FATIMA JINNAH WOMEN UNIVERSITY

SUBMITTED TO:

DR. SIDRA EJAZ

SUBMITTED BY:

BISMA ALI (2021-BSE-008)
RAHEELA NOSHEEN (2021-BSE-027)
TANZEELA ASGHAR (2021-BSE-032)

COURSE TITLE:

MACHINE LEARNING

DATED:

4ST JUNE,2024.

WEATHER FORECASTING

INTRODUCTION

This dataset contains information on various weather parameters recorded over a series of days. These parameters include the type of precipitation (such as rain), temperature in degrees Celsius, apparent temperature, humidity level, wind speed in kilometers per hour, wind bearing in degrees, visibility in kilometers, cloud cover, pressure in millibars, and a brief summary of the daily weather conditions.

The dataset appears to offer a detailed overview of the weather conditions for a specific location over a certain timeframe. It likely presents valuable data for meteorological analysis, forecasting, and climate research. The inclusion of metrics like humidity, wind speed, and pressure provides a comprehensive understanding of the atmospheric conditions during the recorded periods. The dataset's structured format enables easy comparisons between different days and the identification of potential trends or patterns in the weather data.

STEPS:

Connect the google colab.

Upload the dataset weatherHistory.csv from downloads into the colab.

Import necessary libraries:

- **pandas as pd**: Pandas is a popular Python library used for data manipulation and analysis. It provides data structures such as Series (1-dimensional labeled array) and DataFrame (2-dimensional labeled data structure with columns of potentially different types). The as pd syntax allows you to use the alias pd instead of typing pandas every time

- **numpy as np**: NumPy is a library for working with arrays and mathematical operations in Python. It provides support for large, multi-dimensional arrays and matrices, and is the foundation of most scientific computing in Python
- **sklearn.model_selection.train_test_split**: This function from Scikit-learn (a machine learning library) is used to split a dataset into training and testing sets. This is a crucial step in machine learning, as it allows you to train your model on a portion of the data and evaluate its performance on the remaining portion.
- **sklearn.linear_model.LogisticRegression**: This class from Scikit-learn implements logistic regression, a type of supervised learning algorithm that predicts the probability of a binary outcome (e.g., 0 or 1, yes or no, etc.) based on a set of input features.
- **sklearn.metrics.accuracy_score**: This function from Scikit-learn calculates the accuracy score of a classification model, which is the proportion of correctly predicted instances among all instances.

Importing the Dependencies

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Load the dataset.

Data Collection and Processing

```
[2]  # loading the csv data to a panda DataFrame
     wf_data = pd.read_csv('/content/weather.csv')
```

```
# print first five rows of the dataset
wf_data.head()
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

```
# print last five rows of the dataset
wf_data.tail()
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

The dataset has 96453 rows and 12 columns.

```
[5]  # no of columns and rows in dataset
     wf_data.shape

(96453, 12)
```

- Handling missing values (e.g., imputation, interpolation)

```
[6] #checking for missing values
    wf_data.isnull().sum()
```

```
Formatted Date                  0
Summary                         0
Precip Type                   517
Temperature (C)                 0
Apparent Temperature (C)        0
Humidity                        0
Wind Speed (km/h)               0
Wind Bearing (degrees)          0
Visibility (km)                 0
Loud Cover                      0
Pressure (millibars)            0
Daily Summary                   0
dtype: int64
```

wf.data:

- Historical weather data
- Multiple columns (date, precipitation, temperature, humidity, wind speed, etc.)
- CSV format
- Thousands to hundreds of thousands of rows

```
#Statistical measures about the data
wf_data.describe()
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000


```
#Statistical measures about the data
wf_data.describe()
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.0	96453.000000
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	0.0	1003.235956
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	0.0	116.969906
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	0.0	1011.900000
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	0.0	1016.450000
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	0.0	1021.090000
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	0.0	1046.380000

```
#check for missing values and data types
missing_values= wf_data.isnull().sum()
data_types= wf_data.dtypes

#Display the information
print ('Missing Values:')
print (missing_values)
print ('\nData Types:')
print (data_types)
```

```
Missing Values:
Formatted Date              0
Summary                     0
Precip Type               517
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
Daily Summary               0
dtype: int64

Data Types:
Formatted Date               object
Summary                      object
Precip Type                  object
Temperature (C)             float64
Apparent Temperature (C)    float64
Humidity                    float64
Wind Speed (km/h)           float64
Wind Bearing (degrees)      float64
Visibility (km)             float64
Loud Cover                  float64
Pressure (millibars)        float64
Daily Summary                object
dtype: object
```

```
[11] wf_data.dropna(inplace=True)
```

