2306AML138 MORLA VENKATA LAKSHMI AIML 16

ANN MODEL using linear regression , classifiers

```
#importing the libraries
import pandas as pd
import numpy as np
# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.gofplots import ProbPlot

# import libraries for building linear regression model
from statsmodels.formula.api import ols
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression

# import library for preparing data
from sklearn.model_selection import train_test_split

# import library for data preprocessing
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings("ignore")
```

imported libraries which are required

```
data = pd.read_csv('BostonHousing.csv')
data
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
```

```
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

b:

-Bostenhousing consisting of the features crim zn indus chas nox rm age dis rad tax ptratio b lstat medv means -CRIM: per capita crime rate by town // -ZN: proportion of residential land zoned for lots over 25,000 sq.ft.// -INDUS: proportion of non-retail business acres per town// -CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)// -NOX: nitric oxides concentration (parts per 10 million)// -RM: average number of rooms per dwelling// -AGE: proportion of owner-occupied units built prior to 1940// -DIS: weighted distances to five Boston employment centres// -RAD: index of accessibility to radial highways// -TAX: full-value property-tax rate per 10,000 dollars// -PTRATIO: pupil-teacher ratio by town// -B1000: (Bk - 0.63)^2 where Bk is the proportion of blacks by town// -LSTAT: %lower status of the population// -MEDV: Median value of owner-occupied homes in 1000 dollars.//

above data havinng the 506 rows and16 columns

```
print (data.dtypes)
```

```
crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
lstat      float64
medv       float64
dtype: object
```

above information tells about the features of datatypes integers or float,text

```
print(data.describe())
```

```
             crim          zn       indus        chas         nox          rm
\
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
```

```
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              age         dis         rad         tax     ptratio           b
\
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            lstat        medv
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

features have different statistical values to understand the distrubution of features
describe function used here

Observations:____

- Crime rates average is 3.6 with very low crime rates in 50% of towns and extreme high rates in other towns.
- At least 50% of Boston towns have no zoned lands for large lot.
- The mean of CHAS is 0.07, which means that most of the houses are not on riverside.

```
data.isnull().sum()
```

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
```

```
b          0
lstat      0
medv       0
dtype: int64
```

indicates null functions are not there ,all features having the values

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
```

