ABSTRACT

Rainfall Prediction is one of the most important and challenging task in modern world. In general, climate and rainfall are highly non-linear and complicated phenomena which require advanced computer modelling and simulation for their accurate prediction. An Artificial Neural Network (ANN) can be used to predict behaviour of such non-linear system because of having the capability of examining and determining the historical data used for prediction.In this project ANN is applied on a large dataset consisting of weather record and a User Interface (UI) is developed for effective user interaction with the system.

Expected programme outcomes: PO1, PO2, PO3, PO4, PO5, PO6, PO9, PO10, PO11, PO12.

1. INTRODUCTION

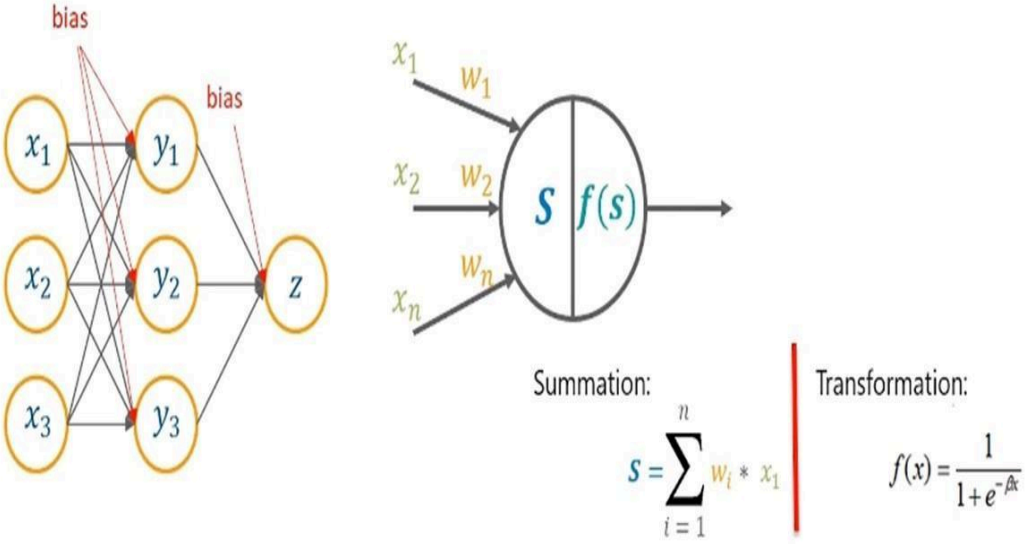
Artificial Neural Networks

Definition:

Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabelled data according to similarities among the example inputs, and they classify data when they have a labelled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression. These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

Components:



Neurons-

ANNs retained the biological concept of artificial neurons, which receive input, combine the input with their internal state (activation) and an optional threshold using an activation function,

and produce output using an output function. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image. The important characteristic of the activation function is that it provides a smooth transition as input values change, i.e. a small change in input produces a small change in output.

Connections and weights-

The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance. A given neuron can have multiple input and output connections.

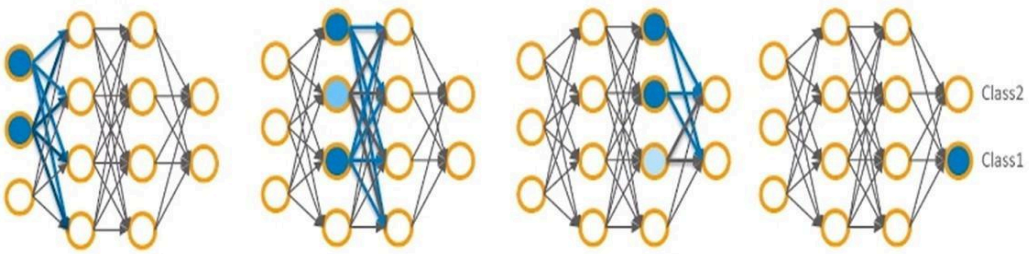
Propagation function-

The propagation function computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum. A bias term can be added to the result of the propagation.

