In [1]:
```
import pandas as pd
import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
print(tf.__version__)
```

```
2.3.0
```

In [3]:
```
raw_dataset=pd.read_csv("shaleporV1.csv",sep=",")
```

In [4]:
```
shaleporV1 = raw_dataset.copy()
shaleporV1.head()
```

Out[4]:

	Porosity_%	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_%	Pyrite_%	MBI_%
0	6.3	0.325	2.5851	5.62762	2.7	1.86	54.2	11.4	27.1	4.7	2.0	70.0
1	7.7	0.326	2.5892	5.63064	2.6	1.86	41.8	12.1	35.3	6.8	2.2	62.0
2	3.9	0.326	2.5934	5.63367	1.4	1.86	47.0	6.2	22.6	21.3	1.4	76.0
3	4.3	0.326	2.5959	5.63669	1.7	1.86	45.9	6.5	23.7	14.2	7.5	72.0
4	6.2	0.327	2.5991	5.63972	2.5	1.86	38.9	9.1	32.9	16.3	2.1	64.0

In [5]:
```
shaleporV1.shape
```

Out[5]:
```
(1148, 12)
```

In [6]:

```
shaleporV1.describe()
```

Out[6]:

	Porosity_%	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_'
count	1148.000000	1148.000000	1148.000000	1148.000000	1148.000000	1148.000000	1148.000000	1148.000000	1148.000000	1148.00000
mean	4.896899	0.274268	2.620440	4.866332	3.465958	1.629373	32.250383	8.772404	32.314983	22.47379
std	1.882841	0.155369	0.899567	2.335575	1.969073	0.747745	11.036565	3.861187	12.887978	20.98937
min	1.660000	0.086000	0.704300	1.794500	1.010000	0.620000	5.700000	1.200000	2.400000	1.00000
25%	3.600000	0.094000	1.907800	3.025600	2.040000	0.960000	23.000000	6.000000	22.000000	7.00000
50%	4.500000	0.331000	2.375300	4.226760	3.040000	1.400000	34.900000	8.000000	36.000000	13.48000
75%	6.000000	0.437000	3.096550	7.606360	4.000000	2.510000	40.000000	11.000000	41.900000	32.00000
max	10.470000	0.456000	4.883320	7.972460	12.800000	2.640000	64.700000	24.900000	71.000000	85.60000

In [7]:
```
shaleporV1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1148 entries, 0 to 1147
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Porosity_%        1148 non-null   float64
 1   Age_BA            1148 non-null   float64
 2   Present_Depth_km  1148 non-null   float64
 3   Burial_Depth_km   1148 non-null   float64
 4   TOC_%             1148 non-null   float64
 5   Requ_%            1148 non-null   float64
 6   Quartz_%          1148 non-null   float64
 7   Feldspar_%        1148 non-null   float64
 8   Clay_%            1148 non-null   float64
 9   Carbonate_%       1148 non-null   float64
 10  Pyrite_%          1148 non-null   float64
 11  MBI_%             1148 non-null   float64
dtypes: float64(12)
memory usage: 107.8 KB
```

In [8]:
```
corr_matrix = shaleporV1.corr()
```

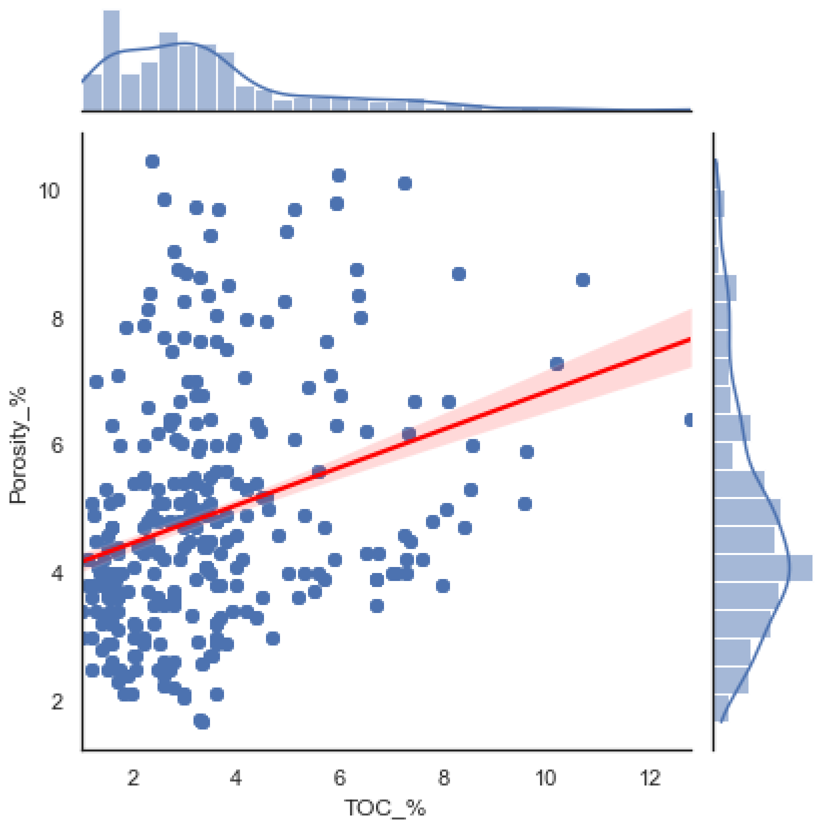
In [9]:
```
corr_matrix["Porosity_%"].sort_values(ascending=False)
```

Out[9]:
```
Porosity_%          1.000000
Present_Depth_km    0.677456
TOC_%               0.309901
Carbonate_%         0.204004
Pyrite_%            0.034403
Burial_Depth_km     0.017547
Requ_%             -0.011152
Clay_%             -0.031191
MBI_%              -0.036460
Feldspar_%         -0.047513
Age_BA             -0.196413
Quartz_%           -0.425452
Name: Porosity_%, dtype: float64
```

In [10]:
```
sns.set_theme(style="white")
plt.figure(figsize = (40,5), dpi = (100))
sns.jointplot(x = shaleporV1['TOC_%'], y = shaleporV1['Porosity_%'], kind='reg', line_kws={"color": "red"})
```

Out[10]:
```
<seaborn.axisgrid.JointGrid at 0x27c329bdfa0>
```

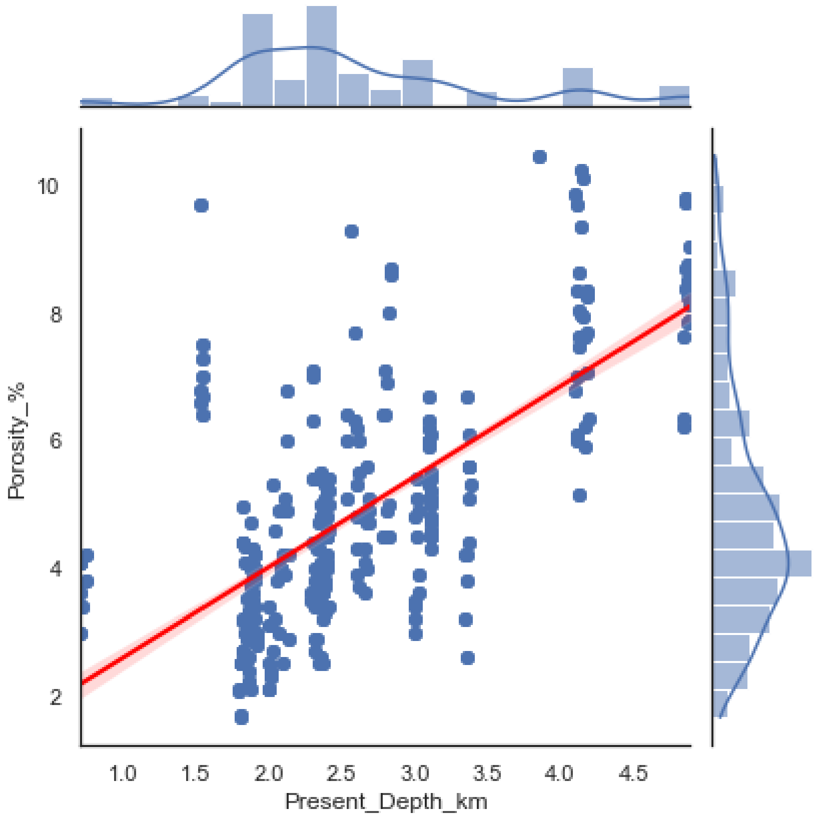
```
<Figure size 4000x500 with 0 Axes>
```



In [11]:
```
sns.set_theme(style="white")
plt.figure(figsize = (40,5), dpi = (100))
sns.jointplot(x = shaleporV1['Present_Depth_km'], y = shaleporV1['Porosity_%'], kind='reg', line_kws={"color": "red"})
```

