Sahil Satyam
RA1911003010675

AI LAB-12

Aim: Applying deep learning methods to solve an application.

Problem Formulation: Solving a dataset using deep learning methods to predict crypto currency prices.

Problem Statement: Predicting the prices of cryptocurrencies like bitcoin and Ethereum using LSTM.

Algorithm used (Problem Solving): Naïve Bayes
LSTM or Long Short Term Memory is a Recurrent Neural Network used for sequence prediction problems. Here we are doing a problem of time series data i.e. it changes with time. Hence LSTM is used for this problem.

Dataset:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
1	1	Ethereum	ETH	2015-08-08 23:59:59	2.7988100051879883	0.7147250175476074	2.793760061264038	0.7533249855041504	674188.0	45486894.2408
2	2	Ethereum	ETH	2015-08-09 23:59:59	0.8798099756240845	0.629190981388092	0.7061359882354736	0.7018970251083374	532170.0	42399573.4991
3	3	Ethereum	ETH	2015-08-10 23:59:59	0.7298539876937866	0.6365460157394409	0.7139890193939209	0.7084479928016663	405283.0	42818364.3945
4	4	Ethereum	ETH	2015-08-11 23:59:59	1.131410002708435	0.6632350087165833	0.7080870270729065	1.0678600072860718	1463100.0	64569288.4328
5	5	Ethereum	ETH	2015-08-12 23:59:59	1.2899399995803833	0.8836079835891724	1.058750033378601	1.2174400091171265	2150620.0	73645010.9863
6	6	Ethereum	ETH	2015-08-13 23:59:59	1.9650700092315674	1.1719900369644165	1.2222399711608887	1.8276699781417847	4068680.0	110607191.674
7	7	Ethereum	ETH	2015-08-14 23:59:59	2.2618799209594727	1.7547500133514404	1.810920000076294	1.8278700113296509	4637030.0	110672321.811
8	8	Ethereum	ETH	2015-08-15 23:59:59	1.8772399425506592	1.5709799528121948	1.8028899431228638	1.6888999938964844	2554360.0	102303608.467
9	9	Ethereum	ETH	2015-08-16 23:59:59	1.6952400207519531	1.0898100137710571	1.6843500137329102	1.5660300254821777	3550790.0	94901005.3503

The dataset has 2161 rows, i.e. 2161 data entries.

Code:

Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense,Dropout,Activation

from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.metrics import mean_absolute_error
```

Entering the Etherium dataset

In [2]:
```
data = '/content/coin_Ethereum.csv'
dataset = pd.read_csv(data)
chosen_col = 'Close'
print(len(dataset))
dataset.head()
```

2160

Out[2]:

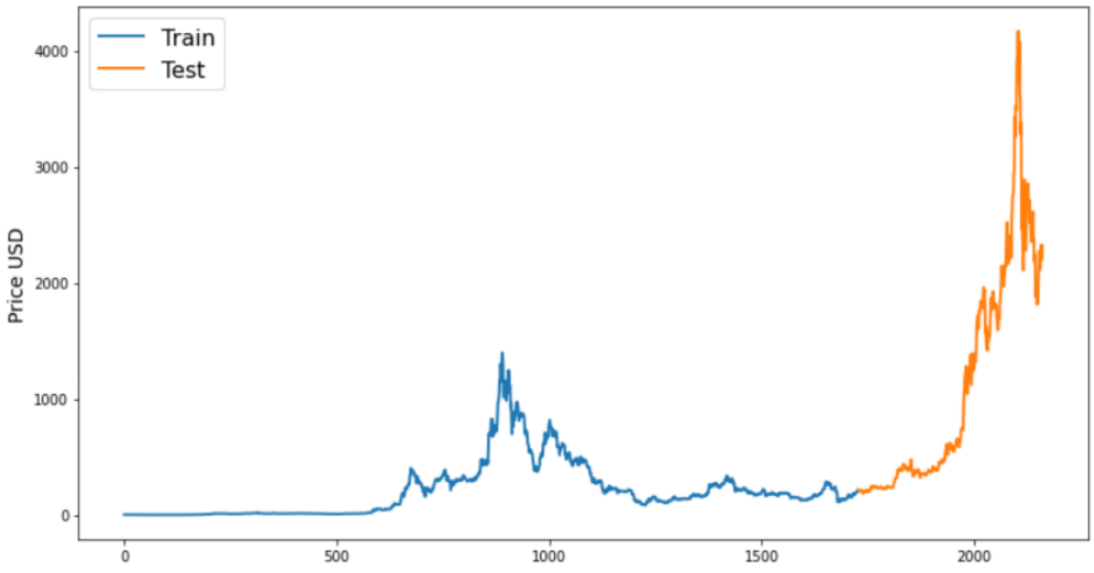
	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	1	Ethereum	ETH	2015-08-08 23:59:59	2.798810	0.714725	2.793760	0.753325	674188.0	4.548689e+07
1	2	Ethereum	ETH	2015-08-09 23:59:59	0.879810	0.629191	0.706136	0.701897	532170.0	4.239957e+07
2	3	Ethereum	ETH	2015-08-10 23:59:59	0.729854	0.636546	0.713989	0.708448	405283.0	4.281836e+07
3	4	Ethereum	ETH	2015-08-11 23:59:59	1.131410	0.663235	0.708087	1.067860	1463100.0	6.456929e+07
4	5	Ethereum	ETH	2015-08-12 23:59:59	1.289940	0.883608	1.058750	1.217440	2150620.0	7.364501e+07

Making Training and Testing data

In [3]:
```
split_row = len(dataset) - int(0.2 * len(dataset))
train_data = dataset.iloc[:split_row]
test_data = dataset.iloc[split_row:]
```

In [4]:
```
fig, ax = plt.subplots(1, figsize=(13, 7))
ax.plot(train_data[chosen_col], label='Train', linewidth=2)
ax.plot(test_data[chosen_col], label='Test', linewidth=2)
ax.set_ylabel('Price USD', fontsize=14)
ax.set_title('', fontsize=16)
ax.legend(loc='best', fontsize=16)
```

Out[4]:
```
<matplotlib.legend.Legend at 0x7f30bbde6e50>
```



Making the model

In [10]:
```
model = Sequential()
model.add(LSTM(units=100, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('linear'))
model.compile(optimizer='adam', loss='mse')
```

Running the model

In [11]:
```
callback = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=32, verbose=1, callbacks=[callback], shuffle=True)
```

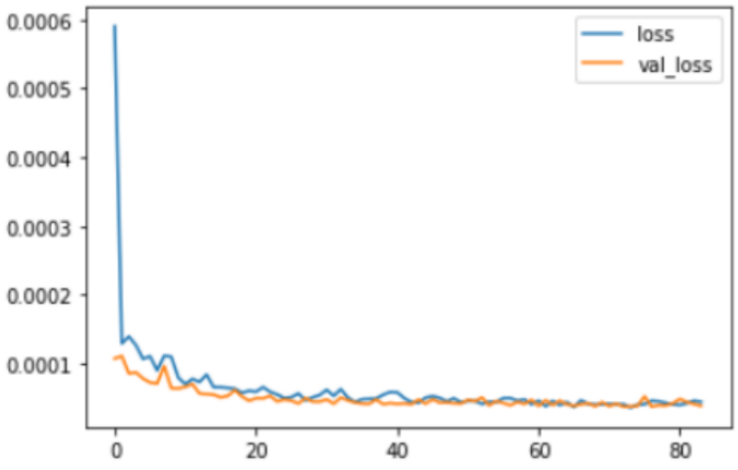
```
Epoch 1/100
42/42 [==============================] - 5s 57ms/step - loss: 5.9006e-04 - val_loss: 1.0731e-04
Epoch 2/100
42/42 [==============================] - 2s 42ms/step - loss: 1.2917e-04 - val_loss: 1.1092e-04
Epoch 3/100
42/42 [==============================] - 2s 43ms/step - loss: 1.3931e-04 - val_loss: 8.5686e-05
Epoch 4/100
42/42 [==============================] - 2s 42ms/step - loss: 1.2668e-04 - val_loss: 8.7286e-05
Epoch 5/100
42/42 [==============================] - 2s 43ms/step - loss: 1.0669e-04 - val_loss: 7.8201e-05


