

A project report on

Vellore Rainfall Forecasting & NLP Dashboard

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology
in

Information Technology

by

Om Hemantkumar Purohit (17BIT0368)



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SITE

May 2021

DECLARATION

I here by declare that the thesis entitled “Vellore Rainfall Forecasting and NLP Dashboard” submitted by me, for the award of the degree of *Bachelor of Technology in Information Technology* to VIT is a record of bonafide work carried out by me under the supervision of Guide Name. I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Om Purohit

Date: 24-May-2021

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “**NLP Dashboard and Rainfall Forecasting**” submitted by **Purohit Om Hemantkumar 17BIT0368 SITE** VIT, for the award of the degree of *Bachelor of Technology in Information Technology* to VIT is a record of bonafide work carried out by him/her under my supervision. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The Project report fulfil the requirements and regulations of VIT and in my opinion meets the necessary standards for submission.

Signature of the Guide

**Signature of the
HoD**

Internal Examiner

External Examiner



CERTIFICATE OF INTERNSHIP

This is to certify that

Om Purohit

has successfully completed his internship at

Twimbit (R&D Division) from **August 01,2020** till **May 1, 2021** as a
Data Science Intern

He has worked proficiently and sincerely on plethora of projects such as:

1. NLP Dashboard with DashPlotly
2. Implemented Topic clustering, Semantic search and Summarization
3. Time-series forecasting
4. Scraping Dynamic websites
5. Graph Database creation and management using Neo4j
6. Creating and managing chromeextensions
7. Visualisations using D3.js

I personally thank him for his work and ensure anyone to have him for his exceptional talent and willpower.

Aman Sharma
CTO Twimbit

Place : **Noida**
Date : **29 April 2021**

Twimbit Research India Pvt. Ltd., Nukleus Co-working,
I.T. Tower, Plot 29, Sector 142, Noida
reachus@twimbit.com
twimbit.com

VELLORE RAINFALL FORECASTING ABSTRACT

Rainfall Forecast will be done on the basis of simple factors such as the date in our time series data. We will even perform some feature engineering and generate new features and try to approach this time series problem as a regular regression problem, obviously with the help of newly generated features that remove any autocorrelation present in the data. etc. On the basis of the available dataset on Vellore, which is a Time-series data ranging from year 2010 to 2019. We are going to perform forecasting techniques such as ARIMA, Exponential Smoothing. We will also create neural networks and apply LSTM as it is a well known fact that LSTM can handle be used to model on Time-series data. Lastly we will see if the problem statement at hand is a regression problem or not? We will also look at regression techniques and how we can manipulate our dataset and remove the time series component in order to perform regression techniques. We will compare the accuracy using a test dataset and decide on which model to use for our website, which will be created on finalizing the model on the basis of not only test accuracy, but how the predictions are being forecasted and is the model able to predict both the higher order and lower order of the forecast.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr.KATHIRAVAN S,Associate Professor Grade 1, SITE, Vellore Institute of Technology, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine learning and AI. I would like to express my gratitude to Dr.G.VISWANATHAN, Dr. SEKAR VISWANATHAN, Dr. RAMBABU KODALI, Dr. NARAYANAN S., and Dr. BALAKRUSHNA TRIPATHY, SITE for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to HOD Dr. USHA DEVI G, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Date: 19-May-21

Purohit Om Hemantkumar

CONTENTS

Declaration.....	2
Certificate.....	3
Internship Certificate.....	4
Abstract.....	5
Acknowledgment.....	6
Contents.....	7
1) Introduction	11
2)Literature Survey And Design	12
2.1)Modules.....	13
2.2) Database.....	13
2.3)Hardware Requirements.....	13
2.4) Software Requirements.....	13
3)Dataset	
3.1)Dataset main sheet.....	14
3.2) Dataset month sheet.....	14
3.3) Dataset yearwise.....	15
4)Preprocessing and cleaning data	
4.1) Collecting and Aggregation.....	16
4.2) Replacing NAN and irregular values.....	17
5)Exploratory Data Analysis	
5.1) Interactive Plot with district and year.....	18
6)Detecting trend and seasonality in Time series data	
6.1) Trend.....	21
6.2) Seasonality.....	21

6.3) Is our time series data stationary.....	21
6.4) Augmented Dickey Fuller test.....	22
6.5) ADF test 2.....	24
7)The ARIMA model	
7.1) Autoregressive.....	25
7.2)Moving Average.....	25
7.3) Test Accuracy.....	26
8)Holt-Winter Exponential smoothing model	
8.1) Test Accuracy.....	28
9)LSTM	
9.1) Basic Structure.....	29
9.2) LSTM Gates.....	30
9.3) LSTM Keras code.....	31
9.4) Test Accuracy.....	32
10) Feature Engineering & XGBoost	
10.1) Problem with our dataset.....	33
10.2)Transforming timeseries to regular dataset.....	33
10.3) Linear Regression Baseline.....	34
11)Support Vector Regressor	
11.1) Understanding SVM first.....	35
11.2) Kernel	35
11.3) Hyperplane.....	35
11.4) Decision Boundary.....	35
11.5) Test Accuracy.....	36
12)XGBoost Regressor	
12.1) About Gradient Boosting.....	37
12.2) Test Accuracy.....	37
12.3) Final Table.....	38