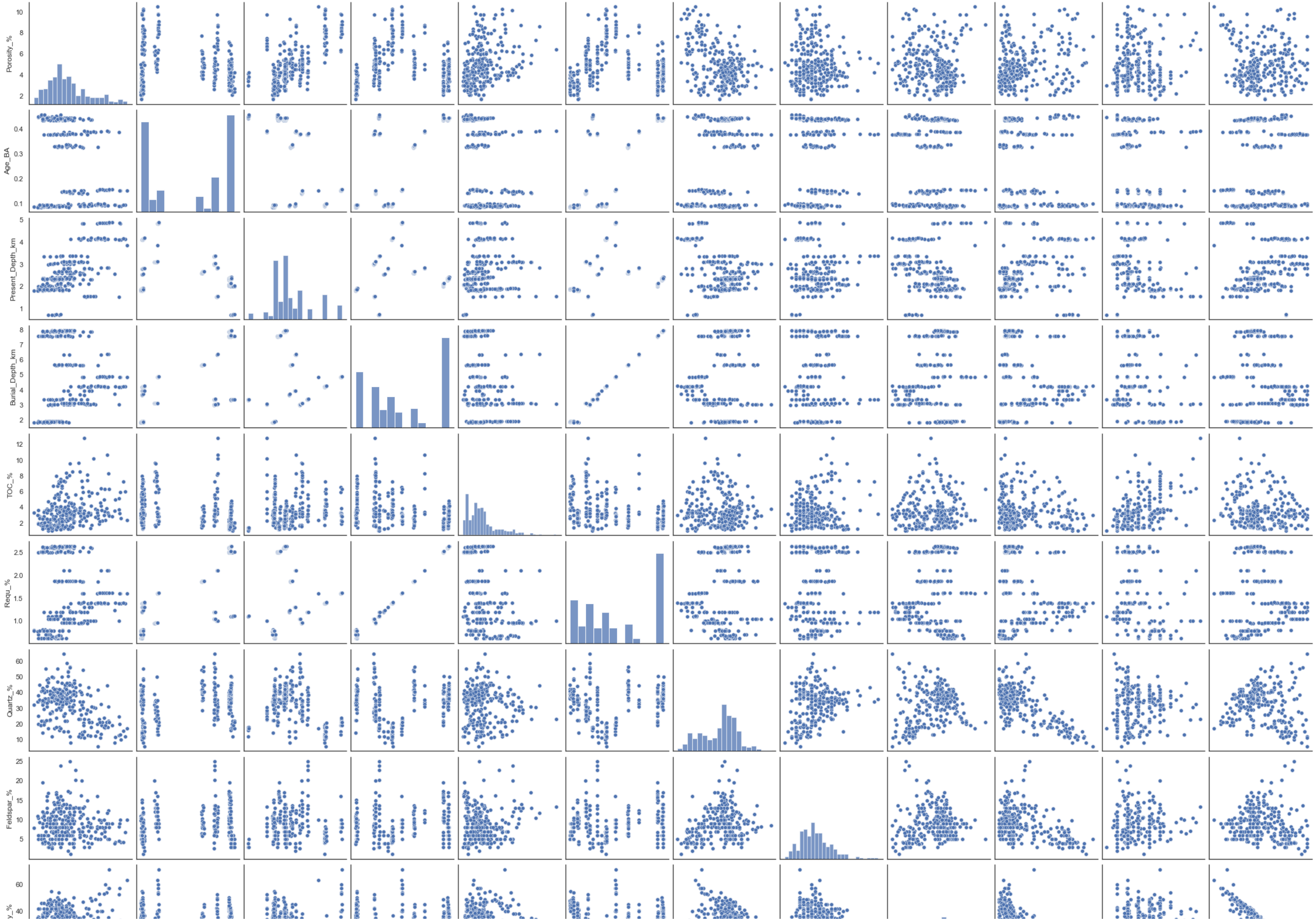
Out[11]: `<seaborn.axisgrid.JointGrid at 0x27c32c13220>`

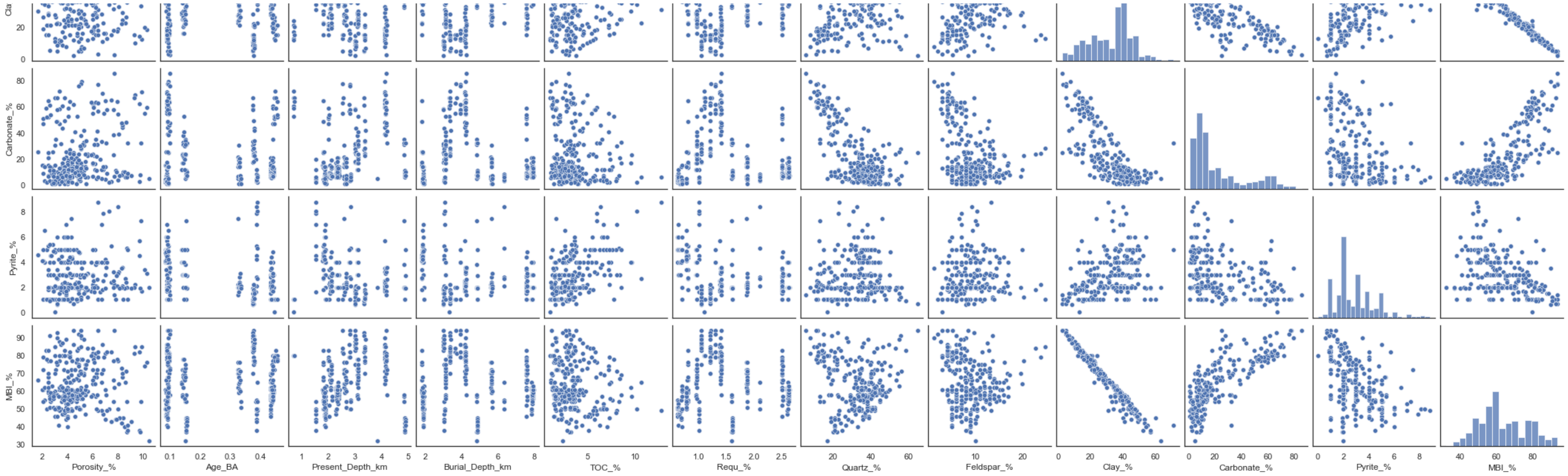
`<Figure size 4000x500 with 0 Axes>`



In [12]:
```
sns.pairplot(shaleporV1)
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x27c329c2790>





In [13]:
```
X = shaleporV1.iloc[:, 1:].values
y = shaleporV1.iloc[:, 0].values
```

In [14]:
```
X
```

Out[14]:
```
array([[ 0.325   ,  2.5851 ,  5.62762, ...,  4.7    ,  2.     , 70.     ],
       [ 0.326   ,  2.5892 ,  5.63064, ...,  6.8    ,  2.2    , 62.     ],
       [ 0.326   ,  2.5934 ,  5.63367, ..., 21.3    ,  1.4    , 76.     ],
       ...,
       [ 0.441   ,  2.125  ,  7.61241, ...,  7.7    ,  4.8    , 74.     ],
       [ 0.441   ,  2.13   ,  7.61544, ...,  6.7    ,  3.1    , 55.     ],
       [ 0.442   ,  2.135  ,  7.61846, ...,  6.6    ,  1.7    , 57.     ]])
```

In [15]:
```
y
```

Out[15]:
```
array([6.3, 7.7, 3.9, ..., 4.9, 4.2, 2.9])
```

In [16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [17]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [18]:
```
X_train
```

Out[18]:
```
array([[-1.18749835, -0.81269193, -1.29425205, ..., -0.49523445,
         1.9689647 , -1.05366511],
       [ 1.07785067, -0.30497404,  1.3042013 , ..., -0.20549509,
        -0.59086408, -0.20040124],
       [ 0.64788357,  0.40428948, -0.81735076, ...,  0.41261555,
        -0.6548598 ,  1.66126539],
       ...,
       [-1.16182867, -0.87653932, -1.31927208, ..., -0.42376541,
         1.02182805, -0.6658179 ],
       [-1.19391577, -0.8275605 , -1.30007864, ..., -0.83326371,
         1.9689647 , -1.28637344],
       [ 1.12919002, -0.67502648,  1.14735491, ...,  1.77439055,
        -1.23082127,  1.5061265 ]])
```

In [19]:
```
from tensorflow.keras.layers import Input, Dense, Activation,Dropout
from tensorflow.keras.models import Model
```

In [20]:
```
input_layer = Input(shape=(X.shape[1],))
dense_layer_1 = Dense(512, activation='relu')(input_layer)
dense_layer_2 = Dense(256, activation='relu')(dense_layer_1)
dense_layer_3 = Dense(128, activation='relu')(dense_layer_2)
dense_layer_4 = Dense(64, activation='relu')(dense_layer_3)
dense_layer_5 = Dense(64, activation='relu')(dense_layer_3)
output = Dense(1)(dense_layer_5)