wf_data.dropna(inplace=True): Drops rows with missing values (NaN) from the wf_data dataset, and modifies the original dataset in-place.

```
[12] #check for missing values and data types
     missing_values= wf_data.isnull().sum()
     data_types= wf_data.dtypes

     #Display the information
     print ('Missing Values:')
     print (missing_values)
     print ('\nData Types:')
     print (data_types)
```

```
Missing Values:
Formatted Date              0
Summary                     0
Precip Type                 0
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
Daily Summary               0
dtype: int64

Data Types:
Formatted Date               object
Summary                      object
Precip Type                  object
Temperature (C)             float64
Apparent Temperature (C)    float64
Humidity                    float64
Wind Speed (km/h)           float64
Wind Bearing (degrees)      float64
Visibility (km)             float64
Loud Cover                  float64
Pressure (millibars)        float64
Daily Summary                object
dtype: object
```

Apply Scaling

Scales the data to have zero mean and unit variance

```
# Assuming your DataFrame is named 'wf_data' and 'DailySummary' is the column to be encoded
label_encoder = LabelEncoder()
wf_data['Daily Summary'] = label_encoder.fit_transform(wf_data['Daily Summary'])

# Display the transformed DataFrame
print(wf_data.head())


##Scaling/Standardizing Numerical Features
```

```
#wf_data[['numerical_feature1', 'numerical_feature2']] =scaler.fit_transform(df[['numerical_feature1', 'numerical_feature2']])
#numerical features are selected excluding date, Summary, Precip Type, Daily Summary.
numerical_features = ['Temperature (C)', 'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Wind Bearing (degrees)','Visibility (km)', 'Loud Cover', 'Pressure (millibars)']

# Applying StandardScaler to scale the selected numerical features
wf_data[numerical_features] = StandardScaler().fit_transform(wf_data[numerical_features])
```

```
                  Formatted Date        Summary Precip Type  Temperature (C)  \
0  2006-04-01 00:00:00.000 +0200  Partly Cloudy        rain         9.472222
1  2006-04-01 01:00:00.000 +0200  Partly Cloudy        rain         9.355556
2  2006-04-01 02:00:00.000 +0200  Mostly Cloudy        rain         9.377778
3  2006-04-01 03:00:00.000 +0200  Partly Cloudy        rain         8.288889
4  2006-04-01 04:00:00.000 +0200  Mostly Cloudy        rain         8.755556

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                  7.388889      0.89            14.1197
1                  7.227778      0.86            14.2646
2                  9.377778      0.89             3.9284
3                  5.944444      0.83            14.1036
4                  6.977778      0.83            11.0446

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                   251.0          15.8263         0.0               1015.13
1                   259.0          15.8263         0.0               1015.63
2                   204.0          14.9569         0.0               1015.94
3                   269.0          15.8263         0.0               1016.41
4                   259.0          15.8263         0.0               1016.51

   Daily Summary
0            197
1            197
2            197
3            197
4            197
```

Apply label encoding

- from sklearn.preprocessing import LabelEncoder
- le = LabelEncoder()
- le.fit(df['column_name']): Maps categorical values to numerical values (0, 1, 2, ...).
- df['column_name'] = le.transform(df['column_name']): Encodes the categorical values in the dataset.

```
# Label encoding
# Define a list of categorical columns to be encoded
categorical_columns = [
  'Summary', 'Precip Type', 'Daily Summary' ]
# Initialize LabelEncoder
label_encoder = LabelEncoder()
# Apply label encoding to each categorical column
for col in categorical_columns:
    wf_data[col] = label_encoder.fit_transform(wf_data[col])
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Assuming wf_data is already loaded as a DataFrame
# Example: Loading a DataFrame from a CSV file
# wf_data = pd.read_csv('path/to/your/data.csv')

# Define a list of categorical columns to be encoded
categorical_columns = ['Summary', 'Precip Type', 'Daily Summary']

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column if it exists in the DataFrame
for col in categorical_columns:
    if col in wf_data.columns:
        wf_data[col] = label_encoder.fit_transform(wf_data[col])
    else:
        print(f"Column '{col}' not found in DataFrame")

# Verify the DataFrame after encoding
print(wf_data.head())

# Selecting independent features and target/dependent feature
X = wf_data[['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)', 'Apparent Temperature (C)',
'Humidity', 'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover', 'Pressure (millibars)']]
y = wf_data['Daily Summary']

# Print shapes to debug
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)

# Ensure 'X' and 'y' have the same number of samples
```

```
if X.shape[0] != y.shape[0]:
    print("Error: The number of samples in X and y are not consistent.")
else:
    # Splitting the data into training (70%) and testing (30%) sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    print("Training and testing sets created successfully.")
```

```
[153]
0s
                   Formatted Date  Summary  Precip Type  Temperature (C)  \
0  2006-04-01 00:00:00.000 +0200       19            0        -0.257951
1  2006-04-01 01:00:00.000 +0200       19            0        -0.270141
2  2006-04-01 02:00:00.000 +0200       17            0        -0.267819
3  2006-04-01 03:00:00.000 +0200       19            0        -0.381594
4  2006-04-01 04:00:00.000 +0200       17            0        -0.332833

   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                 -0.324102  0.792748           0.478964
1                 -0.339134  0.639470           0.499902
2                 -0.138532  0.792748          -0.993620
3                 -0.458873  0.486192           0.476638
4                 -0.362460  0.486192           0.034630


   Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
0                 -0.324102  0.792748           0.478964
1                 -0.339134  0.639470           0.499902
2                 -0.138532  0.792748          -0.993620
3                 -0.458873  0.486192           0.476638
4                 -0.362460  0.486192           0.034630

   Wind Bearing (degrees)  Visibility (km)  Loud Cover  Pressure (millibars)  \
0                0.591157         1.309107         0.0              0.102152
1                0.665655         1.309107         0.0              0.106415
2                0.153478         1.100806         0.0              0.109058
3                0.758778         1.309107         0.0              0.113066
4                0.665655         1.309107         0.0              0.113919

   Daily Summary
0            197
1            197
2            197
3            197
4            197
Shape of X: (95936, 11)
Shape of y: (95936,)
Training and testing sets created successfully.
```