imported the required libraries for regression model

splitting data in to train and test

```
target=data['medv']

X_train, X_test, y_train, y_test = train_test_split(data, target,
test_size=0.25, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build the Regression ANN model
model = keras.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)  # Output layer with one neuron (Regression)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1)
```

```
Epoch 1/100
12/12 [==============================] - 3s 7ms/step - loss: 616.4184
Epoch 2/100
12/12 [==============================] - 0s 5ms/step - loss: 591.6747
Epoch 3/100
12/12 [==============================] - 0s 4ms/step - loss: 566.0430
Epoch 4/100
12/12 [==============================] - 0s 6ms/step - loss: 536.0198
Epoch 5/100
12/12 [==============================] - 0s 3ms/step - loss: 496.6950
Epoch 6/100
```

```
12/12 [==============================] - 0s 5ms/step - loss: 447.2167
Epoch 7/100
12/12 [==============================] - 0s 5ms/step - loss: 384.6385
Epoch 8/100
12/12 [==============================] - 0s 4ms/step - loss: 311.2697
Epoch 9/100
12/12 [==============================] - 0s 5ms/step - loss: 227.2774
Epoch 10/100
12/12 [==============================] - 0s 3ms/step - loss: 145.3581
Epoch 11/100
12/12 [==============================] - 0s 4ms/step - loss: 87.0497
Epoch 12/100
12/12 [==============================] - 0s 5ms/step - loss: 55.4233
Epoch 13/100
12/12 [==============================] - 0s 4ms/step - loss: 40.9755
Epoch 14/100
12/12 [==============================] - 0s 5ms/step - loss: 31.5736
Epoch 15/100
12/12 [==============================] - 0s 3ms/step - loss: 24.8267
Epoch 16/100
12/12 [==============================] - 0s 3ms/step - loss: 20.8123
Epoch 17/100
12/12 [==============================] - 0s 4ms/step - loss: 18.1267
Epoch 18/100
12/12 [==============================] - 0s 3ms/step - loss: 16.1995
Epoch 19/100
12/12 [==============================] - 0s 4ms/step - loss: 14.7293
Epoch 20/100
12/12 [==============================] - 0s 4ms/step - loss: 13.5844
Epoch 21/100
12/12 [==============================] - 0s 4ms/step - loss: 12.6181
Epoch 22/100
12/12 [==============================] - 0s 5ms/step - loss: 11.7994
Epoch 23/100
12/12 [==============================] - 0s 4ms/step - loss: 11.0654
Epoch 24/100
12/12 [==============================] - 0s 6ms/step - loss: 10.4192
Epoch 25/100
12/12 [==============================] - 0s 5ms/step - loss: 9.8784
Epoch 26/100
12/12 [==============================] - 0s 6ms/step - loss: 9.3456
Epoch 27/100
12/12 [==============================] - 0s 4ms/step - loss: 8.9152
Epoch 28/100
12/12 [==============================] - 0s 4ms/step - loss: 8.4453
Epoch 29/100
12/12 [==============================] - 0s 4ms/step - loss: 8.1172
Epoch 30/100
12/12 [==============================] - 0s 5ms/step - loss: 7.7172
Epoch 31/100
```

```
12/12 [==============================] - 0s 3ms/step - loss: 7.4138
Epoch 32/100
12/12 [==============================] - 0s 5ms/step - loss: 7.1354
Epoch 33/100
12/12 [==============================] - 0s 5ms/step - loss: 6.8667
Epoch 34/100
12/12 [==============================] - 0s 4ms/step - loss: 6.6203
Epoch 35/100
12/12 [==============================] - 0s 5ms/step - loss: 6.3902
Epoch 36/100
12/12 [==============================] - 0s 4ms/step - loss: 6.1781
Epoch 37/100
12/12 [==============================] - 0s 5ms/step - loss: 5.9706
Epoch 38/100
12/12 [==============================] - 0s 5ms/step - loss: 5.7903
Epoch 39/100
12/12 [==============================] - 0s 5ms/step - loss: 5.6224
Epoch 40/100
12/12 [==============================] - 0s 4ms/step - loss: 5.4648
Epoch 41/100
12/12 [==============================] - 0s 4ms/step - loss: 5.2944
Epoch 42/100
12/12 [==============================] - 0s 4ms/step - loss: 5.1538
Epoch 43/100
12/12 [==============================] - 0s 4ms/step - loss: 5.0201
Epoch 44/100
12/12 [==============================] - 0s 6ms/step - loss: 4.8712
Epoch 45/100
12/12 [==============================] - 0s 5ms/step - loss: 4.7401
Epoch 46/100
12/12 [==============================] - 0s 5ms/step - loss: 4.6155
Epoch 47/100
12/12 [==============================] - 0s 5ms/step - loss: 4.5071
Epoch 48/100
12/12 [==============================] - 0s 5ms/step - loss: 4.3998
Epoch 49/100
12/12 [==============================] - 0s 5ms/step - loss: 4.2991
Epoch 50/100
12/12 [==============================] - 0s 5ms/step - loss: 4.2174
Epoch 51/100
12/12 [==============================] - 0s 4ms/step - loss: 4.1137
Epoch 52/100
12/12 [==============================] - 0s 6ms/step - loss: 4.0314
Epoch 53/100
12/12 [==============================] - 0s 5ms/step - loss: 3.9392
Epoch 54/100
12/12 [==============================] - 0s 7ms/step - loss: 3.8485
Epoch 55/100
12/12 [==============================] - 0s 6ms/step - loss: 3.7839
Epoch 56/100
```

```
12/12 [==============================] - 0s 4ms/step - loss: 3.7173
Epoch 57/100
12/12 [==============================] - 0s 5ms/step - loss: 3.6302
Epoch 58/100
12/12 [==============================] - 0s 4ms/step - loss: 3.5610
Epoch 59/100
12/12 [==============================] - 0s 6ms/step - loss: 3.4942
Epoch 60/100
12/12 [==============================] - 0s 7ms/step - loss: 3.4269
Epoch 61/100
12/12 [==============================] - 0s 6ms/step - loss: 3.3557
Epoch 62/100
12/12 [==============================] - 0s 8ms/step - loss: 3.2968
Epoch 63/100
12/12 [==============================] - 0s 9ms/step - loss: 3.2223
Epoch 64/100
12/12 [==============================] - 0s 6ms/step - loss: 3.1728
Epoch 65/100
12/12 [==============================] - 0s 7ms/step - loss: 3.1093
Epoch 66/100
12/12 [==============================] - 0s 5ms/step - loss: 3.0891
Epoch 67/100
12/12 [==============================] - 0s 5ms/step - loss: 3.0209
Epoch 68/100
12/12 [==============================] - 0s 5ms/step - loss: 2.9403
Epoch 69/100
12/12 [==============================] - 0s 6ms/step - loss: 2.8953
Epoch 70/100
12/12 [==============================] - 0s 5ms/step - loss: 2.8451
Epoch 71/100
12/12 [==============================] - 0s 5ms/step - loss: 2.7837
Epoch 72/100
12/12 [==============================] - 0s 5ms/step - loss: 2.7153
Epoch 73/100
12/12 [==============================] - 0s 4ms/step - loss: 2.6647
Epoch 74/100
12/12 [==============================] - 0s 4ms/step - loss: 2.6129
Epoch 75/100
12/12 [==============================] - 0s 3ms/step - loss: 2.5790
Epoch 76/100
12/12 [==============================] - 0s 2ms/step - loss: 2.5431
Epoch 77/100
12/12 [==============================] - 0s 3ms/step - loss: 2.5016
Epoch 78/100
12/12 [==============================] - 0s 3ms/step - loss: 2.4374
Epoch 79/100
12/12 [==============================] - 0s 3ms/step - loss: 2.3754
Epoch 80/100
12/12 [==============================] - 0s 3ms/step - loss: 2.3422
Epoch 81/100
```

```
12/12 [==============================] - 0s 3ms/step - loss: 2.3047
Epoch 82/100
12/12 [==============================] - 0s 3ms/step - loss: 2.2706
Epoch 83/100
12/12 [==============================] - 0s 3ms/step - loss: 2.2205
Epoch 84/100
12/12 [==============================] - 0s 4ms/step - loss: 2.1711
Epoch 85/100
12/12 [==============================] - 0s 3ms/step - loss: 2.1310
Epoch 86/100
12/12 [==============================] - 0s 4ms/step - loss: 2.0864
Epoch 87/100
12/12 [==============================] - 0s 3ms/step - loss: 2.0559
Epoch 88/100
12/12 [==============================] - 0s 3ms/step - loss: 2.0203
Epoch 89/100
12/12 [==============================] - 0s 3ms/step - loss: 1.9771
Epoch 90/100
12/12 [==============================] - 0s 3ms/step - loss: 1.9640
Epoch 91/100
12/12 [==============================] - 0s 3ms/step - loss: 1.9149
Epoch 92/100
12/12 [==============================] - 0s 3ms/step - loss: 1.8747
Epoch 93/100
12/12 [==============================] - 0s 3ms/step - loss: 1.8646
Epoch 94/100
12/12 [==============================] - 0s 3ms/step - loss: 1.8201
Epoch 95/100
12/12 [==============================] - 0s 3ms/step - loss: 1.7860
Epoch 96/100
12/12 [==============================] - 0s 4ms/step - loss: 1.7386
Epoch 97/100
12/12 [==============================] - 0s 3ms/step - loss: 1.7069
Epoch 98/100
12/12 [==============================] - 0s 3ms/step - loss: 1.6780
Epoch 99/100
12/12 [==============================] - 0s 3ms/step - loss: 1.6773
Epoch 100/100
12/12 [==============================] - 0s 3ms/step - loss: 1.6294

<keras.src.callbacks.History at 0x25151f45180>
```

```
# Evaluate the model on the test set
loss = model.evaluate(X_test, y_test)
print(f"Mean Squared Error on Test Data: {loss:.2f}")
```

```
4/4 [==============================] - 0s 3ms/step - loss: 2.4941
Mean Squared Error on Test Data: 2.49
```

