```
import tensorflow as tf
from tensorflow.keras.datasets import boston_housing
from sklearn import preprocessing
import plotly.graph_objects as go
import matplotlib.pyplot as plt
```

```
(train_x,train_y),(test_x,test_y)=boston_housing.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57026/57026 [==============================] - 0s 0us/step
```

```
print("Train Shape :",train_x.shape)
print("Test Shape :",test_x.shape)
print("Training Sample :",train_x[0])
print("Training Target Sample :",train_y[0])
```

```
Train Shape : (404, 13)
Test Shape : (102, 13)
Training Sample : [  1.23247   0.        8.14      0.        0.538     6.142    91.7
   3.9769    4.      307.       21.      396.9      18.72   ]
Training Target Sample : 15.2
```

```
mean=train_x.mean(axis=0)
std=train_x.std(axis=0)
```

```
train_x=(train_x-mean)/std
test_x=(test_x-mean)/std
```

```
train_x[0]
```

```
array([-0.27224633, -0.48361547, -0.43576161, -0.25683275, -0.1652266 ,
       -0.1764426 ,  0.81306188,  0.1166983 , -0.62624905, -0.59517003,
        1.14850044,  0.44807713,  0.8252202 ])
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


def HousePricePredictionModel():
  model=Sequential()
  model.add(Dense(128,activation='relu',input_shape=(train_x[0].shape),name='dense_1')) #128 Neurons
  model.add(Dense(64,activation='relu',name='dense_2')) #64 Neurons
  model.add(Dense(1,activation='linear',name='dense_output')) #1 Neuron
  model.compile(optimizer='adam', loss='mse', metrics=['mae'])
  model.summary()
  return model


model=HousePricePredictionModel()
history=model.fit(x=train_x,y=train_y,epochs=100,batch_size=1,verbose=1,validation_data=(test_x,test_y))
```

```
404/404 [==============================] - 1s 2ms/step - loss: 2.7736 - mae: 1.2301 - val_loss: 11.0412 - val_mae: 2.4354
Epoch 83/100
404/404 [==============================] - 1s 3ms/step - loss: 3.0967 - mae: 1.2953 - val_loss: 13.1144 - val_mae: 2.5434
Epoch 84/100
404/404 [==============================] - 1s 3ms/step - loss: 2.9094 - mae: 1.2476 - val_loss: 10.3407 - val_mae: 2.3168
Epoch 85/100
404/404 [==============================] - 1s 2ms/step - loss: 2.3655 - mae: 1.1455 - val_loss: 12.0468 - val_mae: 2.3457