ALGORITHMS TRAINING

- Now we will train forward and backward propagation on the dataset.

- For that we will read the csv file into df.
- Then we will display some basic information about the dataset.
- We again check for any missing values.
- Then we will plot a temperature histogram.
- Then we performed data cleaning and dropped the rows with missing values.
- After that we will convert catagorial column into numerical.
- Then we selected some features and target variable and standardize them.
- Next we have divided it into testing and training sets.

CODE:

```
# Load the data
import pandas as pd

data_path = '/content/weather.csv'
df = pd.read_csv(data_path)
print(df.head())

# Display basic information
print(df.info())
print(df.describe())

# Checking for missing values
print(df.isnull().sum())

# Plot a histogram of temperature
import matplotlib.pyplot as plt

df['Temperature (C)'].hist(bins=50)
plt.title('Temperature Distribution')
plt.xlabel('Temperature (C)')
plt.ylabel('Frequency')
plt.show()

# Data cleaning and transformation
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Droping the rows with missing values
df = df.dropna()

# Converting now categorical column to numerical
le = LabelEncoder()
df['Daily Summary'] = le.fit_transform(df['Daily Summary'])

# Selecting features and target
features = ['Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Pressure
(millibars)', 'Daily Summary']
```

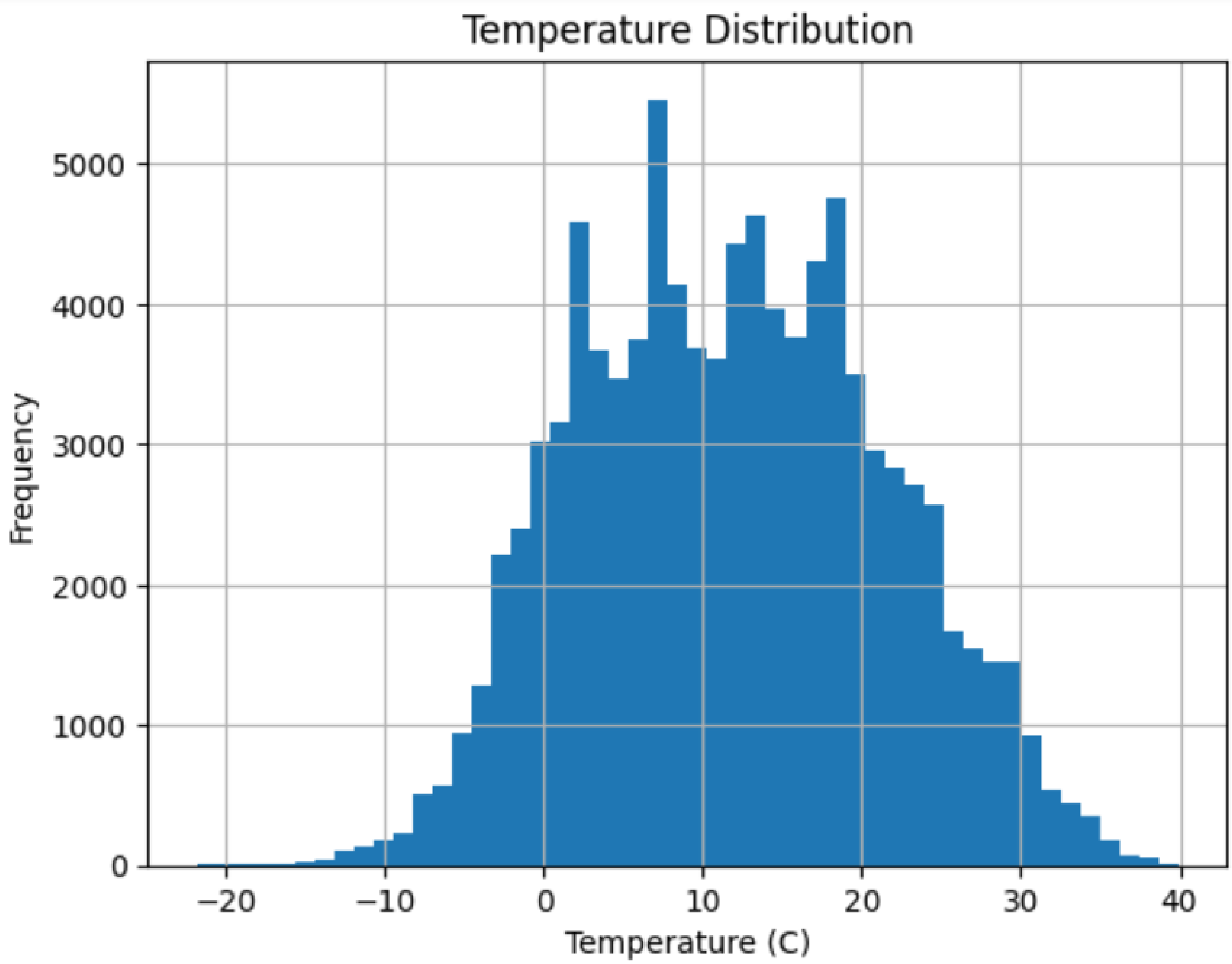
```
target = 'Apparent Temperature (C)'

X = df[features].astype('float32')
y = df[target].astype('float32')

# Standardize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Train-Test Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

# Use a smaller subset of data to fit in memory
X_train = X_train[:5000]
y_train = y_train[:5000]
X_test = X_test[:1000]
y_test = y_test[:1000]
```

OUTPUT:

```
         Summary Precip Type  Temperature (C)  Apparent Temperature (C)  \
0  Partly Cloudy        rain         9.472222                  7.388889
1  Partly Cloudy        rain         9.355556                  7.227778
2  Mostly Cloudy        rain         9.377778                  9.377778
3  Partly Cloudy        rain         8.288889                  5.944444
4  Mostly Cloudy        rain         8.755556                  6.977778

   Humidity  Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  \
0      0.89            14.1197                     251          15.8263
1      0.86            14.2646                     259          15.8263
2      0.89             3.9284                     204          14.9569
3      0.83            14.1036                     269          15.8263
4      0.83            11.0446                     259          15.8263

   Loud Cover  Pressure (millibars)                      Daily Summary
0           0               1015.13  Partly cloudy throughout the day.