Organization-

The neurons are typically organized into multiple layers, especially in deep learning. Neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. The layer that receives external data is the input layer. The layer that produces the ultimate result is the output layer. In between them are zero or more hidden layers. Single layer and unlayered networks are also used. Between two layers, multiple connection patterns are possible. They can be fully connected, with every neuron in one layer connecting to every neuron in the next layer. They can be pooling, where a group of neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer.

Prediction

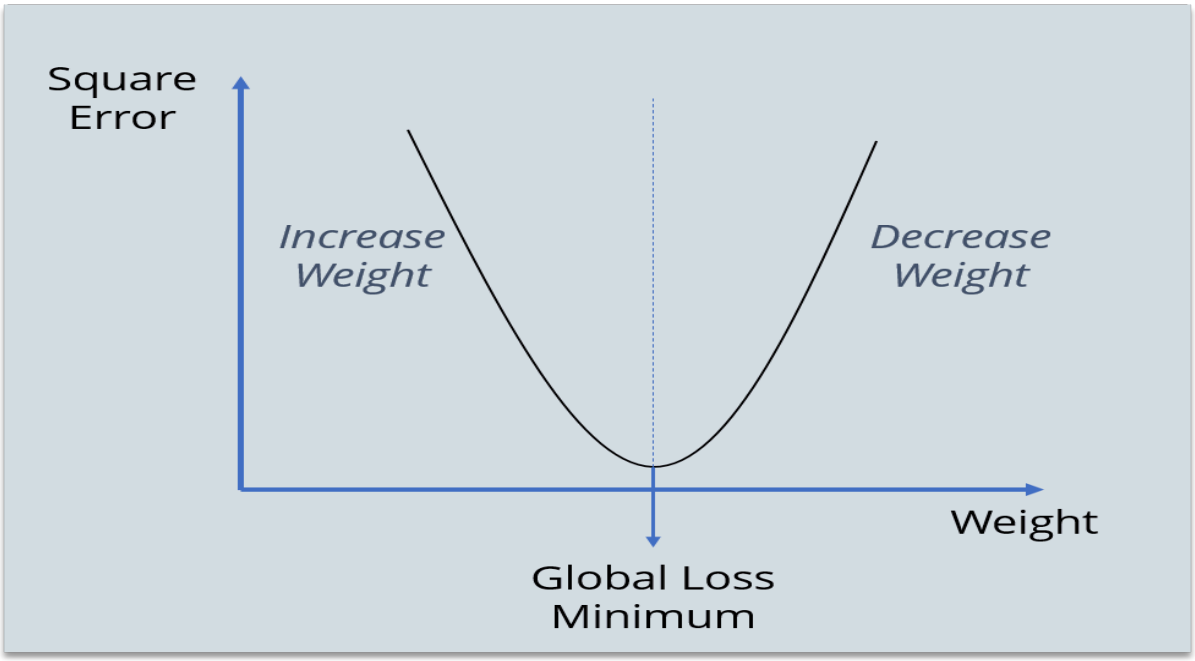


Learning:

Learning is the adaptation of the network to better handle a task by considering sample observations. Learning involves adjusting the weights (and optional thresholds) of the network to improve the accuracy of the result. This is done by minimizing the observed errors. Learning is complete when examining additional observations does not usefully reduce the error rate. Even after learning, the error rate typically does not reach 0. If after learning, the error rates too high, the network typically must be redesigned. Practically this is done by defining a cost function that is evaluated periodically during learning. As long as its output continues to decline, learning continues. The cost is frequently defined as a statistic whose value can only be approximated. The outputs are actually numbers, so when the error is low, the difference between the output (almost certainly a cat) and the correct answer (cat) is small. Learning attempts to reduce the total of the differences across the observations. Most learning models can be viewed as a straightforward application of optimization theory and statistical estimation.