Epoch 80/100
42/42 [==============================] - 2s 41ms/step - loss: 4.0768e-05 - val_loss: 4.2097e-05
Epoch 81/100
42/42 [==============================] - 2s 42ms/step - loss: 3.9795e-05 - val_loss: 4.8531e-05
Epoch 82/100
42/42 [==============================] - 2s 42ms/step - loss: 4.2216e-05 - val_loss: 4.3942e-05
Epoch 83/100
42/42 [==============================] - 2s 42ms/step - loss: 4.5950e-05 - val_loss: 4.1302e-05
Epoch 84/100
42/42 [==============================] - 2s 43ms/step - loss: 4.4286e-05 - val_loss: 3.8486e-05
```

loss and validation loss both are decreasing therefore the model is performing well

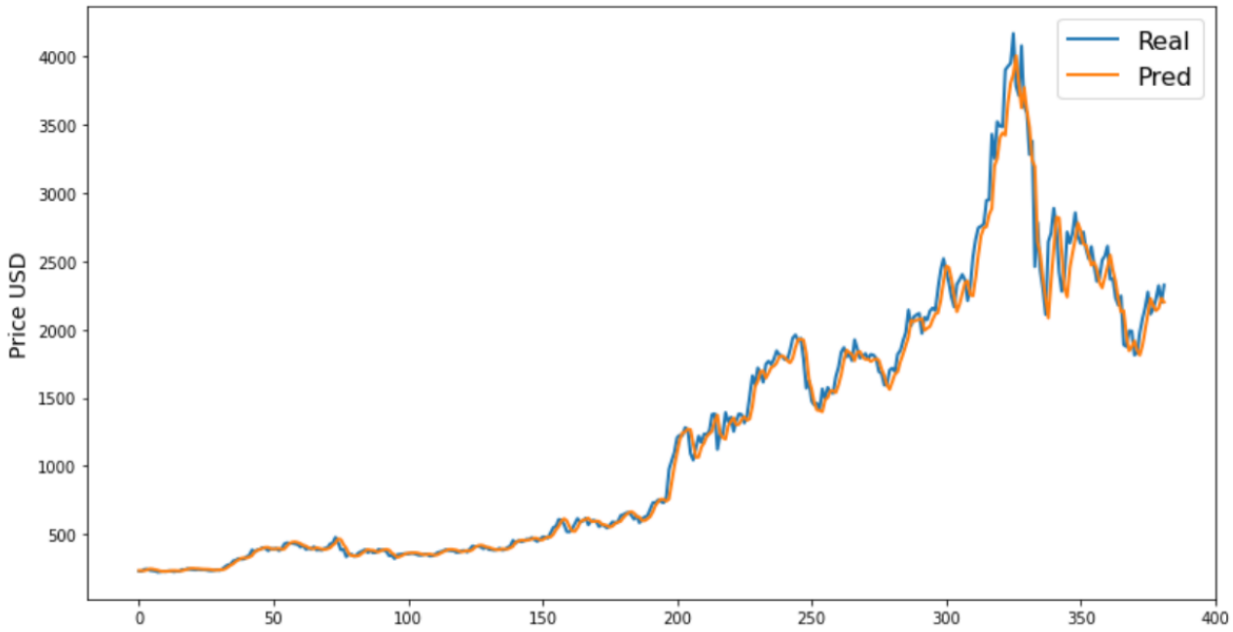
In [12]:
```
plt.plot(history.history['loss'], label = 'loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.legend()
```

Out[12]:
```
<matplotlib.legend.Legend at 0x7f309be81f50>
```



This clearly shows that the model is neither overfitting nor underfitting

Output:



Graph of real values (blue) and predicted values (orange).

Output: Hence Deep Learning method- LSTM is used to predict the prices of cryptocurrency.