13)Website Development

13.1) Tech Stack.....	39
13.2) HTML.....	39
13.3) Css.....	50
13.4) Index.js.....	65
13.5) some_funcs.py.....	79
13.6) app.py.....	82
13.7) Website Images & important graphs.....	86

14)Internship Project Abstract..... 91

15) Literature Survey& Design 92

15.1) Modules.....	93
15.2) Database.....	93
15.3) Hardware requirement.....	93
15.4) Software requirement.....	93

Conclusion..... 94

References..... 95

References(Internship)..... 96

List of Tables

12.3) Final Table.....	38
------------------------	----

List of Images

3.1) Dataset main sheet.....	14
3.2) Dataset month sheet.....	14
3.3) Dataset year wise.....	15
4.2)Replacing irregular and NAN values.....	17
5.1) Interactive plot with year and district selectors.....	18
6.3) Is our time series data stationary.....	21
6.4) Augmented Dickey Fuller test.....	22

6.5) ADF test 2.....	24
7.2)Moving Average.....	25
7.3) Test Accuracy.....	26
8.1) Test Accuracy.....	28
9.1) Basic Structure.....	29
9.2) LSTM Gates.....	30
10.2)Transforming timeseries to regular dataset.....	33
10.3) Linear Regression Baseline.....	34
11.4) Decision Boundary.....	35
11.5) Test Accuracy.....	36
12.1) About Gradient Boosting.....	37
12.2) Test Accuracy.....	37
13.7)Website Images & important graphs.....	86

1)Introduction

The biggest problem we face nowadays is not knowing whether it is going to rain or not. Imagine planning for something big, and all your plan gets canceled just because it starts raining. Sounds pretty irritating, doesn't it? Well, now we have a solution to this problem and we call it the Rain Predictor. I have taken data from Dr. Kathiravan sir in unprocessed format. The dataset is specifically time series data in nature, as only the date and rainfall measure in millimeter are provided from year 2010 to 2019. The challenge is present in the data itself, as the climate of Vellore does not suit heavy rainfall or any pattern, even during the monsoon season. Hence, testing an array of different models is performed. These models range from the old and ancient forecasting methodologies such as ARIMA and Exponential smoothing, to newer deep learning approaches such as LSTM. Furthermore, we will not narrow our mindset nor our data to Time series only, as given the data has sparse values and a larger contribution of randomness than seasonality. In acknowledging so, proper feature engineering will also be applied such that we are able to apply regression techniques on our new transformed dataset. All in all, Rain Predictor is a software which will help us predict whether it is going to rain on a particular day or not and provide us the quantity of rain in millimeter. We will compare the accuracy using a test dataset and decide on which model to use for our website, which will be created on finalizing the model on the basis of not only test accuracy, but how the predictions are being forecasted and is the model able to predict both the higher order and lower order of the forecast.

2)Literature Survey and Design

<u>Title</u>	<u>Dataset</u>	<u>Abstract</u>	<u>Parameters</u>	<u>Advantage</u>
Forecasting Monthly Precipitation In Sylhet City Using Arima Model	rainfall data from 1980 to 2010 of Sylhet station w	Arima model was used in the forecast by first identifying and performing diagnostic checks for stationary before applying ARIMA	ARIMA(1,1,1), autoregression of 1 and lag of 1 was used	95% confidence interval
Using Historical Precipitation Patterns to Forecast Daily Extremes of Rainfall for the Coming Decades in Naples (Italy)	An annual series of daily maximum rainfall spanning the period between 1866 and 2016 was used	Exponential smoothing used for the reason that data stretching very far almost infinitely can get their contribution from it	Delta was used as the basic smoothing parameter which was calculated on training the data	ES is better for larger and historical data than ARIMA, moreover it can detect patterns easily
An artificial neural network model for rainfall forecasting in Bangkok, Thailand	4 years of hourly data from 75 rain gauge stations	The developed ANN model is being applied for real time rainfall forecasting and flood management in Bangkok, Thailand. Aimed at providing forecasts in a near real time schedule.	relative humidity, air pressure, wet bulb temperature and cloudiness	ANN forecasts have superiority over the ones obtained by the persistent model. Rainfall forecasts for Bangkok from 1 to 3 h ahead were highly satisfactory
Rainfall Forecast Model Based on the TabNet Model	5 years of meteorological data from 26 stations in the Beijing–Tianjin–Hebei region of China	this study proposes a rainfall forecast model based on the Attentive Interpretable Tabular Learning neural network (TabNet). This study used self-supervised learning to help the TabNet model speed up convergence and maintain stability. Also used feature engineering for uncertainty caused by seasonality	For tabular data sets, ensemble tree models are still mainly used. In many data-mining competitions, Xgboost and Lightgbm have been widely used. Learning rate of 0.01, layers used is 8.	Combines the advantages of decision trees and neural networks
Evidence for Using Lagged Climate Indices to Forecast Australian Seasonal Rainfall	Australian rainfall data from 1950–2009.	Climate indices were used and some of them even have lagged relationships, which is why Bayesian Join probability is used.	By using the distribution of model parameters inferred from the data record after excluding the event t, the cross-validation predictive density for event t is derived.	Using a lot of indices the model can be trained on a selective indices and hence has a better chance to detect patterns