Back Propagation:

The Back propagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem. Basically, we need to figure out whether we need to increase or decrease the weight value. Once we know that, we keep on updating the weight value in that direction until error becomes minimum. You might reach a point, where if you further update the weight, the error will increase. At that time you need to stop, and that is your final weight value.



Prediction04

2. LITERATURE SURVEY

PAPER	DESCRIPTION	MERITS	DEMERITS
Comparative Study of Rainfall Prediction Modeling Techniques Razeefl Mohd,Muheet Ahmed Butt and Majeed Zaman ISSN,2018	This system uses Naïve Bayesian Classifier algorithm which is based on Bayes rule of Conditional probability. It analysis each attribute individually and assumes that all of them are independent	● It requires a small amount of training data to estimate the parameter necessary for classification	● It is almost impossible to find a set of attributes which are completely independent. ● Performance is sensitive to skewed data.
Rainfall Prediction Using Machine Learning Arnav Garg, Himanshu Pandey ISSN,5 May,2019	This system uses KNN which is a non-parametric strategy utilized for classification and regression.	● Since KNN requires no training predictions, new data can be added which will not impact the accuracy of the algorithm.	● Sensitive to noisy data and missing values. ● Does not work well with large dataset.

3.1 EXISTING SYSTEM:

Naïve Bayes' Classifier algorithm uses a statistical method which assumes an underlying probabilistic model, the Bayes' theorem. This algorithm has drawbacks like zero conditional probability problem, numerical underflow and violation of independence assumption that leads to the chance of loss of accuracy in the prediction results.

Another approach to predictive analysis is the Decision Tree algorithm which has high probability of over fitting, complications in the calculations when there are many class labels that makes this algorithm inefficient for categorical classification.

3.2 PROPOSED SYSTEM

Artificial Neural Network (ANN) can be used to efficiently overcome the disadvantages of the existing system.ANNs have the ability to learn and model non-linear and complex relationships and can be applied to large number of variables or parameters. The network learns by itself using the training data and generates intelligent patterns which are useful for rainfall prediction.

This model is efficient for both binary and categorical classification. .Error minimization in the results is efficiently handled by back propagation technique, thus making ANN efficient and reliable for accurate predictio

5. SYSTEM REQUIREMENT ANALYSIS

4.1 MODULES

- Data Pre-processing Module
- Prediction Module
- User Interface Module

Data Pre-processing Module:

- Import libraries- NumPy, Pandas.
- Import dataset of Austin weather record, Number of samples 1320, Source-Kaggle..
- Handling missing data and null values in the dataset.
- Performing Label Encoding and One hot Encoding on the categorical variables.
- Split the dataset into train set and test set.
- Perform feature scaling.

Prediction Module:

- Import Keras libraries and packages.
- Initialize input, hidden and output layers of ANN model and compile the model.
- Fitting the model to the dataset.
- Make prediction on the test set.

User Interface Module:

It is used to develop an interface for the users to interact with the system and predict the results for the data entered by the users.

6. SOFTWARE DESIGN

6.1 ALGORITHM

Step 1: Start

Step 2: Import the dataset of Austin weather record.

The 20 column parameters in the Austin weather record are: TempHighF,TempAvgF,TempLowF,DewPointHighF,DewPointAvgF,DewPointLowF,HumidityHighPercent,HumidityAvgPercent,HumidityLowPercent,SeaLevelPressureHighInches,SeaLevelPressureAvgInches,SeaLevelPressureLowInches,VisibilityHighMiles,VisibilityAvgMiles,VisibilityLowMiles,WindHighMPH,WindAvgMPH,WindGustMPH,PrecipitationSumInches,Events

Step 3: Check if any null values are present in all the parameter columns using any(), if yes then goto step 4 else goto step 5.

Step 4: Replace the null values with the mean value for each parameter column using mean().

Step 5: Transform the first 19 columns(which has all the independent parameters) into an array (say X) and the remaining one column (which is dependent on the rest columns) into another array(say Y).