model = Model(inputs=input_layer, outputs=output)
model.compile(loss="mean_squared_error" , optimizer="adam", metrics=["mean_squared_error"])
```

In [21]:
```
my_model = model
```

In [22]:
```
my_model.summary()
```

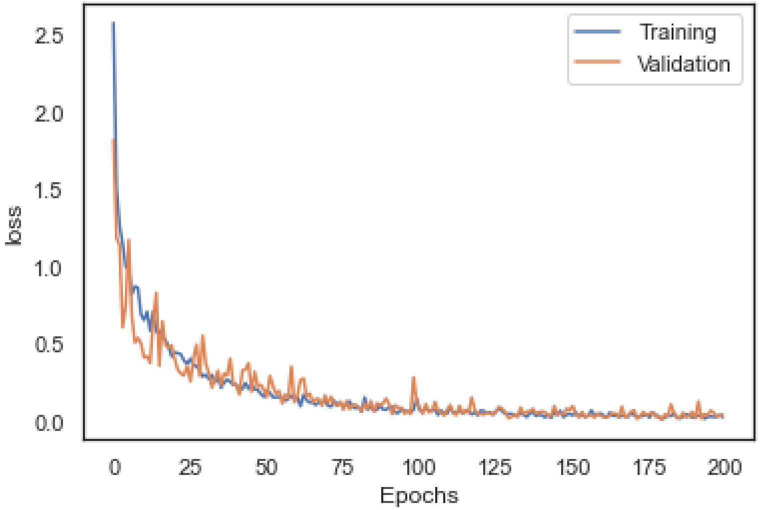
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 11)]              0         
_________________________________________________________________
dense (Dense)                (None, 512)               6144      
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328    
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896     
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65        
=================================================================
Total params: 178,689
Trainable params: 178,689
Non-trainable params: 0
_________________________________________________________________
```

In [23]:
```
history = model.fit(X_train, y_train, batch_size=1, epochs=200, verbose=1, validation_split=0.2)
```

```
Epoch 1/200
688/688 [==============================] - 4s 6ms/step - loss: 2.5764 - mean_squared_error: 2.5764 - val_loss: 1.8244
- val_mean_squared_error: 1.8244
Epoch 2/200
688/688 [==============================] - 4s 6ms/step - loss: 1.5105 - mean_squared_error: 1.5105 - val_loss: 1.1884
- val_mean_squared_error: 1.1884
Epoch 3/200
688/688 [==============================] - 4s 6ms/step - loss: 1.2654 - mean_squared_error: 1.2654 - val_loss: 1.1420
- val_mean_squared_error: 1.1420
Epoch 4/200
688/688 [==============================] - 6s 8ms/step - loss: 1.1515 - mean_squared_error: 1.1515 - val_loss: 0.6092
- val_mean_squared_error: 0.6092
Epoch 5/200
688/688 [==============================] - 7s 10ms/step - loss: 1.0014 - mean_squared_error: 1.0014 - val_loss: 0.750
3 - val_mean_squared_error: 0.7503
Epoch 6/200
688/688 [==============================] - 4s 6ms/step - loss: 0.9856 - mean_squared_error: 0.9856 - val_loss: 1.1763
- val_mean_squared_error: 1.1763
Epoch 7/200
```

In [24]:
```
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.xlabel("Epochs")
plt.ylabel("loss")
plt.legend(["Training","Validation"])
```

Out[24]: <matplotlib.legend.Legend at 0x27c3bfe36d0>



In [25]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

pred_train = model.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train)))

pred = model.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred)))
```

```
0.19449140667299072
0.24116353383621433
```

In [26]:
```
score = model.evaluate(X_test, y_test, verbose=1)

print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
9/9 [==============================] - 0s 12ms/step - loss: 0.0582 - mean_squared_error: 0.0582
Test Score: 0.058159865438938l4
Test Accuracy: 0.05815986543893814
```

In [27]:
```
predictions = model.predict(X_test)
np.set_printoptions(suppress=True)
print('Predicted labels: ', np.round(predictions)[:10])
print('Actual labels    : ' ,y_test[:10])
```

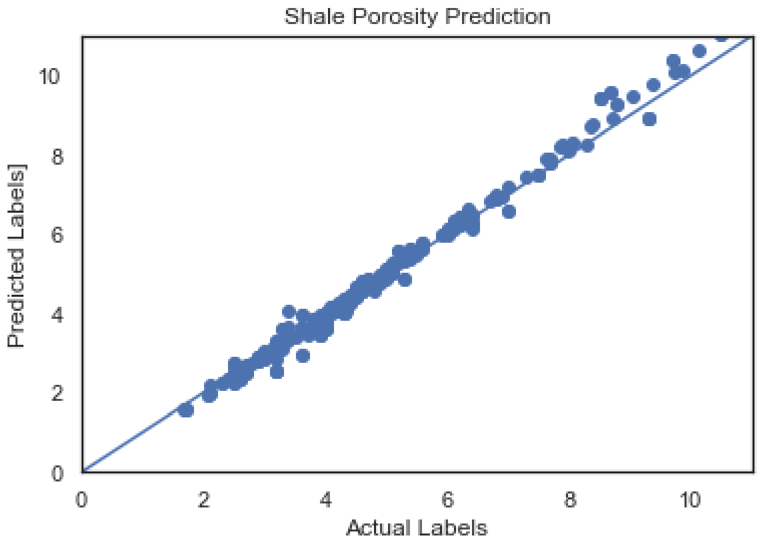
```
Predicted labels:  [[5.]
 [4.]
 [4.]
 [8.]
 [7.]
 [3.]
 [6.]
 [3.]
 [6.]
 [6.]]
Actual labels    :  [5.   3.6  3.3  7.48 7.5  3.9  6.1  3.4  5.6  5.2 ]
```

In [28]:
```
from sklearn.metrics import r2_score

y_true = np.round(predictions)
y_pred = y_test
r2_score(y_true, y_pred)
```

Out[28]:
```
0.9690701242717236
```

In [29]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels]')
plt.title('Shale Porosity Prediction')
lims = [0, 11]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```



In [30]:
```
my_model.save('./saved_models/my_tf_model')
```

```
WARNING:tensorflow:From C:\Users\samil\anaconda3\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111:
Model.state_updates (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future versio
n.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From C:\Users\samil\anaconda3\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111:
Layer.updates (from tensorflow.python.keras.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: ./saved_models/my_tf_model\assets
```

In [31]:
```
my_tf_saved_model = tf.keras.models.load_model(
    './saved_models/my_tf_model')
my_tf_saved_model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 11)]              0
_________________________________________________________________
dense (Dense)                (None, 512)               6144
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 178,689
Trainable params: 178,689
Non-trainable params: 0
_________________________________________________________________
```

In [32]:
```
from tensorflow.keras.models import save_model, load_model
import pandas as pd
```

In [33]:
```
model = load_model('./saved_models/my_tf_model',
        custom_objects=None,
    compile=True)
```

In [34]:
```
raw_dataset=pd.read_csv("eaglefordporositypredict.csv",sep=",")
```

In [35]:
```
 eaglefordporositypredict= raw_dataset.copy()
eaglefordporositypredict.head()
```

Out[35]:

	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_%	Pyrite_%	MBI_%
0	0.09	3.59	3.59	1.71	1.1	11.4	4.8	6.3	76.7	0.5	92
1	0.09	3.60	3.60	2.85	1.1	15.1	4.8	10.0	68.5	1.4	86
2	0.09	3.61	3.61	4.05	1.1	15.6	4.8	20.4	57.8	1.3	75
3	0.09	3.62	3.62	5.50	1.2	19.8	5.0	11.5	62.1	1.5	82
4	0.09	3.63	3.63	2.56	1.2	34.6	6.7	21.2	35.0	2.0	75

In [36]:
```
X_new = eaglefordporositypredict.iloc[:, 0:].values
```

In [37]:
```
X_new
```

Out[37]:
```
array([[ 0.09,  3.59,  3.59,  1.71,  1.1 , 11.4 ,  4.8 ,  6.3 , 76.7 ,
         0.5 , 92.  ],
       [ 0.09,  3.6 ,  3.6 ,  2.85,  1.1 , 15.1 ,  4.8 , 10.  , 68.5 ,
         1.4 , 86.  ],
       [ 0.09,  3.61,  3.61,  4.05,  1.1 , 15.6 ,  4.8 , 20.4 , 57.8 ,
         1.3 , 75.  ],
       [ 0.09,  3.62,  3.62,  5.5 ,  1.2 , 19.8 ,  5.  , 11.5 , 62.1 ,
         1.5 , 82.  ],
       [ 0.09,  3.63,  3.63,  2.56,  1.2 , 34.6 ,  6.7 , 21.2 , 35.  ,
         2.  , 75.  ],
       [ 0.09,  3.63,  3.63,  4.92,  1.2 , 17.5 ,  1.6 , 14.2 , 65.2 ,
         1.5 , 80.  ],
       [ 0.09,  3.64,  3.64,  3.72,  1.2 , 10.1 ,  0.9 , 20.3 , 67.  ,
         1.7 , 75.  ],
       [ 0.09,  3.64,  3.64,  4.63,  1.2 , 17.7 ,  1.4 , 39.6 , 36.9 ,
         3.1 , 54.  ],
       [ 0.09,  3.71,  3.71,  3.39,  1.1 , 16.1 ,  8.3 , 14.3 , 59.8 ,
         1.  , 82.  ],
       [ 0.09,  3.71,  3.71,  3.17,  1.1 , 15.5 ,  6.7 , 10.9 , 65.7 ,
         0.9 , 85.  ],
       [ 0.09,  3.71,  3.71,  3.33,  1.1 , 15.8 ,  6.3 , 12.2 , 63.9 ,
         1.1 , 84.  ],
       [ 0.09,  3.73,  3.73,  1.1 ,  1.2 ,  8.  ,  1.5 ,  1.6 , 88.7 ,
         0.2 , 97.  ],
       [ 0.09,  3.73,  3.73,  1.14,  1.2 ,  6.  ,  1.5 ,  3.2 , 89.  ,
         0.2 , 96.  ],
       [ 0.09,  3.75,  3.75,  3.29,  1.2 , 18.1 ,  4.5 ,  8.4 , 67.1 ,
         1.7 , 87.  ]])
```

In [38]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_new = sc.fit_transform(X_new)
```