A Mean Squared Error (MSE) of 2.49 on the test data indicates how well regression model is performing. In this context, MSE measures the average squared difference between the actual house prices and the predicted house prices.

```
model2 = keras.Sequential([
    keras.layers.Input(shape=(X_train.shape[1],)),
    keras.layers.Dense(64, activation='tanh'),
    keras.layers.Dense(32, activation='tanh'),
    keras.layers.Dense(1) # Output layer with one neuron (Regression)
])
# Change the activation function and continue using the same model

#we are etting the same value lossfunction with out changing for numberof epochs
#  Early Stopping this is adopted for no changing value happend in below models selected
from tensorflow import keras
from tensorflow.keras.callbacks import EarlyStopping

# 3. Model Compilation
model2.compile(loss='mean_squared_error', optimizer='adam')

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

 # 4. Model Training
model2.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
12/12 [==============================] - 1s 4ms/step - loss: 22.3308
Epoch 2/100
12/12 [==============================] - 0s 4ms/step - loss: 21.2961
Epoch 3/100
12/12 [==============================] - 0s 4ms/step - loss: 20.2721
Epoch 4/100
12/12 [==============================] - 0s 4ms/step - loss: 19.4070
Epoch 5/100
12/12 [==============================] - 0s 4ms/step - loss: 18.4641
Epoch 6/100
12/12 [==============================] - 0s 4ms/step - loss: 17.6443
Epoch 7/100
12/12 [==============================] - 0s 4ms/step - loss: 16.8420
Epoch 8/100
12/12 [==============================] - 0s 4ms/step - loss: 16.1105
Epoch 9/100
12/12 [==============================] - 0s 7ms/step - loss: 15.3726
Epoch 10/100
12/12 [==============================] - 0s 4ms/step - loss: 14.7286
Epoch 11/100
12/12 [==============================] - 0s 4ms/step - loss: 14.0730
Epoch 12/100
12/12 [==============================] - 0s 4ms/step - loss: 13.4599
```

```
Epoch 13/100
12/12 [==============================] - 0s 4ms/step - loss: 12.9397
Epoch 14/100
12/12 [==============================] - 0s 5ms/step - loss: 12.3779
Epoch 15/100
12/12 [==============================] - 0s 4ms/step - loss: 11.8583
Epoch 16/100
12/12 [==============================] - 0s 4ms/step - loss: 11.3751
Epoch 17/100
12/12 [==============================] - 0s 5ms/step - loss: 10.9264
Epoch 18/100
12/12 [==============================] - 0s 3ms/step - loss: 10.4494
Epoch 19/100
12/12 [==============================] - 0s 3ms/step - loss: 10.0575
Epoch 20/100
12/12 [==============================] - 0s 4ms/step - loss: 9.6361
Epoch 21/100
12/12 [==============================] - 0s 3ms/step - loss: 9.2710
Epoch 22/100
12/12 [==============================] - 0s 4ms/step - loss: 8.8864
Epoch 23/100
12/12 [==============================] - 0s 4ms/step - loss: 8.5277
Epoch 24/100
12/12 [==============================] - 0s 3ms/step - loss: 8.2216
Epoch 25/100
12/12 [==============================] - 0s 4ms/step - loss: 7.8792
Epoch 26/100
12/12 [==============================] - 0s 4ms/step - loss: 7.5670
Epoch 27/100
12/12 [==============================] - 0s 5ms/step - loss: 7.2699
Epoch 28/100
12/12 [==============================] - 0s 3ms/step - loss: 6.9415
Epoch 29/100
12/12 [==============================] - 0s 4ms/step - loss: 6.6761
Epoch 30/100
12/12 [==============================] - 0s 4ms/step - loss: 6.3742
Epoch 31/100
12/12 [==============================] - 0s 5ms/step - loss: 6.1278
Epoch 32/100
12/12 [==============================] - 0s 5ms/step - loss: 5.8548
Epoch 33/100
12/12 [==============================] - 0s 4ms/step - loss: 5.6169
Epoch 34/100
12/12 [==============================] - 0s 6ms/step - loss: 5.3805
Epoch 35/100
12/12 [==============================] - 0s 5ms/step - loss: 5.1376
Epoch 36/100
12/12 [==============================] - 0s 4ms/step - loss: 4.9181
Epoch 37/100
12/12 [==============================] - 0s 4ms/step - loss: 4.7349
```

```
Epoch 38/100
12/12 [==============================] - 0s 4ms/step - loss: 4.5077
Epoch 39/100
12/12 [==============================] - 0s 4ms/step - loss: 4.3279
Epoch 40/100
12/12 [==============================] - 0s 4ms/step - loss: 4.1162
Epoch 41/100
12/12 [==============================] - 0s 6ms/step - loss: 3.9673
Epoch 42/100
12/12 [==============================] - 0s 5ms/step - loss: 3.7964
Epoch 43/100
12/12 [==============================] - 0s 5ms/step - loss: 3.6069
Epoch 44/100
12/12 [==============================] - 0s 6ms/step - loss: 3.4650
Epoch 45/100
12/12 [==============================] - 0s 5ms/step - loss: 3.3280
Epoch 46/100
12/12 [==============================] - 0s 6ms/step - loss: 3.1668
Epoch 47/100
12/12 [==============================] - 0s 5ms/step - loss: 3.0579
Epoch 48/100