2.1) Modules:

- Data Aggregator(Excel files)
- Preprocessing data
- EDA(Exploratory Data Analysis)
- Detect and remove Trend, Seasonality as per Time Series standards
- Feature Engineering
- Final Prediction using different algorithms
- Creating web app with proper backend API
- Using model with greater training and testing accuracy
- Deploying it on Heroku

2.2) Database: Excel Files, Pandas csv

2.3) Hardware requirements: NONE

2.4) Software requirements:

- Pandas
- Numpy and core preprocessing libraries
- Statistical modelling library
- Scikit-learn
- Keras
- Plotly and matplotlib
- Xgboost
- HTML,CSS and Javascript
- Flask
- D3.js

3) The Dataset

We have Excel files ranging from the year 2010- 2019, below picture is the Raw unprocessed data! The below Image is for the year 2010 Annual Rainfall in millimeter.

3.1) Dataset main sheet











VELLORE DISTRICT RAINFALL ANNUAL REPORT- 2010														
Date	Alangayam	Ambur	Arakkonam	Arcot	Gudiyatham	Kaveripakkam	Melalathur	Sholingur	Tirupattur	Vaniyambad	Vellore	Wallajah	Total	Avg
January	-	-	-	-	14.00	NF	9.20	20.00	3.50	-	44.80	-	91.50	7.63
February	-	-	-	-	-	NF	-	-	-	-	-	-	0.00	0.00
March	-	-	-	-	-	NF	-	-	-	-	-	-	0.00	0.00
April	22.00	-	-	-	73.80	NF	47.60	30.00	12.20	-	5.50	6.00	197.10	16.43
May	258.40	32.30	139.60	27.80	85.00	NF	80.20	92.00	125.80	200.00	75.80	51.20	1168.10	97.34
June	67.00	75.00	70.10	34.80	164.00	NF	192.00	14.00	138.00	128.00	113.50	45.00	1041.40	86.78
July	66.00	17.00	267.00	111.40	151.20	NF	125.90	124.00	91.50	66.00	134.90	133.00	1287.90	107.33
August	52.80	65.40	174.90	97.80	56.60	NF	86.60	166.10	132.40	77.00	119.80	63.90	1093.30	91.11
September	130.00	48.00	134.20	161.60	155.80	NF	158.90	84.00	116.10	82.10	162.70	99.00	1332.40	111.03
October	85.70	16.00	64.30	67.40	113.60	39.20	115.80	116.00	114.70	150.20	102.20	60.00	1045.10	87.09
November	285.00	213.00	272.20	139.30	254.80	287.60	263.60	241.00	204.50	251.20	224.00	153.00	2789.20	232.43
December	48.40	60.00	110.00	44.28	80.80	115.80	97.00	109.40	29.10	60.40	158.40	83.20	996.78	83.07
Total	1015.30	526.70	1232.30	684.38	1149.60	442.60	1176.80	996.50	967.80	1014.90	1141.60	694.30	11042.78	920.23
Season Report														
Season	Alangayam	Ambur	Arakkonam	Arcot	Gudiyatham	Kaveripakkam	Melalathur	Sholingur	Tirupattur	Vaniyambad	Vellore	Wallajah	Total	Avg
Winter	0.00	0.00	0.00	0.00	14.00	0.00	9.20	20.00	3.50	0.00	44.80	0.00	91.50	7.63
Summer	280.40	32.30	139.60	27.80	158.80	0.00	127.80	122.00	138.00	200.00	81.30	57.20	1365.20	113.77
SWM	315.80	205.40	646.20	405.60	527.60	0.00	563.40	388.10	478.00	353.10	530.90	340.90	4755.00	396.25
NEM	419.10	289.00	446.50	250.98	449.20	442.60	476.40	466.40	348.30	461.80	484.60	296.20	4831.08	402.59
District Total			11042.78											
District Avg			920.23											

3.2) Dataset month sheet

The below image is the data for Daily rainfall in mm of month of January 2010

[illegible]

3.3) Dataset yearwise

Name	Date modified	Type	Size
 2010-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	61 KB
 2011-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	64 KB
 2012-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	63 KB
 2013-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	65 KB
 2014-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	67 KB
 2015-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	69 KB
 2016-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	61 KB
 2017-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	63 KB
 2018-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	68 KB
 2019-Vellore District Rainfall	04-10-2020 22:53	XLSX Worksheet	74 KB