In [39]:
```
X_new
```

Out[39]:
```
array([[ 0.        , -1.38384037, -1.38384037, -1.20071641, -1.15470054,
        -0.67849859,  0.25819889, -0.82258264,  0.82183007, -1.08402366,
         0.94922176],
       [ 0.        , -1.19755416, -1.19755416, -0.30606497, -1.15470054,
        -0.10886768,  0.25819889, -0.42022399,  0.26815582,  0.14648968,
         0.3714346 ],
       [ 0.        , -1.01126796, -1.01126796,  0.63567339, -1.15470054,
        -0.03189053,  0.25819889,  0.71073005, -0.45432155,  0.00976598,
        -0.68784186],
       [ 0.        , -0.82498176, -0.82498176,  1.77360725,  0.8660254 ,
         0.61471753,  0.34426519, -0.25710562, -0.16398018,  0.28321339,
        -0.01375684],
       [ 0.        , -0.63869555, -0.63869555, -0.53365174,  0.8660254 ,
         2.89324117,  1.07582871,  0.79772651, -1.99380606,  0.96683191,
        -0.68784186],
       [ 0.        , -0.63869555, -0.63869555,  1.3184337 ,  0.8660254 ,
         0.26062264, -1.11886186,  0.03650744,  0.0453357 ,  0.28321339,
        -0.20635256],
       [ 0.        , -0.45240935, -0.45240935,  0.37669534,  0.8660254 ,
        -0.87863918, -1.42009389,  0.69985549,  0.16687395,  0.5566608 ,
        -0.68784186],
       [ 0.        , -0.45240935, -0.45240935,  1.09084693,  0.8660254 ,
         0.2914135 , -1.20492815,  2.7986452 , -1.86551568,  2.47079267,
        -2.71009691],
       [ 0.        ,  0.85159407,  0.85159407,  0.1177173 , -1.15470054,
         0.04508662,  1.76435908,  0.047382  , -0.31927905, -0.40040514,
        -0.01375684],
       [ 0.        ,  0.85159407,  0.85159407, -0.05493474, -1.15470054,
        -0.04728596,  1.07582871, -0.32235297,  0.07909632, -0.53712884,
         0.27513674],
       [ 0.        ,  0.85159407,  0.85159407,  0.07063038, -1.15470054,
        -0.00109967,  0.90369611, -0.18098372, -0.04244193, -0.26368143,
         0.17883888],
       [ 0.        ,  1.22416648,  1.22416648, -1.67943341,  0.8660254 ,
        -1.20194321, -1.161895  , -1.33368687,  1.63208507, -1.49419478,
         1.43071106],
       [ 0.        ,  1.22416648,  1.22416648, -1.64804213,  0.8660254 ,
        -1.50985181, -1.161895  , -1.15969394,  1.65234145, -1.49419478,
         1.3344132 ],
```

```
       [ 0.        ,  1.59673889,  1.59673889,  0.0392391 ,  0.8660254 ,
         0.35299522,  0.12909944, -0.59421692,  0.17362607,  0.5566608 ,
         0.46773246]])
```

In [40]:
```
print(model.predict(X_new))
```

```
[[ 2.4959989]
 [ 3.4928994]
 [ 4.040115 ]
 [ 4.7786684]
 [ 6.332433 ]
 [ 5.6821074]
 [ 7.274669 ]
 [ 6.469673 ]
 [ 6.114649 ]
 [ 6.115079 ]
 [ 5.726995 ]
 [10.212651 ]
 [11.349706 ]
 [ 6.9119554]]
```

In [41]:
```
from tensorflow.keras.models import save_model, load_model
import pandas as pd
```

In [42]:
```
model = load_model('./saved_models/my_tf_model',
        custom_objects=None,
    compile=True
)
```

In [43]:
```
raw_dataset=pd.read_csv("dadasshaleporositypredict.csv",sep=",")
```

In [44]:
```
dadasshaleporositypredict = raw_dataset.copy()
dadasshaleporositypredict.head()
```

Out[44]:

	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_%	Pyrite_%	MBI_%
0	0.44	2.49	2.49	10.60	0.70	33.7	10.6	22.8	30.5	2.4	70
1	0.44	2.52	2.52	11.80	0.73	22.5	0.0	61.5	14.1	1.9	34
2	0.44	2.54	2.54	9.74	0.71	15.8	0.0	68.6	13.9	1.7	29
3	0.44	2.57	2.57	4.50	0.81	17.0	4.5	15.3	63.2	0.0	81
4	0.44	2.60	2.60	4.20	0.63	47.4	13.1	13.4	22.6	3.4	83