12/12 [==============================] - 0s 5ms/step - loss: 2.9396
Epoch 49/100
12/12 [==============================] - 0s 4ms/step - loss: 2.8196
Epoch 50/100
12/12 [==============================] - 0s 6ms/step - loss: 2.6958
Epoch 51/100
12/12 [==============================] - 0s 5ms/step - loss: 2.5734
Epoch 52/100
12/12 [==============================] - 0s 6ms/step - loss: 2.4442
Epoch 53/100
12/12 [==============================] - 0s 5ms/step - loss: 2.3479
Epoch 54/100
12/12 [==============================] - 0s 5ms/step - loss: 2.2373
Epoch 55/100
12/12 [==============================] - 0s 5ms/step - loss: 2.1435
Epoch 56/100
12/12 [==============================] - 0s 3ms/step - loss: 2.0569
Epoch 57/100
12/12 [==============================] - 0s 5ms/step - loss: 1.9600
Epoch 58/100
12/12 [==============================] - 0s 5ms/step - loss: 1.8731
Epoch 59/100
12/12 [==============================] - 0s 4ms/step - loss: 1.8060
Epoch 60/100
12/12 [==============================] - 0s 5ms/step - loss: 1.7228
Epoch 61/100
12/12 [==============================] - 0s 4ms/step - loss: 1.6584
Epoch 62/100
12/12 [==============================] - 0s 6ms/step - loss: 1.5983
```

```
Epoch 63/100
12/12 [==============================] - 0s 6ms/step - loss: 1.5510
Epoch 64/100
12/12 [==============================] - 0s 4ms/step - loss: 1.4798
Epoch 65/100
12/12 [==============================] - 0s 4ms/step - loss: 1.4125
Epoch 66/100
12/12 [==============================] - 0s 4ms/step - loss: 1.3562
Epoch 67/100
12/12 [==============================] - 0s 4ms/step - loss: 1.2828
Epoch 68/100
12/12 [==============================] - 0s 5ms/step - loss: 1.2407
Epoch 69/100
12/12 [==============================] - 0s 4ms/step - loss: 1.1959
Epoch 70/100
12/12 [==============================] - 0s 3ms/step - loss: 1.1441
Epoch 71/100
12/12 [==============================] - 0s 7ms/step - loss: 1.0863
Epoch 72/100
12/12 [==============================] - 0s 7ms/step - loss: 1.0511
Epoch 73/100
12/12 [==============================] - 0s 4ms/step - loss: 1.0180
Epoch 74/100
12/12 [==============================] - 0s 4ms/step - loss: 1.0092
Epoch 75/100
12/12 [==============================] - 0s 4ms/step - loss: 0.9745
Epoch 76/100
12/12 [==============================] - 0s 5ms/step - loss: 0.9306
Epoch 77/100
12/12 [==============================] - 0s 4ms/step - loss: 0.8647
Epoch 78/100
12/12 [==============================] - 0s 4ms/step - loss: 0.8196
Epoch 79/100
12/12 [==============================] - 0s 4ms/step - loss: 0.7887
Epoch 80/100
12/12 [==============================] - 0s 4ms/step - loss: 0.7533
Epoch 81/100
12/12 [==============================] - 0s 4ms/step - loss: 0.7249
Epoch 82/100
12/12 [==============================] - 0s 5ms/step - loss: 0.7045
Epoch 83/100
12/12 [==============================] - 0s 4ms/step - loss: 0.6754
Epoch 84/100
12/12 [==============================] - 0s 3ms/step - loss: 0.6526
Epoch 85/100
12/12 [==============================] - 0s 4ms/step - loss: 0.6291
Epoch 86/100
12/12 [==============================] - 0s 3ms/step - loss: 0.6133
Epoch 87/100
12/12 [==============================] - 0s 5ms/step - loss: 0.5795
```

```
Epoch 88/100
12/12 [==============================] - 0s 6ms/step - loss: 0.5744
Epoch 89/100
12/12 [==============================] - 0s 5ms/step - loss: 0.5750
Epoch 90/100
12/12 [==============================] - 0s 4ms/step - loss: 0.5401
Epoch 91/100
12/12 [==============================] - 0s 4ms/step - loss: 0.4999
Epoch 92/100
12/12 [==============================] - 0s 6ms/step - loss: 0.4837
Epoch 93/100
12/12 [==============================] - 0s 4ms/step - loss: 0.4590
Epoch 94/100
12/12 [==============================] - 0s 5ms/step - loss: 0.4343
Epoch 95/100
12/12 [==============================] - 0s 4ms/step - loss: 0.4204
Epoch 96/100
12/12 [==============================] - 0s 3ms/step - loss: 0.4015
Epoch 97/100
12/12 [==============================] - 0s 3ms/step - loss: 0.3949
Epoch 98/100
12/12 [==============================] - 0s 3ms/step - loss: 0.3718
Epoch 99/100
12/12 [==============================] - 0s 4ms/step - loss: 0.3494
Epoch 100/100
12/12 [==============================] - 0s 4ms/step - loss: 0.3428

<keras.src.callbacks.History at 0x25157582350>
```

```
# Evaluate the model on the test set
loss = model2.evaluate(X_test, y_test)
print(f"Mean Squared Error on Test Data: {loss:.2f}")
```

```
4/4 [==============================] - 0s 4ms/step - loss: 0.5167
Mean Squared Error on Test Data: 0.52
```

```
#Mean Squared Error on Test Data: 0.52  means it is the good model for the
testing the values
from tensorflow import keras

X_unseen =
np.array([[0.4,4,245.65,0.5,581,5.856,97,94.44,2,188,370.31,250.41,130.3,34]]
)

# Make predictions on the preprocessed unseen data
predictions = model2.predict(X_unseen)
```