4) Preprocessing and Cleaning Data

4.1) Collecting and aggregation

```
import pandas as pd
```

```
# %%  
df10=pd.read_excel("rainfalldata/2010-VelloreDistrict  
Rainfall.xlsx",sheet_name=[0,1,2,3,4,5,6,7,8,9,10,11,12])  
df11=pd.read_excel("rainfalldata/2011-Vellore District Rainfall.xlsx")  
df12=pd.read_excel("rainfalldata/2012-Vellore District Rainfall.xlsx")
```

```
# %%  
df10[1]
```

```
# %%  
df10[0].columns=df10[0].iloc[1,0:]  
df10[0]
```

```
# %%  
newcolumns=df10[0].columns[1:-2]
```

```
# %%  
newcolumns
```

```
# %%  
tempdata={"Date":[],"District":[],"Rain":[]}
```

```
# %%
for year in range(10,20):
    df10=pd.read_excel("rainfalldata/20"+str(year)+"-Vellore
Rainfall.xlsx",sheet_name=[0,1,2,3,4,5,6,7,8,9,10,11,12])

    for i in range(0,12):
        df10[i].columns=df10[i].iloc[1,0:]
        for row,frame in df10[i].iterrows():
            if row<2 or row>(df10[i].shape[0]-5):
                continue
            else:

                newcolumns=df10[i].columns[1:-2]
                for newcol in newcolumns:
                    tempdata["Date"].append(frame["Date"])
                    tempdata["District"].append(newcol)
                    tempdata["Rain"].append(frame[newcol])

# %%
BigDF=pd.DataFrame(tempdata)

# %%
BigDF.to_csv("combined.csv",index=False)
Combining the Data from 2010 to 2019, we create a final dataset called combined.csv
```

	A	B	C
1	Date	District	Rain
2	01-01-2010	Alangayam	-
3	01-01-2010	Ambur	-
4	01-01-2010	Arakkonam	-
5	01-01-2010	Arcot	-
6	01-01-2010	Gudiyatham	-
7	01-01-2010	Melalathur	-
8	01-01-2010	Sholingur	-

4.2) Replacing irregular and NAN values

Dataset Shape ==> 46293*3

Replacing “-” with 0.0 float values, indicating that no rain had occurred.

```
def convertToFloat(x):
    x=x.strip()
    if x=="-" or x=="NR" or x==" " or x=="." or x=="'" or x=="' ":
        return 0
    else:
        thevar[0]=x
        return float(x)
```

	Date	District	Rain	rain
0	2010-01-01	Alangayam	-	0.0
1	2010-01-01	Ambur	-	0.0
2	2010-01-01	Arakkonam	-	0.0
3	2010-01-01	Arcot	-	0.0
4	2010-01-01	Gudiyatham	-	0.0
...
6287	2019-12-31	Natrampalli	-	0.0
6288	2019-12-31	TCS Mill	-	0.0
6289	2019-12-31	ACS Mill	-	0.0
6290	2019-12-31	VCS Mill	-	0.0
6291	2019-12-31	Ponnai	-	0.0

5) Exploratory Data Analysis

Using Altair graphing library, we can create **interactive graphs**. For example one can easily use these graphs to fetch the rainfall in mm of a given input of (month, year). It features rich sliders and dropdown menu as widgets to make a tiny UI for better user experience.

5.1) Interactive plot with year and district selectors

```
# %%
get_ipython().system('pip install altair vega_datasets')

# %%
BigDF=pd.read_csv("combined.csv")

# %%
from datetime import datetime

def getYear(x):
    print(x)

    return int((datetime.strptime(str(x),"%Y-%m-%d").year))
def getMonth(x):
    return int(datetime.strptime(str(x),"%Y-%m-%d").month)
def getDay(x):
    return int(datetime.strptime(str(x),"%Y-%m-%d").day)

# %%
```

```

# %%
BigDF["year"]=BigDF["Date"].apply(getYear)
BigDF["month"]=BigDF["Date"].apply(getMonth)
BigDF["day"]=BigDF["Date"].apply(getDay)

# %%
def ChangeRainToFloat(x):
    try:
        return float(x)
    except:
        return 0

BigDF["Rain"]=BigDF["Rain"].apply(ChangeRainToFloat)

# %%
"Total" in BigDF.Date.unique()

# %%
BigDF

# %%
df_grouped=BigDF.groupby(["District","year","month"]).sum().reset_index()
df_grouped

# %%
df_grouped["District"].unique()

# %%

import altair as alt
slider = alt.binding_range(min=2010, max=2019, step=1)
select_year= alt.selection_single(name='year', fields=['year'],
                                  bind=slider, init={'year': 2011})
slider2 = alt.binding_select(options=df_grouped["District"].unique())

select_district= alt.selection_single(name="District", fields=['District'],
                                      bind=slider2, init={'District': "Vellore"})
alt.Chart(df_grouped).mark_bar().encode(
    x="month",
    y="Rain",
    row="District",