Step 6: Perform the label and one hot encoding on the strings of Y which are Rain, No Rain,Thunderstorm,Fog,[Fog,Rain,Thunderstorm],[Fog,Rain],[Rain,Thunderstorm],[Rain ,Snow].

Step 7: Split the X and Y in X_train, X_test, Y_train and Y_test using 70% as training size.

Step 8: Initialization of model

- Initialise input layer with input dimension 19, output dimension as 21 using relu activation function.
- Initialise hidden layer with output dimension as 11 using relu activation function.
- Initialise output layer with output dimension as 8 using softmax activation function.

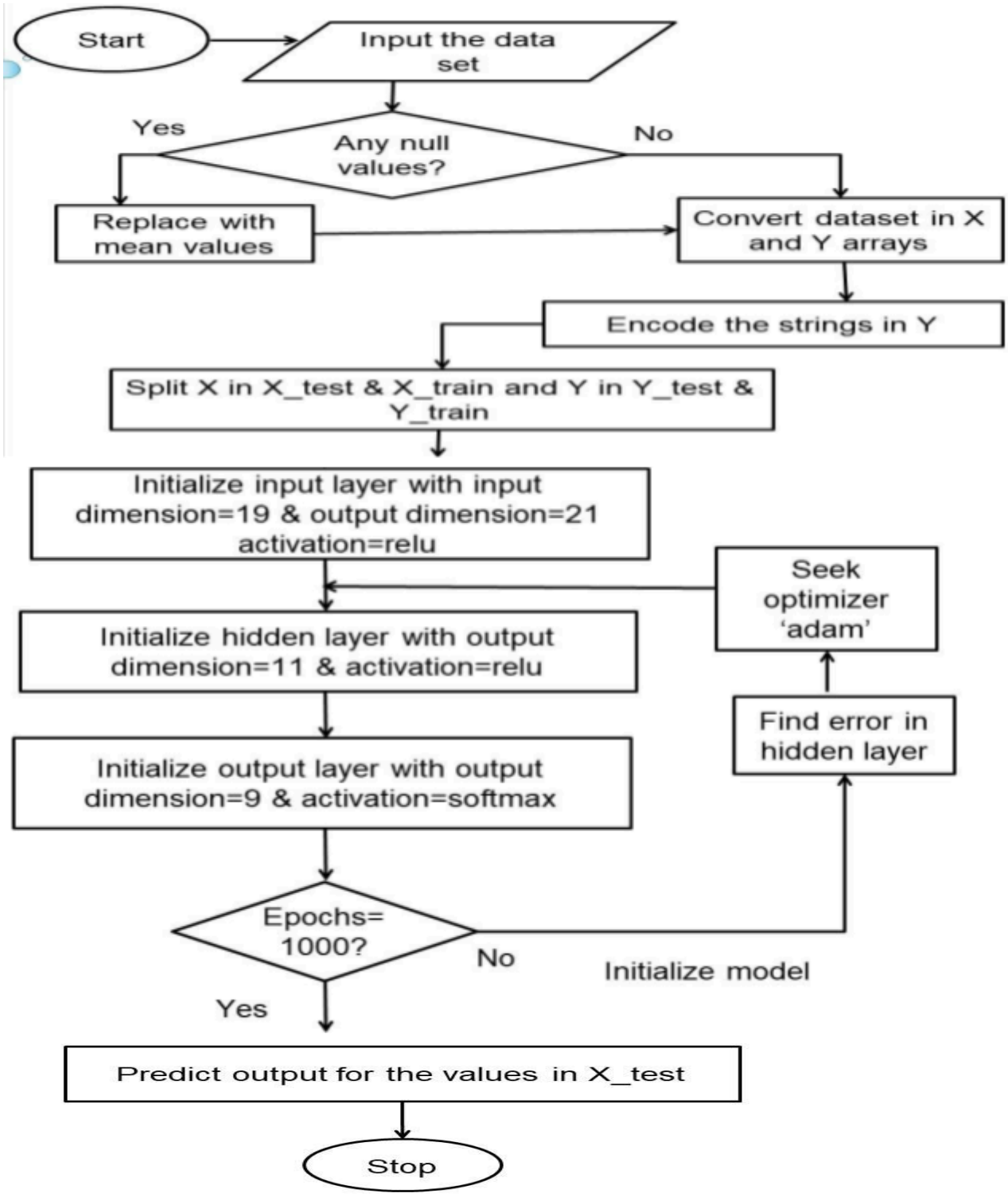
Step 9: Train the model using 128 as batch size for 1000 epochs.