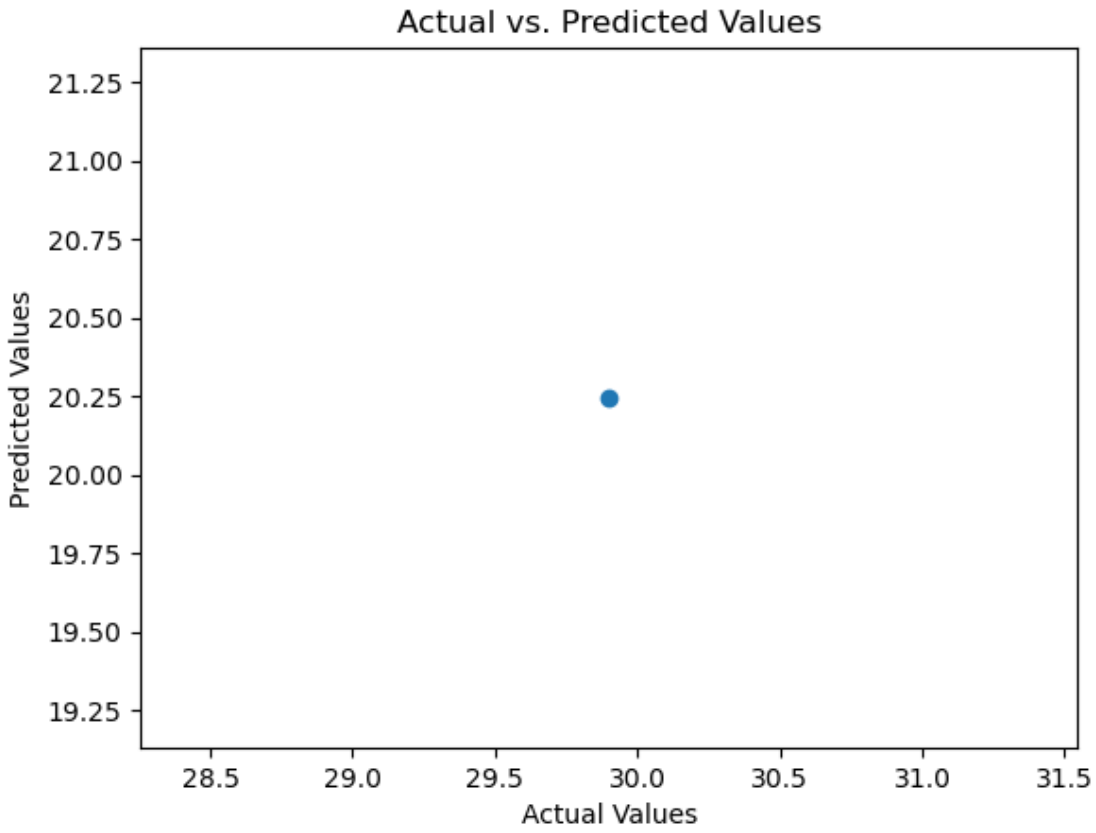
```
1/1 [==============================] - 0s 154ms/step

y_unseen=[29.9]
```

```
from sklearn.metrics import mean_squared_error
# Step 4: Evaluate Model Performance
mse = mean_squared_error(y_unseen, predictions)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 93.20022598220396
```

```
# Step 5: Visualize or Analyze Results (Optional)
plt.scatter(y_unseen, predictions)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.show()
```




```
#above figure information is showing that there is near relation between the predicted and actual value
```