1           0               1015.63  Partly cloudy throughout the day.
2           0               1015.94  Partly cloudy throughout the day.
3           0               1016.41  Partly cloudy throughout the day.
4           0               1016.51  Partly cloudy throughout the day.
```

```
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Summary                   96453 non-null  object
```

```
 1   Precip Type               95936 non-null  object
 2   Temperature (C)           96453 non-null  float64
 3   Apparent Temperature (C)  96453 non-null  float64
 4   Humidity                  96453 non-null  float64
 5   Wind Speed (km/h)         96453 non-null  float64
 6   Wind Bearing (degrees)    96453 non-null  int64
 7   Visibility (km)           96453 non-null  float64
 8   Loud Cover                96453 non-null  int64
 9   Pressure (millibars)      96453 non-null  float64
 10  Daily Summary             96453 non-null  object
dtypes: float64(6), int64(2), object(3)
memory usage: 8.1+ MB
None
       Temperature (C)  Apparent Temperature (C)      Humidity  \
count     96453.000000              96453.000000  96453.000000
mean         11.932678                 10.855029      0.734899
std           9.551546                 10.696847      0.195473
min         -21.822222                -27.716667      0.000000
25%           4.688889                  2.311111      0.600000
50%          12.000000                 12.000000      0.780000
75%          18.838889                 18.838889      0.890000
max          39.905556                 39.344444      1.000000

       Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
count       96453.000000            96453.000000     96453.000000     96453.0
mean           10.810640              187.509232        10.347325         0.0
std             6.913571              107.383428         4.192123         0.0
min             0.000000                0.000000         0.000000         0.0
25%             5.828200              116.000000         8.339800         0.0
50%             9.965900              180.000000        10.046400         0.0
75%            14.135800              290.000000        14.812000         0.0
max            63.852600              359.000000        16.100000         0.0

       Pressure (millibars)
count          96453.000000
mean            1003.235956
std              116.969906
min                0.000000
25%             1011.900000
50%             1016.450000
75%             1021.090000
max             1046.380000
Summary                       0
Precip Type                 517
Temperature (C)               0
Apparent Temperature (C)      0
Humidity                      0
Wind Speed (km/h)             0
Wind Bearing (degrees)        0
Visibility (km)               0
Loud Cover                    0
Pressure (millibars)          0
Daily Summary                 0
   dtype: int64
```