Epoch 86/100
404/404 [==============================] - 1s 2ms/step - loss: 2.6953 - mae: 1.2196 - val_loss: 11.7692 - val_mae: 2.4890
Epoch 87/100
404/404 [==============================] - 1s 2ms/step - loss: 2.5780 - mae: 1.1963 - val_loss: 12.1750 - val_mae: 2.3008
Epoch 88/100
404/404 [==============================] - 1s 2ms/step - loss: 3.0786 - mae: 1.3137 - val_loss: 10.8993 - val_mae: 2.3241
Epoch 89/100
404/404 [==============================] - 1s 2ms/step - loss: 2.2300 - mae: 1.0989 - val_loss: 11.0223 - val_mae: 2.3422
Epoch 90/100
404/404 [==============================] - 1s 2ms/step - loss: 2.4482 - mae: 1.1902 - val_loss: 11.3606 - val_mae: 2.4105
Epoch 91/100
404/404 [==============================] - 1s 2ms/step - loss: 2.3593 - mae: 1.1642 - val_loss: 10.3123 - val_mae: 2.3654
Epoch 92/100
404/404 [==============================] - 1s 2ms/step - loss: 2.3273 - mae: 1.1181 - val_loss: 10.1262 - val_mae: 2.2096
Epoch 93/100
404/404 [==============================] - 1s 2ms/step - loss: 2.4862 - mae: 1.1905 - val_loss: 12.5619 - val_mae: 2.4429
Epoch 94/100
404/404 [==============================] - 1s 2ms/step - loss: 2.6926 - mae: 1.2653 - val_loss: 9.8133 - val_mae: 2.2528
Epoch 95/100
404/404 [==============================] - 1s 2ms/step - loss: 2.4357 - mae: 1.1561 - val_loss: 10.3101 - val_mae: 2.2520
Epoch 96/100
404/404 [==============================] - 1s 2ms/step - loss: 2.4437 - mae: 1.1551 - val_loss: 11.7081 - val_mae: 2.3610
Epoch 97/100
404/404 [==============================] - 1s 2ms/step - loss: 2.3008 - mae: 1.1241 - val_loss: 11.3845 - val_mae: 2.3941
Epoch 98/100
404/404 [==============================] - 1s 3ms/step - loss: 2.2561 - mae: 1.1368 - val_loss: 9.9193 - val_mae: 2.2567
Epoch 99/100
404/404 [==============================] - 1s 3ms/step - loss: 2.5401 - mae: 1.2102 - val_loss: 10.0739 - val_mae: 2.2588
Epoch 100/100
404/404 [==============================] - 1s 2ms/step - loss: 2.4294 - mae: 1.1355 - val_loss: 10.3713 - val_mae: 2.3987
```

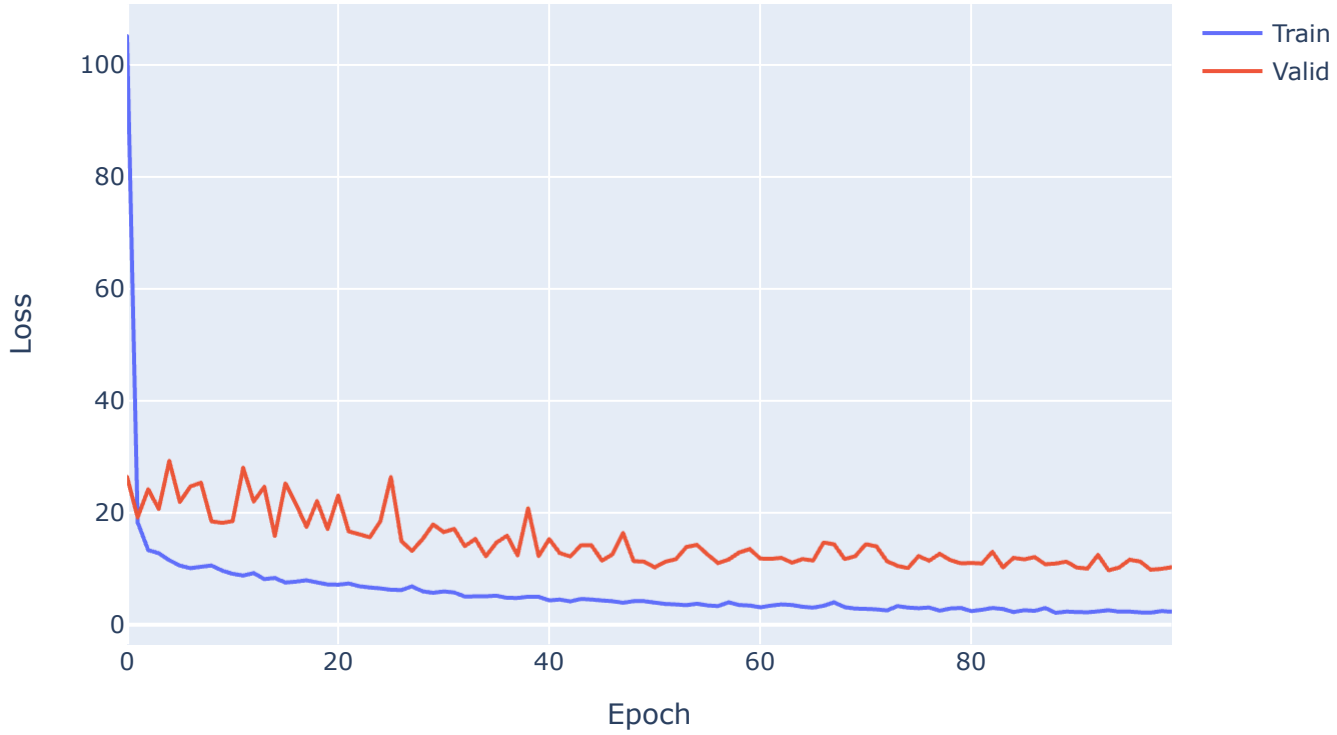
```
test_x[8]
```

```
array([-0.39570978, -0.48361547,  2.13815109, -0.25683275,  0.20183093,
       -0.43176465,  0.85606329, -0.81539201, -0.85646254, -1.31131055,