Classification (Categorizing House Prices)

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
# Define price range categories based on thresholds
price_bins = [0, 20, 30, 40, 50, np.inf]
price_labels = ["Very Low", "Low", "Medium", "High", "Very High"]
data['Price Range'] = pd.cut(target, bins=price_bins, labels=price_labels)

# Encode the price range labels to numeric values
label_encoder = LabelEncoder()
data['Price Range'] = label_encoder.fit_transform(data['Price Range'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.drop('Price Range',
axis=1), data['Price Range'], test_size=0.2, random_state=42)

# Initialize classifiers
logistic_classifier = LogisticRegression()
tree_classifier = DecisionTreeClassifier()
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train classifiers
logistic_classifier.fit(X_train, y_train)
tree_classifier.fit(X_train, y_train)
rf_classifier.fit(X_train, y_train)

RandomForestClassifier(random_state=42)

# Make predictions
logistic_preds = logistic_classifier.predict(X_test)
tree_preds = tree_classifier.predict(X_test)
rf_preds = rf_classifier.predict(X_test)

# Evaluate classifiers
print("Logistic Regression Accuracy:", accuracy_score(y_test,
logistic_preds))
print("Decision Tree Classifier Accuracy:", accuracy_score(y_test,
tree_preds))
print("Random Forest Classifier Accuracy:", accuracy_score(y_test, rf_preds))

Logistic Regression Accuracy: 0.8823529411764706
Decision Tree Classifier Accuracy: 1.0
Random Forest Classifier Accuracy: 1.0

# Create a Linear Regression model
model2 = LinearRegression()

# Train the model on the training data
model2.fit(X_train, y_train)

LinearRegression()

# Make predictions on the test set
predictions = model2.predict(X_test)
```

```
# Evaluate the model using a suitable metric (e.g., Mean Squared Error)
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 0.4866917066973078
```

predicting for unseen data

```
unseen_data =
np.array([[0.14455,12.5,7.87,0,0.524,65.172,6790.1,5.9505,5,31,15.2,9.9,17.1,
3]])  # Replace with the feature values for the unseen house
scaled_unseen_data = scaler.transform(unseen_data)  # If you used feature
scaling during training
predicted_price = model2.predict(scaled_unseen_data)
print(f"Predicted House Price: {predicted_price}")
```

```
Predicted House Price: [10.36723235]
```

above parameter are showing the for the expected values

```
unseen_data2 =
np.array([[0.15038,0,25.65,0,0.581,54.856,97,1.9444,2,188,19.1,370.31,17.3,34
]])  # Replace with the feature values for the unseen house
scaled_unseen_data = scaler.transform(unseen_data2)  # If you used feature
scaling during training
predicted_price = model2.predict(scaled_unseen_data)
print(f"Predicted House Price: {predicted_price}")
```

```
Predicted House Price: [7.47723173]
```

RNN MODEL

FAKE NEWS DETECTION

```
#IMPORTIN REQUIRED LIBRERIES
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import re
import string

#Read the data using pandas
datafake=pd.read_csv("fake.csv")
```

```
datafake
```

```
                                                   title  \
0       Donald Trump Sends Out Embarrassing New Year’...
1       Drunk Bragging Trump Staffer Started Russian ...
2       Sheriff David Clarke Becomes An Internet Joke...
```

```
3       Trump Is So Obsessed He Even Has Obama’s Name...
4       Pope Francis Just Called Out Donald Trump Dur...
...                                                   ...
23476  McPain: John McCain Furious That Iran Treated ...
23477  JUSTICE? Yahoo Settles E-mail Privacy Class-ac...
23478  Sunnistan: US and Allied ‘Safe Zone’ Plan to T...
23479  How to Blow $700 Million: Al Jazeera America F...
23480  10 U.S. Navy Sailors Held by Iranian Military ...