Step 10: Save the model as mymodel.h5.

Step 11: Make prediction on X_test.

Step 12: Stop

6.2 FLOWCHART:



7. IMPLEMENTATION

7.1 CODING:

```
  import numpy as np import matplotlib.pyplot as plt
  import pandas as pd import types import types import pandas as pd
from botocore.client import Config import ibm_boto3

def __iter__(self): return 0
  if not hasattr(body, "__iter__"):body.__iter__= types.MethodType(
  __iter__, body ) dataset = pd.read_csv(body)
  dataset.head() dataset.head() type(dataset) dataset.describe()
  dataset.replace({'PrecipitationSumInches': {'T': 0}},inplace=True)
dataset.replace({'WindGustMPH': {'-':np.nan}},inplace=True)
dataset.isnull().any()
  dataset.replace({'WindAvgMPH': {'-':np.nan}},inplace=True)
dataset.replace({'WindHighMPH': {'-':np.nan}},inplace=True)
dataset.replace({'VisibilityLowMiles': {'-':np.nan}},inplace=True)
dataset.replace({'VisibilityAvgMiles': {'-':np.nan}},inplace=True)
dataset.replace({'VisibilityHighMiles': {'-':np.nan}},inplace=True)
dataset.replace({'SeaLevelPressureLowInches':
  {'-':np.nan}},inplace=True)
dataset.replace({'SeaLevelPressureAvgInches':
  {'-':np.nan}},inplace=True)
dataset.replace({'SeaLevelPressureHighInches':
  {'-':np.nan}},inplace=True)
  dataset.replace({'DewPointLowF': {'-':np.nan}},inplace=True)
dataset.replace({'DewPointAvgF': {'-':np.nan}},inplace=True)
dataset.replace({'DewPointHighF': {'-':np.nan}},inplace=True)
dataset.replace({'HumidityHighPercent': {'-':np.nan}},inplace=True)
dataset.replace({'HumidityAvgPercent': {'-':np.nan}},inplace=True)
dataset.replace({'HumidityLowPercent': {'-':np.nan}},inplace=True)
dataset.head() dataset.head(30)
```

```
   dataset.isnull().any()
   from pandas.api.types import is_numeric_dtype
is_numeric_dtype(dataset['DewPointHighF']) from pandas.api.types import
is_string_dtype is_string_dtype(dataset['DewPointHighF'])
   dataset[["PrecipitationSumInches","HumidityHighPercent","HumidityAvgP
ercent","HumidityLowPercent","TempAvgF","TempLowF","TempHighF","DewPo
intHighF",
   "DewPointAvgF","DewPointLowF","SeaLevelPressureHighInches","SeaLevelP
ressureAvgInches","SeaLevelPressureLowInches","VisibilityHighMiles","
VisibilityAvgMiles","VisibilityLowMiles","WindHighMPH","WindAvgMPH","
WindGustMPH"]] =
   dataset[["PrecipitationSumInches","HumidityHighPercent","HumidityAvgP
ercent","HumidityLowPercent","TempAvgF","TempLowF","TempHighF","DewPo
intHighF",
"DewPointAvgF","DewPointLowF","SeaLevelPressureHighInches","SeaLevelP
ressureAvgInches","SeaLevelPressureLowInches","VisibilityHighMiles","
VisibilityAvgMiles","VisibilityLowMiles","WindHighMPH","WindAvgMPH","
   WindGustMPH"]].apply(pd.to_numeric)
   dataset['DewPointHighF'].fillna(dataset['DewPointHighF'].mean(),inpla
ce=True)
   dataset['DewPointAvgF'].fillna(dataset['DewPointAvgF'].mean(),inplace
   =
   True)
dataset['DewPointLowF'].fillna(dataset['DewPointLowF'].mean(),inplace
   =
   True)
dataset['SeaLevelPressureHighInches'].fillna(dataset['SeaLevelPressur
eHighInches'].mean(),inplace=True)
   dataset['SeaLevelPressureAvgInches'].fillna(dataset['SeaLevelPressure
AvgInches'].mean(),inplace=True)
   dataset['SeaLevelPressureLowInches'].fillna(dataset['SeaLevelPressure
LowInches'].mean(),inplace=True)
dataset['VisibilityHighMiles'].fillna(dataset['VisibilityHighMiles'].
   mean(),inplace=True)
dataset['VisibilityAvgMiles'].fillna(dataset['VisibilityAvgMiles'].
   mean(),inplace=True)
dataset['VisibilityLowMiles'].fillna(dataset['VisibilityLowMiles'].
   mean(),inplace=True)
dataset['HumidityHighPercent'].fillna(dataset['VisibilityLowMiles'].
   mean(),inplace=True)
dataset['HumidityAvgPercent'].fillna(dataset['VisibilityLowMiles'].
   mean(),inplace=True)
dataset['HumidityLowPercent'].fillna(dataset['VisibilityLowMiles'].
```

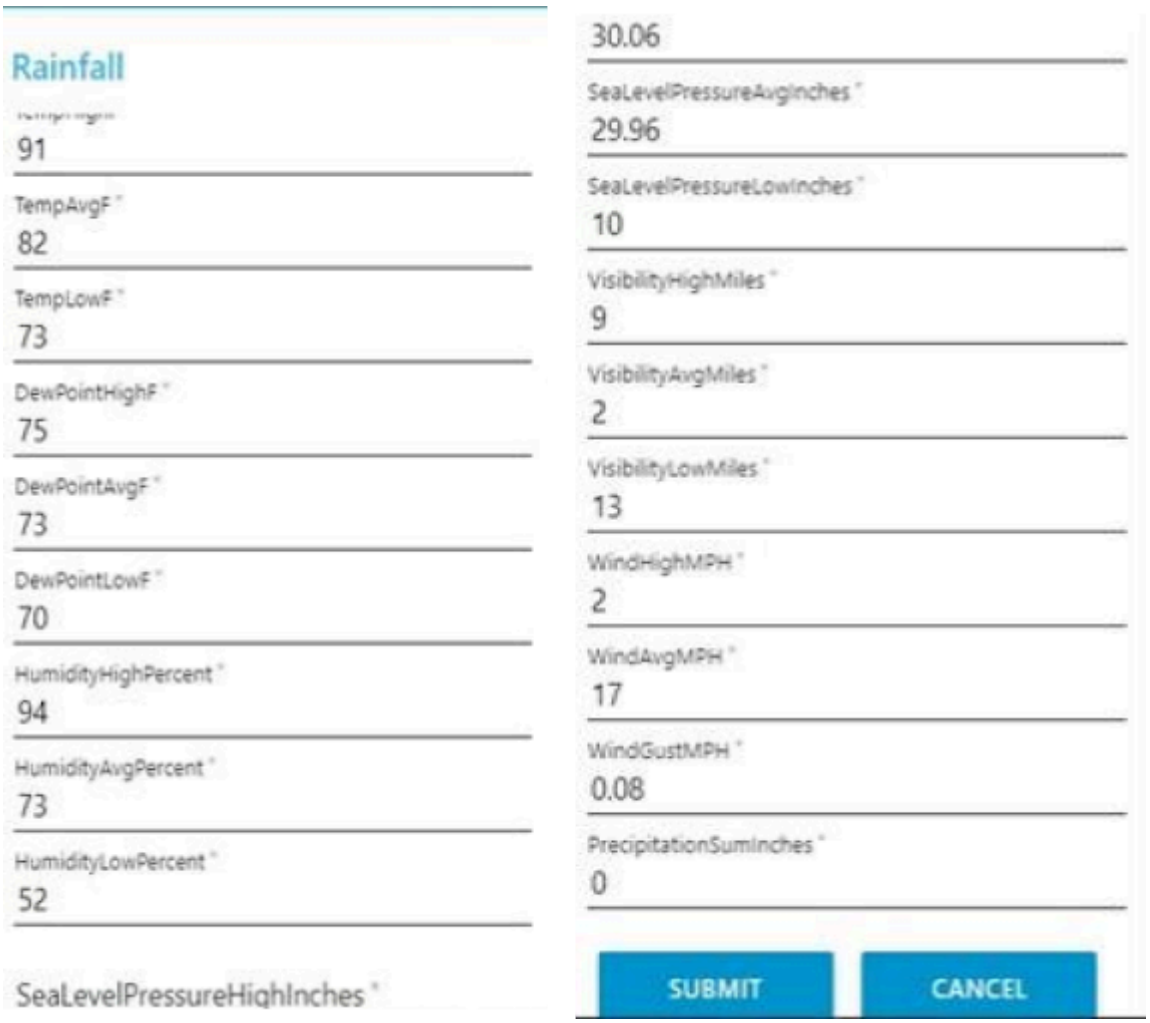
```
   mean(),inplace=True)
   dataset['WindHighMPH'].fillna(dataset['WindHighMPH'].mean(),inplace=True
   dataset['WindAvgMPH'].fillna(dataset['WindAvgMPH'].mean(),inplace=Tru e)
   dataset['WindGustMPH'].fillna(dataset['WindGustMPH'].mean(),inplace=
True)
   dataset.isnull().any()
   from pandas.api.types import is_numeric_dtype
is_numeric_dtype(dataset['DewPointHighF']) x = dataset.iloc[:, 1:20].values y
= dataset.iloc[:, 20:21].values x
   x.shape y
   y.shape
   from sklearn.preprocessing import LabelEncoder
   lb1=LabelEncoder() y[:,0]=lb1.fit_transform(y[:,0]) y
   from sklearn.preprocessing import OneHotEncoder
oh=OneHotEncoder(categorical_features=[0]) y=oh.fit_transform(y).toarray()
from pandas.api.types import is_numeric_dtype
   is_numeric_dtype(y) is_numeric_dtype(x)
   from sklearn.model_selection import train_test_split
   x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s
tate=0)
   from sklearn.preprocessing import StandardScaler sc =StandardScaler()
x_train =sc.fit_transform(x_train) x_test =sc.transform(x_test)
y_test.shape
   y.shape
   from sklearn.model_selection import train_test_split
   x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_s
tate=0)
   from sklearn.preprocessing import StandardScaler sc =StandardScaler()
x_train =sc.fit_transform(x_train) x_test =sc.transform(x_test) x.shape
x_train.shape y_train.shape
```

