#Importing Python Libraries

```
import pandas as pd
import numpy as np
import matplotlib as plt
from keras.models import Sequential
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from keras.optimizers import Adamax
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import r2_score

x_train = pd.read_csv("ALL_X_train_p.csv")
y_train = pd.read_csv("ALL_y_train_p.csv")
x_test = pd.read_csv("ALL_X_test_p.csv")
y_test = pd.read_csv("ALL_y_test_p.csv")
```

#EXPLORING

```
x_train
```

```
       Unnamed: 0      r1t1      r1t2      r1t3      r1t4      r2t1
r2t2  \
0               0  0.857143  0.785714  0.916667  0.846154  0.900000
0.875000
1               1  0.555556  0.928571  0.250000  0.857143  0.928571
0.444444
2               2  0.857143  0.428571  0.583333  0.692308  0.500000
0.875000
3               3  0.714286  0.857143  0.333333  0.846154  0.500000
0.750000
4               4  0.642857  0.714286  0.833333  0.846154  0.800000
0.750000
...           ...       ...       ...       ...       ...       ...
...
13948       13948  0.500000  0.785714  0.750000  0.923077  0.800000
0.875000
13949       13949  0.571429  0.785714  0.750000  0.846154  0.700000
0.875000
13950       13950  0.285714  0.785714  0.400000  0.933333  0.533333
0.625000
13951       13951  0.888889  0.428571  0.750000  0.571429  0.714286
1.000000
13952       13952  0.785714  0.923077  0.812500  0.687500  0.857143
0.833333
```

```
           r2t3      r2t4
0      0.666667  0.583333
1      0.900000  0.571429
2      0.666667  0.833333
3      0.888889  0.666667
4      0.444444  0.500000
...         ...       ...
13948  0.555556  0.583333
13949  0.555556  0.666667
13950  0.625000  0.500000
13951  0.600000  0.857143
13952  0.800000  0.700000

[13953 rows x 9 columns]
```

```
x_train.describe()
```

```
         Unnamed: 0          r1t1          r1t2          r1t3
r1t4  \
count  13953.000000  13953.000000  13953.000000  13953.000000
13953.000000
mean    6976.000000      0.732111      0.736229      0.661648
0.704418
std     4028.028488      0.149544      0.173437      0.197857
0.188444
min        0.000000      0.000000      0.000000      0.000000
0.000000
25%     3488.000000      0.642857      0.642857      0.533333
0.571429
50%     6976.000000      0.714286      0.785714      0.666667
0.733333
75%    10464.000000      0.857143      0.857143      0.812500
0.866667
max    13952.000000      1.000000      1.000000      1.000000
1.000000

               r2t1          r2t2          r2t3          r2t4
count  13953.000000  13953.000000  13953.000000  13953.000000
mean       0.636129      0.736301      0.687203      0.644405
std        0.225658      0.162209      0.147712      0.239129
min        0.000000      0.000000      0.000000      0.000000
25%        0.500000      0.625000      0.625000      0.500000
50%        0.700000      0.750000      0.666667      0.666667
75%        0.800000      0.875000      0.800000      0.833333
max        1.000000      1.000000      1.000000      1.000000
```

```
x_train.info()
y_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13953 entries, 0 to 13952
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  13953 non-null  int64  
 1   r1t1        13953 non-null  float64
 2   r1t2        13953 non-null  float64
 3   r1t3        13953 non-null  float64
 4   r1t4        13953 non-null  float64
 5   r2t1        13953 non-null  float64
 6   r2t2        13953 non-null  float64
 7   r2t3        13953 non-null  float64
 8   r2t4        13953 non-null  float64
dtypes: float64(8), int64(1)
memory usage: 981.2 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13953 entries, 0 to 13952
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  13953 non-null  int64
 1   N_People    13953 non-null  int64
dtypes: int64(2)
memory usage: 218.1 KB
```

```
x_train.shape
```

```
(13953, 9)
```

```
print(x_train.isnull().sum())
```

```
Unnamed: 0    0
r1t1          0
r1t2          0
r1t3          0
r1t4          0
r2t1          0
r2t2          0
r2t3          0
r2t4          0
dtype: int64
```

```
y_train.head(5)
```

```
   Unnamed: 0  N_People
0           0         7
1           1         0
2           2         0
3           3         9
4           4         7
```

```
print(y_train.isnull().sum())
```

```
Unnamed: 0    0
N_People      0
dtype: int64
```

```
x_test.head(5)
```

```
   Unnamed: 0      r1t1      r1t2      r1t3      r1t4      r2t1
r2t2  \
0           0  0.642857  0.714286  0.733333  0.866667  0.466667
0.625000
1           1  0.714286  0.923077  0.812500  0.625000  0.857143
0.916667
2           2  0.571429  0.714286  0.916667  0.923077  0.800000
0.875000
3           3  0.857143  0.500000  0.666667  0.692308  0.500000
0.875000
4           4  0.785714  0.538462  0.812500  0.625000  0.857143
0.750000

       r2t3      r2t4
0  0.875000  0.500000
1  0.600000  0.600000
2  0.666667  0.666667
3  0.666667  0.833333
4  0.400000  0.600000
```

```
print(x_test.isnull().sum())
```

```
Unnamed: 0    0
r1t1          0
r1t2          0
r1t3          0
r1t4          0
r2t1          0
r2t2          0
r2t3          0
r2t4          0
dtype: int64
```

```
y_test.head(5)
```

```
   Unnamed: 0  N_People
0           0         5
1           1         8
2           2         7
3           3         0
4           4         2
```

```
print(y_test.isnull().sum())
```

```
Unnamed: 0    0
N_People      0
dtype: int64
```

```
x_train.columns[:].duplicated()
```

```
array([False, False, False, False, False, False, False, False, False])
```

```
y_train.columns[:].duplicated()
```

```
array([False, False])
```

```
x_test.columns[:].duplicated()
```

```
array([False, False, False, False, False, False, False, False, False])
```

```
x_test.columns[:].duplicated()
```

```
array([False, False, False, False, False, False, False, False, False])
```

```
x_train = x_train.drop(['Unnamed: 0'], axis=1)
```

```
x_train.head(5)
```

```
       r1t1      r1t2      r1t3      r1t4      r2t1      r2t2
r2t3  \
0  0.857143  0.785714  0.916667  0.846154  0.900000  0.875000
0.666667
1  0.555556  0.928571  0.250000  0.857143  0.928571  0.444444
0.900000
2  0.857143  0.428571  0.583333  0.692308  0.500000  0.875000
0.666667
3  0.714286  0.857143  0.333333  0.846154  0.500000  0.750000
0.888889
4  0.642857  0.714286  0.833333  0.846154  0.800000  0.750000
0.444444

       r2t4
0  0.583333
1  0.571429
2  0.833333
3  0.666667
4  0.500000
```

```
x_train.head(5)
```

```
       r1t1      r1t2      r1t3      r1t4      r2t1      r2t2
r2t3  \
0  0.857143  0.785714  0.916667  0.846154  0.900000  0.875000
0.666667
1  0.555556  0.928571  0.250000  0.857143  0.928571  0.444444
0.900000
```

```
2  0.857143  0.428571  0.583333  0.692308  0.500000  0.875000
0.666667
3  0.714286  0.857143  0.333333  0.846154  0.500000  0.750000
0.888889
4  0.642857  0.714286  0.833333  0.846154  0.800000  0.750000
0.444444

       r2t4
0  0.583333
1  0.571429
2  0.833333
3  0.666667
4  0.500000
```

```
x_train.shape
```

```
(13953, 8)
```

```
y_train = y_train.drop(['Unnamed: 0'], axis = 1)
```

```
y_train.head(5)
```

```
   N_People
0         7
1         0
2         0
3         9
4         7
```

```
y_train.shape
```

```
(13953, 1)
```

```
x_test = x_test.drop(['Unnamed: 0'], axis=1)
```

```
x_test.head(5)
```

```
       r1t1      r1t2      r1t3      r1t4      r2t1      r2t2
r2t3  \
0  0.642857  0.714286  0.733333  0.866667  0.466667  0.625000
0.875000
1  0.714286  0.923077  0.812500  0.625000  0.857143  0.916667
0.600000
2  0.571429  0.714286  0.916667  0.923077  0.800000  0.875000
0.666667
3  0.857143  0.500000  0.666667  0.692308  0.500000  0.875000
0.666667
4  0.785714  0.538462  0.812500  0.625000  0.857143  0.750000
0.400000

       r2t4
0  0.500000
```

```
1  0.600000
2  0.666667
3  0.833333
4  0.600000
```

```
x_test.shape
```

```
(4652, 8)
```

```
y_test = y_test.drop(['Unnamed: 0'], axis=1)
```

```
y_test.head(5)
```

```
   N_People
0         5
1         8
2         7
3         0
4         2
```

```
y_test.shape
```

```
(4652, 1)
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
features_to_scale = y_train.columns


scaler = MinMaxScaler()


y_train_scaled = 
pd.DataFrame(scaler.fit_transform(y_train[features_to_scale]), 
columns=features_to_scale)
```

```
print(y_train_scaled)
```

```
       N_People
0      0.636364
1      0.000000
2      0.000000
3      0.818182
4      0.636364
...         ...
13948  0.545455
13949  0.636364
13950  0.636364
13951  1.000000
13952  0.727273

[13953 rows x 1 columns]
```

```
from sklearn.preprocessing import MinMaxScaler



features_to_scale = y_test.columns


scaler = MinMaxScaler()


y_test_scaled =
pd.DataFrame(scaler.fit_transform(y_test[features_to_scale]),
columns=features_to_scale)

print(y_test_scaled)
```

```
      N_People
0     0.454545
1     0.727273
2     0.636364
3     0.000000
4     0.181818
...        ...
4647  0.818182
4648  0.545455
4649  0.272727
4650  1.000000
4651  0.454545

[4652 rows x 1 columns]
```

#TRAINING

```
x_train, x_val, y_train_scaled, y_val = train_test_split(x_train,
y_train_scaled, test_size=0.2, random_state=42)

model = Sequential()
model.add(Dense(28, activation='relu',
kernel_initializer='he_uniform', input_dim=8))
model.add(Dense(21, kernel_initializer='he_uniform',
activation='relu'))
model.add(Dense(7, kernel_initializer='he_uniform',
activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(optimizer=Adamax(learning_rate=0.001),
loss='mean_absolute_error', metrics=['RootMeanSquaredError'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 28)                252

 dense_1 (Dense)             (None, 21)                609

 dense_2 (Dense)             (None, 7)                 154

 dense_3 (Dense)             (None, 1)                 8

=================================================================
Total params: 1023 (4.00 KB)
Trainable params: 1023 (4.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

#VALIDATING

```
history = model.fit(x_train, y_train_scaled, epochs=150,
validation_data=(x_val, y_val))
```

```
Epoch 1/150
349/349 [==============================] - 2s 3ms/step - loss: 0.4449
- root_mean_squared_error: 0.7041 - val_loss: 0.2306 -
val_root_mean_squared_error: 0.2895
Epoch 2/150
349/349 [==============================] - 1s 2ms/step - loss: 0.2065
- root_mean_squared_error: 0.2633 - val_loss: 0.1795 -
val_root_mean_squared_error: 0.2322
Epoch 3/150
349/349 [==============================] - 1s 2ms/step - loss: 0.1669
- root_mean_squared_error: 0.2211 - val_loss: 0.1454 -
val_root_mean_squared_error: 0.2018
Epoch 4/150
349/349 [==============================] - 1s 2ms/step - loss: 0.1402
- root_mean_squared_error: 0.1989 - val_loss: 0.1273 -
val_root_mean_squared_error: 0.1879
Epoch 5/150
349/349 [==============================] - 1s 2ms/step - loss: 0.1274
- root_mean_squared_error: 0.1875 - val_loss: 0.1163 -
val_root_mean_squared_error: 0.1780
Epoch 6/150
349/349 [==============================] - 0s 1ms/step - loss: 0.1177
- root_mean_squared_error: 0.1771 - val_loss: 0.1091 -
val_root_mean_squared_error: 0.1705
Epoch 7/150
349/349 [==============================] - 1s 2ms/step - loss: 0.1112
```

```
- root_mean_squared_error: 0.1692 - val_loss: 0.1051 -
val_root_mean_squared_error: 0.1649
Epoch 8/150
349/349 [==============================] - 0s 1ms/step - loss: 0.1069
- root_mean_squared_error: 0.1627 - val_loss: 0.1038 -
val_root_mean_squared_error: 0.1585
Epoch 9/150
349/349 [==============================] - 0s 1ms/step - loss: 0.1035
- root_mean_squared_error: 0.1575 - val_loss: 0.1002 -
val_root_mean_squared_error: 0.1546
Epoch 10/150
349/349 [==============================] - 0s 1ms/step - loss: 0.1001
- root_mean_squared_error: 0.1528 - val_loss: 0.0985 -
val_root_mean_squared_error: 0.1502
Epoch 11/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0970
- root_mean_squared_error: 0.1482 - val_loss: 0.1011 -
val_root_mean_squared_error: 0.1495
Epoch 12/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0948
- root_mean_squared_error: 0.1450 - val_loss: 0.0932 -
val_root_mean_squared_error: 0.1437
Epoch 13/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0929
- root_mean_squared_error: 0.1428 - val_loss: 0.0916 -
val_root_mean_squared_error: 0.1418
Epoch 14/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0916
- root_mean_squared_error: 0.1410 - val_loss: 0.0911 -
val_root_mean_squared_error: 0.1410
Epoch 15/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0907
- root_mean_squared_error: 0.1398 - val_loss: 0.0903 -
val_root_mean_squared_error: 0.1414
Epoch 16/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0896
- root_mean_squared_error: 0.1384 - val_loss: 0.0892 -
val_root_mean_squared_error: 0.1390
Epoch 17/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0885
- root_mean_squared_error: 0.1370 - val_loss: 0.0883 -
val_root_mean_squared_error: 0.1376
Epoch 18/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0880
- root_mean_squared_error: 0.1359 - val_loss: 0.0874 -
val_root_mean_squared_error: 0.1360
Epoch 19/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0870
- root_mean_squared_error: 0.1349 - val_loss: 0.0866 -
```

```
val_root_mean_squared_error: 0.1347
Epoch 20/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0862
- root_mean_squared_error: 0.1339 - val_loss: 0.0857 -
val_root_mean_squared_error: 0.1334
Epoch 21/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0851
- root_mean_squared_error: 0.1323 - val_loss: 0.0841 -
val_root_mean_squared_error: 0.1327
Epoch 22/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0842
- root_mean_squared_error: 0.1315 - val_loss: 0.0837 -
val_root_mean_squared_error: 0.1314
Epoch 23/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0839
- root_mean_squared_error: 0.1308 - val_loss: 0.0830 -
val_root_mean_squared_error: 0.1308
Epoch 24/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0829
- root_mean_squared_error: 0.1295 - val_loss: 0.0829 -
val_root_mean_squared_error: 0.1308
Epoch 25/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0820
- root_mean_squared_error: 0.1288 - val_loss: 0.0825 -
val_root_mean_squared_error: 0.1302
Epoch 26/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0816
- root_mean_squared_error: 0.1280 - val_loss: 0.0805 -
val_root_mean_squared_error: 0.1278
Epoch 27/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0809
- root_mean_squared_error: 0.1273 - val_loss: 0.0804 -
val_root_mean_squared_error: 0.1275
Epoch 28/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0802
- root_mean_squared_error: 0.1262 - val_loss: 0.0797 -
val_root_mean_squared_error: 0.1262
Epoch 29/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0798
- root_mean_squared_error: 0.1256 - val_loss: 0.0808 -
val_root_mean_squared_error: 0.1276
Epoch 30/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0789
- root_mean_squared_error: 0.1249 - val_loss: 0.0796 -
val_root_mean_squared_error: 0.1258
Epoch 31/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0787
- root_mean_squared_error: 0.1246 - val_loss: 0.0778 -
val_root_mean_squared_error: 0.1247
```

```
Epoch 32/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0777
- root_mean_squared_error: 0.1234 - val_loss: 0.0817 -
val_root_mean_squared_error: 0.1278
Epoch 33/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0773
- root_mean_squared_error: 0.1230 - val_loss: 0.0788 -
val_root_mean_squared_error: 0.1249
Epoch 34/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0770
- root_mean_squared_error: 0.1226 - val_loss: 0.0766 -
val_root_mean_squared_error: 0.1229
Epoch 35/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0762
- root_mean_squared_error: 0.1215 - val_loss: 0.0762 -
val_root_mean_squared_error: 0.1222
Epoch 36/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0762
- root_mean_squared_error: 0.1213 - val_loss: 0.0756 -
val_root_mean_squared_error: 0.1220
Epoch 37/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0756
- root_mean_squared_error: 0.1209 - val_loss: 0.0756 -
val_root_mean_squared_error: 0.1218
Epoch 38/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0751
- root_mean_squared_error: 0.1205 - val_loss: 0.0786 -
val_root_mean_squared_error: 0.1233
Epoch 39/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0746
- root_mean_squared_error: 0.1200 - val_loss: 0.0764 -
val_root_mean_squared_error: 0.1208
Epoch 40/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0737
- root_mean_squared_error: 0.1192 - val_loss: 0.0739 -
val_root_mean_squared_error: 0.1197
Epoch 41/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0733
- root_mean_squared_error: 0.1186 - val_loss: 0.0740 -
val_root_mean_squared_error: 0.1201
Epoch 42/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0732
- root_mean_squared_error: 0.1182 - val_loss: 0.0730 -
val_root_mean_squared_error: 0.1194
Epoch 43/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0729
- root_mean_squared_error: 0.1183 - val_loss: 0.0750 -
val_root_mean_squared_error: 0.1216
Epoch 44/150
```

```
349/349 [==============================] - 1s 2ms/step - loss: 0.0721
- root_mean_squared_error: 0.1173 - val_loss: 0.0724 -
val_root_mean_squared_error: 0.1187
Epoch 45/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0718
- root_mean_squared_error: 0.1173 - val_loss: 0.0731 -
val_root_mean_squared_error: 0.1199
Epoch 46/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0715
- root_mean_squared_error: 0.1171 - val_loss: 0.0713 -
val_root_mean_squared_error: 0.1184
Epoch 47/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0710
- root_mean_squared_error: 0.1165 - val_loss: 0.0723 -
val_root_mean_squared_error: 0.1191
Epoch 48/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0710
- root_mean_squared_error: 0.1164 - val_loss: 0.0732 -
val_root_mean_squared_error: 0.1190
Epoch 49/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0706
- root_mean_squared_error: 0.1162 - val_loss: 0.0702 -
val_root_mean_squared_error: 0.1168
Epoch 50/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0700
- root_mean_squared_error: 0.1158 - val_loss: 0.0725 -
val_root_mean_squared_error: 0.1181
Epoch 51/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0700
- root_mean_squared_error: 0.1156 - val_loss: 0.0705 -
val_root_mean_squared_error: 0.1167
Epoch 52/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0698
- root_mean_squared_error: 0.1152 - val_loss: 0.0704 -
val_root_mean_squared_error: 0.1172
Epoch 53/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0697
- root_mean_squared_error: 0.1155 - val_loss: 0.0708 -
val_root_mean_squared_error: 0.1174
Epoch 54/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0695
- root_mean_squared_error: 0.1151 - val_loss: 0.0708 -
val_root_mean_squared_error: 0.1161
Epoch 55/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0687
- root_mean_squared_error: 0.1145 - val_loss: 0.0703 -
val_root_mean_squared_error: 0.1172
Epoch 56/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0688
```

```
- root_mean_squared_error: 0.1145 - val_loss: 0.0687 -
val_root_mean_squared_error: 0.1159
Epoch 57/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0686
- root_mean_squared_error: 0.1140 - val_loss: 0.0684 -
val_root_mean_squared_error: 0.1156
Epoch 58/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0685
- root_mean_squared_error: 0.1138 - val_loss: 0.0692 -
val_root_mean_squared_error: 0.1156
Epoch 59/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0680
- root_mean_squared_error: 0.1136 - val_loss: 0.0682 -
val_root_mean_squared_error: 0.1151
Epoch 60/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0679
- root_mean_squared_error: 0.1136 - val_loss: 0.0687 -
val_root_mean_squared_error: 0.1152
Epoch 61/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0675
- root_mean_squared_error: 0.1132 - val_loss: 0.0687 -
val_root_mean_squared_error: 0.1162
Epoch 62/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0675
- root_mean_squared_error: 0.1135 - val_loss: 0.0675 -
val_root_mean_squared_error: 0.1145
Epoch 63/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0673
- root_mean_squared_error: 0.1133 - val_loss: 0.0675 -
val_root_mean_squared_error: 0.1150
Epoch 64/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0671
- root_mean_squared_error: 0.1126 - val_loss: 0.0685 -
val_root_mean_squared_error: 0.1153
Epoch 65/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0672
- root_mean_squared_error: 0.1127 - val_loss: 0.0673 -
val_root_mean_squared_error: 0.1150
Epoch 66/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0671
- root_mean_squared_error: 0.1126 - val_loss: 0.0678 -
val_root_mean_squared_error: 0.1148
Epoch 67/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0667
- root_mean_squared_error: 0.1124 - val_loss: 0.0673 -
val_root_mean_squared_error: 0.1148
Epoch 68/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0662
- root_mean_squared_error: 0.1121 - val_loss: 0.0663 -
```

```
val_root_mean_squared_error: 0.1141
Epoch 69/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0663
- root_mean_squared_error: 0.1119 - val_loss: 0.0662 -
val_root_mean_squared_error: 0.1142
Epoch 70/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0662
- root_mean_squared_error: 0.1118 - val_loss: 0.0661 -
val_root_mean_squared_error: 0.1144
Epoch 71/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0656
- root_mean_squared_error: 0.1115 - val_loss: 0.0717 -
val_root_mean_squared_error: 0.1180
Epoch 72/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0660
- root_mean_squared_error: 0.1117 - val_loss: 0.0690 -
val_root_mean_squared_error: 0.1144
Epoch 73/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0656
- root_mean_squared_error: 0.1115 - val_loss: 0.0663 -
val_root_mean_squared_error: 0.1145
Epoch 74/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0651
- root_mean_squared_error: 0.1112 - val_loss: 0.0660 -
val_root_mean_squared_error: 0.1145
Epoch 75/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0654
- root_mean_squared_error: 0.1113 - val_loss: 0.0659 -
val_root_mean_squared_error: 0.1143
Epoch 76/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0653
- root_mean_squared_error: 0.1112 - val_loss: 0.0687 -
val_root_mean_squared_error: 0.1158
Epoch 77/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0650
- root_mean_squared_error: 0.1111 - val_loss: 0.0666 -
val_root_mean_squared_error: 0.1133
Epoch 78/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0652
- root_mean_squared_error: 0.1110 - val_loss: 0.0659 -
val_root_mean_squared_error: 0.1131
Epoch 79/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0646
- root_mean_squared_error: 0.1107 - val_loss: 0.0656 -
val_root_mean_squared_error: 0.1132
Epoch 80/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0649
- root_mean_squared_error: 0.1109 - val_loss: 0.0668 -
val_root_mean_squared_error: 0.1139
```

```
Epoch 81/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0647
- root_mean_squared_error: 0.1107 - val_loss: 0.0672 -
val_root_mean_squared_error: 0.1135
Epoch 82/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0640
- root_mean_squared_error: 0.1101 - val_loss: 0.0648 -
val_root_mean_squared_error: 0.1134
Epoch 83/150
349/349 [==============================] - 1s 1ms/step - loss: 0.0642
- root_mean_squared_error: 0.1101 - val_loss: 0.0648 -
val_root_mean_squared_error: 0.1124
Epoch 84/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0639
- root_mean_squared_error: 0.1101 - val_loss: 0.0670 -
val_root_mean_squared_error: 0.1141
Epoch 85/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0637
- root_mean_squared_error: 0.1101 - val_loss: 0.0644 -
val_root_mean_squared_error: 0.1125
Epoch 86/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0638
- root_mean_squared_error: 0.1099 - val_loss: 0.0645 -
val_root_mean_squared_error: 0.1124
Epoch 87/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0633
- root_mean_squared_error: 0.1096 - val_loss: 0.0643 -
val_root_mean_squared_error: 0.1120
Epoch 88/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0632
- root_mean_squared_error: 0.1096 - val_loss: 0.0643 -
val_root_mean_squared_error: 0.1130
Epoch 89/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0631
- root_mean_squared_error: 0.1095 - val_loss: 0.0644 -
val_root_mean_squared_error: 0.1133
Epoch 90/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0631
- root_mean_squared_error: 0.1094 - val_loss: 0.0647 -
val_root_mean_squared_error: 0.1121
Epoch 91/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0633
- root_mean_squared_error: 0.1097 - val_loss: 0.0635 -
val_root_mean_squared_error: 0.1119
Epoch 92/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0630
- root_mean_squared_error: 0.1093 - val_loss: 0.0639 -
val_root_mean_squared_error: 0.1129
Epoch 93/150
```

```
349/349 [==============================] - 1s 2ms/step - loss: 0.0628
- root_mean_squared_error: 0.1091 - val_loss: 0.0647 -
val_root_mean_squared_error: 0.1117
Epoch 94/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0626
- root_mean_squared_error: 0.1093 - val_loss: 0.0632 -
val_root_mean_squared_error: 0.1119
Epoch 95/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0625
- root_mean_squared_error: 0.1089 - val_loss: 0.0648 -
val_root_mean_squared_error: 0.1130
Epoch 96/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0626
- root_mean_squared_error: 0.1089 - val_loss: 0.0636 -
val_root_mean_squared_error: 0.1129
Epoch 97/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0623
- root_mean_squared_error: 0.1084 - val_loss: 0.0632 -
val_root_mean_squared_error: 0.1113
Epoch 98/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0622
- root_mean_squared_error: 0.1086 - val_loss: 0.0633 -
val_root_mean_squared_error: 0.1115
Epoch 99/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0620
- root_mean_squared_error: 0.1083 - val_loss: 0.0640 -
val_root_mean_squared_error: 0.1114
Epoch 100/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0623
- root_mean_squared_error: 0.1086 - val_loss: 0.0627 -
val_root_mean_squared_error: 0.1118
Epoch 101/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0618
- root_mean_squared_error: 0.1083 - val_loss: 0.0630 -
val_root_mean_squared_error: 0.1108
Epoch 102/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0620
- root_mean_squared_error: 0.1082 - val_loss: 0.0633 -
val_root_mean_squared_error: 0.1115
Epoch 103/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0617
- root_mean_squared_error: 0.1080 - val_loss: 0.0629 -
val_root_mean_squared_error: 0.1110
Epoch 104/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0615
- root_mean_squared_error: 0.1077 - val_loss: 0.0626 -
val_root_mean_squared_error: 0.1106
Epoch 105/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0615
```

```
- root_mean_squared_error: 0.1081 - val_loss: 0.0645 -
val_root_mean_squared_error: 0.1121
Epoch 106/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0617
- root_mean_squared_error: 0.1080 - val_loss: 0.0622 -
val_root_mean_squared_error: 0.1109
Epoch 107/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0614
- root_mean_squared_error: 0.1075 - val_loss: 0.0645 -
val_root_mean_squared_error: 0.1102
Epoch 108/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0615
- root_mean_squared_error: 0.1073 - val_loss: 0.0627 -
val_root_mean_squared_error: 0.1122
Epoch 109/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0615
- root_mean_squared_error: 0.1077 - val_loss: 0.0620 -
val_root_mean_squared_error: 0.1108
Epoch 110/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0610
- root_mean_squared_error: 0.1073 - val_loss: 0.0620 -
val_root_mean_squared_error: 0.1106
Epoch 111/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0614
- root_mean_squared_error: 0.1075 - val_loss: 0.0626 -
val_root_mean_squared_error: 0.1108
Epoch 112/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0606
- root_mean_squared_error: 0.1069 - val_loss: 0.0621 -
val_root_mean_squared_error: 0.1106
Epoch 113/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0609
- root_mean_squared_error: 0.1071 - val_loss: 0.0656 -
val_root_mean_squared_error: 0.1122
Epoch 114/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0610
- root_mean_squared_error: 0.1067 - val_loss: 0.0635 -
val_root_mean_squared_error: 0.1116
Epoch 115/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0606
- root_mean_squared_error: 0.1069 - val_loss: 0.0632 -
val_root_mean_squared_error: 0.1112
Epoch 116/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0607
- root_mean_squared_error: 0.1069 - val_loss: 0.0629 -
val_root_mean_squared_error: 0.1112
Epoch 117/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0607
- root_mean_squared_error: 0.1069 - val_loss: 0.0626 -
```

```
val_root_mean_squared_error: 0.1096
Epoch 118/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0605
- root_mean_squared_error: 0.1066 - val_loss: 0.0621 -
val_root_mean_squared_error: 0.1103
Epoch 119/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0605
- root_mean_squared_error: 0.1066 - val_loss: 0.0623 -
val_root_mean_squared_error: 0.1100
Epoch 120/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0606
- root_mean_squared_error: 0.1061 - val_loss: 0.0620 -
val_root_mean_squared_error: 0.1107
Epoch 121/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0604
- root_mean_squared_error: 0.1066 - val_loss: 0.0623 -
val_root_mean_squared_error: 0.1116
Epoch 122/150
349/349 [==============================] - 1s 1ms/step - loss: 0.0604
- root_mean_squared_error: 0.1067 - val_loss: 0.0626 -
val_root_mean_squared_error: 0.1100
Epoch 123/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0606
- root_mean_squared_error: 0.1068 - val_loss: 0.0611 -
val_root_mean_squared_error: 0.1098
Epoch 124/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0599
- root_mean_squared_error: 0.1061 - val_loss: 0.0612 -
val_root_mean_squared_error: 0.1094
Epoch 125/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0601
- root_mean_squared_error: 0.1064 - val_loss: 0.0623 -
val_root_mean_squared_error: 0.1103
Epoch 126/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0598
- root_mean_squared_error: 0.1061 - val_loss: 0.0627 -
val_root_mean_squared_error: 0.1109
Epoch 127/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0602
- root_mean_squared_error: 0.1064 - val_loss: 0.0609 -
val_root_mean_squared_error: 0.1095
Epoch 128/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0599
- root_mean_squared_error: 0.1060 - val_loss: 0.0606 -
val_root_mean_squared_error: 0.1091
Epoch 129/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0601
- root_mean_squared_error: 0.1064 - val_loss: 0.0623 -
val_root_mean_squared_error: 0.1098
Epoch 130/150
```

```
349/349 [==============================] - 1s 2ms/step - loss: 0.0599
- root_mean_squared_error: 0.1062 - val_loss: 0.0610 -
val_root_mean_squared_error: 0.1090
Epoch 131/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0596
- root_mean_squared_error: 0.1059 - val_loss: 0.0625 -
val_root_mean_squared_error: 0.1094
Epoch 132/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0599
- root_mean_squared_error: 0.1059 - val_loss: 0.0635 -
val_root_mean_squared_error: 0.1116
Epoch 133/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0594
- root_mean_squared_error: 0.1063 - val_loss: 0.0613 -
val_root_mean_squared_error: 0.1099
Epoch 134/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0600
- root_mean_squared_error: 0.1056 - val_loss: 0.0609 -
val_root_mean_squared_error: 0.1083
Epoch 135/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0596
- root_mean_squared_error: 0.1058 - val_loss: 0.0603 -
val_root_mean_squared_error: 0.1081
Epoch 136/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0595
- root_mean_squared_error: 0.1058 - val_loss: 0.0617 -
val_root_mean_squared_error: 0.1089
Epoch 137/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0592
- root_mean_squared_error: 0.1056 - val_loss: 0.0615 -
val_root_mean_squared_error: 0.1083
Epoch 138/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0595
- root_mean_squared_error: 0.1055 - val_loss: 0.0608 -
val_root_mean_squared_error: 0.1093
Epoch 139/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0594
- root_mean_squared_error: 0.1055 - val_loss: 0.0603 -
val_root_mean_squared_error: 0.1087
Epoch 140/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0587
- root_mean_squared_error: 0.1050 - val_loss: 0.0600 -
val_root_mean_squared_error: 0.1082
Epoch 141/150
349/349 [==============================] - 1s 1ms/step - loss: 0.0590
- root_mean_squared_error: 0.1054 - val_loss: 0.0602 -
val_root_mean_squared_error: 0.1091
Epoch 142/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0590
```

```
- root_mean_squared_error: 0.1054 - val_loss: 0.0600 -
val_root_mean_squared_error: 0.1100
Epoch 143/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0586
- root_mean_squared_error: 0.1051 - val_loss: 0.0621 -
val_root_mean_squared_error: 0.1092
Epoch 144/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0590
- root_mean_squared_error: 0.1053 - val_loss: 0.0611 -
val_root_mean_squared_error: 0.1086
Epoch 145/150
349/349 [==============================] - 1s 1ms/step - loss: 0.0590
- root_mean_squared_error: 0.1054 - val_loss: 0.0601 -
val_root_mean_squared_error: 0.1087
Epoch 146/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0589
- root_mean_squared_error: 0.1052 - val_loss: 0.0598 -
val_root_mean_squared_error: 0.1082
Epoch 147/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0586
- root_mean_squared_error: 0.1049 - val_loss: 0.0599 -
val_root_mean_squared_error: 0.1082
Epoch 148/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0588
- root_mean_squared_error: 0.1053 - val_loss: 0.0604 -
val_root_mean_squared_error: 0.1090
Epoch 149/150
349/349 [==============================] - 0s 1ms/step - loss: 0.0590
- root_mean_squared_error: 0.1050 - val_loss: 0.0594 -
val_root_mean_squared_error: 0.1084
Epoch 150/150
349/349 [==============================] - 1s 2ms/step - loss: 0.0581
- root_mean_squared_error: 0.1045 - val_loss: 0.0604 -
val_root_mean_squared_error: 0.1091
```

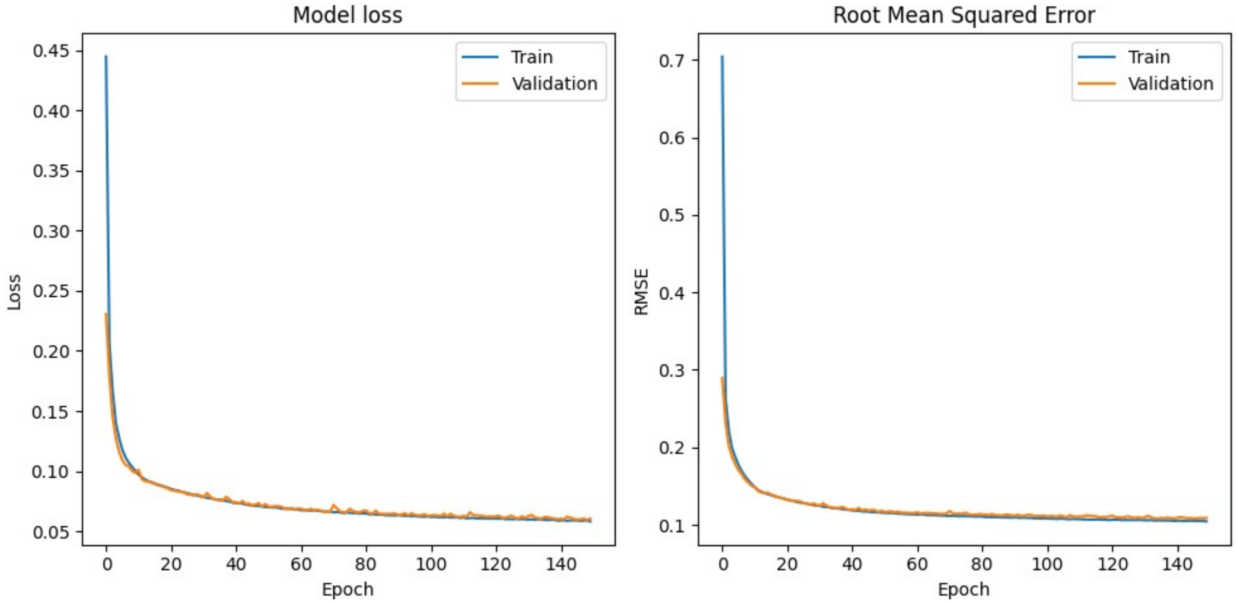
#VISUALISING

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train')
plt.plot(history.history['val_loss'], label='Validation')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['root_mean_squared_error'], label='Train')
plt.plot(history.history['val_root_mean_squared_error'],
label='Validation')
plt.title('Root Mean Squared Error')
plt.ylabel('RMSE')
plt.xlabel('Epoch')
plt.legend()

plt.tight_layout()
plt.show()
```



#EVALUATING

```
predictions = model.predict(x_test)
mae = mean_absolute_error(y_test_scaled, predictions)
accuracy_like_metric = 1 - (mae / (np.max(y_test_scaled) -
np.min(y_test_scaled)))


print("Accuracy-like Metric:", accuracy_like_metric)
```

```
146/146 [==============================] - 0s 853us/step
Accuracy-like Metric: N_People    0.940542
dtype: float64

/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
FutureWarning: In a future version, DataFrame.max(axis=None) will
return a scalar max over the entire DataFrame. To retain the old
behavior, use 'frame.max(axis=0)' or just 'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
```

```
FutureWarning: In a future version, DataFrame.min(axis=None) will
return a scalar min over the entire DataFrame. To retain the old
behavior, use 'frame.min(axis=0)' or just 'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
```

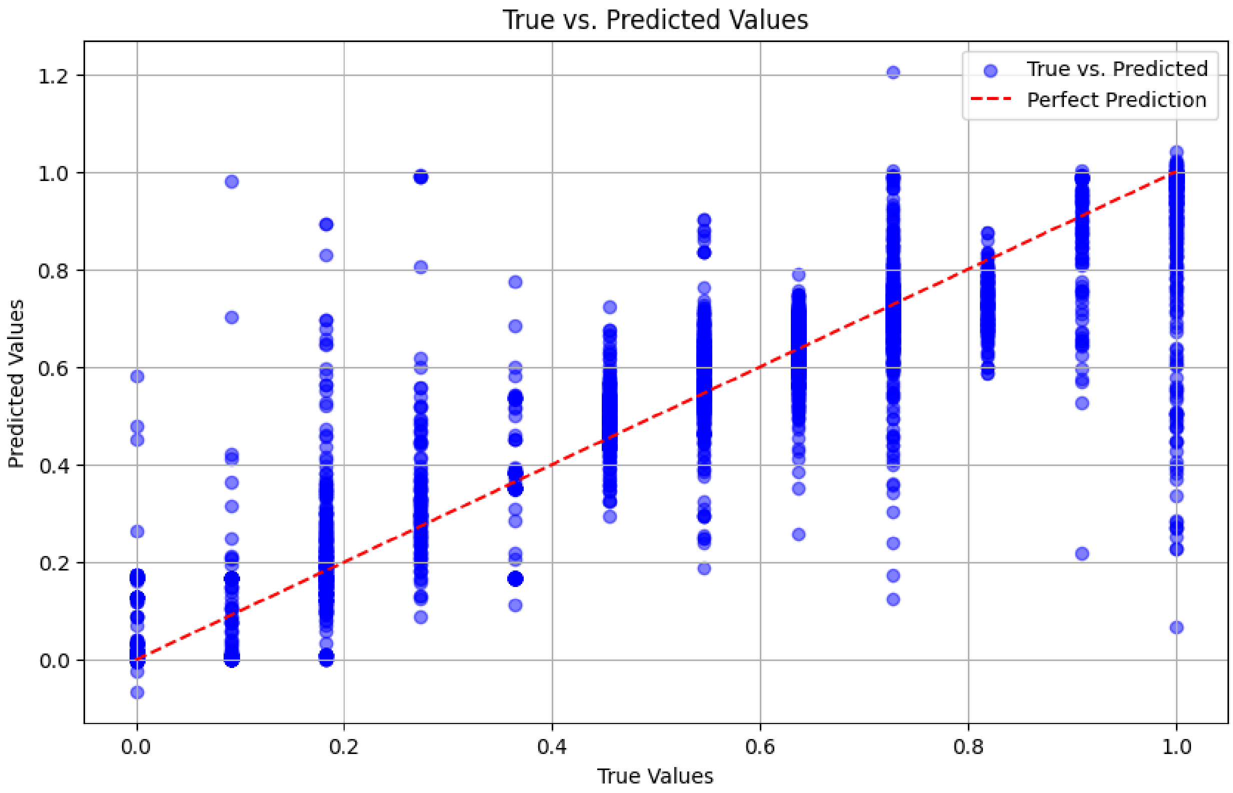
```
plt.figure(figsize=(10, 6))

plt.scatter(y_test_scaled, predictions, alpha=0.5, color='blue',
label='True vs. Predicted')
plt.plot([np.min(y_test_scaled), np.max(y_test_scaled)],
[np.min(y_test_scaled), np.max(y_test_scaled)], color='red',
linestyle='--', label='Perfect Prediction')

plt.title('True vs. Predicted Values')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)
plt.show()

# Display the accuracy-like metric
print("Accuracy-like Metric:", accuracy_like_metric)
```

```
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
FutureWarning: In a future version, DataFrame.min(axis=None) will
return a scalar min over the entire DataFrame. To retain the old
behavior, use 'frame.min(axis=0)' or just 'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
FutureWarning: In a future version, DataFrame.max(axis=None) will
return a scalar max over the entire DataFrame. To retain the old
behavior, use 'frame.max(axis=0)' or just 'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
FutureWarning: In a future version, DataFrame.min(axis=None) will
return a scalar min over the entire DataFrame. To retain the old
behavior, use 'frame.min(axis=0)' or just 'frame.min()'
  return reduction(axis=axis, out=out, **passkwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:84:
FutureWarning: In a future version, DataFrame.max(axis=None) will
return a scalar max over the entire DataFrame. To retain the old
behavior, use 'frame.max(axis=0)' or just 'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
```



```
Accuracy-like Metric: N_People    0.940542
dtype: float64
```

```python
# Display the accuracy-like metric
print("Accuracy-like Metric:", accuracy_like_metric)
```

```
Accuracy-like Metric: N_People    0.940542
dtype: float64
```

```python
from sklearn.metrics import r2_score
r2 = r2_score(y_test_scaled, predictions)
print(f'R-squared Score: {r2}')
```

```
R-squared Score: 0.8881847807871238
```

```python
from sklearn.metrics import mean_squared_error, mean_absolute_error
score = mean_absolute_error(y_test_scaled,predictions)
print("The Mean Absolute Error of our Model is {}".format(round(score,
2)))
```

```
The Mean Absolute Error of our Model is 0.06
```

```python
y_train_predicted = model.predict(x_train)
y_test_predicted = model.predict(x_test)

def accuracy(y_true, y_pred, threshold):
    correct_predictions = np.sum(np.abs(y_true - y_pred) <= threshold)
    total_predictions = len(y_true)
    accuracy = correct_predictions / total_predictions
    return accuracy

threshold = 0.5

accuracy_train = accuracy(y_train_scaled, y_train_predicted,
threshold)
accuracy_test = accuracy(y_test_scaled, y_test_predicted, threshold)

print('Accuracy:')
print('Train: {:.2%}'.format(float(accuracy_train)))
print('Test: {:.2%}'.format(float(accuracy_test)))



349/349 [==============================] - 1s 2ms/step
146/146 [==============================] - 0s 2ms/step
Accuracy:
Train: 99.20%
Test: 99.14%

print('Accuracy ')
print('Train : ' ,accuracy_train)
print('-------->')
print('Test : ', accuracy_test)

Accuracy
Train :  N_People    0.992027
dtype: float64
-------->
Test :  N_People    0.991402
dtype: float64
```