                                                    text      subject  \
0      Donald Trump just couldn t wish all Americans ...         News
1      House Intelligence Committee Chairman Devin Nu...         News
2      On Friday, it was revealed that former Milwauk...         News
3      On Christmas day, Donald Trump announced that ...         News
4      Pope Francis used his annual Christmas Day mes...         News
...                                                  ...          ...
23476  21st Century Wire says As 21WIRE reported earl...  Middle-east
23477  21st Century Wire says It s a familiar theme. ...  Middle-east
23478  Patrick Henningsen  21st Century WireRemember ...  Middle-east
23479  21st Century Wire says Al Jazeera America will...  Middle-east
23480  21st Century Wire says As 21WIRE predicted in ...  Middle-east

                   date
0      December 31, 2017
1      December 31, 2017
2      December 30, 2017
3      December 29, 2017
4      December 25, 2017
...                 ...
23476   January 16, 2016
23477   January 16, 2016
23478   January 15, 2016
23479   January 14, 2016
23480   January 12, 2016

[23481 rows x 4 columns]
```

```
datafake.isnull().sum()
```

```
title      0
text       0
subject    0
date       0
dtype: int64
```

```
datafake.shape
```

```
(23481, 4)
```

```
#insert a column "class"
datafake['class']=0
```

```
datafake.shape
```

```
(23481, 5)
```

```
datafake.head()
```

```
                                                title  \
0   Donald Trump Sends Out Embarrassing New Year’...
1   Drunk Bragging Trump Staffer Started Russian ...
2   Sheriff David Clarke Becomes An Internet Joke...
3   Trump Is So Obsessed He Even Has Obama’s Name...
4   Pope Francis Just Called Out Donald Trump Dur...

                                                text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                date  class
0  December 31, 2017      0
1  December 31, 2017      0
2  December 30, 2017      0
3  December 29, 2017      0
4  December 25, 2017      0
```

```
datatrue=pd.read_csv('True.csv')
```

```
datatrue['class']=1
```

```
datatrue.head()
```

```
                                                title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                 date  class
0  December 31, 2017       1
1  December 29, 2017       1
2  December 31, 2017       1
```

```
3  December 30, 2017       1
4  December 29, 2017       1
```

```
datatrue.shape
```

```
(21417, 5)
```

```
datatrue.shape,datafake.shape
```

```
((21417, 5), (23481, 5))
```

```
# merging the true ,fake files
fakenewsdetection=pd.concat([datatrue,datafake], ignore_index=True,
sort=False)
df=fakenewsdetection
df.head()
```

```
                                                title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                 date  class
0  December 31, 2017       1
1  December 29, 2017       1
2  December 31, 2017       1
3  December 30, 2017       1
4  December 29, 2017       1
```

```
df.tail(5)
```

```
                                                   title  \
44893  McPain: John McCain Furious That Iran Treated ...
44894  JUSTICE? Yahoo Settles E-mail Privacy Class-ac...
44895  Sunnistan: US and Allied ‘Safe Zone’ Plan to T...
44896  How to Blow $700 Million: Al Jazeera America F...
44897  10 U.S. Navy Sailors Held by Iranian Military ...

                                                   text      subject  \
44893  21st Century Wire says As 21WIRE reported earl...  Middle-east
44894  21st Century Wire says It s a familiar theme. ...  Middle-east
44895  Patrick Henningsen  21st Century WireRemember ...  Middle-east
```

```
44896  21st Century Wire says Al Jazeera America will...  Middle-east
44897  21st Century Wire says As 21WIRE predicted in ...  Middle-east

                  date  class
44893  January 16, 2016      0
44894  January 16, 2016      0
44895  January 15, 2016      0
44896  January 14, 2016      0
44897  January 12, 2016      0
```

```
#random mixing of two samples
df = df.sample(frac = 1)
```

```
df
```

```
                                                    title  \
37362  10-Year Old Writes to Trump Asking to Mow The ...
25588   Trump’s #1 Deplorable Has A Question For You ...
2120   Factbox: Trump on Twitter (Aug 17) - Stonewall...
20361  Japan PM Abe's ratings regain 50 percent amid ...
38011  BREAKING! INVESTIGATION: Hillary Clinton Did N...
...                                                  ...
42282  HARVARD BULLIED INTO Dropping 80 Year Old “Rac...
17137  Greek police cut down to size: EU court rules ...
6275   Senate Democrats ask Trump attorney general pi...
43735  Obama Joins Comedy Central Host to Push ‘Laugh...
30409   Oregon Right-Wing Terrorist Makes CHILLING Co...

                                                    text          subject  \
37362  The National Park Service might be out of a jo...  Government News
25588  Jim Stachowiak is a staunch Trump supporter wh...             News
2120   The following statements were posted to the ve...    politicsNews
20361  TOKYO (Reuters) - Japanese Prime Minister Shin...       worldnews
38011   Was Hillary Clinton negligent or was she doin...  Government News
...                                                  ...              ...
42282  Harvard is agreeing to erase the history of th...       left-news
17137  LUXEMBOURG (Reuters) - European police forces ...       worldnews
6275   WASHINGTON (Reuters) - Nine Democratic senator...    politicsNews
43735  21st Century Wire says President Obama is once...         US_News
30409  If anyone questions whether or not the Bundy-l...            News

                      date  class
37362         Sep 15, 2017      0
25588     October 16, 2016      0
2120      August 17, 2017       1
20361  September 12, 2017       1
38011         May 25, 2016      0
...                    ...    ...
42282         Mar 16, 2016      0
17137    October 18, 2017       1
```

```
6275     January 17, 2017       1
43735    December 15, 2016      0
30409      January 3, 2016      0

[44898 rows x 5 columns]
```

```
df.head()
```

```
                                                   title  \
37362  10-Year Old Writes to Trump Asking to Mow The ...
25588   Trump’s #1 Deplorable Has A Question For You ...
2120   Factbox: Trump on Twitter (Aug 17) - Stonewall...
20361  Japan PM Abe's ratings regain 50 percent amid ...
38011  BREAKING! INVESTIGATION: Hillary Clinton Did N...

                                                    text          subject  \
37362  The National Park Service might be out of a jo...  Government News
25588  Jim Stachowiak is a staunch Trump supporter wh...             News
2120   The following statements were posted to the ve...    politicsNews
20361  TOKYO (Reuters) - Japanese Prime Minister Shin...       worldnews
38011   Was Hillary Clinton negligent or was she doin...  Government News

                      date  class
37362         Sep 15, 2017      0
25588     October 16, 2016      0
2120      August 17, 2017       1
20361  September 12, 2017       1
38011         May 25, 2016      0
```

```
#drop the un necessary columns
df1=df.drop(['title','subject','date'],axis=1)
```

```
df1.reset_index(inplace = True)
df1.drop(["index"], axis = 1, inplace = True)
```

```
df1.head()
```

```
                                                text  class
0  The National Park Service might be out of a jo...      0
1  Jim Stachowiak is a staunch Trump supporter wh...      0
2  The following statements were posted to the ve...      1
3  TOKYO (Reuters) - Japanese Prime Minister Shin...      1
4   Was Hillary Clinton negligent or was she doin...      0
```

Creating a function to process the texts

```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W"," ",text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
```

```
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

df["text"] = df["text"].apply(wordopt)
```

Defining dependent and independent variables

```
x = df["text"]
y = df["class"]
```

Splitting Training and Testing

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

Convert text to vectors

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

logistic regression

```
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression()
LR.fit(xv_train,y_train)

LogisticRegression()

pred_lr=LR.predict(xv_test)

LR.score(xv_test, y_test)

0.987260579064588

print(classification_report(y_test, pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5884
           1       0.99      0.99      0.99      5341

    accuracy                           0.99     11225
   macro avg       0.99      0.99      0.99     11225
weighted avg       0.99      0.99      0.99     11225
```

multinomial classification Navie Basian

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
```

```
NB=MultinomialNB()
NB.fit(xv_train,y_train)
```

```
MultinomialNB()
```

```
pred_lrnb=NB.predict(xv_test)
```

```
NB.score(xv_test, y_test)
```

```
0.9357683741648107
```

```
print(classification_report(y_test, pred_lrnb))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      5884
           1       0.94      0.92      0.93      5341

    accuracy                           0.94     11225
   macro avg       0.94      0.94      0.94     11225
weighted avg       0.94      0.94      0.94     11225
```

Decision Tree Classification

```
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

```
DecisionTreeClassifier()
```

```
pred_dt = DT.predict(xv_test)
```

```
DT.score(xv_test, y_test)
```

```
0.9965256124721603
```

```
print(classification_report(y_test, pred_dt))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5884
           1       1.00      1.00      1.00      5341

    accuracy                           1.00     11225
   macro avg       1.00      1.00      1.00     11225
weighted avg       1.00      1.00      1.00     11225
```

Model Testing

```
def output_lable(n):
    if n == 0:
```

```
        return "Fake News"
    elif n == 1:
        return "Not A Fake News"

def manual_testing(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_DT = DT.predict(new_xv_test)
    pred_NB = NB.predict(new_xv_test)
    return print("\n\nLR Prediction: {} \nDT Prediction: {} \NB Prediction:
{} .
                 format(output_lable(pred_LR[0]),
                   output_lable(pred_DT[0]),
                    output_lable(pred_NB[0]), ))


news = str(input())
manual_testing(news)

news = str(input())
manual_testing(news)
```