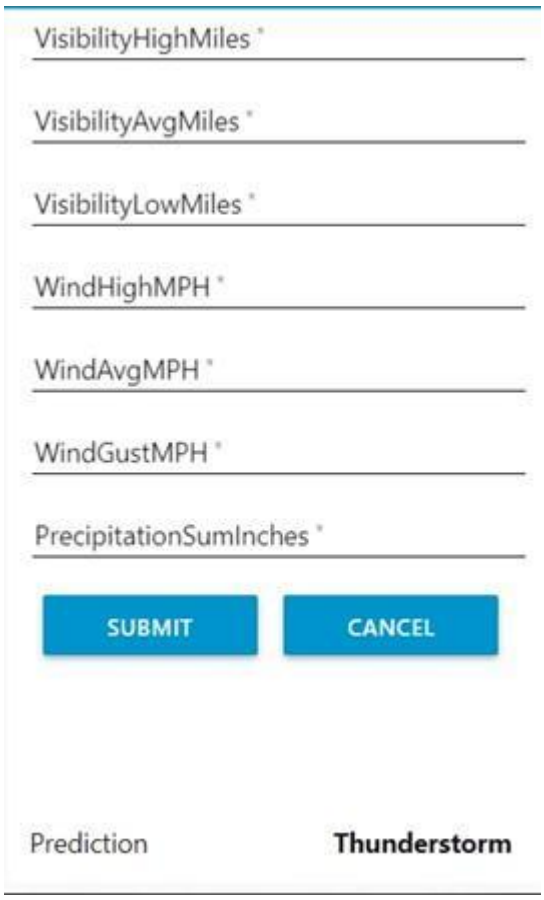
```
import keras from keras.models import Sequential from keras.layers import
Dense
   model=Sequential()
   model.add(Dense(input_dim = 19,init = 'uniform',activation
   ='relu',output_dim =21 ))
   model.add(Dense(input_dim = 21,init = 'uniform',activation
   ='relu',output_dim =11 ))
model.add(Dense(output_dim = 9, init = 'uniform', activation = 'softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metric
s=['accuracy'])       model.fit(x_train,y_train,epochs=1000,batch_size=128)
y_pred=model.predict(x_test)   y_pred   y_pred=model.predict_classes(x_test)
y_pred    y_pred    model.save("mymodel.h5")    y_pred=model.predict(x_test)
get_ipython().system(u'tar      -zcvf      mymodel.tgz      mymodel.h5')       y_pred
y_pred=model.predict_classes(x_test) y_pred y_pred.shape
   y.shape
   import matplotlib.pyplot as plt plt.hist(y_pred) from
watson_machine_learning_client import WatsonMachineLearningAPIClient
   client = WatsonMachineLearningAPIClient( wml_credentials ) model_id =
model_details["metadata"]["guid"] model_deployment_details =
client.deployments.create( artifact_uid=model_id, name="deployment" )
scoring_endpoint =
   client.deployments.get_scoring_url(model_deployment_details)
scoring_endpoint scoring_payload = {"fields": ["Prediction"],"values":
([[0,1,100,1,25,8,0,0,1,0,200,23,45,56,67,78,89,90,67]])} predictions =
client.deployments.score(scoring_endpoint, scoring_payload)
scoring_endpoint predictions
```