In [45]:
```
X_new = dadasshaleporositypredict.iloc[:, 0:].values
```

In [46]:
```
X_new
```

Out[46]:
```
array([[ 0.44,  2.49,  2.49, 10.6 ,  0.7 , 33.7 , 10.6 , 22.8 , 30.5 ,
         2.4 , 70.  ],
       [ 0.44,  2.52,  2.52, 11.8 ,  0.73, 22.5 ,  0.  , 61.5 , 14.1 ,
         1.9 , 34.  ],
       [ 0.44,  2.54,  2.54,  9.74,  0.71, 15.8 ,  0.  , 68.6 , 13.9 ,
         1.7 , 29.  ],
       [ 0.44,  2.57,  2.57,  4.5 ,  0.81, 17.  ,  4.5 , 15.3 , 63.2 ,
         0.  , 81.  ],
       [ 0.44,  2.6 ,  2.6 ,  4.2 ,  0.63, 47.4 , 13.1 , 13.4 , 22.6 ,
         3.4 , 83.  ],
       [ 0.44,  2.62,  2.62,  7.1 ,  0.68, 25.  ,  0.  , 63.3 , 11.8 ,
         0.  , 34.  ],
       [ 0.44,  2.64,  2.64,  5.08,  0.71, 32.2 , 11.8 , 23.1 , 32.9 ,
         0.  , 73.  ],
       [ 0.44,  3.22,  3.22,  2.35,  0.76, 14.5 ,  0.  , 21.1 , 62.2 ,
         2.2 , 77.  ],
       [ 0.44,  3.22,  3.22,  3.25,  0.76, 20.5 ,  0.  , 23.7 , 53.6 ,
         2.2 , 74.  ],
       [ 0.44,  3.23,  3.23,  4.4 ,  0.84, 33.9 ,  0.  , 39.5 , 22.3 ,
         4.3 , 58.  ],
       [ 0.44,  3.25,  3.25,  6.68,  0.84, 13.1 ,  0.  , 20.2 , 63.1 ,
         3.6 , 75.  ],
       [ 0.44,  3.26,  3.26,  6.3 ,  0.84, 22.9 ,  7.5 , 21.4 , 46.1 ,
         2.1 , 74.  ],
       [ 0.44,  3.28,  3.28,  5.05,  0.81, 16.7 ,  0.  , 69.9 , 10.8 ,
         2.5 , 29.  ],
       [ 0.44,  3.31,  3.31,  4.16,  0.81, 31.1 ,  0.  , 37.3 , 27.5 ,
         4.1 , 60.  ],
       [ 0.44,  3.33,  3.33,  4.67,  0.86, 17.1 ,  0.  , 66.7 , 15.2 ,
         1.1 , 32.  ],
       [ 0.44,  3.33,  3.33,  3.84,  0.83, 17.3 ,  5.6 , 71.3 ,  4.  ,
         1.7 , 28.  ],
       [ 0.44,  3.34,  3.34,  3.41,  0.83, 20.1 ,  0.  , 73.9 ,  3.8 ,
         2.2 , 25.  ],
       [ 0.44,  3.36,  3.36,  3.24,  0.84, 26.2 ,  0.  , 64.9 ,  7.2 ,
         1.7 , 34.  ],
       [ 0.44,  3.23,  3.23,  4.32,  0.81,  8.9 ,  0.  , 25.4 , 65.7 ,
         0.  , 72.  ],
       [ 0.44,  3.24,  3.24,  5.18,  0.81, 18.6 ,  0.  , 58.1 , 20.8 ,
```

```
        2.5 , 40.  ],
      [ 0.44,  3.25,  3.25,  3.95,  0.88, 29.2 ,  0.  , 36.2 , 34.2 ,
        0.5 , 61.  ],
      [ 0.44,  3.25,  3.25,  2.68,  0.88, 30.  ,  0.  , 43.4 , 26.6 ,
        0.  , 55.  ],
      [ 0.44,  3.26,  3.26,  2.64,  0.88, 15.6 ,  0.  , 47.  , 37.5 ,
        0.  , 52.  ],
      [ 0.44,  3.27,  3.27,  2.27,  0.84, 41.4 ,  0.  , 45.8 , 10.9 ,
        1.9 , 53.  ],
      [ 0.44,  3.28,  3.28,  1.53,  0.83, 36.3 ,  0.  , 20.4 , 43.2 ,
        0.  , 78.  ],
      [ 0.44,  3.29,  3.29,  2.25,  0.88, 28.6 ,  0.  , 64.1 ,  4.9 ,
        2.4 , 35.  ],
      [ 0.44,  3.31,  3.31,  2.96,  0.86, 47.7 , 12.4 , 36.2 ,  0.  ,
        3.8 , 62.  ],
      [ 0.44,  3.32,  3.32,  3.44,  0.89, 33.2 ,  0.  , 64.4 ,  0.  ,
        2.4 , 34.  ]])
```

In [47]:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_new = sc.fit_transform(X_new)
```

In [48]:
```
X_new
```

Out[48]:
```
array([[ 1.        , -1.96798131, -1.96798131,  2.38205998, -1.55282516,
         0.81009603,  1.89583859, -1.03103887,  0.18487442,  0.44871421,
         0.82876607],
       [ 1.        , -1.87125198, -1.87125198,  2.86651743, -1.11066477,
        -0.30855705, -0.53686739,  0.89356702, -0.62057467,  0.07028054,
        -1.03595759],
       [ 1.        , -1.80676575, -1.80676575,  2.03486547, -1.40543836,
        -0.9777513 , -0.53686739,  1.24666009, -0.63039722, -0.08109293,
        -1.29494699],
       [ 1.        , -1.71003642, -1.71003642, -0.08059872,  0.06842958,
        -0.85789562,  0.49588515, -1.40402451,  1.79086134, -1.3677674 ,
         1.39854274],
       [ 1.        , -1.61330709, -1.61330709, -0.20171309, -2.58453272,
         2.17844845,  2.46959   , -1.4985142 , -0.2031163 ,  1.20558154,
         1.5021385 ],
       [ 1.        , -1.54882087, -1.54882087,  0.96905908, -1.84759875,
        -0.0588577 , -0.53686739,  0.98308357, -0.733534  , -1.3677674 ,
        -1.03595759],
       [ 1.        , -1.48433464, -1.48433464,  0.15355571, -1.40543836,
         0.66027642,  2.17123927, -1.01611944,  0.30274502, -1.3677674 ,
         0.98415971],
       [ 1.        ,  0.38576579,  0.38576579, -0.94858499, -0.66850439,
        -1.10759496, -0.53686739, -1.11558228,  1.74174859,  0.29734074,
         1.19135123],
       [ 1.        ,  0.38576579,  0.38576579, -0.5852419 , -0.66850439,
        -0.50831653, -0.53686739, -0.98628059,  1.31937894,  0.29734074,
         1.03595759],
       [ 1.        ,  0.4180089 ,  0.4180089 , -0.12097018,  0.51058997,
         0.83007197, -0.53686739, -0.20052418, -0.21785013,  1.88676215,
         0.20719152],
       [ 1.        ,  0.48249512,  0.48249512,  0.79949898,  0.51058997,
        -1.2474266 , -0.53686739, -1.16034056,  1.78595006,  1.35695501,
         1.08775547],
       [ 1.        ,  0.51473824,  0.51473824,  0.64608745,  0.51058997,
        -0.26860516,  1.18438684, -1.10066286,  0.95103332,  0.22165401,
         1.03595759],
       [ 1.        ,  0.57922446,  0.57922446,  0.14144427,  0.06842958,
        -0.88785954, -0.53686739,  1.31131094, -0.78264675,  0.52440094,
        -1.29494699],
```

```
       [ 1.        ,  0.67595379,  0.67595379, -0.21786167,  0.06842958,
         0.5504087 , -0.53686739, -0.3099333 ,  0.03753617,  1.73538868,
         0.31078728],
       [ 1.        ,  0.74044001,  0.74044001, -0.01196725,  0.80536356,
        -0.84790764, -0.53686739,  1.1521704 , -0.56655065, -0.53521333,
        -1.13955335],
       [ 1.        ,  0.74044001,  0.74044001, -0.34705032,  0.36320317,
        -0.82793169,  0.74833577,  1.38093492, -1.11661344, -0.08109293,
        -1.34674487],
       [ 1.        ,  0.77268312,  0.77268312, -0.52064757,  0.36320317,
        -0.54826842, -0.53686739,  1.51023661, -1.12643599,  0.29734074,
        -1.5021385 ],
       [ 1.        ,  0.83716935,  0.83716935, -0.58927905,  0.51058997,
         0.06099798, -0.53686739,  1.06265384, -0.95945265, -0.08109293,
        -1.03595759],
       [ 1.        ,  0.4180089 ,  0.4180089 , -0.15326734,  0.06842958,
        -1.6669215 , -0.53686739, -0.90173718,  1.91364321, -1.3677674 ,
         0.93236183],
       [ 1.        ,  0.45025201,  0.45025201,  0.19392716,  0.06842958,
        -0.69808803, -0.53686739,  0.7244802 , -0.29151925,  0.52440094,
        -0.72517031],
       [ 1.        ,  0.48249512,  0.48249512, -0.30264172,  1.10013714,
         0.3606372 , -0.53686739, -0.36463786,  0.3665916 , -0.98933373,
         0.36258516],
       [ 1.        ,  0.48249512,  0.48249512, -0.81535919,  1.10013714,
         0.44054099, -0.53686739, -0.00657165, -0.0066653 , -1.3677674 ,
         0.05179788],
       [ 1.        ,  0.51473824,  0.51473824, -0.83150777,  1.10013714,
        -0.99772725, -0.53686739,  0.17246145,  0.52866367, -1.3677674 ,
        -0.10359576],
       [ 1.        ,  0.54698135,  0.54698135, -0.98088215,  0.51058997,
         1.57917002, -0.53686739,  0.11278375, -0.77773547,  0.07028054,
        -0.05179788],
       [ 1.        ,  0.57922446,  0.57922446, -1.27963091,  0.36320317,
         1.06978335, -0.53686739, -1.15039427,  0.80860634, -1.3677674 ,
         1.24314911],
       [ 1.        ,  0.61146757,  0.61146757, -0.98895644,  1.10013714,
         0.30070936, -0.53686739,  1.02286871, -1.07241197,  0.44871421,
        -0.98415971],
       [ 1.        ,  0.67595379,  0.67595379, -0.70231912,  0.80536356,
         2.20841237,  2.30893961, -0.36463786, -1.31306444,  1.50832848,
         0.41438304],
```

```
       [ 1.        ,  0.7081969 ,  0.7081969 , -0.50853614,  1.24752394,
         0.76015616, -0.53686739,  1.03778813, -1.31306444,  0.44871421,
        -1.03595759]])
```

In [49]:

```python
print(model.predict(X_new))
```

```
[[ 4.7260814]
 [ 8.461251 ]
 [10.225585 ]
 [ 3.7220874]
 [ 3.0028062]
 [ 5.840834 ]
 [ 3.3608336]
 [ 2.9984736]
 [ 3.2286205]
 [ 5.4510465]
 [ 9.151768 ]
 [ 5.58838  ]
 [ 8.749031 ]
 [ 5.495323 ]
 [ 7.853764 ]
 [ 7.124008 ]
 [ 7.7048497]
 [ 4.9856296]
 [ 2.4334407]
 [ 6.9832726]
 [ 4.024339 ]
 [ 6.2241836]
 [ 8.51217  ]
 [ 5.1714935]
 [ 3.2360349]
 [ 5.1440187]
 [ 5.540882 ]
 [ 3.596858 ]]
```

In [50]:

```python
from tensorflow.keras.models import save_model, load_model
import pandas as pd
```

In [51]:
```
model = load_model('./saved_models/my_tf_model',
        custom_objects=None,
    compile=True
)
```

In [52]:
```
raw_dataset=pd.read_csv("niobraraporositypredict.csv",sep=",")
```

In [53]:
```
niobraraporositypredict= raw_dataset.copy()
niobraraporositypredict.head()
```

Out[53]:

	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_%	Pyrite_%	MBI_%
0	0.085	2.085619	2.085619	2.54	0.76	35.23	6.81	43.45	4	4.01	50.27
1	0.085	2.096409	2.096409	4.46	0.71	32.46	10.77	43.59	2	7.29	48.64
2	0.085	2.102993	2.102993	4.33	0.76	34.86	5.11	41.56	9	4.85	51.65
3	0.085	2.111467	2.111467	4.73	0.76	12.78	2.39	16.67	51	6.38	75.56
4	0.085	2.120733	2.120733	3.74	0.74	9.28	2.54	11.39	63	3.09	83.21

In [54]:
```
X_new = niobraraporositypredict.iloc[:, 0:].values
```

In [55]:
```
X_new
```

Out[55]:
```
array([[ 0.085   ,  2.085619,  2.085619,  2.54    ,  0.76    , 35.23    ,
         6.81   , 43.45    ,  4.      ,  4.01    , 50.27    ],
       [ 0.085   ,  2.096409,  2.096409,  4.46    ,  0.71    , 32.46    ,
        10.77   , 43.59    ,  2.      ,  7.29    , 48.64    ],
       [ 0.085   ,  2.102993,  2.102993,  4.33    ,  0.76    , 34.86    ,
         5.11   , 41.56    ,  9.      ,  4.85    , 51.65    ],
       [ 0.085   ,  2.111467,  2.111467,  4.73    ,  0.76    , 12.78    ,
         2.39   , 16.67    , 51.      ,  6.38    , 75.56    ],
       [ 0.085   ,  2.120733,  2.120733,  3.74    ,  0.74    ,  9.28    ,
         2.54   , 11.39    , 63.      ,  3.09    , 83.21    ],
       [ 0.085   ,  2.128079,  2.128079,  3.68    ,  0.71    ,  9.65    ,
         2.15   ,  9.31    , 68.      ,  2.26    , 86.01    ],
       [ 0.085   ,  2.132132,  2.132132,  7.77    ,  0.78    ,  7.6     ,
         2.96   , 13.74    , 53.      ,  5.83    , 74.58    ],
       [ 0.085   ,  2.133474,  2.133474,  6.9     ,  0.74    ,  9.38    ,
         3.59   , 15.99    , 52.      ,  5.98    , 73.84    ],
       [ 0.085   ,  2.136308,  2.136308, 10.3     ,  0.72    ,  7.19    ,
         3.31   , 15.45    , 52.      ,  6.03    , 70.96    ],
       [ 0.085   ,  2.142801,  2.142801,  1.87    ,  0.8     , 12.96    ,
         2.84   , 21.12    , 55.      ,  1.58    , 75.61    ],
       [ 0.085   ,  2.146214,  2.146214,  2.11    ,  0.8     , 13.42    ,
         2.53   , 13.64    , 61.      ,  2.89    , 82.93    ],
       [ 0.085   ,  2.151487,  2.151487,  2.32    ,  0.83    , 19.31    ,
         2.78   , 18.75    , 48.      ,  1.66    , 76.99    ],
       [ 0.085   ,  2.161272,  2.161272,  2.92    ,  0.81    , 16.09    ,
         2.32   , 15.9     , 55.      ,  1.3     , 79.7     ],
       [ 0.085   ,  2.165539,  2.165539,  4.27    ,  0.81    , 20.98    ,
         4.16   , 20.32    , 40.      ,  2.45    , 72.55    ],
       [ 0.085   ,  2.172854,  2.172854,  6.45    ,  0.8     , 19.21    ,
         8.05   , 19.18    , 38.      ,  6.33    , 71.74    ],
       [ 0.085   ,  2.174012,  2.174012,  7.3     ,  0.8     , 30.03    ,
         6.1    , 21.13    , 25.      ,  6.02    , 68.3     ],
       [ 0.085   ,  2.198122,  2.198122,  1.11    ,  0.78    , 27.59    ,
         6.94   , 28.23    , 29.      ,  0.94    , 68.43    ],
       [ 0.085   ,  2.199098,  2.199098,  1.42    ,  0.8     , 28.3     ,
         7.32   , 29.26    , 25.      ,  1.19    , 66.51    ],
       [ 0.085   ,  1.70492 ,  1.70492 ,  4.31    ,  0.71    , 28.56    ,
        10.44   , 41.69    ,  4.      ,  9.62    , 48.32    ],
       [ 0.085   ,  1.708851,  1.708851,  6.01    ,  0.63    , 40.69    ,
```

```
        11.16    , 33.99    ,  3.      ,  8.5     , 57.87    ],
       [ 0.085    ,  1.711442,  1.711442,  4.91    ,  0.81    , 42.21    ,
         6.18    , 46.45    ,  0.      ,  2.74    , 48.56    ],
       [ 0.085    ,  1.716563,  1.716563,  3.61    ,  0.71    ,  9.78    ,
         2.88    , 12.63    , 60.      ,  9.32    , 81.65    ],
       [ 0.085    ,  1.720769,  1.720769,  5.45    ,  0.67    , 12.37    ,
         5.58    , 19.      , 44.      , 10.1     , 71.55    ],
       [ 0.085    ,  1.724275,  1.724275,  3.9     ,  0.71    ,  9.17    ,
         8.83    , 34.47    , 28.      , 14.99    , 54.69    ],
       [ 0.085    ,  1.730523,  1.730523,  3.79    ,  0.67    , 11.55    ,
         2.83    , 12.24    , 58.      ,  7.48    , 81.96    ],
       [ 0.085    ,  1.735339,  1.735339,  5.48    ,  0.74    , 10.99    ,
         4.57    , 16.48    , 52.      ,  5.02    , 75.54    ],
       [ 0.085    ,  1.737777,  1.737777,  3.08    ,  0.71    ,  5.47    ,
         1.66    ,  6.55    , 74.      ,  1.93    , 89.43    ],
       [ 0.085    ,  1.746921,  1.746921,  8.54    ,  0.8     , 10.23    ,
         5.73    , 24.17    , 35.      , 11.6     , 60.71    ],
       [ 0.085    ,  1.748202,  1.748202,  6.89    ,  0.72    , 13.48    ,
         5.1     , 26.48    , 35.      ,  7.22    , 61.66    ],
       [ 0.085    ,  1.74936 ,  1.74936 ,  9.47    ,  0.78    ,  7.04    ,
         3.75    , 15.55    , 54.      ,  5.91    , 72.01    ],
       [ 0.085    ,  1.754237,  1.754237,  3.33    ,  0.8     ,  7.34    ,
         2.63    , 16.09    , 62.      ,  2.25    , 78.85    ],
       [ 0.085    ,  1.760241,  1.760241,  2.75    ,  0.78    , 11.      ,
         1.58    , 10.31    , 69.      ,  1.87    , 86.17    ],
       [ 0.085    ,  1.766155,  1.766155,  2.45    ,  0.78    ,  5.09    ,
         0.49    ,  6.38    , 82.      ,  1.41    , 90.86    ],
       [ 0.085    ,  1.767069,  1.767069,  3.18    ,  0.76    , 13.19    ,
         1.61    , 13.2     , 63.      ,  2.72    , 82.55    ],
       [ 0.085    ,  1.770605,  1.770605,  3.92    ,  0.8     ,  9.74    ,
         1.92    , 13.19    , 61.      ,  3.07    , 81.01    ],
       [ 0.085    ,  1.778011,  1.778011,  2.43    ,  0.74    , 18.11    ,
         2.97    , 18.49    , 50.      ,  2.16    , 77.32    ],
       [ 0.085    ,  1.781334,  1.781334,  2.36    ,  0.81    , 19.97    ,
         3.24    , 17.9     , 49.      ,  2.26    , 78.05    ],
       [ 0.085    ,  1.785205,  1.785205,  4.71    ,  0.8     , 24.78    ,
         3.67    , 17.05    , 41.      ,  4.4     , 76.25    ],
       [ 0.085    ,  1.791728,  1.791728,  6.35    ,  0.74    , 25.22    ,
         4.82    , 14.69    , 42.      ,  6.38    , 77.36    ],
       [ 0.085    ,  1.815441,  1.815441,  1.09    ,  0.83    , 39.87    ,
         9.8     , 31.24    , 11.      ,  1.28    , 65.41    ],
```