```

```

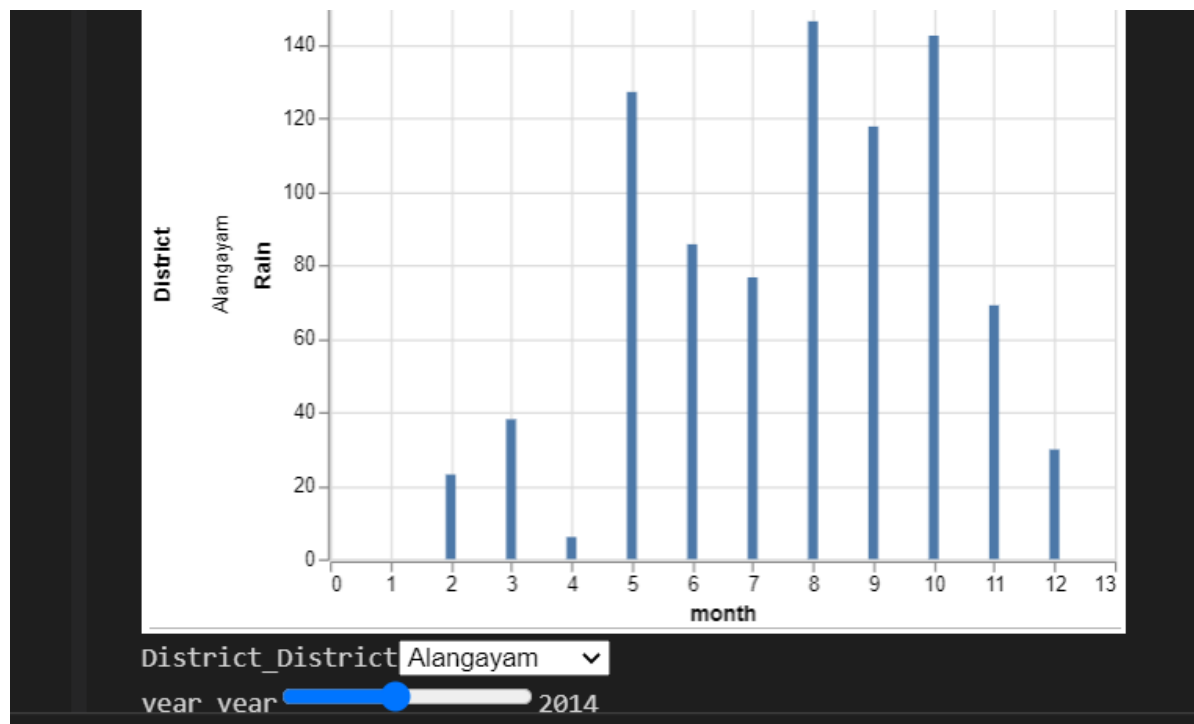
        tooltip=["Rain","month"]
    ).add_selection(select_year,select_district).transform_filter(select_year).transform_filter(select_district)

```

```

# %%
BigDF[(BigDF["year"]==2010)      &      (BigDF["District"]=="Kaveripakkam")&
(BigDF["month"]==11) ]

```



6) Detecting Trend and Seasonality in the Time-Series Data

6.1)Trend

The trend is the component of a time series that represents variations of low frequency in a time series, the high and medium frequency fluctuations having been filtered out. This component can be viewed as those variations with a period longer than a chosen threshold (usually 8 years is considered as the maximum length of the business cycle)

6.2)Seasonality

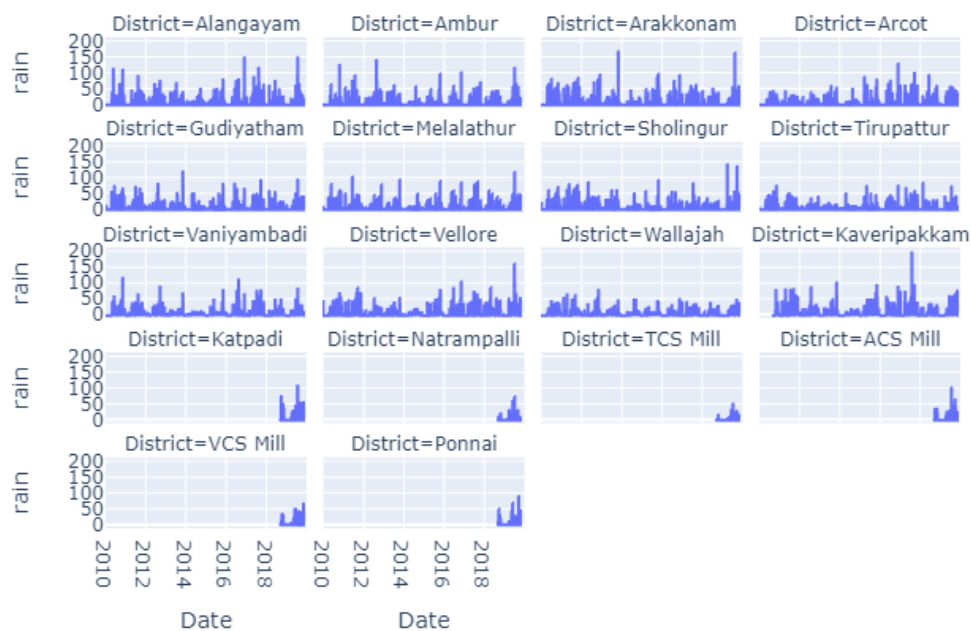
Seasonal variation, or seasonality, are cycles that repeat regularly over time. A repeating pattern within each year is known as seasonal variation, although the term is applied more generally to repeating patterns within any fixed period.

Why we need to check for seasonality and trend? What does it mean a time-series to be stationary?

A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time. In general, a stationary time series will have no predictable patterns in the long-term. Time plots will show the series to be roughly horizontal (although some cyclic behaviour is possible), with constant variance.

6.3) Is our Time Series Data Stationary ?

```
import plotly.express as px
fig = px.line(df, x="Date", y="rain", facet_col="District", facet_col_wrap=4)
```



Although detecting seasonality from visualizations is not practical, it helps in detecting if there are any trends in the data. From the above visualization we can easily conclude that there is **no increasing or decreasing trend**.

6.4) Augmented Dickey Fuller Test

The Augmented Dickey Fuller Test (ADF) is unit root test for stationarity. Unit roots can cause unpredictable results in your time series analysis.

- The [null hypothesis](#) for this test is that there is a unit root.
- The [alternate hypothesis](#) differs slightly according to which equation you're using. The basic alternate is that the time series is stationary (or trend-stationary).

Unit root tests can be used to determine if trending data should be first differenced or regressed on deterministic functions of time to render the data stationary. Moreover, economic and finance theory often suggests the existence of long-run equilibrium relationships among nonstationary time series variables.

$$y_t = D_t + z_t + \varepsilon_t$$

Dt-> Deterministic component

Zt-> Random(Stochastic component)

Et-> Stationary Error

We are only concerned with the result of AD-Fuller test to detect whether our series is stationary or not.

We perform the test below.

Using AutoLag

Lag in a time series data is displacement of the data by an order of **t**, where **t** is the assumed time period where seasonality occurs. We will be using AutoLag in our Augmented Dicky Fuller test, where the test will automatically detect a lag for testing.

```
from statsmodels.tsa.stattools import adfuller
```

```
def test_stationarity(timeseries):
```

```
    print('Results of Dickey-Fuller Test:')
```

```
    dftest = adfuller(timeseries)
```

```
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value',
```

```
    '#Lags Used','Number of Observations Used'])
```

```
    for key,value in dftest[4].items():
```

```
        dfoutput['Critical Value (%s)'%key] = value
```

```
    print(dfoutput)
```

```
test_stationarity(traindf["rain"])
```

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries)
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
test_stationarity(traindf["rain"])
```

```
Results of Dickey-Fuller Test:
Test Statistic      -1.527302e+01
p-value             4.688703e-28
#Lags Used          7.000000e+00
Number of Observations Used  2.914000e+03
Critical value (1%)  -3.432596e+00
Critical value (5%)  -2.862532e+00
Critical value (10%) -2.567298e+00
dtype: float64
```

From the above test the important results for us are the **Test Statistic** and **Critical Value(10%)**(this is the benchmark critical value) We observe that the critical value - **3.432** is greater than Test statistic **-15.273**. The lag automatically detected is **7**

$-15.273 < -3.432$

Test Statistic < Critical value

We reject null hypothesis and accept the alternate hypothesis.

The result is astonishing! The test is concluding that our data is already **stationary**! It is surprising as rainfall data should always contain **seasonality** and be **non-stationary**.

6.5)Using MaxLag=400 and AutoLag=None

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag=None, maxlag=400)
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
test_stationarity(traindf["rain"])
```

```
Results of Dickey-Fuller Test:
Test Statistic          -2.242520
p-value                 0.191176
#Lags Used              400.000000
Number of Observations Used  2521.000000
Critical Value (1%)      -3.432947
Critical Value (5%)      -2.862687
Critical Value (10%)     -2.567381
dtype: float64
```

From the above test the important results for us are the **Test Statistic** and **Critical Value(10%)**(this is the benchmark critical value) We observe that the critical value - **2.56** is greater than Test statistic **-3**.

$-2.24 < -2.56$

Test Statistic < Critical value

We accept the null hypothesis.

Using lag of 400 which is approximately a time period of 1 year, the test is able to detect the seasonality. In order to remove seasonality we use **Differencing** with interval of 400

```
def difference(dataset, interval=1):
```

```
    diff = list()
```

```
    for i in range(interval, len(dataset)):
```

```

    print(i)

    value = dataset[i] - dataset[i - interval]

    diff.append(value)

return diff

# invert differenced forecast

def inverse_difference(last_ob, value):

    return value + last_ob

```

```

# create a differenced series
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        print(i)
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return diff

# invert differenced forecast
def inverse_difference(last_ob, value):
    return value + last_ob

```

7)The ARIMA model

ARIMA stands for Auto Regressive Integrated Moving Average model. An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

7.1) Autoregressive

A statistical model is Autoregressive if it predicts the time series data from previous data. It seeks to use a lag to shift the time series data. A lag of k means it will predict the values using previous k terms of the dataset.

7.2) Moving Average

Moving average is a simple statistical measure to calculate averages in a moving window. It is also used in technical analysis of stock data, which is a well known time series data.

The formula is $ARIMA(x,y,z)$. x determines the Autoregression lag to be used, moving average is determined by a lag z and y is used for integration, i.e the differencing order. We already have performed differencing hence we initialise this to 0.

$ARIMA(7,0,1)$

```
from statsmodels.tsa.arima_model import ARIMA
# train_log_v=list(traindf["rain"].values)
# test_log_v=list(testdf["rain"].values)
predictions=list()
model = ARIMA(traindf_diff, order=(7,0,1))
# for t in range(len(test_log_v)):
#     # LAG-3, MOVING AVG-2
#     model_fit = model.fit(dis=0)
#     output = model_fit.forecast()
#     yhat = output[0]
#     predictions.append(yhat)
#     obs = test_log_v[t]
#     train_log_v.append(obs)
```

7.3) TEST ACCURACY

We will be using a common metric for measuring error of various models in the future. **MAE, mean absolute error** is a great choice compared to other metrics. As some metrics tend to divide the error by the true value, for the case of data containing 0's it can lead to error reaching **infinity**.

For MAE the error more close to 0.0 is better!

```
mean_absolute_error(testdf["rain"],inverted)
```

4.143719004261452

Mean Absolute Error-> 4.143

The error is large and **inefficient model**.

	orig	pred
Date		
2018-01-01	0.0	0.111796
2018-01-02	0.0	0.120688
2018-01-03	0.0	0.134962
2018-01-04	0.0	0.137757
2018-01-05	0.0	0.143894
...
2019-12-27	0.0	6.855810
2019-12-28	0.0	7.755810
2019-12-29	0.0	0.155810
2019-12-30	0.0	0.855810
2019-12-31	0.0	0.155810

It is obvious from the above image, the model is not able to predict rainfall of more than **1mm** accuracy.

8)The Holt-Winter Exponential Smoothing Model

Exponential smoothing was proposed in the late 1950s (Brown, 1959; Holt, 1957; Winters, 1960), and has motivated some of the most successful forecasting methods. Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1 - \alpha)y_{T-1} + \alpha(1 - \alpha)^2 y_{T-2} + \dots,$$

Here the alpha value is the smoothing parameter. The algorithm gives more preference to the nearest predecessor, and the further the predecessor, the less impact it has on the forecast.

Holt-Winters' additive method

The component form for the additive method is:

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\ \ell_t &= \alpha(y_t - s_{t-m}) + (1 - \alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1 - \beta^*)b_{t-1} \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1 - \gamma)s_{t-m},\end{aligned}$$

```
testexp=[]
traindf2=traindf.copy()
for i in range(len(testdf)):
    fit2=ExponentialSmoothing(traindf2,seasonal_periods=4).fit()
    testexp.append(fit2.forecast(1)[0])
    d={"rain":[fit2.forecast(1)[0]], "Date":[str(fit2.forecast(1).index.date[0])]}
    temp=pd.DataFrame(d)
    temp.index=temp.Date
    temp.pop("Date")
    traindf2=pd.concat([traindf2,temp])
```

```
mean_absolute_error(testdf["rain"],testexp)|

3.5206459947827082
```

8.1) TEST ACCURACY

Mean Absolute Error-> 3.5206

Although it is less than ARIMA model, the model is actually worse.

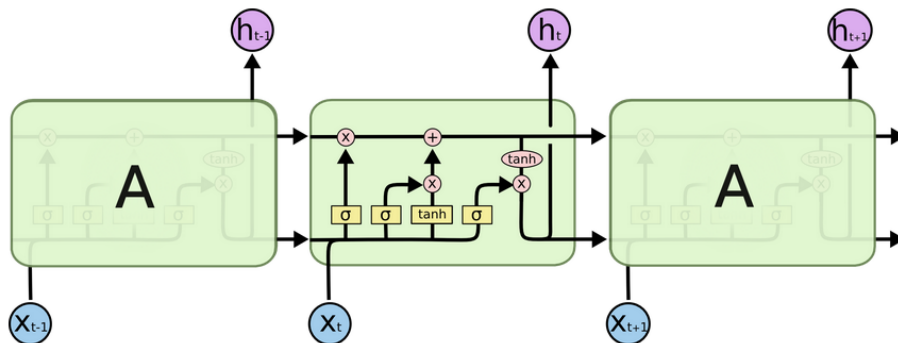
	orig	pred
Date		
2018-01-01	0.0	1.771350
2018-01-02	0.0	1.771350
2018-01-03	0.0	1.771350
2018-01-04	0.0	1.771350
2018-01-05	0.0	1.771349

Predicted values do not even exceed 2 mm!

9) LSTM-Long-Short Term Memory

9.1) Basic Structure

We enter the world of Deep learning to get better results using a well known implementation of a neural network for time series data, LSTM. It is an advancement of Recurrent neural networks which use forget gates to keep on the required information stored. It has been widely applied in Stock market technical analysis. The fully connected layers in the network are in fact cells in LSTM, which provide error flow to be smooth between gates. This allows us to bridge previous cells.