FORWARD AND BACKWARD PROPAGATION CODE:

Then we did manual ANN Implementation with Batch Gradient Descent.

```
# Manual ANN Implementation with Batch Gradient Descent
import numpy as np

# Activation functions and their derivatives
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

# Loss function and its derivative
def mse_loss(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

def mse_loss_derivative(y_true, y_pred):
    return 2 * (y_pred - y_true) / y_true.size
```

```
# ANN class
Now we have trained forward and backward propagation on the dataset.
class ANN:
    def __init__(self, input_size, hidden_size, output_size):
        # Initialize weights and biases
        np
        self.weights_input_hidden = np.random.rand(input_size,
hidden_size).astype('float32')
        self.bias_hidden = np.zeros((1, hidden_size)).astype('float32')
        self.weights_hidden_output = np.random.rand(hidden_size,
output_size).astype('float32')
        self.bias_output = np.zeros((1, output_size)).astype('float32')

    def forward(self, X):
        # Forward propagation
        self.hidden_input = np.dot(X, self.weights_input_hidden) +
self.bias_hidden
        self.hidden_output = sigmoid(self.hidden_input)
        self.output_input = np.dot(self.hidden_output, self.weights_hidden_output)
+ self.bias_output
        self.output = sigmoid(self.output_input)
        return self.output

    def backward(self, X, y, output):
        # Backward propagation
        output_error = mse_loss_derivative(y, output)
        output_delta = output_error * sigmoid_derivative(output)

        hidden_error = output_delta.dot(self.weights_hidden_output.T)
        hidden_delta = hidden_error * sigmoid_derivative(self.hidden_output)

        # Update weights and biases
        self.weights_hidden_output -= self.hidden_output.T.dot(output_delta)
        self.bias_output -= np.sum(output_delta, axis=0, keepdims=True)
        self.weights_input_hidden -= X.T.dot(hidden_delta)
        self.bias_hidden -= np.sum(hidden_delta, axis=0, keepdims=True)

    def train(self, X, y, epochs, batch_size, learning_rate):
        for epoch in range(epochs):
            for i in range(0, X.shape[0], batch_size):
                X_batch = X[i:i+batch_size]
                y_batch = y[i:i+batch_size].values.reshape(-1, 1)
                output = self.forward(X_batch)
                self.backward(X_batch, y_batch, output)
            if epoch % 10 == 0:
                loss = mse_loss(y_batch, output)
                print(f'Epoch {epoch}, Loss: {loss}')

# Initialize the ANN
```

```
input_size = X_train.shape[1]
hidden_size = 10
output_size = 1
ann = ANN(input_size=input_size, hidden_size=hidden_size, output_size=output_size)

# Train the ANN
ann.train(X_train, y_train, epochs=100, batch_size=32, learning_rate=0.01)

# Test the ANN
output = ann.forward(X_test)
print("Predicted output:")
print(output)
```

OUTPUT:

Epoch 0, Loss: 216.7237091064453
Epoch 10, Loss: 216.59634399414062
Epoch 20, Loss: 216.58184814453125
Epoch 30, Loss: 216.57647705078125
Epoch 40, Loss: 216.57386779785156
Epoch 50, Loss: 216.57266235351562
Epoch 60, Loss: 216.5713653564453
Epoch 70, Loss: 216.57131958007812
Epoch 80, Loss: 216.57052612304688
Epoch 90, Loss: 216.57066345214844
Predicted output:
[[2.14898591e-06]
 [1.00000000e+00]
 [1.00000000e+00]
 [9.77875113e-01]
 [2.03529745e-03]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [8.80707502e-02]
 [3.90460912e-07]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [7.43861847e-06]
 [1.00000000e+00]
 [3.17625701e-01]
 [1.00000000e+00]
 [4.50584622e-07]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [1.00000000e+00]
 [9.42766474e-06]
 [1.00000000e+00]
 [1.00000000e+00]
 [2.08828951e-07]
 [1.00000000e+00]

