IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Temperature Prediction Using Various Machine Learning Algorithms

[1] Shasanka Roy, [2] Dr. Pramod Kumar Maurya
School of Computer Science & Engineering, Vellore Institute of Technology,

Vellore – 632014, India

Abstract

The Gross Domestic Product (GDP) of a developing country like India significantly depends on agriculture and the associated industry namely agribusiness industry. These sectors are highly vulnerable with respect to extreme weather events (EWE). It is reported in various literatures that the frequency of EWEs has been increased in recent years over the globe which affects different sectors of agriculture and living beings. Therefore, it demands an accurate prediction prior to a season which would be helpful in this direction. Deep Learning (DL) and Machine Learning (ML) approaches have been popular in the field of prediction. However, it is also proved the performance of ML and DL models varies according to the data sets. In this project work, several ML and DL models such as Linear regression, Decision tree regression, Random Forest regressor, and Neural Network will be used for comparison of the performance in accurate prediction. For this purpose, temperature data of India at a monthly scale spanning over 1901-2015 considered for the analysis. Towards end of the project, the best model will be chosen and will be used for prediction of temperature at monthly scale. Further, significant effort will be given for the development of new strategy which can predict the extreme events.

Keywords – Deep learning, Machine learning, LSTM, Linear regression, Random forest regression, Decision tree.

1.1 INTRODUCTION

Temperature prediction is a perplexing interaction and a difficult task for specialists. It remembers skill for different disciplines. The prediction of climatic boundaries is fundamental for different applications. Some of them incorporate environment observing, dry season location, serious climate prediction, horticulture, and creation, arranging in energy industry, aeronautics industry, correspondence, contamination dispersal and so forth Exact prediction of climate boundaries is a troublesome errand because of the unique idea of climate. Different procedures like linear regression, auto regression,

Multi-facet Perceptron, Radial Basis Function networks are applied to foresee barometrical boundaries like temperature, wind speed, precipitation, meteorological contamination and so forth. Machine Learning approach that is being experimented in a wide variety of climate change studies or prediction problems. Regression incorporates numerous strategies to display few variables when the attention is on the connection between a needy variable and at least one free variable. Most usually, regression is used for assumption for the reliant variable given the free variables On the whole cases, the assessment target is a component of the free variables called the regression work.

1.2 OBJECTIVE

To find a suitable algorithm which provides more accurate temperature prediction results. To test all the possible outcomes and compare them to get the best output. The comparison of the results among all the different modules is the major factor of this project.

1.3 PROBLEM STATEMENT

The problem with the previous model is that not many algorithms were used in execution which resulted in less comparison among the results. The existing system provides less accuracy compared to the current model used in this project.

1.4 PURPOSE

The main purpose of this project is applying various machine learning modules and comparing the results among them to get the suitable module which provides less error. Also the dataset is collected if of Assam region which plays a major roles in agriculture. This project will help many agricultural projects in the Assam state in the near future.

2.1 LITERATURE SURVEY

We have selected various papers related to temperature prediction model which contains various machine learning algorithms to get an idea about the various algorithms which we can implement and get better results.

In our related work we have taken various papers based on temperature prediction using various algorithms. The most commonly used algorithm is the linear regression model which shows the relation between two factors by using a condition. One variable is seen as an enlightening variable, and the other is seen as a dependent variable. The instance of one useful variable is called straightforward direct regression. For more than one enlightening variable, the interaction is called various linear regression.

We also went through papers which has other algorithms like random forest, decision tree, ANN, etc. e found out that ANN was not providing accurate results in most of the models. Decision tree model uses features of an article and readies a model in the development of a tree to expect data later to convey huge unending yield. Constant yield suggests that the outcome isn't discrete, i.e., it isn't tended to simply by a discrete, known course of action of numbers or characteristics. Random forest regression is the algorithm which helps us in setting the random state value and utilizations averaging to improve the prescient precision and power over-fitting.

LSTM models using optimizers are a new addition to this project where we used various optimizers like adam and RMS Prop to get better results and compare with the other module outputs.

2.3 ISSUES IN EXISTING SYSTEM

The existing system needs more computational power because it has more data as input and tensor flow was not used in the model for execution which delays the system. It provides less accuracy and is less efficient because less modules where used in it which resulted in less comparison of results.

2.4 EXISTING SYSTEM

In the literature survey many various papers are taken which includes various machine learning and deep learning modules. Various types of data also used which provides different results. Many algorithms like SVM, LSTM, linear regression, decision tree, random forest regression etc are used in the papers. The existing system had many issues like low accuracy, needs more computation power, less efficiency which effects the results.

3.1 SECIFICATIONS

In his chapter an introduction to the specification about the system software and the language used is provided which is jupyter notebook and python language.

3.2 USER INTERFACE

1.User interface segments

When opening bug reports or sending messages to the Jupyter list , it's valuable to comprehend the names of different UI parts all together that different designers and clients have a simpler time helping you analyze your issues. This segment will acquaint you with the names of UI components inside the Journal and thusly the distinctive Note pad modes.