```
        [ 0.085    ,  1.819952,  1.819952,  1.21     ,  0.81     , 34.88    ,
          8.85     , 34.63     , 13.       ,  1.09     , 61.39     ]])
```

In [56]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_new = sc.fit_transform(X_new)
```

In [57]:
```
X_new
```

Out[57]:
```
array([[ 0.        ,  0.82637881,  0.82637881, -0.79593897, -0.03090813,
         1.55225242,  0.75490946,  2.04905265, -1.74128103, -0.21726524,
        -1.85025107],
       [ 0.        ,  0.88154559,  0.88154559,  0.05885388, -1.08693596,
         1.29578684,  2.17597727,  2.06212251, -1.8330452 ,  0.79059366,
        -1.99147435],
       [ 0.        ,  0.91520806,  0.91520806,  0.00097728, -0.03090813,
         1.51799528,  0.14485509,  1.87260961, -1.51187061,  0.04084497,
        -1.7306878 ],
       [ 0.        ,  0.95853366,  0.95853366,  0.17905912, -0.03090813,
        -0.52632237, -0.8312319 , -0.45102384,  0.4151769 ,  0.51097427,
         0.34087576],
       [ 0.        ,  1.00590858,  1.00590858, -0.26169344, -0.45331926,
        -0.85037635, -0.77740357, -0.94394407,  0.96576191, -0.49995737,
         1.00367214],
       [ 0.        ,  1.04346698,  1.04346698, -0.28840572, -1.08693596,
        -0.81611921, -0.91735722, -1.13812477,  1.19517233, -0.75499484,
         1.24626427],
       [ 0.        ,  1.06418903,  1.06418903,  1.53248112,  0.391503  ,
```

In [58]:
```
print(model.predict(X_new))
```

```
[[ 5.1572604]
 [ 6.771973 ]
 [ 4.899841 ]
 [ 7.217488 ]
 [ 6.2704635]
 [ 6.7040405]
 [10.921588 ]
 [ 8.559395 ]
 [ 9.411206 ]
 [ 9.043764 ]
 [ 7.7415957]
 [ 7.570786 ]
 [ 7.3525257]
 [ 4.386225 ]
 [ 7.280806 ]
 [ 7.062373 ]
 [ 4.951488 ]
 [ 4.7500696]
 [ 3.3394938]
 [ 5.378327 ]
 [ 3.7587953]
 [ 3.7122278]
 [ 2.639886 ]
 [ 3.360816 ]
 [ 3.7227616]
 [ 5.086318 ]
 [ 3.7258735]
 [ 6.1243067]
 [ 5.978765 ]
 [ 7.427363 ]
 [ 6.0369115]
 [ 3.3170733]
 [ 3.340549 ]
 [ 3.633749 ]
 [ 5.813089 ]
 [ 3.204146 ]
 [ 2.878603 ]
 [ 2.8228621]
 [ 2.6095471]
```

```
 [ 4.873797 ]
 [ 4.1978693]]
```

In [1]:
```
from tensorflow.keras.models import save_model, load_model
import pandas as pd
```

In [2]:
```
model = load_model('./saved_models/my_tf_model',
        custom_objects=None,
    compile=True
)
```

In [4]:
```
raw_dataset=pd.read_csv("niobrarawell3porosityprediction.csv",sep=",")
```

In [5]:
```
niobrarawell3porosityprediction= raw_dataset.copy()
niobrarawell3porosityprediction.head()
```

Out[5]:

	Age_BA	Present_Depth_km	Burial_Depth_km	TOC_%	Requ_%	Quartz_%	Feldspar_%	Clay_%	Carbonate_%	Pyrite_%	MBI_%
0	0.085	1.985	1.985	5.30	0.91	26.98	9.19	50.3	4	9.93	47.21
1	0.085	1.986	1.986	6.74	0.91	28.94	8.15	50.1	5	7.57	46.75
2	0.085	1.992	1.992	5.20	0.91	17.72	4.59	27.7	44	5.72	68.72
3	0.085	1.993	1.993	5.59	0.91	31.13	7.56	51.5	6	3.67	45.92
4	0.085	1.995	1.995	3.25	0.91	11.52	3.55	19.2	59	6.59	78.25

In [7]:
```
X_new = niobrarawell3porosityprediction.iloc[:, 0:].values
```

In [8]:
```
X_new
```

Out[8]:
```
array([[8.500e-02, 1.985e+00, 1.985e+00, 5.300e+00, 9.100e-01, 2.698e+01,
         9.190e+00, 5.030e+01, 4.000e+00, 9.930e+00, 4.721e+01],
        [8.500e-02, 1.986e+00, 1.986e+00, 6.740e+00, 9.100e-01, 2.894e+01,
         8.150e+00, 5.010e+01, 5.000e+00, 7.570e+00, 4.675e+01],
        [8.500e-02, 1.992e+00, 1.992e+00, 5.200e+00, 9.100e-01, 1.772e+01,
         4.590e+00, 2.770e+01, 4.400e+01, 5.720e+00, 6.872e+01],
        [8.500e-02, 1.993e+00, 1.993e+00, 5.590e+00, 9.100e-01, 3.113e+01,
         7.560e+00, 5.150e+01, 6.000e+00, 3.670e+00, 4.592e+01],
        [8.500e-02, 1.995e+00, 1.995e+00, 3.250e+00, 9.100e-01, 1.152e+01,
         3.550e+00, 1.920e+01, 5.900e+01, 6.590e+00, 7.825e+01],
        [8.500e-02, 2.001e+00, 2.001e+00, 1.610e+00, 9.100e-01, 2.980e+00,
         0.000e+00, 6.800e+00, 8.800e+01, 2.590e+00, 9.172e+01],
        [8.500e-02, 2.002e+00, 2.002e+00, 1.210e+00, 9.100e-01, 2.040e+00,
         0.000e+00, 2.900e+00, 9.300e+01, 1.620e+00, 9.594e+01],
        [8.500e-02, 2.003e+00, 2.003e+00, 1.280e+00, 9.100e-01, 2.270e+00,
         0.000e+00, 7.600e+00, 8.800e+01, 1.840e+00, 9.124e+01],
        [8.500e-02, 2.004e+00, 2.004e+00, 1.210e+00, 9.100e-01, 2.760e+00,
         0.000e+00, 5.800e+00, 9.000e+01, 1.760e+00, 9.308e+01],
        [8.500e-02, 2.004e+00, 2.004e+00, 1.610e+00, 9.100e-01, 2.840e+00,
         0.000e+00, 6.800e+00, 8.800e+01, 2.450e+00, 9.172e+01],
        [8.500e-02, 2.007e+00, 2.007e+00, 3.820e+00, 9.100e-01, 1.071e+01,
         0.000e+00, 1.820e+01, 6.300e+01, 7.910e+00, 7.879e+01],
        [8.500e-02, 2.010e+00, 2.010e+00, 3.150e+00, 9.100e-01, 1.023e+01,
         0.000e+00, 2.050e+01, 6.400e+01, 5.500e+00, 7.706e+01],
        [8.500e-02, 2.020e+00, 2.020e+00, 2.870e+00, 9.100e-01, 1.403e+01,
         0.000e+00, 2.850e+01, 5.400e+01, 3.670e+00, 6.949e+01],
        [8.500e-02, 2.026e+00, 2.026e+00, 2.020e+00, 9.100e-01, 2.600e+00,
         0.000e+00, 7.900e+00, 8.900e+01, 7.300e-01, 9.027e+01],
        [8.500e-02, 2.029e+00, 2.029e+00, 1.510e+00, 9.100e-01, 1.830e+00,
         0.000e+00, 4.800e+00, 9.300e+01, 2.700e-01, 9.378e+01],
        [8.500e-02, 2.029e+00, 2.029e+00, 1.530e+00, 9.100e-01, 1.770e+00,
         0.000e+00, 2.600e+00, 9.500e+01, 4.700e-01, 9.593e+01],
        [8.500e-02, 2.030e+00, 2.030e+00, 1.590e+00, 9.100e-01, 2.090e+00,
         0.000e+00, 3.900e+00, 9.400e+01, 3.300e-01, 9.460e+01],
        [8.500e-02, 2.030e+00, 2.030e+00, 1.540e+00, 9.100e-01, 1.440e+00,
         0.000e+00, 3.200e+00, 9.500e+01, 2.400e-01, 9.533e+01],
        [8.500e-02, 2.033e+00, 2.033e+00, 3.770e+00, 9.100e-01, 1.287e+01,
         3.330e+00, 2.420e+01, 5.700e+01, 3.010e+00, 7.303e+01],
        [8.500e-02, 2.037e+00, 2.037e+00, 3.220e+00, 9.100e-01, 8.560e+00,
```

```
         1.780e+00, 1.800e+01, 7.000e+01, 1.670e+00, 7.945e+01],
        [8.500e-02, 2.040e+00, 2.040e+00, 2.060e+00, 9.100e-01, 7.600e+00,
         0.000e+00, 1.150e+01, 8.000e+01, 8.400e-01, 8.671e+01],
        [8.500e-02, 2.046e+00, 2.046e+00, 1.740e+00, 9.100e-01, 4.150e+00,
         0.000e+00, 7.100e+00, 8.800e+01, 5.400e-01, 9.131e+01],
        [8.500e-02, 2.052e+00, 2.052e+00, 2.690e+00, 9.100e-01, 1.310e+01,
         0.000e+00, 2.700e+01, 5.700e+01, 3.170e+00, 7.108e+01],
        [8.500e-02, 2.058e+00, 2.058e+00, 4.760e+00, 9.100e-01, 1.674e+01,
         0.000e+00, 3.270e+01, 4.700e+01, 3.850e+00, 6.423e+01],
        [8.500e-02, 2.059e+00, 2.059e+00, 4.450e+00, 9.100e-01, 1.572e+01,
         0.000e+00, 2.880e+01, 5.200e+01, 3.800e+00, 6.818e+01],
        [8.500e-02, 2.062e+00, 2.062e+00, 2.460e+00, 9.100e-01, 7.380e+00,
         0.000e+00, 1.350e+01, 7.800e+01, 1.050e+00, 8.442e+01],
        [8.500e-02, 2.062e+00, 2.062e+00, 1.460e+00, 9.100e-01, 8.790e+00,
         0.000e+00, 1.280e+01, 7.800e+01, 7.700e-01, 8.594e+01]])
```