        0.28394328,  0.24795926,  0.71618792])
```

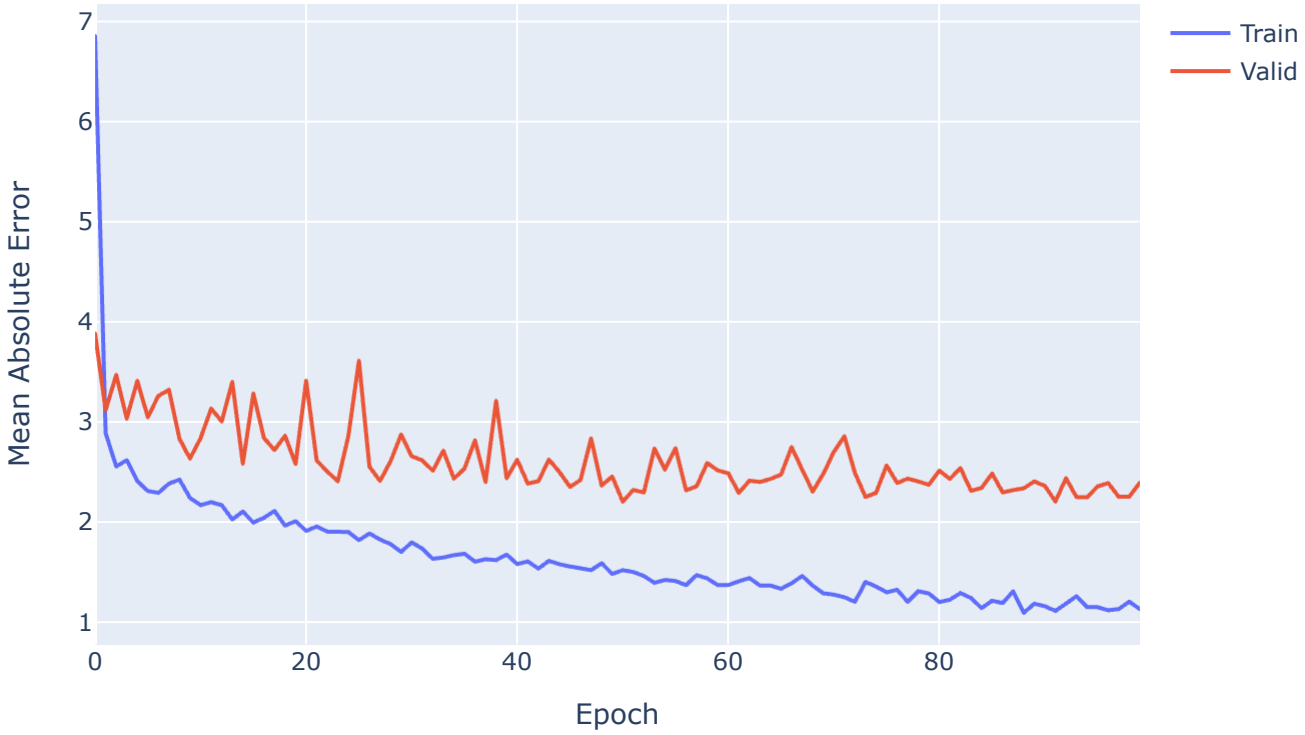
```
test_input=[[-0.39570978, -0.48361547,  2.13815109, -0.25683275,  0.20183093,
       -0.43176465,  0.85606329, -0.81539201, -0.85646254, -1.31131055,
        0.28394328,  0.24795926,  0.71618792]]
print("Actual Output :",test_y[8])
print("Predicted Output :",model.predict(test_input))
```

```
Actual Output : 20.5
1/1 [==============================] - 0s 112ms/step
Predicted Output : [[17.897112]]
```

```
fig = go.Figure()
fig.add_trace(go.Scattergl(y=history.history['loss'],name='Train'))
fig.add_trace(go.Scattergl(y=history.history['val_loss'],name='Valid'))
fig.update_layout(height=500, width=700,xaxis_title='Epoch',yaxis_title='Loss')
fig.show()
```




```
fig = go.Figure()
fig.add_trace(go.Scattergl(y=history.history['mae'],name='Train'))
fig.add_trace(go.Scattergl(y=history.history['val_mae'],name='Valid'))
fig.update_layout(height=500, width=700,xaxis_title='Epoch',yaxis_title='Mean Absolute Error')
fig.show()
```




```
mse_nn,mae_nn=model.evaluate(test_x,test_y)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 10.3713 - mae: 2.3987
```

```
print('Mean squared error on test data :',mse_nn)
print('Mean absolute error on test data :',mae_nn)
```

```
Mean squared error on test data : 10.371269226074219
Mean absolute error on test data : 2.398691415786743
```

```
from sklearn.metrics import r2_score
y_dl=model.predict(test_x)
r2=r2_score(test_y,y_dl)
print('R2 Score :',r2)
```

```
4/4 [==============================] - 0s 6ms/step
R2 Score : 0.87541098462976
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score


lr_model=LinearRegression()
lr_model.fit(train_x,train_y)
```

▾ LinearRegression

LinearRegression()

```
y_pred=lr_model.predict(test_x)


mse_lr=mean_squared_error(test_y,y_pred)
mae_lr=mean_absolute_error(test_y,y_pred)
r2=r2_score(test_y,y_pred)
print('Mean squared error on test data :',mse_lr)
print('Mean absolute error on test data :',mae_lr)
print('R2 Score :',r2)
```

```
Mean squared error on test data : 23.19559925642298
Mean absolute error on test data : 3.4641858124067175
R2 Score : 0.7213535934621552
```