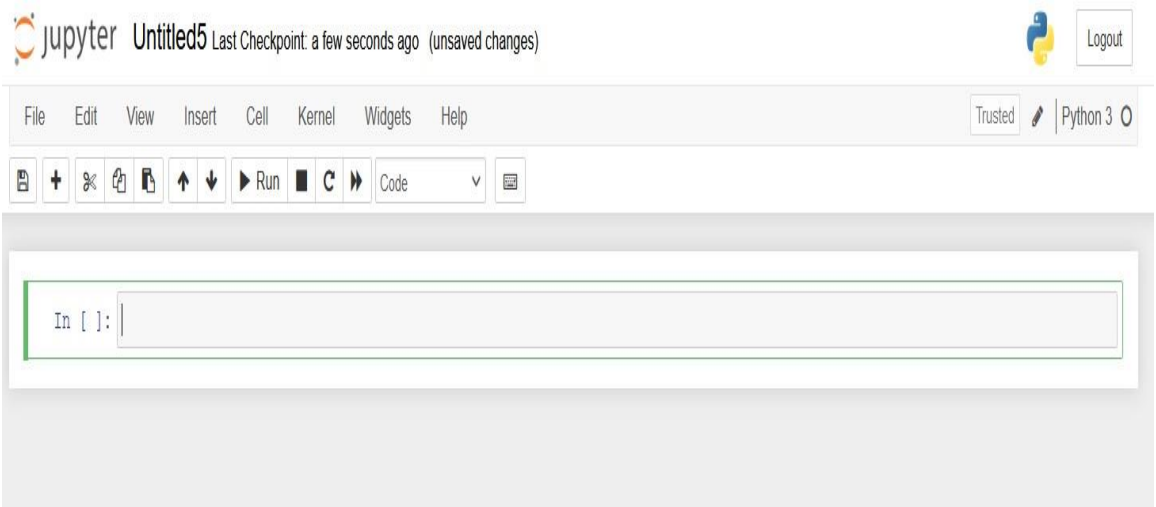

Fig 3.2.1 – User interface

2.Notebook Dashboard

At the point when you dispatch jupyter scratch pad the principal page that you experience is the Note pad Dashboard.

Fig 3.2.2 – Notebook dashboard

3.Notebook Supervisor

When you've chosen a Note pad to alter, the Scratch pad will open in the Note pad Supervisor.

4.Interactive UI Voyage through the Scratch pad

On the off chance that you'd wish to search out increasingly about the exact components inside the Scratch pad Editorial manager, you'll experience the interface visit by choosing Help inside the menu-bar at that point choosing interface Visit.

5. Alter Mode and Note pad Editorial manager

At the point when a cell is in alter mode, the Cell Mode Pointer will change to mirror the cell's state. This state is demonstrated by a little pencil symbol on the most noteworthy right of the interface. At the point when the cell is in order mode, there's no symbol in that area.

3.3 SOFTWARE INTERFACE

Essentially, Anaconda constrictor Pilot is additionally a work area realistic UI (GUI). In which, Anaconda constrictor appropriation grants us to dispatch applications and oversee different bundles without utilizing order line orders. Guide can search for bundles on Anaconda constrictor Cloud or during a nearby Anaconda constrictor Vault.
It is accessible for Windows, macOS, and Linux.

For pilot, first go to the Guide Cheat Sheet and afterward introduce Anaconda constrictor.

The Beginning with Guide area demonstrates the best approach to begin Pilot from the alternate ways or from a terminal window.

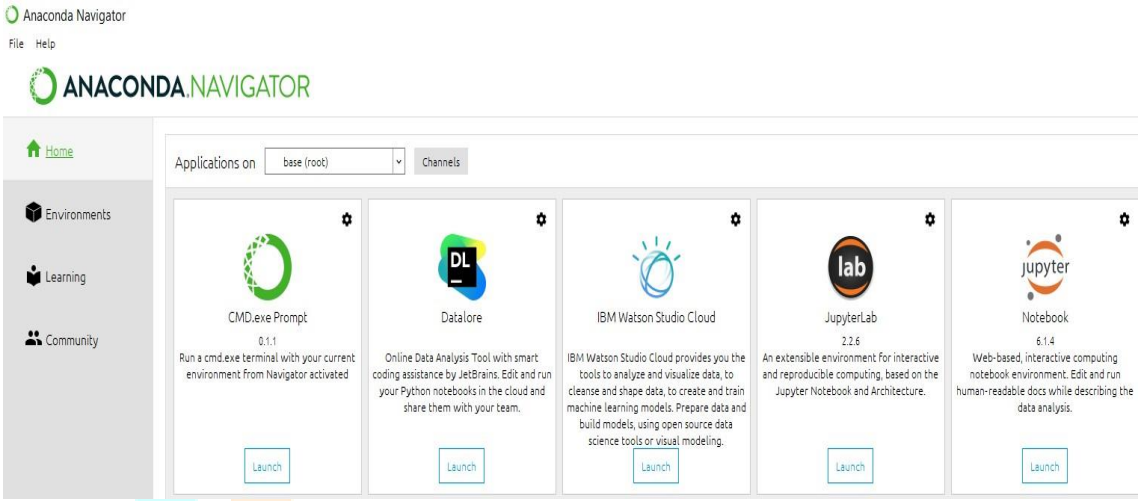

Fig 3.3- Anaconda navigator

Why use Navigator?

So as to run, numerous logical bundles rely on explicit variants of different packages. Information researchers frequently utilize various adaptations of the numerous packages and utilize numerous conditions to isolate these various forms.

The order line program conda is consider as both chief as package and condition. This enables information researchers to ensure that every variant of each bundle has all the conditions it requires and works accurately.

Navigator is a straightforward, point-and-snap on account of work with bundles and conditions without composing conda orders during a terminal window. You can utilize it to search out the bundles you might want, introduce them in a situation, run the packages, and updates them – all inside Navigator.

How can I run code with Navigator?

Jupyter Notebook play out a similar capacity and can be use for its benefit. Jupyter

Notebook consolidate different tasks like the code, graphic content, yield and intelligent pictures into a solitary notebook file. It can likewise be altered, seen and utilized in the internet browser. Jupyter Note pads became well known framework because of such progressed and productive capacities.

3.4 PERFORMANCE REQUIREMENTS

1. Python as a language has an extraordinary network behind it. Any issues experienced can be effortlessly settled with an excursion to Stack Flood. Python is one of the most famous dialects on the site which makes a positive response to any question.

2. Python has numerous integral assets prepared for logical registering. Packages, for example, Numpy, Pandas, and SciPy are unreservedly accessible, produced, and all around recorded. Packages like these can be shockingly decreased, and improve the code expected to compose a given program. This makes emphasis quicker to be pseudo-code. This is significant when the pseudo code gave in the scholastic papers should be structured and tried. Utilizing Python, this progression is typically aimless. Notwithstanding, Python doesn't remain imperfect. The language is staggeringly adaptable and the packages are well known with Duck composing. This can be baffling when a package strategy returns something that, for instance, resembles a cluster instead of a real assortment. In mix with the way that standard Python contents don't indicate the kind of return, this can prompt a great deal of preliminary blunder and mistake that won't be conceivable with another strong content. This is an issue that makes figuring out how to utilize another Python package or library more troublesome than it would some way or another be.

Python was the language of decision for this activity. This was a simple choice for some reasons.

3.4.1 SECURITY REQUIREMENT

Contingent upon the idea of the issue, now and then security@python.org may not be the best revealing channel.

The degree of hazard is regularly controlled by the result of the impact that was once abused, and the probability of misuse happening. At the end of the day, if a bug can cause genuine harm, yet it takes a higher option to misuse the bug, at that point the bug is anything but a genuine risk. Additionally, on the off chance that a bug is effortlessly utilized, however its effect is constrained, at that point it is anything but a major issue.

There is no immovable standard to decide whether a bug merits detailing. General principle any assault that has the right to be accounted for with a security address ought to permit the assailant to bargain the protection, honesty and accessibility of the Python program or its working framework where the aggressor has no influence over it.

To show this point, here are a few instances of past issues and what the Python Security Reaction Group (PSRT) is pondering. On the off chance that you have any questions, be that as it may, if it's not too much trouble send us a report in any case. In the event that the respondent doesn't locate any legitimate code, the report can be resubmitted openly.

4.1 CONCEPT AND DESIGN

Here we can see the system architecture diagram of our model it starts with importing all the required libraries and importing the dataset, taking to the second step is show the relation among the parameters which are most effective to perform the task and get more accurate results. The next step is data pre processing where we clean the dataset and make it suitable for our model. After completing pre processing our dataset is ready to be executed in the algorithms. The dataset is divided into two parts which is training and testing datasets which is implemented in the algorithms to get the results.

4.2 SYSTEM ARCHITECTURE

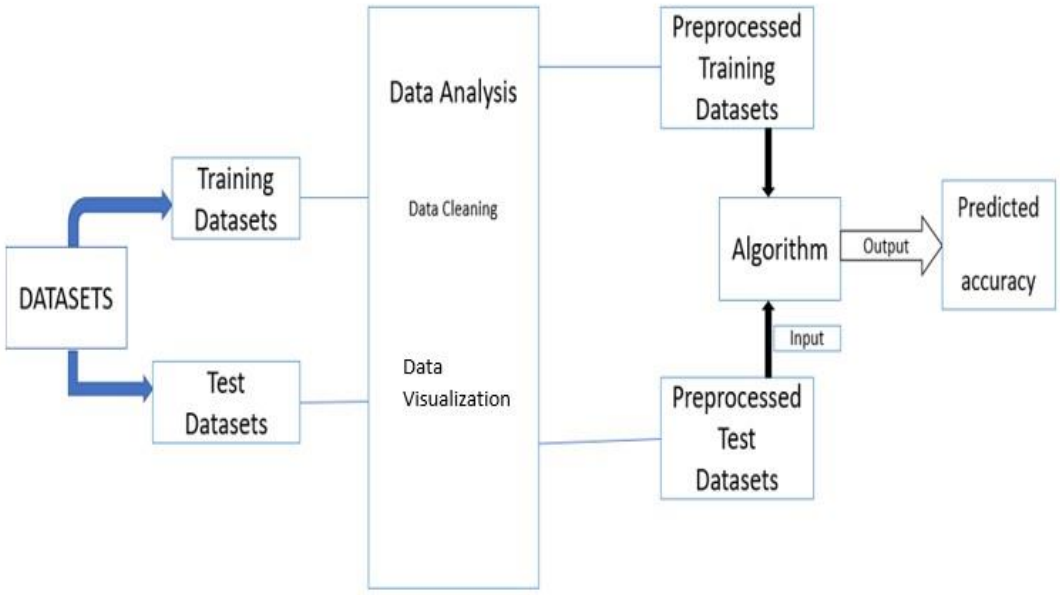

Fig 4.2- System architecture

4.3 CONCEPT

The system architecture shows the basic idea behind the working module. The first step is to import the libraries and the dataset . Tensorflow has been used for faster execution of the module as it handles large data inputs and provides quick and accurate results. Data pre-processing is done in order to clean the data to check if the dataset has nay null values which will effect the model during execution.

Data visualization is done to get an overview of the dataset and to get an idea about the relation among the columns. Later the dataset is divided into two parts. Training set and test set which is later passed on to the modules to get the results.

5.1 MODULES

In this part all the required modules are explained and discussed on how the modules work under various circumstances.

Linear regression- Linear regression shows the relation between two factors by using a condition. One variable is seen as an enlightening variable, and the other is seen as a dependent variable. The instance of one useful variable is called straightforward direct regression. For more than one enlightening variable, the interaction is called various linear regression.

Decision tree regression- Decision tree regression sees features of an article and readies a model in the development of a tree to expect data later to convey huge unending yield. Constant yield suggests that the outcome isn't discrete, i.e., it isn't tended to simply by a discrete, known course of action of numbers or characteristics.

Random forest regressor- A random forest is a meta assessor that fits various ordering decision trees on different sub-examples of the dataset and utilizations averaging to improve the prescient precision and power over-fitting.

Long Short Term Memory- It is a sort of recurrent neural organization. In RNN yield from the last advance is taken care of as contribution to the current advance. It handled the issue of long haul conditions of RNN in which the RNN can't anticipate the worth put away in the drawn out memory however can give additional exact forecasts from the new data. As the hole length builds RNN doesn't give proficient execution. LSTM can as a matter of course hold the data for extensive stretch of time. It is utilized for handling, foreseeing and ordering based on time series information.

Stacked LSTM model : LSTM model compromised of multiple LSTM layers which makes the model more deeper and accurate.

LSTM with Dropout Layer (Optimizer=RMS Prop) : It is used for training neural networks to improve the capability of the optimization. It also decreases the number of function evaluation required.

LSTM with Dropout Layer (Optimizer=Adam) : Adam is replacement optimizer. It combines the best properties of the AdaGrad and RMSProp to handle noisy problems.

6.1 SYSTEM IMPLEMENTATION

In this chapter implementation of system is described in detail. Here the detailed view of code to implement the prediction is described. Jupyter notebook plays a major role in the implementation of the code.

6.2 DATASET DESCRIPTION:

The dataset is of Assam region which contains monthly data from the year 2010-2020 and is taken from Climate Research Unit . It has nine variables

	Year	Month	tmax(degF)	tmin(degF)	ppt(in)	soil(in)	ws(mph)	vap(kPa)	PDSI(unitless)
0	2010	1	77.41	48.78	0.00	4.31	1.83	1.413	-3.48
1	2010	2	81.21	53.55	0.07	3.00	2.62	1.518	-3.58
2	2010	3	88.74	62.60	5.28	3.35	2.93	1.816	-1.77
3	2010	4	90.18	67.80	9.38	7.91	3.04	2.245	3.86
4	2010	5	88.54	73.81	10.39	8.64	3.51	2.718	5.14

Fig 6.2.1 - Dataset

We used describe() function which generates statistics to summarize the central tendency, dispresion and shape of a dataset's distribution, excluding NaN values.

	Year	Month	tmax(degF)	tmin(degF)	ppt(in)	soil(in)	ws(mph)	vap(kPa)	PDSI(unitless)
count	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000
mean	2015.000000	6.500000	84.826136	66.254015	4.880455	5.759242	2.163864	2.329765	-1.487273
std	3.174324	3.465203	5.749630	9.833576	4.700114	2.593938	0.749486	0.691765	2.503978
min	2010.000000	1.000000	70.160000	47.460000	0.000000	1.430000	0.450000	1.217000	-6.500000
25%	2012.000000	3.750000	80.870000	56.885000	0.497500	3.210000	1.610000	1.670500	-3.147500
50%	2015.000000	6.500000	86.715000	68.195000	4.245000	5.775000	2.055000	2.446000	-2.005000
75%	2018.000000	9.250000	89.272500	75.410000	8.187500	8.640000	2.747500	3.037000	-0.280000
max	2020.000000	12.000000	93.740000	80.600000	22.650000	8.640000	3.980000	3.478000	6.010000

Fig 6.2.2 – Dataset description

Once done with the dataset description we perform data pre-processing to clean the dataset like checking for null values which can effect the results. We use data.isnull().any() function to check if there is any null values in the dataset. As we can in the below image there is no null values in the dataset.

```
#check if there is any null value
weather_df.isnull().any()
```

```
Year              False
Month             False
tmax(degF)        False
tmin(degF)        False
ppt(in)           False
soil(in)          False
ws(mph)           False
vap(kPa)          False
PDSI(unitless)    False
dtype: bool
```

Fig 6.2.3 – Checking null values

In the next step we perform to print the summary of the dataset where it shows the datatype and non null count in the dataset. As we can see year and month are the only columns with int64 datatype and rest are float64. Every column has 132 non null count.

```
#prints summary of dataframe
weather_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 9 columns):
 #    Column            Non-Null Count   Dtype
---   ------            --------------   -----
 0    Year              132 non-null     int64
 1    Month             132 non-null     int64
 2    tmax(degF)        132 non-null     float64
 3    tmin(degF)        132 non-null     float64
 4    ppt(in)           132 non-null     float64
 5    soil(in)          132 non-null     float64
 6    ws(mph)           132 non-null     float64
 7    vap(kPa)          132 non-null     float64
 8    PDSI(unitless)    132 non-null     float64
dtypes: float64(7), int64(2)
memory usage: 9.4 KB
```

Fig 6.2.4 – Dataset summary

6.3 DATA VISUALIZATION:

In this section we use various plots and graphs to visualize the dataset. Data visualization is important because it provides the complete overview of the dataset which is very helpful for understanding the dataset it shows the relation among the columns which is very important factor in machine learning algorithms. Firstly we are visualizing the ppt(in) column because we are extraction it to a new data frame.

```
plt.plot(df1)
```

```
[<matplotlib.lines.Line2D at 0x2138267a970>]
```

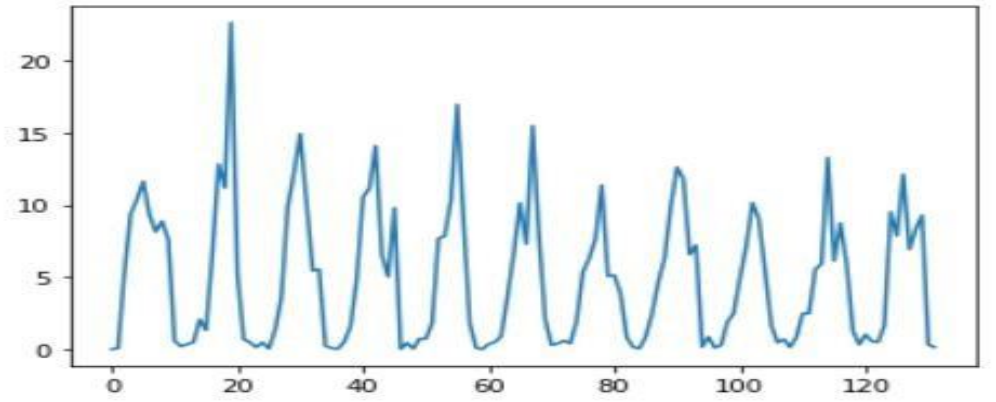

Fig 6.3.1 – ppt(in) plot

In the second step we perform a bar plot to show the visualization of relation among the columns ppt(in) and tmax to show how the ppt(in) changes along with the rise and fall in temperature.

