****

This a report about our project, here are the steps that we follow to get the final output:

**1)Data Preprocessing**

**1.We know some information about data throw reading it using pandas library**

**2.**Knowing Data type for each column, number of null values of each one , table of statistics for each column , column named “loud cover” has 0 values , copy data to another data , # of null values in the copied data , col named “ Precip Type” has 517 rows null value , replacing these null values with the mode of the data called “rain” , then drop column named “loud cover” , col named“Formatted Date “ extract it into (year , month , day) , then drop it , showing histograms for each col , Notice that col named “Pressure” contain outliers , compute mean for this col then fill table with it , convert every col its datatype is “objects” to “integer” data type ,drop col named “daily Summary”, Col named “summary” minimize its length and convert its data type to integer , making dictionary for these data set using “key” , then replace col summary to Summary of dictionary.

Precip Type col replace value rain 🡪 1 , snow 🡪 0

Then split dataset into training data set and testing dataset.

---------------------------------------------------------------------------------------------------------

**2)Creation a Multi-Layer Neurol Network**

-**To create MLNN using Back Propagation, we use these functions: -**

**-initialize network () 🡪** Initialize the network takes 3 parameters (num of inputs, num of hidden layers, num of output layers) ,initialize it as an empty list to be filled with layers , hidden layer also an empty array, neuron is a dictionary has weights of each neuron , this dictionary has information about each neuron , layer is a list of dictionary, outer loop from zero to the number of hidden layer, then we have inner loop from zero to the number of inputs +1 and this one is the bias, in the inner loop we loop on each input and at end loops on bias, then append random number into weight of the dictionary, append neural dictionary to the hidden layer, append the hidden layer to the network.

**Output layer🡪** outer loop from zero to the number of output layer, then we have inner loop from zero to the number of head layer+1 and this one is the bias, in the inner loop we loop on each hidden layer and at end loops on bias, then append random number into weight of the dictionary, append the output layer to the network, then we return network.

-Then pin the value of seed which it means first random number.

**activate () 🡪** takes two parameters weight and input, then make initialize to activation with bias =-1, loop on all neural and calculate the activation by multiplying each weight with input for each neural and make summation for all things above and return the activation.

**- transfer () 🡪** take parameter activation and calculate sigmate function.

**- forward propagate () 🡪** takes two parameters network and row of inputs then outer loop on all layers exist in network and make new empty input, then making inner loop loops on all neural exist in layer, then calculate activation function takes two parameters weight and input, transfer and that will be the output, in this level the neural will have its weight and output (Update the dictionary of neural), then append neural (output) and append it to new input (update new input), then return input.

**-transfer\_derivative ()🡪** Calculate the derivative of a neuron output (derivative of sigmate function).

**- backward\_propagate\_error ()🡪** takes 2 parameters network and expected output, outer for loop loops in reversed way on network, then initialize layer = network, then initialize empty list named error, then calculate error if the loop doesn’t at the end of the neural network the code in to inner loop loops on each neural of the layer and initialize error = 0 and make loop on the inner loop and calculate error = summation of weight of each neural multiplying neural of delta and append error to errors list else if it the last item append expected -neural output in errors list, then calculate delta neural of delta = errors of neural \* transfer derivative of neural output.

---------------------------------------------------------------------------------------------------------

**3) Set weights randomly**

In the above part we set the weights and bias randomly.

**MLNN architecture:**

**Inputs**

–-

0.44

0.002

0.76

0.43

0.83

0.02

0.09

0.78

0.65

0.44

0.49

0.25

0.76

0.13

0.84

**Output**

**Hidden Layer**

**Layer**

**Bias**