RESULTS

8.1 OUTPUT

SCREENS:





UI Link:
https://rainfallpredicting.eu-gb.mybluemix.net/ui/#!/0?socketid=2NPiBxJCI3GO1D_5AAAZ

8. CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The "Rainfall Prediction" system using Artificial Neural Networks has been developed successfully with User Interface(UI) for effective user interaction with the built system where the user can predict the weather condition of Austin on day to day basis. Further, the accuracy achieved by the developed ANN model is recorded to be 89%.

Since rainfall is natural climatic phenomena whose prediction is challenging and demanding, ANN algorithm becomes an inductive approach in rainfall prediction owing to its high flexibility, data driven learning in building models and its ability to model both linear and non-linear systems without the need to make assumptions as are implicit in most traditional statistical approaches.

The attained Programme Outcomes in the project are:
PO1, PO2, PO3, PO4, PO5, PO6, PO9, PO10, PO11, PO12.

Future Enhancement

In future this project can be enhanced by using wireless sensors or devices that transmit weather conditions to the system and by maintaining a database to record the weather conditions.

The functionality of system predicting short-term and long-term results can be added.

BIBLIOGRAPHY AND REFERENCES

Bibliography

[1] John D. Kelleher, Aolfe D'Arcy and Brian Mac Namee, Fundamentals of Machine Learning for Predictive Data Analytics, 2015.

[2] Antonio Gulli, Deep Learning with Keras, Packt Publishers, 2017.

[3] Tariq Rashid, Make Your Own Neural Network, 2016.

References

[1] https://nptel.ac.in/courses/106/106/106106213/

[2] https://towardsdatascience.com