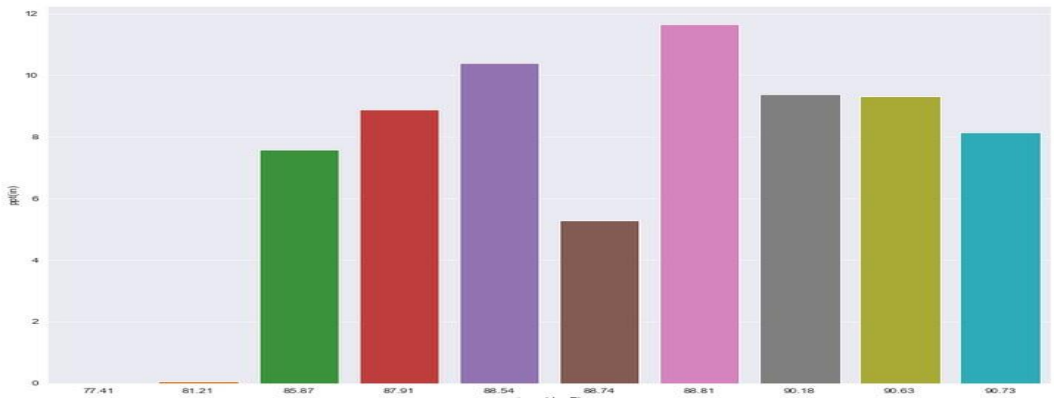

Fig 6.3.2 – Visualization of ppt(in) /Tmax

We also perform a bar plot to show the visualization of relation among the columns ppt(in) and tmin to show how the ppt(in) changes along with the rise and fall in temperature.

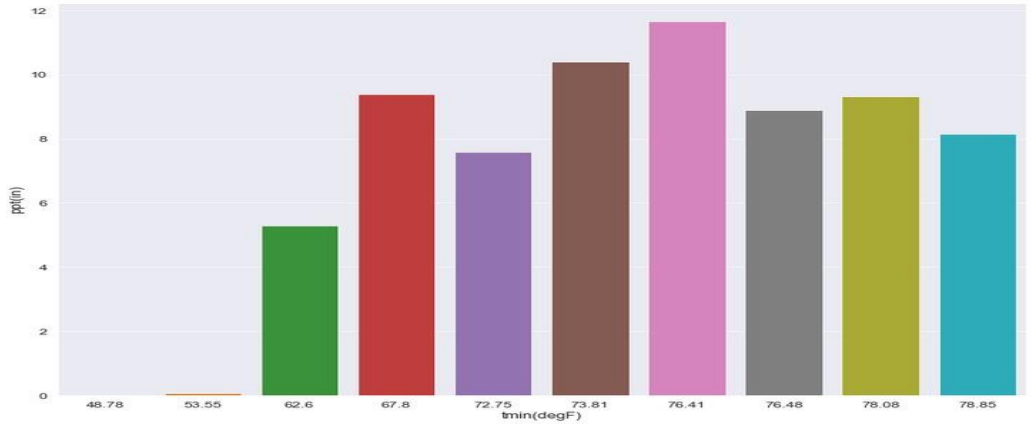

Fig 6.3.3 – Visualization of ppt(in)/Tmin

We also performed a bar plot to visualize the columns ppt(in) and month to show the relation among them of which month has the most amount of ppt(in) in the past years according to the dataset.

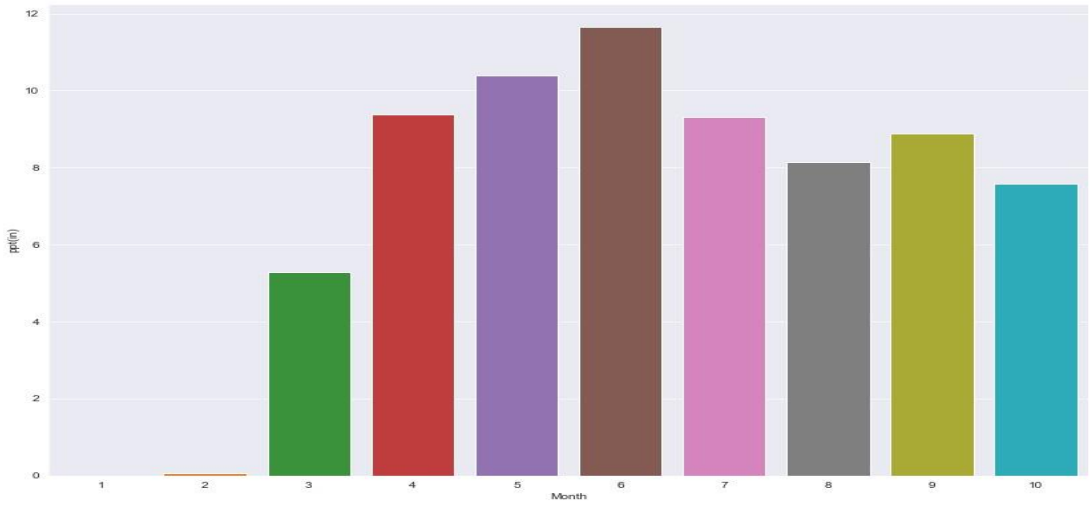

Fig 6.3.4 – Visualization of ppt(in)/Month

We also created a heatmap with data you must have data set up as a matrix where variables are on the columns and rows. Correlation tells you how influential a variable is on the result. So we see that n previous accident is heavily correlated with accidents, while the insurance premium is not.

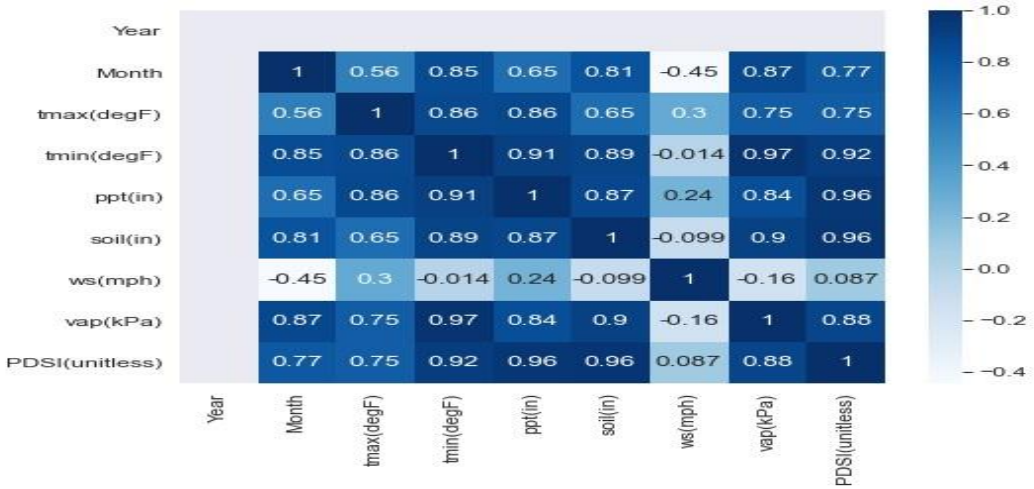

Fig 6.3.5 – Correlation matrix

6.4 METHODOLOGY:

Firstly the required libraries are imported which is required to perform the implementation. We are using sklearn to import all the modules required to perform the task. We are using sklearn model selection to import train test split which helps in dividing the dataset. We also import matplotlib and seaborn to visualize the dataset. We are using tensorflow which helps in faster execution of inputs with accurate results.

In this project various models are used to get a better comparison on which algorithm provides better results. The algorithms are linear regression, decision tree regression, random forest regression, Stacked LSTM model, LSTM with dropout layer using RMS Prop optimizer and LSTM with dropout layer using Adam optimizer.

Linear regression shows the relation between two factors by using a condition. One variable is seen as an enlightening variable, and the other is seen as a dependent variable. The instance of one useful variable is called straightforward direct regression.

For more than one enlightening variable, the interaction is called various linear regression.

Decision tree regression sees features of an article and readies a model in the development of a tree to expect data later to convey huge unending yield. Constant yield suggests that the outcome isn't discrete, i.e., it isn't tended to simply by a discrete, known course of action of numbers or characteristics.

Random forest regressor is a meta assessor that fits various ordering decision trees on different sub-examples of the dataset and utilizations averaging to improve the prescient precision and power over-fitting.

Long Short Term Memory is a sort of recurrent neural organization. In RNN yield from the last advance is taken care of as contribution to the current advance. It handled the issue of long haul conditions of RNN in which the RNN can't anticipate the worth put away in the drawn out memory however can give additional exact forecasts from the new data. As the hole length builds RNN doesn't give proficient execution. LSTM can as a matter of course hold the data for extensive stretch of time. It is utilized for handling, foreseeing and ordering based on time series information.

Stacked LSTM model compromised of multiple LSTM layers which makes the model more deeper and accurate.

LSTM with Dropout Layer (Optimizer=RMS Prop) is used for training neural networks to improve the capability of the optimization. It also decreases the number of function evaluation required.

LSTM with Dropout Layer (Optimizer=Adam) is replacement optimizer. It combines the best properties of the AdaGrad and RMSProp to handle noisy problems.

At first the dataset is collected from Climate Research Unit. The dataset has nine parameters like Year, Month, tmax(degF), tmin(degF), ppt(in), soil(in), ws(mph), vap(kPa), PDSI(unitless).

	Year	Month	tmax(degF)	tmin(degF)	ppt(in)	soil(in)	ws(mph)	vap(kPa)	PDSI(unitless)
0	2010	1	77.41	48.78	0.00	4.31	1.83	1.413	-3.48
1	2010	2	81.21	53.55	0.07	3.00	2.62	1.518	-3.58
2	2010	3	88.74	62.60	5.28	3.35	2.93	1.816	-1.77
3	2010	4	90.18	67.80	9.38	7.91	3.04	2.245	3.86
4	2010	5	88.54	73.81	10.39	8.64	3.51	2.718	5.14

Fig 6.4.1 - Dataset

Firstly the dataset and the required libraries are loaded and pre-processing is performed in the dataset to check if there's a missing value in the dataset. As we know null values in the dataset will effect the results making it less accurate. We use df.isnull() to check if the dataset has any null values. The result shows it has zero null values.

```
round(100*(weather_df.isnull().sum()/len(weather_df.index)),2)
```

```
Year              0.0
Month             0.0
tmax(degF)        0.0
tmin(degF)        0.0
ppt(in)           0.0
soil(in)          0.0
ws(mph)           0.0
vap(kPa)          0.0
PDSI(unitless)    0.0
dtype: float64
```

Fig 6.4.2 – No null values

After performing all the required pre-processing tasks the dataset is passed on to the next stage where the data visualization is done to show the relation among the columns which provides us an overview of the dataset.

Once the data pre-processing and visualization is done we move on to the next stage where the various models are applied to get the results. We use sklearn.model_selection to import train_test_split which helps in splitting the dataset into train and test where the train dataset is applied in the machine learning model and test dataset is used to predict the output. In our case we are using the split of 80 percent train set and 20 percent test set as it is providing more accurate results when algorithms are applied to this dataset because it is the standard advice and the more data we hold to train model is better for predictive power and the more data we hold for test model the better the performance estimation. After splitting the dataset we introduce weather_df_num which includes all the columns except ppt(in) and we assign weather_df_num to X train and test where as we assign ppt(in) to Y train and test.

We tested many different dataset split to check which split provides more accurate results. If we use test case as 0.3 with a random state 4 we get rmse value of "3.247", with test case as

0.4 and random state 4 we get rmse value of "6.9217" and the error margin keeps on increasing as we increase the test set size.

The first model used is the linear regression where we divide the dataset into two parts train and test which is 80 percent train and 20 percent test with a random state of 4 and after performing the execution we get rmse value '3.172255520085546'.

```
model = LinearRegression()
model.fit(train_X,train_y)
```

```
LinearRegression()
```

```
prediction = model.predict(test_X)
```

```
np.mean((prediction-test_y)**2)
```

```
1.0063205084713218e-29
```

```
from sklearn.metrics import mean_squared_error
```

```
mse = mean_squared_error(test_y, prediction)
rmse = np.sqrt(mse)

print('MSE = ', mse)
print('RMSE =', rmse)
```

```
MSE =  1.0063205084713218e-29
RMSE = 3.172255520085546e-15
```

Fig 6.4.3 – Linear regression

The next model is decision tree regression where we also test it with different random state like using random state as 2 we get rmse value of "1.6523" , with random state 3 we get rmse value of "1.4997", with random state as 4 we get rmse value of "1.6560" it is the same thing here again the error margin increases as we increase the random state. So, we take the same split which is 80 percent train and 20 percent test with a random state 1. After execution of the algorithm we get rmse value '1.104837846322557' which is lesser than the linear regression. So we can say that in this case decision tree is providing more accurate results compared to linear regression.

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=1)
regressor.fit(train_X,train_y)
```

```
DecisionTreeRegressor(random_state=1)
```

```
prediction3 = regressor.predict(test_X)
np.mean((prediction3-test_y)**2)
```

```
1.2206666666666661
```

```
mse = mean_squared_error(test_y, prediction3)
rmse = np.sqrt(mse)

print('MSE = ', mse)
print('RMSE =', rmse)
```

```
MSE =  1.2206666666666661
RMSE = 1.104837846322557
```

Fig 6.4.4 – Decision tree

The next model used is random forest regression where we test it with different random states like using random state 2 we get rmse value of "1.5055", with random state of 4 we get rmse value "1.4973" and using random state 5 we take rmse value "1.5051". So, we take the same split which is 80 percent train and 20 percent test with a random state 3 and max depth 50.

After execution of the algorithm we get rmse value '1.4737102707867038 ' which is higher than decision tree but lesser than linear regression. So for now decision tree is providing more accurate results among the three algorithms.

```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(max_depth=50, random_state=3,n_estimators=100)
regr.fit(train_X, train_y)
```

```
RandomForestRegressor(max_depth=50, random_state=3)
```

```
prediction4 = regr.predict(test_X)
np.mean((prediction4-test_y)**2)
```

```
2.17182196222222
```

```
mse = mean_squared_error(test_y, prediction4)
rmse = np.sqrt(mse)

print('MSE = ', mse)
print('RMSE =', rmse)
```

```
MSE =  2.17182196222222
RMSE = 1.4737102707867038
```

Fig 6.4.5 – Random forest

LSTM models are also used with different optimizers to get a better result and compare them with other algorithms. To perform the LSTM algorithm we firstly extract the ppt(in) in a new data frame. Then we perform feature scaling to normalize the distance between the data. Here we also take the same training and test split which is 80-20 split. We also perform an array of values to dataset matrix and reshape the dataset. The first LSTM model is stacked LSTM model where we use tensorflow keras layers and models to import sequential, dense and LSTM.

We run the dataset in the model and run 300 epochs whoch gives us a rmse value "0.11125377960500066".

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 50)            10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 50)            20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

```
In [23]: ### Calculate RMSE performance metrics
         math.sqrt(mean_squared_error(y_train,train_predict))
```

```
Out[23]: 0.09917324263220037
```

```
In [24]: ### Test Data RMSE
         math.sqrt(mean_squared_error(ytest,test_predict))
```

```
Out[24]: 0.11125377960500066
```

Fig 6.4.6 – Stacked LSTM

Second model is LSTM with Dropout Layer (Optimizer=RMS Prop) where we import dropout layer and use RMS Prop as an optimizer. The reason behind using optimizer is it helps in changing the attributes of the neural network and provides increasing accuracy results. We run the dataset in the model and run 300 epochs which gives us rmse value "0.11040159743991015".

```
from tensorflow.keras.layers import Dropout
```

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='RMSprop')
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 10, 50)            10400
_________________________________________________________________
lstm_4 (LSTM)                (None, 10, 50)            20200
_________________________________________________________________
dropout (Dropout)            (None, 10, 50)            0
_________________________________________________________________
lstm_5 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

```
In [32]: ### Calculate RMSE performance metrics
         math.sqrt(mean_squared_error(y_train,train_predict))
Out[32]: 0.11121046142759106

In [33]: ### Test Data RMSE
         math.sqrt(mean_squared_error(ytest,test_predict))
Out[33]: 0.11040159743991015
```

Fig 6.4.7 – LSTM (RMS Prop)

Third model is LSTM with Dropout Layer (Optimizer=Adam) where we import dropout layer from tensorflow keras layer and use adam optimizer. We run the dataset in the model and run
300 epochs which gives us rmse value "0.10674302858568045".

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 10, 50)            10400
_________________________________________________________________
lstm_7 (LSTM)                (None, 10, 50)            20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 50)            0
_________________________________________________________________
lstm_8 (LSTM)                (None, 100)               60400
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 91,101
Trainable params: 91,101
Non-trainable params: 0
```

```
In [40]: ### Calculate RMSE performance metrics
         math.sqrt(mean_squared_error(y_train,train_predict))
Out[40]: 0.09891988077974016

In [41]: ### Test Data RMSE
         math.sqrt(mean_squared_error(ytest,test_predict))
Out[41]: 0.10674302858568045
```

Fig 6.4.8 – LSTM (adam)

From the results we can see that LSTM with Dropout Layer (Optimizer=Adam) provides less error and is more suitable for this dataset.