[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[4.89092410e-01]
[1.00000000e+00]
[8.83526027e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.32789107e-02]
[1.00000000e+00]
[9.99979734e-01]
[9.99899745e-01]
[1.00000000e+00]
[3.93560100e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999523e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.98476005e-06]
[1.00000000e+00]
[1.00000000e+00]
[8.84316087e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.66634509e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.89837551e-05]
[6.11924799e-03]
[1.00000000e+00]
[1.00000000e+00]
[9.99870181e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[9.99992609e-01]
[8.42180699e-02]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.56874450e-02]
[6.67278755e-06]
[1.00000000e+00]
[9.99994636e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.43693702e-04]
[1.00000000e+00]
[1.74694322e-03]
[1.00000000e+00]
[2.16123895e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.44701153e-06]
[1.00000000e+00]
[1.47494461e-07]
[1.00000000e+00]
[1.00000000e+00]
[8.75556886e-01]
[1.00000000e+00]
[9.99997139e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.32339130e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99666929e-01]
[1.00000000e+00]
[1.00000000e+00]
[2.03760067e-07]
[1.00000000e+00]
[3.63235642e-07]
[1.01036233e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.26692719e-04]
[1.00000000e+00]
[1.98874739e-04]
[9.99348104e-01]
[1.00000000e+00]
[8.55085790e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

```
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.03048084e-02]
[1.00000000e+00]
[1.12437199e-04]
[1.00000000e+00]
[7.11543635e-02]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.45344472e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.26439243e-05]
[9.51384008e-01]
[5.78850063e-07]
[1.00000000e+00]
[1.49760169e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99996066e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[9.98452544e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
```

[1.00000000e+00]
[2.21905624e-03]
[1.00000000e+00]
[1.00000000e+00]
[3.81172640e-07]
[5.61143588e-06]
[4.30156469e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99963403e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.45078462e-07]
[9.99998569e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.33897447e-07]
[1.00000000e+00]
[9.99983907e-01]
[1.00000000e+00]
[1.00000000e+00]
[9.99950051e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.75493333e-06]
[9.99999881e-01]
[1.00000000e+00]
[1.26063023e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.82301549e-07]
[4.66570555e-07]
[4.26104486e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[7.23009929e-02]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[1.00000000e+00]
[5.15394561e-07]
[1.00000000e+00]
[1.00000000e+00]
[9.99999642e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999642e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.23577674e-03]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.34768772e-01]
[2.04484593e-07]
[2.85024689e-05]
[1.00000000e+00]
[4.18117935e-07]
[8.65980255e-05]
[1.00000000e+00]
[9.99815881e-01]
[2.95653054e-03]
[1.00000000e+00]
[9.96168554e-01]
[1.00000000e+00]
[1.00000000e+00]
[5.53323787e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.29604246e-04]
[1.00000000e+00]
[1.00000000e+00]
[2.55390637e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[4.74839032e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[9.96844053e-01]
[9.99642849e-01]
[1.00000000e+00]
[9.99989390e-01]
[1.26731002e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.36926919e-01]
[1.00000000e+00]

[1.79054274e-03]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.56166971e-01]
[1.40338926e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.32484979e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.34379799e-02]
[1.00000000e+00]
[1.00000000e+00]
[1.47969260e-07]
[1.35528356e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.13424801e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99836922e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[6.08881976e-07]
[1.00000000e+00]
[1.00000000e+00]
[2.45763744e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[7.35696801e-07]
[1.00000000e+00]
[1.00000000e+00]
[9.99997616e-01]
[9.99982238e-01]
[1.00000000e+00]

[8.54603648e-01]
[1.61331582e-06]
[5.51764071e-01]
[1.00000000e+00]
[9.97778475e-01]
[4.71312960e-04]
[1.29518526e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[4.53232224e-06]
[1.00000000e+00]
[1.00000000e+00]
[9.99998808e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.19168145e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.47564708e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.73492935e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.74382699e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.72847806e-07]
[9.99946117e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.34286267e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[1.04038243e-03]
[8.84811997e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[6.43210951e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.25040984e-04]
[1.00000000e+00]
[1.38374321e-06]
[1.30471633e-07]
[2.26879791e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.50355417e-03]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99999166e-01]
[9.98225629e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.55027446e-04]
[2.05095517e-07]
[1.91760932e-07]
[2.68572185e-05]
[3.47924579e-05]
[9.99643803e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.98419523e-01]
[4.24271366e-06]
[2.19401301e-07]
[1.00000000e+00]
[1.00000000e+00]
[9.92017508e-01]
[1.00000000e+00]
[9.98929679e-01]
[1.00000000e+00]
[3.25524779e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.68318432e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.86426890e-01]
[1.00000000e+00]
[1.00000000e+00]
[9.98905659e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[7.69646761e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.32727171e-02]
[4.88484966e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99996424e-01]
[1.00000000e+00]
[2.17394427e-05]
[1.00000000e+00]
[1.00000000e+00]
[2.97382911e-07]
[1.00000000e+00]

