])
```

```
# 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:  1097
```



```
#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
2019-06-01  0.057403
2019-06-02  0.060262
2019-06-03  0.051732
2019-06-04  0.043725
2019-06-05  0.045585
...              ...
2022-05-28  0.380920
2022-05-29  0.391002
2022-05-30  0.427433
2022-05-31  0.428486
2022-06-01  0.396643

[1097 rows x 1 columns]
```

```
#split data into train and test set
totaldata = dataset.values
totaldatatrain = int(len(totaldata)*0.7)
val = int(len(totaldata)*0.1)
totaldatatest = int(len(totaldata)*0.2)

training_set = dataset_norm[0:totaldatatrain]
test_set = dataset_norm[totaldatatrain+val:]
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In [1], line 1
----> 1 totaldata = dataset.values
      2 totaldatatrain = int(len(totaldata)*0.7)
      3 val = int(len(totaldata)*0.1)

NameError: name 'dataset' is not defined
```

```
#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)
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call 
last)
Cell In [2], line 12
      9     return np.array(x),np.array(y)
     11 # Formating data into array for create sliding windows
---> 12 array_training_set = np.array(training_set)
     13 array_test_set = np.array(test_set)
     15 # Create sliding windows into training data

NameError: name 'np' is not defined
```

```
#Apply train and test set into the model
model_rf = RandomForestRegressor(n_estimators=500, oob_score=True, 
random_state=100)
model_rf.fit(x_train, y_train)
pred_train_rf= model_rf.predict(x_train)

pred_test_rf = model_rf.predict(x_test)
```

```
<class 'numpy.ndarray'>
```

```
set_test = dataset["Close"]
set_test
```

```
Date
2019-06-01     8564.016602
2019-06-02     8742.958008
2019-06-03     8208.995117
2019-06-04     7707.770996
2019-06-05     7824.231445
                  ...     
2022-05-28    28814.900391
2022-05-29    29445.957031
```

```
2022-05-30    31726.390625
2022-05-31    31792.310547
2022-06-01    29799.080078
Name: Close, Length: 1097, dtype: float64
```

```
#Inverse normalize
y_pred_invert_norm =
scaler.inverse_transform(pred_test_rf.reshape(219, 1))
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In [3], line 1
----> 1 y_pred_invert_norm =
scaler.inverse_transform(pred_test_rf.reshape(219, 1))

NameError: name 'scaler' is not defined
```

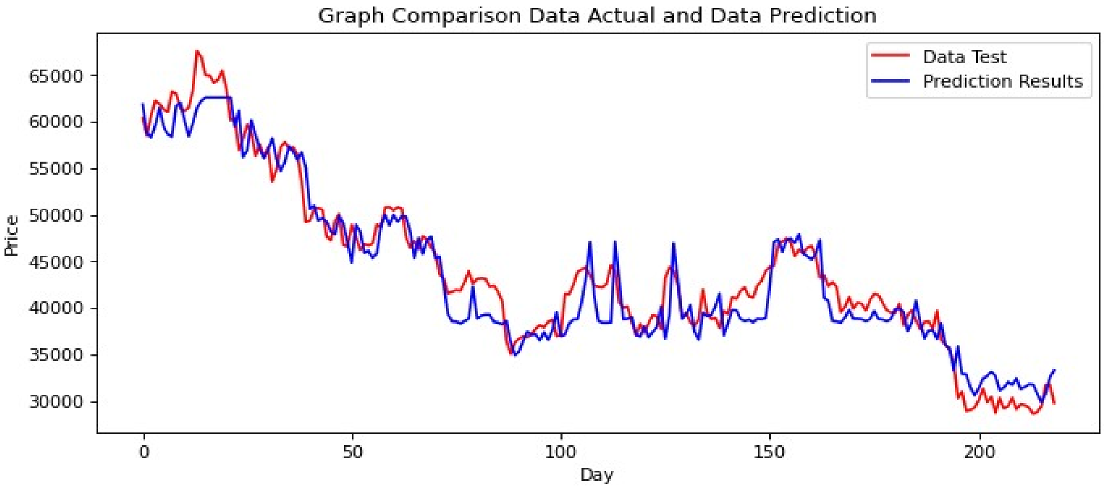
```
#Compare table
datacompare = pd.DataFrame()
datatest=np.array(set_test[totaldatatrain+val+lag:])
datapred= y_pred_invert_norm

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

```
        Data Test  Prediction Results
0    60363.792969        61843.125227
1    58482.386719        58795.665992
2    60622.136719        58278.568086
3    62227.964844        59583.012672
4    61888.832031        61472.468547
..            ...                 ...
214  28814.900391        30780.249641
215  29445.957031        29853.797359
216  31726.390625        30795.809930
217  31792.310547        32521.994715
218  29799.080078        33284.927055

[219 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 mae(y_true, predictions):
    y_true, predictions = np.array(y_true), np.array(predictions)
    return np.mean(np.abs(y_true - predictions))
print(mae(datatest, datapred))
```

```
10045.591960523981
```