7.1 CONCLUSION

By applying all the required modules and performing all the required tasks to get the results we get the following rmse values.

	Model	RMSE
1.	Linear regression	3.172255520085546
2.	Decision tree regression	1.104837846322557
3.	Random forest regression	1.4737102707867038
4.	Stacked LSTM model	0.11125377960500066
5.	LSTM with Dropout Layer (Optimizer=RMS Prop)	0.11040159743991015
6.	LSTM with Dropout Layer (Optimizer=Adam)	0.10674302858568045

Table 7.1 - RMSE comparison

The above table shows the comparison of the outputs gained after executing each model as the results show that LSTM using Adam optimizer provides more accuracy in the dataset and is more suitable for providing accurate outputs.

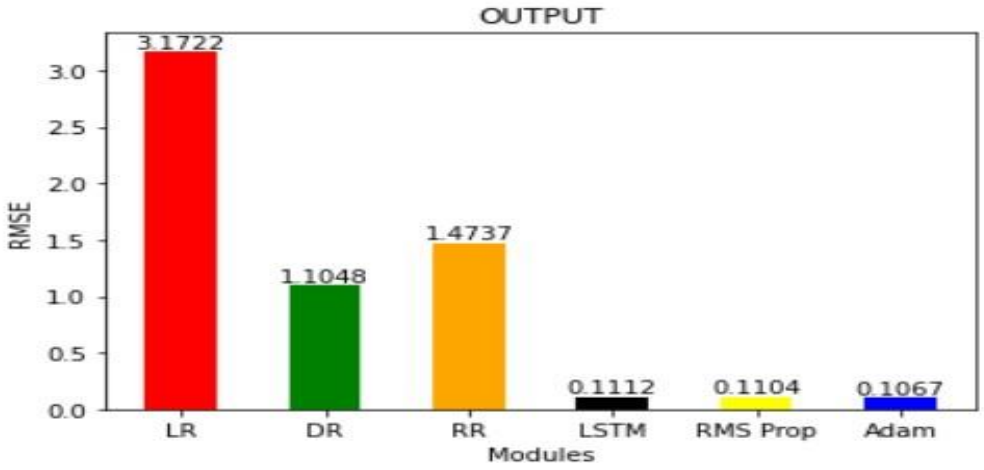

Fig 7.1 – Bar plot

In the above bar plot we can visualize the results gained from the models which shows LSTM using Adam as optimizer is providing accurate results.

REFERENCES

[1] Fernando Mateo1 , Juan J. Carrasco1 ,M´onica Mill´an-Giraldo1 2 , Abderrahim Sellami1 , Pablo Escandell-Montero1 , Jos´e M. Mart´ınez-Mart´ınez1 and Emilio Soria-Olivas1, Temperature Forecast in Buildings Using Machine Learning Techniques, ESANN 2013.

[2] Wint Thida Zaw, Thinn Thu Naing, Modeling of Rainfall Prediction over Myanmar Using Polynomial Regression, International Conference on Computer Engineering and Technology, 2009.

[3] Anjali T, Chandini K, Anoop K, Lajish V L,Temperature Prediction using Machine Learning Approaches, 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT).

[4] 1S. Karthick, 2D. Malathi, 3C.Arun, International Journal of Pure and Applied Mathematics, Weather Prediction Analysis Using Random Forest Algorithm.

[5] Y.Radhika and M.Shashi, Atmospheric Temperature Prediction using Support Vector Machine, International Journal of Computer Theory and Engineering, Vol. 1, No. 1, April 2009. [6] Sindhu P. Menon, Ramith Bharadwaj, Pooja Shetty, Prajwal Sanu, Sai Nagendra, Prediction of Temperature using Linear Regression, 2017 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT).

[7] Kaicheng Zhang, Akhil Guliani, Seda Ogrenci-Memik, Gokhan Memik, Kazutomo Yoshii, Rajesh Sankaran, Pete Beckman, Machine Learning-Based Temperature Prediction for Runtime Thermal Management across System Components, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTE, VOL. XX, NO. YY, MARCH 2016.

[8] Rajesh Kumar , Decision Tree for the Weather Forecasting, International Journal of Computer Applications (0975 – 8887) Volume 76– No.2, August 2013.

[9] Mark Holmstrom, Dylan Liu, Christopher Vo , Machine Learning Applied to Weather Forecasting, Stanford University (Dated: December 15, 2016).

[10] Ike Sri Rahayu, Esmeralda C Djamal, Ridwan Ilyas, Abdul Talib Bon, Daily Temperature Prediction Using Recurrent Neural Networks and Long-Short Term Memory, IEOM Society International, August 10 - 14, 2020.

[11] S. Salcedo-Sanz1 · R. C. Deo2 · L. Carro-Calvo1 · B. Saavedra-Moreno, Monthly prediction of air temperature in Australia and New Zealand with machine learning algorithms, Research Gate, 10 May 2015.

[12] UJJAL PROTIM DUTTA, PARTHA PRATIM GOGOI, PARTHA PRATIM SENGUPTA, PARAG PHUKON, Time Series Analysis of Temperature and Rainfall in the Brahmaputra Basin, Assam, Research Gate, 23 June 2019.