[2.03608536e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.89482973e-03]
[9.31556406e-06]
[8.84464767e-04]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.20606301e-05]
[9.99996543e-01]
[9.84721623e-07]
[1.00000000e+00]
[2.67026663e-01]
[9.99949455e-01]
[1.00000000e+00]
[3.84074639e-07]
[9.06000821e-07]
[1.00000000e+00]
[2.54734507e-04]
[1.00000000e+00]
[1.00000000e+00]
[5.84828595e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.32731671e-07]
[1.00000000e+00]
[3.15625039e-05]
[1.00000000e+00]
[2.35825544e-04]
[1.75771291e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99969363e-01]
[1.00000000e+00]
[2.53481016e-06]
[1.00000000e+00]
[9.80994880e-01]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.09124778e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.03471497e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[9.88730133e-01]
[1.00000000e+00]
[6.60646498e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99941468e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[5.17618202e-04]
[1.00000000e+00]
[1.00000000e+00]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.25285276e-04]
[9.99999642e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.14300692e-05]
[1.07131859e-06]
[1.64184746e-06]
[1.00000000e+00]
[1.69683187e-06]
[1.00000000e+00]
[1.00000000e+00]
[9.99999762e-01]
[9.99999285e-01]
[9.98691976e-01]
[1.00000000e+00]
[9.99855042e-01]
[1.00000000e+00]
[1.09620996e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.24548696e-05]
[3.65550280e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.87898586e-05]
[1.00000000e+00]

[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.11649919e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[8.19348323e-04]
[8.60774253e-06]
[9.99997973e-01]
[1.00000000e+00]
[1.00000000e+00]
[3.42731969e-03]
[9.99999642e-01]
[9.99896646e-01]
[1.00000000e+00]
[1.00000000e+00]
[3.86906380e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.64384517e-07]
[6.47388399e-04]
[2.24534432e-07]
[4.74408618e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.68105159e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.83388960e-01]
[3.31686788e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.41550004e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.98577714e-01]
[9.84482408e-01]
[1.00000000e+00]
[1.00000000e+00]
[3.09364577e-06]
[1.00000000e+00]
[1.00000000e+00]
[1.03974377e-03]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]

[1.00000000e+00]
[9.99999762e-01]
[1.00000000e+00]
[4.77098752e-07]
[1.00000000e+00]
[7.54189631e-03]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[4.52230006e-01]
[9.99998450e-01]
[1.00000000e+00]
[1.00000000e+00]
[7.55620306e-07]
[9.99998808e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.22720211e-07]
[9.99818146e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.87871769e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[5.01478382e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.99150145e-02]
[1.00000000e+00]
[9.99716222e-01]
[1.00000000e+00]
[1.00000000e+00]

[1.00000000e+00]
[9.98099148e-01]
[1.90046293e-07]
[1.00000000e+00]
[4.16986868e-02]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.03828861e-07]
[9.99393582e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.34542356e-02]
[1.00000000e+00]
[1.00000000e+00]
[4.15457208e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.85212609e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[5.30913984e-03]
[1.00000000e+00]
[1.00000000e+00]
[4.05264274e-07]
[1.00000000e+00]
[7.23140402e-05]
[8.69197777e-07]
[1.00000000e+00]
[1.00000000e+00]
[9.99999762e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.79375696e-03]
[1.71397062e-06]
[9.99441803e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.58411353e-06]
[1.00000000e+00]

[1.00000000e+00]
[1.00000000e+00]
[3.55937227e-04]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.65389013e-01]
[1.00000000e+00]
[9.99937057e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.98012320e-04]
[1.00000000e+00]
[9.99973059e-01]
[5.35420077e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.48344634e-05]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[4.52241686e-04]
[1.00000000e+00]
[1.00000000e+00]
[3.73382471e-04]
[3.69121693e-02]
[9.99850512e-01]
[3.84498667e-03]
[1.39415164e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99971747e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[7.86709785e-03]
[1.00000000e+00]
[9.99680042e-01]
[2.83931822e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.66256399e-07]
[9.99028206e-01]
[1.00000000e+00]
[1.00000000e+00]

```
[1.00000000e+00]
[1.09162748e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.04584110e-01]
[2.75814962e-02]
[9.98623610e-01]
[1.00000000e+00]
[6.79226100e-01]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[9.99996424e-01]
[1.15383136e-06]
[1.00000000e+00]
[9.90585625e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.26976789e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.51660561e-05]
[1.00000000e+00]
[1.00000000e+00]
[9.99994993e-01]
[5.15344937e-07]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[3.13105481e-03]
[3.60719711e-01]
[9.99999881e-01]
[1.00000000e+00]
[1.00000000e+00]
[9.96800900e-01]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[1.00000000e+00]
[2.86293738e-02]
[1.00000000e+00]
[1.00000000e+00]]
```

MULTI LAYER PERCEPTRON

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
import numpy as np
import time

start_time = time.time()

# Load the dataset
file_path = '/content/weather.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its structure
print(data.head())

# Check for NaNs in the dataset
print("NaNs in the dataset before preprocessing:", data.isna().sum().sum())

# Separate features and target
X = data.iloc[:, :-1]  # Assuming all columns except the last are features
y = data.iloc[:, -1]   # Assuming the last column is the target

# Handle NaNs in the target variable if any
if y.isna().sum() > 0:
    if y.dtype == 'object':
        y.fillna(y.mode()[0], inplace=True)
    else:
        y.fillna(y.mean(), inplace=True)

# Check for NaNs in the target after filling
print("NaNs in the target variable after preprocessing:", y.isna().sum())

# Determine if the task is classification or regression based on the target column
data type
if y.dtype == 'object' or len(y.unique()) < 20:  # Simple heuristic:
classification if target has fewer unique values
    task = 'classification'
else:
    task = 'regression'

# Split the 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)

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object']).columns
numerical_cols = X.select_dtypes(include=['float64', 'int64']).columns
```

```python
# Create a preprocessor for both categorical and numerical features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='mean')),  # Impute missing
numerical values
            ('scaler', StandardScaler())]), numerical_cols),
        ('cat', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='most_frequent')),  # Impute
missing categorical values
            ('onehot', OneHotEncoder(handle_unknown='ignore'))]),
categorical_cols)
    ])

# Fit the preprocessor on the training data
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)

# Convert transformed data back to DataFrame to check for NaNs
X_train_transformed_df = pd.DataFrame.sparse.from_spmatrix(X_train_transformed)
X_test_transformed_df = pd.DataFrame.sparse.from_spmatrix(X_test_transformed)

# Check for NaNs in the transformed data
print("NaNs in X_train_transformed after preprocessing:",
X_train_transformed_df.isna().sum().sum())
print("NaNs in X_test_transformed after preprocessing:",
X_test_transformed_df.isna().sum().sum())

# Ensure there are no NaNs in the transformed data
X_train_transformed_df = X_train_transformed_df.sparse.to_dense().replace([np.inf,
-np.inf], np.nan).fillna(0)
X_test_transformed_df = X_test_transformed_df.sparse.to_dense().replace([np.inf, -
np.inf], np.nan).fillna(0)

# Convert back to numpy arrays for model training
X_train_transformed = X_train_transformed_df.values
X_test_transformed = X_test_transformed_df.values

# Choose the MLP model based on the task type
if task == 'classification':
    mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=100, random_state=42)
else:
    mlp = MLPRegressor(hidden_layer_sizes=(50,), max_iter=100, random_state=42)

# Train the model using the preprocessed training data
mlp.fit(X_train_transformed, y_train)

# Make predictions
```

```python
y_pred = mlp.predict(X_test_transformed)

# Evaluate the model
if task == 'classification':
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy: {accuracy * 100:.2f}%')
else:
    mse = mean_squared_error(y_test, y_pred)
    print(f'Mean Squared Error: {mse:.2f}')

end_time = time.time()
print(f'Total execution time: {end_time - start_time:.2f} seconds')
```

OUTPUT:

```
Summary Precip Type  Temperature (C)  Apparent Temperature (C)  \
0  Partly Cloudy        rain        9.472222                  7.388889
1  Partly Cloudy        rain        9.355556                  7.227778
2  Mostly Cloudy        rain        9.377778                  9.377778
3  Partly Cloudy        rain        8.288889                  5.944444
4  Mostly Cloudy        rain        8.755556                  6.977778

   Humidity  Wind Speed (km/h)  Wind Bearing (degrees)  Visibility (km)  \
0      0.89            14.1197                     251          15.8263
1      0.86            14.2646                     259          15.8263
2      0.89             3.9284                     204          14.9569
3      0.83            14.1036                     269          15.8263
4      0.83            11.0446                     259          15.8263

   Loud Cover  Pressure (millibars)                      Daily Summary
0           0               1015.13  Partly cloudy throughout the day.
1           0               1015.63  Partly cloudy throughout the day.
2           0               1015.94  Partly cloudy throughout the day.
3           0               1016.41  Partly cloudy throughout the day.
4           0               1016.51  Partly cloudy throughout the day.
NaNs in the dataset before preprocessing: 517
NaNs in the target variable after preprocessing: 0
NaNs in X_train_transformed after preprocessing: 0
NaNs in X_test_transformed after preprocessing: 0
```