In [9]:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_new = sc.fit_transform(X_new)
```

In [10]:
```
X_new
```

Out[10]:
```
array([[ 1.        , -1.54029275, -1.54029275,  1.56321614,  0.        ,
         2.01904587,  2.84289385,  2.19647909, -2.32905765,  2.66043378,
        -2.1471308 ],
       [ 1.        , -1.49870485, -1.49870485,  2.49168915,  0.        ,
         2.25149068,  2.46272212,  2.18275427, -2.29230455,  1.75170775,
        -2.17759108],
       [ 1.        , -1.24917742, -1.24917742,  1.49873884,  0.        ,
         0.92086272,  1.16136505,  0.64557474, -0.85893359,  1.03935896,
        -0.72278169],
       [ 1.        , -1.20758952, -1.20758952,  1.75020028,  0.        ,
         2.51121218,  2.24704777,  2.27882799, -2.25555145,  0.24999949,
        -2.23255202],
       [ 1.        , -1.12441371, -1.12441371,  0.24143164,  0.        ,
         0.18557811,  0.78119333,  0.06227   , -0.30763707,  1.37435542,
        -0.09172419],
       [ 1.        , -0.87488628, -0.87488628, -0.81599596,  0.        ,
        -0.82721715, -0.51650824, -0.78866867,  0.75820287, -0.16585819,
         0.80023223],
       [ 1.        , -0.83329838, -0.83329838, -1.07390513,  0.        ,
        -0.93869579, -0.51650824, -1.0563026 ,  0.94196838, -0.53935999,
         1.07967217],
       [ 1.        , -0.79171047, -0.79171047, -1.02877102,  0.        ,
        -0.9114191 , -0.51650824, -0.7337694 ,  0.75820287, -0.45464824,
         0.76844759],
       [ 1.        , -0.75012257, -0.75012257, -1.07390513,  0.        ,
        -0.8533079 , -0.51650824, -0.85729275,  0.83170907, -0.48545251,
         0.8902887 ],
       [ 1.        , -0.75012257, -0.75012257, -0.81599596,  0.        ,
        -0.84382035, -0.51650824, -0.78866867,  0.75820287, -0.21976566,
         0.80023223],
       [ 1.        , -0.62535886, -0.62535886,  0.60895221,  0.        ,
         0.08951673, -0.51650824, -0.00635408, -0.16062467,  1.88262591,
        -0.05596647],
       [ 1.        , -0.50059514, -0.50059514,  0.17695435,  0.        ,
         0.03259147, -0.51650824,  0.15148132, -0.12387156,  0.95464721,
        -0.1705236 ],
       [ 1.        , -0.0847161 , -0.0847161 , -0.00358207,  0.        ,
         0.48324978, -0.51650824,  0.70047401, -0.49140258,  0.24999949,
        -0.67179384],
```

```
       [ 1.        ,  0.16481132,  0.16481132, -0.55163906,  0.        ,
        -0.87228298, -0.51650824, -0.71318217,  0.79495597, -0.88205751,
         0.70421613],
       [ 1.        ,  0.28957504,  0.28957504, -0.88047325,  0.        ,
        -0.96360059, -0.51650824, -0.92591684,  0.94196838, -1.05918208,
         0.9366413 ],
       [ 1.        ,  0.28957504,  0.28957504, -0.86757779,  0.        ,
        -0.97071625, -0.51650824, -1.07688983,  1.01547458, -0.9821714 ,
         1.07900999],
       [ 1.        ,  0.33116294,  0.33116294, -0.82889142,  0.        ,
        -0.93276607, -0.51650824, -0.98767852,  0.97872148, -1.03607888,
         0.99094006],
       [ 1.        ,  0.33116294,  0.33116294, -0.86113006,  0.        ,
        -1.00985236, -0.51650824, -1.03571538,  1.01547458, -1.07073368,
         1.03927919],
       [ 1.        ,  0.45592665,  0.45592665,  0.57671356,  0.        ,
         0.3456804 ,  0.70077239,  0.40539044, -0.38114327, -0.00413576,
        -0.43738213],
       [ 1.        ,  0.62227827,  0.62227827,  0.22208845,  0.        ,
        -0.165461  ,  0.13417029, -0.0200789 ,  0.09664704, -0.52010732,
        -0.01226259],
       [ 1.        ,  0.74704198,  0.74704198, -0.52584814,  0.        ,
        -0.27931152, -0.51650824, -0.46613546,  0.46417806, -0.83970164,
         0.46848006],
       [ 1.        ,  0.99656941,  0.99656941, -0.73217548,  0.        ,
        -0.68846183, -0.51650824, -0.76808144,  0.75820287, -0.95521766,
         0.77308285],
       [ 1.        ,  1.24609683,  1.24609683, -0.1196412 ,  0.        ,
         0.37295709, -0.51650824,  0.59753788, -0.38114327,  0.05747279,
        -0.56650722],
       [ 1.        ,  1.49562426,  1.49562426,  1.21503876,  0.        ,
         0.80464031, -0.51650824,  0.98869517, -0.74867429,  0.3193091 ,
        -1.0201005 ],
       [ 1.        ,  1.53721216,  1.53721216,  1.01515915,  0.        ,
         0.68367413, -0.51650824,  0.72106123, -0.56490878,  0.30005643,
        -0.75853941],
       [ 1.        ,  1.66197588,  1.66197588, -0.26793897,  0.        ,
        -0.30540227, -0.51650824, -0.32888729,  0.39067186, -0.75884043,
         0.31684085],
       [ 1.        ,  1.66197588,  1.66197588, -0.9127119 ,  0.        ,
        -0.13818431, -0.51650824, -0.37692415,  0.39067186, -0.86665538,
         0.41749221]])
```

In [11]:
```
print(model.predict(X_new))
```

```
[[5.531833 ]
 [5.4376135]
 [5.9039893]
 [5.1634874]
 [3.8118262]
 [3.702528 ]
 [3.2508602]
 [3.4736876]
 [3.3046923]
 [3.649736 ]
 [5.1691422]
 [5.414252 ]
 [5.294323 ]
 [5.1910753]
 [5.8651037]
 [4.886865 ]
 [5.468531 ]
 [5.55519  ]
 [5.029869 ]
 [4.8994575]
 [5.1985645]
 [5.645905 ]
 [4.6765723]
 [5.3729963]
 [5.1820374]
 [3.9444346]
 [6.4773374]]
```

In [ ]: