Data Preprocessing

```python
#Importing lbraries

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#Read CSV File

df=pd.read_csv('Service.csv')
df
```

```
     Demand Values  Service level during disruption  \
0                0                               79
1               41                               83
2              119                               86
3               68                               84
4               35                               82
..             ...                              ...
595              0                               80
596             75                               84
597             60                               83
598             41                               83
599             33                               82

     Percentage of  Distruption  Unnamed: 3  \
0                            60         NaN
1                            64         NaN
2                            67         NaN
3                            65         NaN
4                            63         NaN
..                          ...         ...
595                          61         NaN
596                          65         NaN
597                          65         NaN
598                          64         NaN
599                          63         NaN

     OUTPUT: Service level at 100%:  Average.93.69
0                                               89
1                                               92
2                                               96
3                                               94
4                                               92
..                                             ...
595                                             90
```

```
596                                                94
597                                                93
598                                                92
599                                                92

[600 rows x 5 columns]
```

```python
#Remove irrelevant columns

df=df.drop(columns=['Unnamed: 3'])
df
```

```
     Demand Values  Service level during disruption  \
0                0                                79
1               41                                83
2              119                                86
3               68                                84
4               35                                82
..             ...                               ...
595              0                                80
596             75                                84
597             60                                83
598             41                                83
599             33                                82

     Percentage of  Distruption  OUTPUT: Service level at 100%:
Average.93.69
0                            60
89
1                            64
92
2                            67
96
3                            65
94
4                            63
92
..                          ...
...
595                          61
90
596                          65
94
597                          65
93
598                          64
92
599                          63
92
```

```
[600 rows x 4 columns]
```

```python
#rename 4th column as target
df = df.rename(columns={df.columns[-1]: 'target'})
df
```

```
     Demand Values  Service level during disruption  \
0                0                               79
1               41                               83
2              119                               86
3               68                               84
4               35                               82
..             ...                              ...
595              0                               80
596             75                               84
597             60                               83
598             41                               83
599             33                               82

     Percentage of  Distruption  target
0                            60      89
1                            64      92
2                            67      96
3                            65      94
4                            63      92
..                          ...     ...
595                          61      90
596                          65      94
597                          65      93
598                          64      92
599                          63      92

[600 rows x 4 columns]
```

```python
#create function to check for missing values in the data

def count_of_null(df):
    count=df.isnull().sum().sum()
    return count

# print missing value count in the data
count_null = count_of_null(df)
print(count_null)
```

```
0
```

```python
#create function to check duplicate values in the data
def check_duplicates(df):
    count=df.duplicated().sum().sum()
    return count
```

```
# print duplicate values in the data
count_duplicates = check_duplicates(df)
print(count_duplicates)

447

#create a function to check information about the data
def about_data (df):
        about=df.info()
        return about

#print information about the data
info=about_data (df)
print(info)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 4 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Demand Values                    600 non-null    int64
 1   Service level during disruption  600 non-null    int64
 2   Percentage of  Distruption       600 non-null    int64
 3   target                           600 non-null    int64
dtypes: int64(4)
memory usage: 18.9 KB
None

# create function to print unique values in the dataframe

def print_unique_values(df):
    for column in df.columns:
        unique_values = df[column].unique()
        print(f"Column '{column}' has {len(unique_values)} unique
values:")
        print(unique_values)
print_unique_values(df)

Column 'Demand Values' has 138 unique values:
[  0  41 119  68  35  21 103  66  56 101  34 109  23   8  87  84  38
37
  64  94  91 100  62  20  45  17  97  22  30  90  81 113  77  49  95
25
  43  86  53 117  85  71  27 124 127 111  61 129  36  93  44 125  60
58
 110  40  31  59  54  72  16 115  12  79  18  73 123 136  52 107 154
19
  96  89  63  83 121  65  28 112  50  32   3  51  14  26  69  78  42
99
  76  48  92 132 108   6  80  57  75 105 143   1  11 114  47 118 104
```

```
130
 102  46  88  33  70 147  67 140 160 138  74 106  82 137  15  29 153
24
 185 145   5 120  39  55 122  10 131 128 126   9]
Column 'Service level during disruption' has 11 unique values:
[79 83 86 84 82 85 81 80 87 88 89]
Column 'Percentage of  Distruption' has 11 unique values:
[60 64 67 65 63 66 62 61 68 69 70]
Column 'target' has 11 unique values:
[89 92 96 94 91 95 93 90 97 98 99]
```

```
#Create function to check statistical summary of the dataset
def stat_summary (df):
    statistics_sum = df.describe()
    return statistics_sum
#print statistical summary of data
summary = stat_summary (df)
summary
```

```
       Demand Values  Service level during disruption  \
count     600.000000                       600.000000
mean       69.751667                        83.870000
std        35.132280                         1.730537
min         0.000000                        79.000000
25%        43.750000                        83.000000
50%        72.000000                        84.000000
75%        93.250000                        85.000000
max       185.000000                        89.000000

       Percentage of  Distruption      target
count                  600.000000  600.000000
mean                    64.973333   93.650000
std                      1.740979    1.723112
min                     60.000000   89.000000
25%                     64.000000   92.000000
50%                     65.000000   94.000000
75%                     66.000000   95.000000
max                     70.000000   99.000000
```

```
#create function to find columns with Categorical Values

def categorcal_columns (df):
    cat_cols=
df.select_dtypes(exclude=['int','float']).columns.tolist()
    return cat_cols
#print categorical columns
categorical = categorcal_columns (df)
categorical
```

```
[]
```

```
#create function to find columns with Numerical Values

def numerical_columns (df):
    num_cols =
df.select_dtypes(include=['int','float']).columns.tolist()
    return num_cols
#print numerical columns
numerical = numerical_columns (df)
numerical
```

```
['Demand Values',
 'Service level during disruption',
 'Percentage of  Distruption',
 'target']
```

```
#Removing extra spaces from the features Using strip()
df.columns = df.columns.str.strip()
```

```
# load cleaned dataframe into CSV file.
df.to_csv('Fixed_Service.csv', index=False, header=True)
```

Outlier detection

```
#Finding Outliers using IQR score

#The IQR is the first quartile subtracted from the third quartile;
#these quartiles can be clearly seen on a box plot on the data.
# calculate IQR score
#where Q3 is the 75th percentile of the data and Q1 is the 25th
percentile of the data.

Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

print(IQR)

# IQR of each column
```

```
Demand Values                      49.5
Service level during disruption     2.0
Percentage of  Distruption          2.0
target                              3.0
dtype: float64
```

Outlier removal

```
#remove outlier
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

```
#Once the IQR is calculated, we can use it to identify outliers by
defining a threshold range as follows;
#Lower threshold = Q1 - 1.5 * IQR
#Upper threshold = Q3 + 1.5 * IQR
#Any data points that fall outside of this range are considered
outliers and removed from the dataset

df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 *
IQR))).any(axis=1)]
df.shape
```

```
(596, 4)
```

Feature Scaling

```
#Standardize input variables

from sklearn.preprocessing import StandardScaler

# Create a StandardScaler object
scaler = StandardScaler()
import warnings
from pandas.core.common import SettingWithCopyWarning
warnings.simplefilter(action="ignore",
category=SettingWithCopyWarning)

# Standardize variables Demand Values, Service level during
disruption, and Percentage of  Distruption in the dataframe

df[['Demand Values', 'Service level during disruption', 'Percentage of
Distruption']] = scaler.fit_transform(df[['Demand Values',

'Service level during disruption',

'Percentage of  Distruption']])

#view standardized dataframe

df


#The standard range for feature scaling is usually between 0 and 1 or
-1 and 1.
#This is because it helps to normalize the data and make it easier for
machine learning algorithms to converge
#during training.
#Standardizing the features to a particular range can also help to
prevent some features from dominating others in the learning process.
```

```
     Demand Values  Service level during disruption  \
1        -0.836745                        -0.525130
2         1.420864                         1.253151
3        -0.055265                         0.067630
4        -1.010408                        -1.117891
5        -1.415620                        -1.117891
..             ...                              ...
595      -2.023438                        -2.303412
596       0.147341                         0.067630
597      -0.286815                        -0.525130
598      -0.836745                        -0.525130
599      -1.068295                        -1.117891

     Percentage of  Distruption  target
1                     -0.583415      92
2                      1.184630      96
3                      0.005933      94
4                     -1.172764      92
5                     -1.172764      91
..                          ...     ...
595                   -2.351461      90
596                    0.005933      94
597                    0.005933      93
598                   -0.583415      92
599                   -1.172764      92

[596 rows x 4 columns]
```

Train_Test_Split

```
from sklearn.model_selection import train_test_split

# Split data into input (X) and output (y)

X = df.drop(columns=['target']) #drop target/output variable column
from the dataframe/input variable
y = df['target'] #set output feature as y

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

#test_size set split as 80:20 with training set as 80 and test set as
20.
#set random_state to 42 to ensure that the results are reproducible.
```

```
#training set

X_train.shape
```

```
(476, 3)
```

```
# load train set into CSV file.
X_train.to_csv('x_train.csv', index=False, header=True)
```

```
#testng set

X_test.shape
```

```
(120, 3)
```

```
# load test set into CSV file.
X_test.to_csv('x_test.csv', index=False, header=True)
```

Neural Network Model Architecture

```
#Importing libraries
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
import math

#define the model architecture
#using a sequential model with three dense layers. The first layer has
16 neurons and expects an input of 3 features
#second layer has 8 neurons, and the output layer has just 1 neuron

model = Sequential()
model.add(Dense(16, input_dim=3, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='linear'))

# compile the model
model.compile(loss='mean_squared_error', optimizer='adam',
metrics=['mae','mse'])

# print the model summary
print(model.summary())


#using the 'relu' activation function for the hidden layers
#'linear' activation function for the output layer since we are
dealing with continuous variables.
#The loss function used is mean squared error, which is commonly used
for regression problems.
```

```
#The optimizer used is Adam, which is an efficient stochastic gradient
descent algorithm.
#mean absolute error and mean squared error as evaluation metrics

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                64

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________________________
None
```

```
# Fit the model to the training data
history = model.fit(X_train, y_train,
                    validation_data=(X_test, y_test),
                    epochs=100, batch_size=16)
```

```
Epoch 1/100
30/30 [==============================] - 1s 9ms/step - loss: 8709.5879
- mae: 93.3105 - mse: 8709.5879 - val_loss: 8662.1250 - val_mae:
93.0525 - val_mse: 8662.1250
Epoch 2/100
30/30 [==============================] - 0s 4ms/step - loss: 8631.8828
- mae: 92.8911 - mse: 8631.8828 - val_loss: 8569.6738 - val_mae:
92.5517 - val_mse: 8569.6738
Epoch 3/100
30/30 [==============================] - 0s 4ms/step - loss: 8520.3330
- mae: 92.2859 - mse: 8520.3320 - val_loss: 8430.3623 - val_mae:
91.7911 - val_mse: 8430.3623
Epoch 4/100
30/30 [==============================] - 0s 5ms/step - loss: 8360.6016
- mae: 91.4112 - mse: 8360.6016 - val_loss: 8238.2568 - val_mae:
90.7300 - val_mse: 8238.2568
Epoch 5/100
30/30 [==============================] - 0s 4ms/step - loss: 8136.7100
- mae: 90.1674 - mse: 8136.7100 - val_loss: 7960.5752 - val_mae:
89.1700 - val_mse: 7960.5752
Epoch 6/100
30/30 [==============================] - 0s 3ms/step - loss: 7805.0386
- mae: 88.2899 - mse: 7805.0386 - val_loss: 7554.7964 - val_mae:
86.8302 - val_mse: 7554.7964
Epoch 7/100
```

```
30/30 [==============================] - 0s 4ms/step - loss: 7348.7964
- mae: 85.6241 - mse: 7348.7964 - val_loss: 7017.0400 - val_mae:
83.6091 - val_mse: 7017.0400
Epoch 8/100
30/30 [==============================] - 0s 4ms/step - loss: 6750.7725
- mae: 81.9651 - mse: 6750.7725 - val_loss: 6336.5601 - val_mae:
79.3012 - val_mse: 6336.5601
Epoch 9/100
30/30 [==============================] - 0s 4ms/step - loss: 6010.0674
- mae: 77.1367 - mse: 6010.0674 - val_loss: 5514.6670 - val_mae:
73.6599 - val_mse: 5514.6670
Epoch 10/100
30/30 [==============================] - 0s 3ms/step - loss: 5156.2246
- mae: 71.0543 - mse: 5156.2246 - val_loss: 4610.4214 - val_mae:
66.7357 - val_mse: 4610.4214
Epoch 11/100
30/30 [==============================] - 0s 4ms/step - loss: 4254.1865
- mae: 63.8332 - mse: 4254.1865 - val_loss: 3714.0598 - val_mae:
58.8021 - val_mse: 3714.0598
Epoch 12/100
30/30 [==============================] - 0s 4ms/step - loss: 3392.0034
- mae: 55.8095 - mse: 3392.0034 - val_loss: 2914.5115 - val_mae:
50.5507 - val_mse: 2914.5115
Epoch 13/100
30/30 [==============================] - 0s 4ms/step - loss: 2646.8518
- mae: 47.9661 - mse: 2646.8518 - val_loss: 2274.1284 - val_mae:
43.3044 - val_mse: 2274.1284
Epoch 14/100
30/30 [==============================] - 0s 3ms/step - loss: 2078.1218
- mae: 41.3628 - mse: 2078.1218 - val_loss: 1820.1849 - val_mae:
37.7473 - val_mse: 1820.1849
Epoch 15/100
30/30 [==============================] - 0s 4ms/step - loss: 1685.6046
- mae: 36.7581 - mse: 1685.6046 - val_loss: 1533.6879 - val_mae:
33.6485 - val_mse: 1533.6879
Epoch 16/100
30/30 [==============================] - 0s 3ms/step - loss: 1439.5800
- mae: 33.3484 - mse: 1439.5800 - val_loss: 1360.0294 - val_mae:
30.8657 - val_mse: 1360.0294
Epoch 17/100
30/30 [==============================] - 0s 3ms/step - loss: 1283.7113
- mae: 30.8582 - mse: 1283.7113 - val_loss: 1244.8020 - val_mae:
29.1319 - val_mse: 1244.8020
Epoch 18/100
30/30 [==============================] - 0s 4ms/step - loss: 1179.9535
- mae: 29.2723 - mse: 1179.9535 - val_loss: 1157.1799 - val_mae:
28.3829 - val_mse: 1157.1799
Epoch 19/100
30/30 [==============================] - 0s 4ms/step - loss: 1099.7269
- mae: 28.4269 - mse: 1099.7269 - val_loss: 1077.1290 - val_mae:
```

```
27.8093 - val_mse: 1077.1290
Epoch 20/100
30/30 [==============================] - 0s 3ms/step - loss: 1026.5901
- mae: 27.6467 - mse: 1026.5901 - val_loss: 1005.1143 - val_mae:
27.0363 - val_mse: 1005.1143
Epoch 21/100
30/30 [==============================] - 0s 3ms/step - loss: 959.1307
- mae: 26.8310 - mse: 959.1307 - val_loss: 930.2449 - val_mae: 26.1994
- val_mse: 930.2449
Epoch 22/100
30/30 [==============================] - 0s 3ms/step - loss: 889.0327
- mae: 25.8955 - mse: 889.0327 - val_loss: 857.7523 - val_mae: 25.2172
- val_mse: 857.7523
Epoch 23/100
30/30 [==============================] - 0s 4ms/step - loss: 819.7033
- mae: 24.9068 - mse: 819.7033 - val_loss: 777.3906 - val_mae: 24.0126
- val_mse: 777.3906
Epoch 24/100
30/30 [==============================] - 0s 4ms/step - loss: 747.0585
- mae: 23.7497 - mse: 747.0585 - val_loss: 703.0905 - val_mae: 22.8547
- val_mse: 703.0905
Epoch 25/100
30/30 [==============================] - 0s 3ms/step - loss: 673.4082
- mae: 22.5016 - mse: 673.4082 - val_loss: 627.1707 - val_mae: 21.5130
- val_mse: 627.1707
Epoch 26/100
30/30 [==============================] - 0s 4ms/step - loss: 598.6097
- mae: 21.2148 - mse: 598.6097 - val_loss: 550.3431 - val_mae: 20.1361
- val_mse: 550.3431
Epoch 27/100
30/30 [==============================] - 0s 3ms/step - loss: 523.7824
- mae: 19.7721 - mse: 523.7824 - val_loss: 477.3322 - val_mae: 18.6963
- val_mse: 477.3322
Epoch 28/100
30/30 [==============================] - 0s 4ms/step - loss: 452.3430
- mae: 18.3129 - mse: 452.3430 - val_loss: 407.5080 - val_mae: 17.2523
- val_mse: 407.5080
Epoch 29/100
30/30 [==============================] - 0s 3ms/step - loss: 390.6744
- mae: 16.9883 - mse: 390.6744 - val_loss: 347.5778 - val_mae: 15.9262
- val_mse: 347.5778
Epoch 30/100
30/30 [==============================] - 0s 4ms/step - loss: 332.7297
- mae: 15.5891 - mse: 332.7297 - val_loss: 299.1316 - val_mae: 14.6219
- val_mse: 299.1316
Epoch 31/100
30/30 [==============================] - 0s 4ms/step - loss: 282.8120
- mae: 14.2672 - mse: 282.8120 - val_loss: 250.5676 - val_mae: 13.3082
- val_mse: 250.5676
Epoch 32/100
```

```
30/30 [==============================] - 0s 3ms/step - loss: 237.1282
- mae: 12.9508 - mse: 237.1282 - val_loss: 209.0284 - val_mae: 12.0227
- val_mse: 209.0284
Epoch 33/100
30/30 [==============================] - 0s 4ms/step - loss: 197.3064
- mae: 11.7271 - mse: 197.3064 - val_loss: 171.7274 - val_mae: 10.8227
- val_mse: 171.7274
Epoch 34/100
30/30 [==============================] - 0s 4ms/step - loss: 162.9605
- mae: 10.5040 - mse: 162.9605 - val_loss: 143.0635 - val_mae: 9.7137
- val_mse: 143.0635
Epoch 35/100
30/30 [==============================] - 0s 3ms/step - loss: 134.0101
- mae: 9.4501 - mse: 134.0101 - val_loss: 115.0649 - val_mae: 8.6473 -
val_mse: 115.0649
Epoch 36/100
30/30 [==============================] - 0s 4ms/step - loss: 109.3296
- mae: 8.4347 - mse: 109.3296 - val_loss: 93.5619 - val_mae: 7.6972 -
val_mse: 93.5619
Epoch 37/100
30/30 [==============================] - 0s 3ms/step - loss: 89.2063 -
mae: 7.5186 - mse: 89.2063 - val_loss: 77.4245 - val_mae: 6.8468 -
val_mse: 77.4245
Epoch 38/100
30/30 [==============================] - 0s 3ms/step - loss: 73.0379 -
mae: 6.6841 - mse: 73.0379 - val_loss: 62.9441 - val_mae: 6.1127 -
val_mse: 62.9441
Epoch 39/100
30/30 [==============================] - 0s 4ms/step - loss: 59.6072 -
mae: 6.0323 - mse: 59.6072 - val_loss: 50.9854 - val_mae: 5.4745 -
val_mse: 50.9854
Epoch 40/100
30/30 [==============================] - 0s 3ms/step - loss: 48.8588 -
mae: 5.4176 - mse: 48.8588 - val_loss: 41.9412 - val_mae: 4.8554 -
val_mse: 41.9412
Epoch 41/100
30/30 [==============================] - 0s 3ms/step - loss: 40.2664 -
mae: 4.7730 - mse: 40.2664 - val_loss: 35.0932 - val_mae: 4.3310 -
val_mse: 35.0932
Epoch 42/100
30/30 [==============================] - 0s 3ms/step - loss: 33.4752 -
mae: 4.3587 - mse: 33.4752 - val_loss: 28.8277 - val_mae: 3.8818 -
val_mse: 28.8277
Epoch 43/100
30/30 [==============================] - 0s 3ms/step - loss: 27.7885 -
mae: 3.8646 - mse: 27.7885 - val_loss: 24.8095 - val_mae: 3.4358 -
val_mse: 24.8095
Epoch 44/100
30/30 [==============================] - 0s 3ms/step - loss: 23.3469 -
mae: 3.4612 - mse: 23.3469 - val_loss: 20.7153 - val_mae: 3.1363 -
```

```
val_mse: 20.7153
Epoch 45/100
30/30 [==============================] - 0s 3ms/step - loss: 19.8628 -
mae: 3.1838 - mse: 19.8628 - val_loss: 17.8748 - val_mae: 2.8589 -
val_mse: 17.8748
Epoch 46/100
30/30 [==============================] - 0s 3ms/step - loss: 17.0417 -
mae: 2.8552 - mse: 17.0417 - val_loss: 15.8025 - val_mae: 2.6125 -
val_mse: 15.8025
Epoch 47/100
30/30 [==============================] - 0s 3ms/step - loss: 14.9290 -
mae: 2.6178 - mse: 14.9290 - val_loss: 13.6643 - val_mae: 2.4100 -
val_mse: 13.6643
Epoch 48/100
30/30 [==============================] - 0s 3ms/step - loss: 13.1750 -
mae: 2.4236 - mse: 13.1750 - val_loss: 12.2277 - val_mae: 2.2620 -
val_mse: 12.2277
Epoch 49/100
30/30 [==============================] - 0s 3ms/step - loss: 11.7908 -
mae: 2.2207 - mse: 11.7908 - val_loss: 11.2961 - val_mae: 2.1326 -
val_mse: 11.2961
Epoch 50/100
30/30 [==============================] - 0s 3ms/step - loss: 10.6777 -
mae: 2.1516 - mse: 10.6777 - val_loss: 10.3243 - val_mae: 2.0818 -
val_mse: 10.3243
Epoch 51/100
30/30 [==============================] - 0s 3ms/step - loss: 9.8191 -
mae: 2.0868 - mse: 9.8191 - val_loss: 9.7360 - val_mae: 2.0349 -
val_mse: 9.7360
Epoch 52/100
30/30 [==============================] - 0s 3ms/step - loss: 9.1751 -
mae: 2.0217 - mse: 9.1751 - val_loss: 9.1659 - val_mae: 2.0086 -
val_mse: 9.1659
Epoch 53/100
30/30 [==============================] - 0s 3ms/step - loss: 8.6382 -
mae: 1.9879 - mse: 8.6382 - val_loss: 8.7183 - val_mae: 1.9878 -
val_mse: 8.7183
Epoch 54/100
30/30 [==============================] - 0s 3ms/step - loss: 8.2215 -
mae: 1.9713 - mse: 8.2215 - val_loss: 8.4538 - val_mae: 2.0023 -
val_mse: 8.4538
Epoch 55/100
30/30 [==============================] - 0s 3ms/step - loss: 7.8840 -
mae: 1.9414 - mse: 7.8840 - val_loss: 8.0945 - val_mae: 1.9908 -
val_mse: 8.0945
Epoch 56/100
30/30 [==============================] - 0s 3ms/step - loss: 7.5554 -
mae: 1.9492 - mse: 7.5554 - val_loss: 7.8791 - val_mae: 1.9849 -
val_mse: 7.8791
Epoch 57/100
```

```
30/30 [==============================] - 0s 3ms/step - loss: 7.3140 -
mae: 1.9260 - mse: 7.3140 - val_loss: 7.5304 - val_mae: 1.9536 -
val_mse: 7.5304
Epoch 58/100
30/30 [==============================] - 0s 3ms/step - loss: 7.0843 -
mae: 1.9114 - mse: 7.0843 - val_loss: 7.3530 - val_mae: 1.9551 -
val_mse: 7.3530
Epoch 59/100
30/30 [==============================] - 0s 3ms/step - loss: 6.8832 -
mae: 1.8698 - mse: 6.8832 - val_loss: 7.1463 - val_mae: 1.9428 -
val_mse: 7.1463
Epoch 60/100
30/30 [==============================] - 0s 3ms/step - loss: 6.6265 -
mae: 1.9130 - mse: 6.6265 - val_loss: 7.0281 - val_mae: 1.9485 -
val_mse: 7.0281
Epoch 61/100
30/30 [==============================] - 0s 3ms/step - loss: 6.3809 -
mae: 1.8457 - mse: 6.3809 - val_loss: 6.7348 - val_mae: 1.8871 -
val_mse: 6.7348
Epoch 62/100
30/30 [==============================] - 0s 3ms/step - loss: 6.2299 -
mae: 1.8565 - mse: 6.2299 - val_loss: 6.6205 - val_mae: 1.9046 -
val_mse: 6.6205
Epoch 63/100
30/30 [==============================] - 0s 3ms/step - loss: 6.0434 -
mae: 1.8155 - mse: 6.0434 - val_loss: 6.4087 - val_mae: 1.8711 -
val_mse: 6.4087
Epoch 64/100
30/30 [==============================] - 0s 3ms/step - loss: 5.8584 -
mae: 1.7982 - mse: 5.8584 - val_loss: 6.2202 - val_mae: 1.8373 -
val_mse: 6.2202
Epoch 65/100
30/30 [==============================] - 0s 3ms/step - loss: 5.6996 -
mae: 1.7697 - mse: 5.6996 - val_loss: 6.0783 - val_mae: 1.8303 -
val_mse: 6.0783
Epoch 66/100
30/30 [==============================] - 0s 3ms/step - loss: 5.5521 -
mae: 1.7738 - mse: 5.5521 - val_loss: 5.9068 - val_mae: 1.7973 -
val_mse: 5.9068
Epoch 67/100
30/30 [==============================] - 0s 3ms/step - loss: 5.3861 -
mae: 1.7701 - mse: 5.3861 - val_loss: 5.7060 - val_mae: 1.7719 -
val_mse: 5.7060
Epoch 68/100
30/30 [==============================] - 0s 3ms/step - loss: 5.2402 -
mae: 1.6989 - mse: 5.2402 - val_loss: 5.5507 - val_mae: 1.7292 -
val_mse: 5.5507
Epoch 69/100
30/30 [==============================] - 0s 3ms/step - loss: 5.1212 -
mae: 1.7105 - mse: 5.1212 - val_loss: 5.3293 - val_mae: 1.7015 -
```

```
val_mse: 5.3293
Epoch 70/100
30/30 [==============================] - 0s 3ms/step - loss: 4.8878 -
mae: 1.6805 - mse: 4.8878 - val_loss: 5.2067 - val_mae: 1.6866 -
val_mse: 5.2067
Epoch 71/100
30/30 [==============================] - 0s 3ms/step - loss: 4.7125 -
mae: 1.6337 - mse: 4.7125 - val_loss: 5.0320 - val_mae: 1.6637 -
val_mse: 5.0320
Epoch 72/100
30/30 [==============================] - 0s 3ms/step - loss: 4.5754 -
mae: 1.6407 - mse: 4.5754 - val_loss: 4.8729 - val_mae: 1.6307 -
val_mse: 4.8729
Epoch 73/100
30/30 [==============================] - 0s 3ms/step - loss: 4.4186 -
mae: 1.6058 - mse: 4.4186 - val_loss: 4.7399 - val_mae: 1.6248 -
val_mse: 4.7399
Epoch 74/100
30/30 [==============================] - 0s 3ms/step - loss: 4.2484 -
mae: 1.5528 - mse: 4.2484 - val_loss: 4.5719 - val_mae: 1.5915 -
val_mse: 4.5719
Epoch 75/100
30/30 [==============================] - 0s 3ms/step - loss: 4.1137 -
mae: 1.5643 - mse: 4.1137 - val_loss: 4.4675 - val_mae: 1.5889 -
val_mse: 4.4675
Epoch 76/100
30/30 [==============================] - 0s 3ms/step - loss: 3.9793 -
mae: 1.5578 - mse: 3.9793 - val_loss: 4.2477 - val_mae: 1.5446 -
val_mse: 4.2477
Epoch 77/100
30/30 [==============================] - 0s 3ms/step - loss: 3.8544 -
mae: 1.4918 - mse: 3.8544 - val_loss: 4.1481 - val_mae: 1.5330 -
val_mse: 4.1481
Epoch 78/100
30/30 [==============================] - 0s 3ms/step - loss: 3.7512 -
mae: 1.5340 - mse: 3.7512 - val_loss: 4.0105 - val_mae: 1.5196 -
val_mse: 4.0105
Epoch 79/100
30/30 [==============================] - 0s 3ms/step - loss: 3.6378 -
mae: 1.4744 - mse: 3.6378 - val_loss: 3.8792 - val_mae: 1.4897 -
val_mse: 3.8792
Epoch 80/100
30/30 [==============================] - 0s 3ms/step - loss: 3.5031 -
mae: 1.4756 - mse: 3.5031 - val_loss: 3.7193 - val_mae: 1.4690 -
val_mse: 3.7193
Epoch 81/100
30/30 [==============================] - 0s 3ms/step - loss: 3.4083 -
mae: 1.4606 - mse: 3.4083 - val_loss: 3.6301 - val_mae: 1.4622 -
val_mse: 3.6301
Epoch 82/100
```

```
30/30 [==============================] - 0s 3ms/step - loss: 3.2874 -
mae: 1.4280 - mse: 3.2874 - val_loss: 3.4900 - val_mae: 1.4248 -
val_mse: 3.4900
Epoch 83/100
30/30 [==============================] - 0s 3ms/step - loss: 3.1945 -
mae: 1.4225 - mse: 3.1945 - val_loss: 3.3866 - val_mae: 1.4119 -
val_mse: 3.3866
Epoch 84/100
30/30 [==============================] - 0s 3ms/step - loss: 3.1029 -
mae: 1.4049 - mse: 3.1029 - val_loss: 3.2767 - val_mae: 1.3924 -
val_mse: 3.2767
Epoch 85/100
30/30 [==============================] - 0s 3ms/step - loss: 3.0195 -
mae: 1.3818 - mse: 3.0195 - val_loss: 3.1882 - val_mae: 1.3691 -
val_mse: 3.1882
Epoch 86/100
30/30 [==============================] - 0s 3ms/step - loss: 2.9373 -
mae: 1.3732 - mse: 2.9373 - val_loss: 3.0949 - val_mae: 1.3563 -
val_mse: 3.0949
Epoch 87/100
30/30 [==============================] - 0s 3ms/step - loss: 2.8748 -
mae: 1.3666 - mse: 2.8748 - val_loss: 3.0187 - val_mae: 1.3430 -
val_mse: 3.0187
Epoch 88/100
30/30 [==============================] - 0s 3ms/step - loss: 2.7798 -
mae: 1.3342 - mse: 2.7798 - val_loss: 2.9017 - val_mae: 1.3177 -
val_mse: 2.9017
Epoch 89/100
30/30 [==============================] - 0s 3ms/step - loss: 2.7073 -
mae: 1.3386 - mse: 2.7073 - val_loss: 2.8083 - val_mae: 1.2993 -
val_mse: 2.8083
Epoch 90/100
30/30 [==============================] - 0s 3ms/step - loss: 2.6397 -
mae: 1.3141 - mse: 2.6397 - val_loss: 2.7533 - val_mae: 1.2846 -
val_mse: 2.7533
Epoch 91/100
30/30 [==============================] - 0s 3ms/step - loss: 2.5919 -
mae: 1.2982 - mse: 2.5919 - val_loss: 2.7034 - val_mae: 1.2802 -
val_mse: 2.7034
Epoch 92/100
30/30 [==============================] - 0s 3ms/step - loss: 2.4809 -
mae: 1.2847 - mse: 2.4809 - val_loss: 2.5646 - val_mae: 1.2557 -
val_mse: 2.5646
Epoch 93/100
30/30 [==============================] - 0s 3ms/step - loss: 2.4427 -
mae: 1.2685 - mse: 2.4427 - val_loss: 2.5326 - val_mae: 1.2394 -
val_mse: 2.5326
Epoch 94/100
30/30 [==============================] - 0s 3ms/step - loss: 2.3954 -
mae: 1.2681 - mse: 2.3954 - val_loss: 2.4493 - val_mae: 1.2229 -
```

```
val_mse: 2.4493
Epoch 95/100
30/30 [==============================] - 0s 3ms/step - loss: 2.3039 -
mae: 1.2421 - mse: 2.3039 - val_loss: 2.3671 - val_mae: 1.2040 -
val_mse: 2.3671
Epoch 96/100
30/30 [==============================] - 0s 3ms/step - loss: 2.2542 -
mae: 1.2330 - mse: 2.2542 - val_loss: 2.3147 - val_mae: 1.1926 -
val_mse: 2.3147
Epoch 97/100
30/30 [==============================] - 0s 3ms/step - loss: 2.1940 -
mae: 1.2185 - mse: 2.1940 - val_loss: 2.2475 - val_mae: 1.1737 -
val_mse: 2.2475
Epoch 98/100
30/30 [==============================] - 0s 3ms/step - loss: 2.1447 -
mae: 1.2062 - mse: 2.1447 - val_loss: 2.1697 - val_mae: 1.1586 -
val_mse: 2.1697
Epoch 99/100
30/30 [==============================] - 0s 3ms/step - loss: 2.0866 -
mae: 1.1893 - mse: 2.0866 - val_loss: 2.1355 - val_mae: 1.1529 -
val_mse: 2.1355
Epoch 100/100
30/30 [==============================] - 0s 3ms/step - loss: 2.0547 -
mae: 1.1875 - mse: 2.0547 - val_loss: 2.0605 - val_mae: 1.1313 -
val_mse: 2.0605
```

```python
# Define a list to store the MSE scores for each epoch
train_mse_scores = []
test_mse_scores = []
```

```python
# Evaluate the model on the training data
y_train_pred = model.predict(X_train)
train_mse = mean_squared_error(y_train, y_train_pred)
train_mse_scores.append(train_mse)
print('Train MSE:', train_mse)
```

```
Train MSE: 2.014217738896983
```

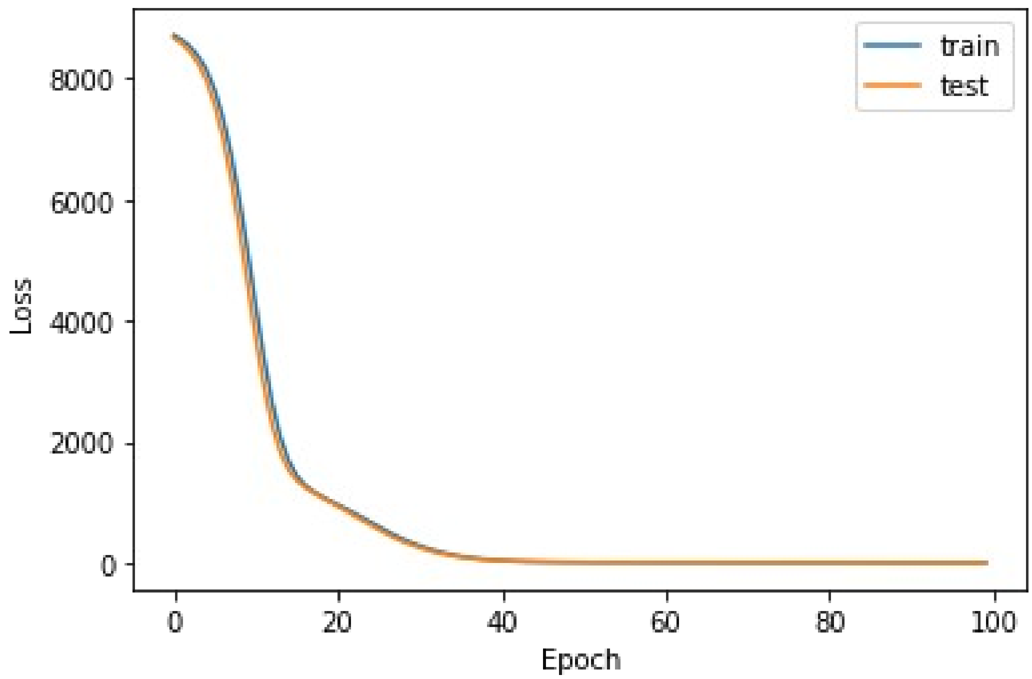
```python
# Evaluate the model on the test data
y_test_pred = model.predict(X_test)
test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse= math.sqrt(test_mse)
test_mse_scores.append(test_mse)


print('Test MSE:', test_mse)
print('Test RMSE:', test_rmse)
```

```
Test MSE: 2.060493790153123
Test RMSE: 1.4354420190844084
```

```
test_mae = mean_absolute_error(y_test, y_test_pred)
print('Test MAE:', test_mae)
```

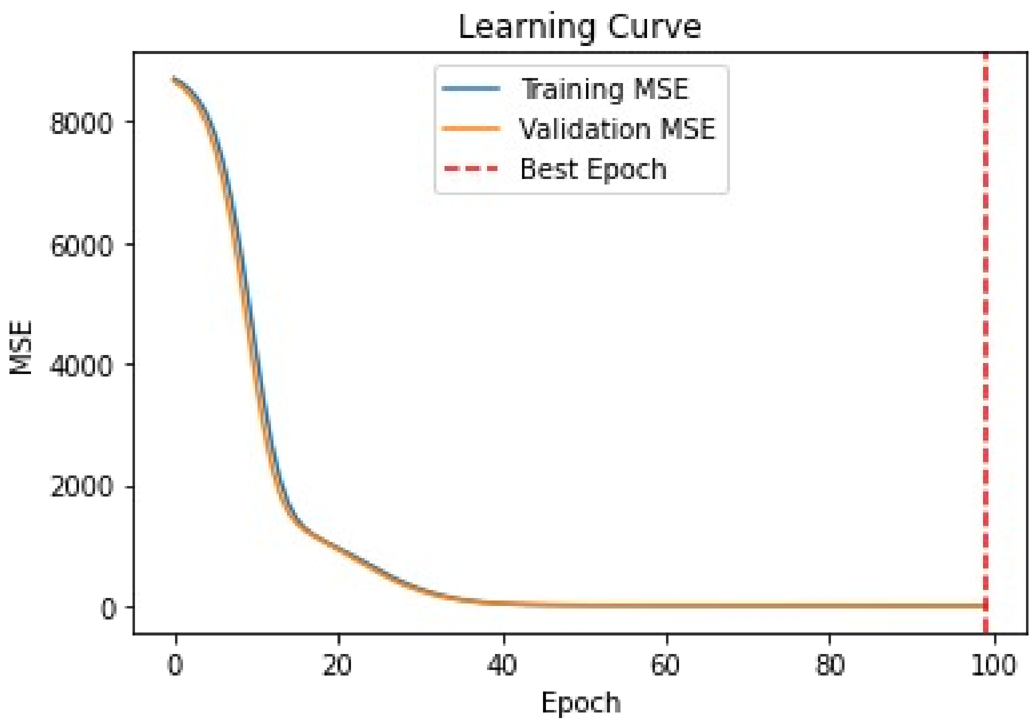
```
Test MAE: 1.1313496271769206
```

```
# Plot the learning curve
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
#check if error rate will be reducing and at what nth epoch
#Extract the MSE scores for the training and test data at each epoch
train_mse_scores = history.history['mse']
test_mse_scores = history.history['val_mse']

# Find the epoch with the lowest test MSE
best_epoch = np.argmin(test_mse_scores)
best_test_mse = test_mse_scores[best_epoch]

# Plot the learning curve
plt.plot(train_mse_scores, label='Training MSE')
plt.plot(test_mse_scores, label='Validation MSE')
plt.axvline(x=best_epoch, linestyle='--', color='r', label='Best
Epoch')
plt.title('Learning Curve')
plt.xlabel('Epoch')
```

```
plt.ylabel('MSE')
plt.legend()
plt.show()

print(f"Best epoch: {best_epoch+1}")
print(f"Best test MSE: {best_test_mse:.4f}")


#This code will plot the learning curve of the model and show the best
epoch
#(i.e., the epoch with the lowest test MSE) and the corresponding test
MSE value.
#The plot will show if the error rate is reducing over epochs and at
which epoch the model performance is optimal.

#so the lower the loss becomes,the better the model performance wll
be.
```




```
Best epoch: 100
Best test MSE: 2.0605
```

Learning curve analysis

#Based on the learning curve plot, the model's performance improves during the first few epochs,

#but eventually reaches a plateau around epoch 30. The loss (MSE) on the training data continues to decrease over time,

#while the loss on the validation data appears to level off around epoch 60.

#Based on the plot, the best epoch seems to be around epoch 100, as this is where the validation loss is lowest.

#The resulting test MSE of 2.0605 suggests that the model is able to predict the target variable with a reasonable degree of accuracy

#From the plot, we can see that the MSE decreases as the number of epochs increases, indicating that the model is improving over time.

```
#Training Time

%time history =  model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=16)
```

```
Epoch 1/100
30/30 [==============================] - 0s 5ms/step - loss: 2.0117 - mae: 1.1696 - mse: 2.0117 - val_loss: 2.0421 - val_mae: 1.1237 - val_mse: 2.0421
Epoch 2/100
30/30 [==============================] - 0s 4ms/step - loss: 1.9536 - mae: 1.1623 - mse: 1.9536 - val_loss: 1.9592 - val_mae: 1.1082 - val_mse: 1.9592
Epoch 3/100
30/30 [==============================] - 0s 4ms/step - loss: 1.9184 - mae: 1.1458 - mse: 1.9184 - val_loss: 1.9387 - val_mae: 1.1031 - val_mse: 1.9387
Epoch 4/100
30/30 [==============================] - 0s 3ms/step - loss: 1.8916 - mae: 1.1525 - mse: 1.8916 - val_loss: 1.8676 - val_mae: 1.0864 - val_mse: 1.8676
Epoch 5/100
30/30 [==============================] - 0s 3ms/step - loss: 1.8338 - mae: 1.1182 - mse: 1.8338 - val_loss: 1.8273 - val_mae: 1.0696 - val_mse: 1.8273
Epoch 6/100
30/30 [==============================] - 0s 3ms/step - loss: 1.7983 - mae: 1.1189 - mse: 1.7983 - val_loss: 1.8004 - val_mae: 1.0663 - val_mse: 1.8004
Epoch 7/100
30/30 [==============================] - 0s 4ms/step - loss: 1.7526 - mae: 1.1084 - mse: 1.7526 - val_loss: 1.7569 - val_mae: 1.0575 - val_mse: 1.7569
Epoch 8/100
30/30 [==============================] - 0s 3ms/step - loss: 1.7178 - mae: 1.0926 - mse: 1.7178 - val_loss: 1.6981 - val_mae: 1.0408 - val_mse: 1.6981
```

```
Epoch 9/100
30/30 [==============================] - 0s 4ms/step - loss: 1.6990 -
mae: 1.0870 - mse: 1.6990 - val_loss: 1.6600 - val_mae: 1.0349 -
val_mse: 1.6600
Epoch 10/100
30/30 [==============================] - 0s 4ms/step - loss: 1.6620 -
mae: 1.0760 - mse: 1.6620 - val_loss: 1.6437 - val_mae: 1.0351 -
val_mse: 1.6437
Epoch 11/100
30/30 [==============================] - 0s 4ms/step - loss: 1.6129 -
mae: 1.0651 - mse: 1.6129 - val_loss: 1.5806 - val_mae: 1.0071 -
val_mse: 1.5806
Epoch 12/100
30/30 [==============================] - 0s 4ms/step - loss: 1.5900 -
mae: 1.0442 - mse: 1.5900 - val_loss: 1.5544 - val_mae: 1.0007 -
val_mse: 1.5544
Epoch 13/100
30/30 [==============================] - 0s 3ms/step - loss: 1.5560 -
mae: 1.0514 - mse: 1.5560 - val_loss: 1.5285 - val_mae: 0.9933 -
val_mse: 1.5285
Epoch 14/100
30/30 [==============================] - 0s 4ms/step - loss: 1.5397 -
mae: 1.0257 - mse: 1.5397 - val_loss: 1.4875 - val_mae: 0.9813 -
val_mse: 1.4875
Epoch 15/100
30/30 [==============================] - 0s 4ms/step - loss: 1.5153 -
mae: 1.0358 - mse: 1.5153 - val_loss: 1.4589 - val_mae: 0.9691 -
val_mse: 1.4589
Epoch 16/100
30/30 [==============================] - 0s 4ms/step - loss: 1.4726 -
mae: 1.0176 - mse: 1.4726 - val_loss: 1.4360 - val_mae: 0.9623 -
val_mse: 1.4360
Epoch 17/100
30/30 [==============================] - 0s 4ms/step - loss: 1.4407 -
mae: 1.0083 - mse: 1.4407 - val_loss: 1.4222 - val_mae: 0.9613 -
val_mse: 1.4222
Epoch 18/100
30/30 [==============================] - 0s 4ms/step - loss: 1.4146 -
mae: 0.9972 - mse: 1.4146 - val_loss: 1.3966 - val_mae: 0.9519 -
val_mse: 1.3966
Epoch 19/100
30/30 [==============================] - 0s 4ms/step - loss: 1.3927 -
mae: 0.9875 - mse: 1.3927 - val_loss: 1.3911 - val_mae: 0.9531 -
val_mse: 1.3911
Epoch 20/100
30/30 [==============================] - 0s 3ms/step - loss: 1.3709 -
mae: 0.9845 - mse: 1.3709 - val_loss: 1.3423 - val_mae: 0.9344 -
val_mse: 1.3423
Epoch 21/100
30/30 [==============================] - 0s 5ms/step - loss: 1.3647 -
```

```
mae: 0.9823 - mse: 1.3647 - val_loss: 1.3581 - val_mae: 0.9409 -
val_mse: 1.3581
Epoch 22/100
30/30 [==============================] - 0s 4ms/step - loss: 1.3209 -
mae: 0.9626 - mse: 1.3209 - val_loss: 1.3151 - val_mae: 0.9336 -
val_mse: 1.3151
Epoch 23/100
30/30 [==============================] - 0s 3ms/step - loss: 1.3125 -
mae: 0.9603 - mse: 1.3125 - val_loss: 1.2713 - val_mae: 0.9119 -
val_mse: 1.2713
Epoch 24/100
30/30 [==============================] - 0s 4ms/step - loss: 1.2755 -
mae: 0.9515 - mse: 1.2755 - val_loss: 1.3073 - val_mae: 0.9216 -
val_mse: 1.3073
Epoch 25/100
30/30 [==============================] - 0s 3ms/step - loss: 1.2624 -
mae: 0.9392 - mse: 1.2624 - val_loss: 1.2535 - val_mae: 0.9024 -
val_mse: 1.2535
Epoch 26/100
30/30 [==============================] - 0s 4ms/step - loss: 1.2425 -
mae: 0.9390 - mse: 1.2425 - val_loss: 1.2149 - val_mae: 0.8921 -
val_mse: 1.2149
Epoch 27/100
30/30 [==============================] - 0s 3ms/step - loss: 1.2186 -
mae: 0.9239 - mse: 1.2186 - val_loss: 1.1934 - val_mae: 0.8801 -
val_mse: 1.1934
Epoch 28/100
30/30 [==============================] - 0s 4ms/step - loss: 1.1979 -
mae: 0.9228 - mse: 1.1979 - val_loss: 1.1762 - val_mae: 0.8713 -
val_mse: 1.1762
Epoch 29/100
30/30 [==============================] - 0s 4ms/step - loss: 1.2027 -
mae: 0.9139 - mse: 1.2027 - val_loss: 1.1771 - val_mae: 0.8675 -
val_mse: 1.1771
Epoch 30/100
30/30 [==============================] - 0s 4ms/step - loss: 1.1719 -
mae: 0.9032 - mse: 1.1719 - val_loss: 1.1432 - val_mae: 0.8649 -
val_mse: 1.1432
Epoch 31/100
30/30 [==============================] - 0s 3ms/step - loss: 1.1445 -
mae: 0.8888 - mse: 1.1445 - val_loss: 1.1781 - val_mae: 0.8697 -
val_mse: 1.1781
Epoch 32/100
30/30 [==============================] - 0s 3ms/step - loss: 1.1245 -
mae: 0.8892 - mse: 1.1245 - val_loss: 1.1442 - val_mae: 0.8549 -
val_mse: 1.1442
Epoch 33/100
30/30 [==============================] - 0s 3ms/step - loss: 1.1379 -
mae: 0.8939 - mse: 1.1379 - val_loss: 1.0911 - val_mae: 0.8387 -
val_mse: 1.0911
```

```
Epoch 34/100
30/30 [==============================] - 0s 4ms/step - loss: 1.0970 -
mae: 0.8652 - mse: 1.0970 - val_loss: 1.1070 - val_mae: 0.8506 -
val_mse: 1.1070
Epoch 35/100
30/30 [==============================] - 0s 4ms/step - loss: 1.0796 -
mae: 0.8681 - mse: 1.0796 - val_loss: 1.1027 - val_mae: 0.8391 -
val_mse: 1.1027
Epoch 36/100
30/30 [==============================] - 0s 3ms/step - loss: 1.0628 -
mae: 0.8601 - mse: 1.0628 - val_loss: 1.0668 - val_mae: 0.8306 -
val_mse: 1.0668
Epoch 37/100
30/30 [==============================] - 0s 3ms/step - loss: 1.0626 -
mae: 0.8549 - mse: 1.0626 - val_loss: 1.0598 - val_mae: 0.8371 -
val_mse: 1.0598
Epoch 38/100
30/30 [==============================] - 0s 3ms/step - loss: 1.0336 -
mae: 0.8520 - mse: 1.0336 - val_loss: 1.0723 - val_mae: 0.8290 -
val_mse: 1.0723
Epoch 39/100
30/30 [==============================] - 0s 5ms/step - loss: 1.0224 -
mae: 0.8425 - mse: 1.0224 - val_loss: 1.0290 - val_mae: 0.8069 -
val_mse: 1.0290
Epoch 40/100
30/30 [==============================] - 0s 3ms/step - loss: 1.0085 -
mae: 0.8341 - mse: 1.0085 - val_loss: 1.0032 - val_mae: 0.8008 -
val_mse: 1.0032
Epoch 41/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9936 -
mae: 0.8291 - mse: 0.9936 - val_loss: 1.0116 - val_mae: 0.8059 -
val_mse: 1.0116
Epoch 42/100
30/30 [==============================] - 0s 3ms/step - loss: 1.0166 -
mae: 0.8314 - mse: 1.0166 - val_loss: 0.9901 - val_mae: 0.8015 -
val_mse: 0.9901
Epoch 43/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9864 -
mae: 0.8322 - mse: 0.9864 - val_loss: 1.0227 - val_mae: 0.7929 -
val_mse: 1.0227
Epoch 44/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9632 -
mae: 0.8057 - mse: 0.9632 - val_loss: 0.9591 - val_mae: 0.7804 -
val_mse: 0.9591
Epoch 45/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9489 -
mae: 0.8029 - mse: 0.9489 - val_loss: 0.9546 - val_mae: 0.7804 -
val_mse: 0.9546
Epoch 46/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9351 -
```

```
mae: 0.8110 - mse: 0.9351 - val_loss: 0.9715 - val_mae: 0.7734 -
val_mse: 0.9715
Epoch 47/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9485 -
mae: 0.8030 - mse: 0.9485 - val_loss: 0.9954 - val_mae: 0.7794 -
val_mse: 0.9954
Epoch 48/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9194 -
mae: 0.7926 - mse: 0.9194 - val_loss: 0.9267 - val_mae: 0.7625 -
val_mse: 0.9267
Epoch 49/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9158 -
mae: 0.7864 - mse: 0.9158 - val_loss: 0.9121 - val_mae: 0.7688 -
val_mse: 0.9121
Epoch 50/100
30/30 [==============================] - 0s 3ms/step - loss: 0.9046 -
mae: 0.7856 - mse: 0.9046 - val_loss: 0.8966 - val_mae: 0.7523 -
val_mse: 0.8966
Epoch 51/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8913 -
mae: 0.7759 - mse: 0.8913 - val_loss: 0.9231 - val_mae: 0.7502 -
val_mse: 0.9231
Epoch 52/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8759 -
mae: 0.7790 - mse: 0.8759 - val_loss: 0.9099 - val_mae: 0.7462 -
val_mse: 0.9099
Epoch 53/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8712 -
mae: 0.7683 - mse: 0.8712 - val_loss: 0.8926 - val_mae: 0.7478 -
val_mse: 0.8926
Epoch 54/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8642 -
mae: 0.7694 - mse: 0.8642 - val_loss: 0.8984 - val_mae: 0.7360 -
val_mse: 0.8984
Epoch 55/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8432 -
mae: 0.7523 - mse: 0.8432 - val_loss: 0.8706 - val_mae: 0.7382 -
val_mse: 0.8706
Epoch 56/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8347 -
mae: 0.7463 - mse: 0.8347 - val_loss: 0.8492 - val_mae: 0.7386 -
val_mse: 0.8492
Epoch 57/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8259 -
mae: 0.7527 - mse: 0.8259 - val_loss: 0.8846 - val_mae: 0.7240 -
val_mse: 0.8846
Epoch 58/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8275 -
mae: 0.7429 - mse: 0.8275 - val_loss: 0.8545 - val_mae: 0.7178 -
val_mse: 0.8545
```

```
Epoch 59/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8187 -
mae: 0.7423 - mse: 0.8187 - val_loss: 0.8430 - val_mae: 0.7157 -
val_mse: 0.8430
Epoch 60/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7950 -
mae: 0.7343 - mse: 0.7950 - val_loss: 0.8180 - val_mae: 0.7104 -
val_mse: 0.8180
Epoch 61/100
30/30 [==============================] - 0s 3ms/step - loss: 0.8082 -
mae: 0.7378 - mse: 0.8082 - val_loss: 0.8115 - val_mae: 0.7312 -
val_mse: 0.8115
Epoch 62/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7798 -
mae: 0.7219 - mse: 0.7798 - val_loss: 0.8464 - val_mae: 0.7100 -
val_mse: 0.8464
Epoch 63/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7724 -
mae: 0.7211 - mse: 0.7724 - val_loss: 0.8173 - val_mae: 0.7036 -
val_mse: 0.8173
Epoch 64/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7623 -
mae: 0.7179 - mse: 0.7623 - val_loss: 0.7783 - val_mae: 0.6999 -
val_mse: 0.7783
Epoch 65/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7657 -
mae: 0.7131 - mse: 0.7657 - val_loss: 0.7832 - val_mae: 0.6928 -
val_mse: 0.7832
Epoch 66/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7498 -
mae: 0.7125 - mse: 0.7498 - val_loss: 0.7774 - val_mae: 0.6841 -
val_mse: 0.7774
Epoch 67/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7383 -
mae: 0.7051 - mse: 0.7383 - val_loss: 0.7977 - val_mae: 0.7012 -
val_mse: 0.7977
Epoch 68/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7364 -
mae: 0.7010 - mse: 0.7364 - val_loss: 0.7716 - val_mae: 0.6809 -
val_mse: 0.7716
Epoch 69/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7389 -
mae: 0.7051 - mse: 0.7389 - val_loss: 0.7478 - val_mae: 0.6735 -
val_mse: 0.7478
Epoch 70/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7180 -
mae: 0.6939 - mse: 0.7180 - val_loss: 0.7507 - val_mae: 0.7031 -
val_mse: 0.7507
Epoch 71/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7187 -
```

```
mae: 0.6935 - mse: 0.7187 - val_loss: 0.7457 - val_mae: 0.6673 -
val_mse: 0.7457
Epoch 72/100
30/30 [==============================] - 0s 3ms/step - loss: 0.7068 -
mae: 0.6908 - mse: 0.7068 - val_loss: 0.7287 - val_mae: 0.6604 -
val_mse: 0.7287
Epoch 73/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6987 -
mae: 0.6857 - mse: 0.6987 - val_loss: 0.7405 - val_mae: 0.6598 -
val_mse: 0.7405
Epoch 74/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6942 -
mae: 0.6829 - mse: 0.6942 - val_loss: 0.7329 - val_mae: 0.6635 -
val_mse: 0.7329
Epoch 75/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6876 -
mae: 0.6817 - mse: 0.6876 - val_loss: 0.7129 - val_mae: 0.6677 -
val_mse: 0.7129
Epoch 76/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6791 -
mae: 0.6798 - mse: 0.6791 - val_loss: 0.7161 - val_mae: 0.6508 -
val_mse: 0.7161
Epoch 77/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6827 -
mae: 0.6794 - mse: 0.6827 - val_loss: 0.6968 - val_mae: 0.6507 -
val_mse: 0.6968
Epoch 78/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6647 -
mae: 0.6642 - mse: 0.6647 - val_loss: 0.7877 - val_mae: 0.6788 -
val_mse: 0.7877
Epoch 79/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6581 -
mae: 0.6649 - mse: 0.6581 - val_loss: 0.7093 - val_mae: 0.6561 -
val_mse: 0.7093
Epoch 80/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6487 -
mae: 0.6609 - mse: 0.6487 - val_loss: 0.6747 - val_mae: 0.6453 -
val_mse: 0.6747
Epoch 81/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6444 -
mae: 0.6553 - mse: 0.6444 - val_loss: 0.7709 - val_mae: 0.6668 -
val_mse: 0.7709
Epoch 82/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6548 -
mae: 0.6601 - mse: 0.6548 - val_loss: 0.6998 - val_mae: 0.6472 -
val_mse: 0.6998
Epoch 83/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6239 -
mae: 0.6445 - mse: 0.6239 - val_loss: 0.6673 - val_mae: 0.6302 -
val_mse: 0.6673
```

```
Epoch 84/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6277 -
mae: 0.6470 - mse: 0.6277 - val_loss: 0.6607 - val_mae: 0.6207 -
val_mse: 0.6607
Epoch 85/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6166 -
mae: 0.6439 - mse: 0.6166 - val_loss: 0.6995 - val_mae: 0.6403 -
val_mse: 0.6995
Epoch 86/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6034 -
mae: 0.6393 - mse: 0.6034 - val_loss: 0.6502 - val_mae: 0.6225 -
val_mse: 0.6502
Epoch 87/100
30/30 [==============================] - 0s 3ms/step - loss: 0.6106 -
mae: 0.6360 - mse: 0.6106 - val_loss: 0.6410 - val_mae: 0.6351 -
val_mse: 0.6410
Epoch 88/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5990 -
mae: 0.6389 - mse: 0.5990 - val_loss: 0.6530 - val_mae: 0.6306 -
val_mse: 0.6530
Epoch 89/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5980 -
mae: 0.6392 - mse: 0.5980 - val_loss: 0.6438 - val_mae: 0.6085 -
val_mse: 0.6438
Epoch 90/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5848 -
mae: 0.6298 - mse: 0.5848 - val_loss: 0.6447 - val_mae: 0.6163 -
val_mse: 0.6447
Epoch 91/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5707 -
mae: 0.6158 - mse: 0.5707 - val_loss: 0.6070 - val_mae: 0.6108 -
val_mse: 0.6070
Epoch 92/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5730 -
mae: 0.6267 - mse: 0.5730 - val_loss: 0.6062 - val_mae: 0.6281 -
val_mse: 0.6062
Epoch 93/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5662 -
mae: 0.6171 - mse: 0.5662 - val_loss: 0.6246 - val_mae: 0.6020 -
val_mse: 0.6246
Epoch 94/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5593 -
mae: 0.6077 - mse: 0.5593 - val_loss: 0.6224 - val_mae: 0.6061 -
val_mse: 0.6224
Epoch 95/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5581 -
mae: 0.6157 - mse: 0.5581 - val_loss: 0.5962 - val_mae: 0.6197 -
val_mse: 0.5962
Epoch 96/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5486 -
```

```
mae: 0.6059 - mse: 0.5486 - val_loss: 0.5747 - val_mae: 0.5997 -
val_mse: 0.5747
Epoch 97/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5362 -
mae: 0.5999 - mse: 0.5362 - val_loss: 0.5762 - val_mae: 0.5904 -
val_mse: 0.5762
Epoch 98/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5428 -
mae: 0.6026 - mse: 0.5428 - val_loss: 0.6191 - val_mae: 0.6013 -
val_mse: 0.6191
Epoch 99/100
30/30 [==============================] - 0s 3ms/step - loss: 0.5321 -
mae: 0.6027 - mse: 0.5321 - val_loss: 0.5912 - val_mae: 0.5830 -
val_mse: 0.5912
Epoch 100/100
30/30 [==============================] - 0s 4ms/step - loss: 0.5165 -
mae: 0.5883 - mse: 0.5165 - val_loss: 0.5676 - val_mae: 0.5726 -
val_mse: 0.5676
Wall time: 10.5 s
```

```python
#convert the nth epoch value to an equivalent time duration.
import datetime

total_training_time = 10.5
num_epochs = 100
time_per_epoch = total_training_time / num_epochs
epoch_of_interest = 100

time_duration = datetime.timedelta(seconds=epoch_of_interest *
time_per_epoch)
print("Equivalent time duration for {}th epoch:
{}".format(epoch_of_interest, time_duration))

#That means that the 100th epoch took approximately 10 seconds to
complete
```

```
Equivalent time duration for 100th epoch: 0:00:10.500000
```