The repeating module in an LSTM contains four interacting layers.

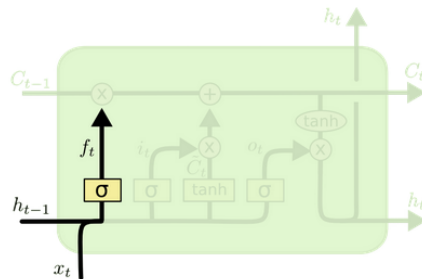
Pink circles -> denote pointwise operation

Yellow boxes -> denotes neural network layers

Black arrows-> carries vectors(Cell state->analogous to conveyor belt)

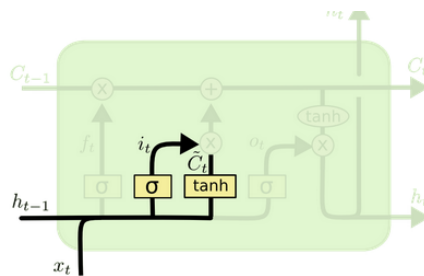
The cell state carries information seamlessly throughout cells. Gates such as the Sigmoid activation layer which usually outputs either a 0 or 1. a 0 will mean that no data can be passed to the pink circle, which is where the pointwise operation happens such as a dot product.

9.2) Gates in LSTM



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

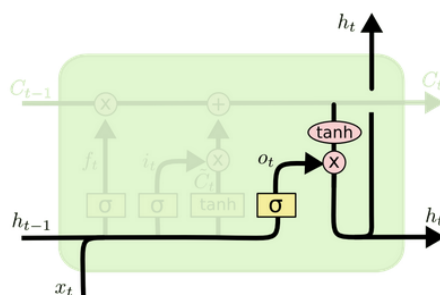
Here the previous cell output($h[t-1]$) and input($x[t]$) are concatenated and flow through the sigmoid gate which either outputs 0 or 1. If 0 is outputted then the information is rejected and not used, hence this gate is called **forget gate**.



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

We go over our next gate, which involves using the previous value we have forgotten to create a new candidate for cell state C_t . Using the tanh activation aids in creating this new vector and updating our cell state.



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

The last stage of the LSTM, is to decide what to output. Again the input goes through a sigmoid gate to decide what information to keep and a final tanh activation layer to convert the values in between -1 to 1.

Below is the implementation of our LSTM. I have decided to use look_back=30, as we will use past 30 values to predict the future.

9.3) Code for LSTM Using Keras

```
from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean_squared_error

from tensorflow import keras

def CreateDataset(dataset,lookback):

    DataX=[]

    DataY=[]

    for i in range(len(dataset)-lookback-1):

        DataX.append(dataset[i:(lookback+i),0])

        DataY.append(dataset[(lookback+i),0])

    return np.array(DataX),np.array(DataY)

trainsize=3287

testsize=365

train, test = dataset[0:trainsize,:], dataset[trainsize:len(dataset),:]

look_back=30

trainX,trainY=CreateDataset(train,look_back)

testX,testY=CreateDataset(test,look_back)

# reshape input to be [samples, time steps, features]

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```



```

model = Sequential()

opt = keras.optimizers.Adam()

model.add(keras.layers.Bidirectional(LSTM(4, input_shape=(1, look_back))))

model.add(Dense(1,activation="relu"))

model.compile(loss='mean_absolute_error', optimizer=opt)

model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)

trainPredict = model.predict(trainX)

testPredict = model.predict(testX)


from sklearn.metrics import mean_squared_error


ypred=testPredict.reshape(334)

ty=testY.reshape(334)

```

The training error produced is lower than previous models(mean absolute error) 1.9907

9.4) TEST ACCURACY

Mean Absolute Error(Test Dataset)->2.955

The problem in our forecast? Only 30 values out of 334 are Non-zero, hence our prediction is Sparse!

10) Feature Engineering and Xgboost gradient boosting

10.1) problems with our dataset

- The data is sparse, as many months pass in Vellore without a millimeter of rainfall.
- Seasonality should be present but is not present within a specific period, hence lag values differ from district to district and year to year. Indicating that the stochastic component(Z_t) must be high
- **You cannot apply regression techniques to time series data!**

The last point is problematic as it limits our modelling process to only statistical models and deep learning models. Regression cannot be applied to data which is **autocorrelated**.

10.2) Transforming timeseries to regular dataset

We generate new features from our dataset using feature engineering. A total of **12 new features** are to be generated, the first 3 being easily obtained from the date feature.

- Month
- Year
- Day
- Is_rain
- Previous1
- Previous2
- Previous3
- Previous4
- Previous5
- Previous6
- Previous7
- Previous8

Month, Day and Year are obtained from stripping the date column using datetime library. Whereas for **is_rain** column, we use a `get_season` function, which is used to classify the month of the input variable is seasonal or not. It ranges from [0,1,2] its mapping as follows

- Dec-April=> 0
- May-September=>2
- October-November=>1

Now for the **previousN** column, we need to create a dictionary called **DayMon**. DayMon's keys are

Key: "District_month_day"

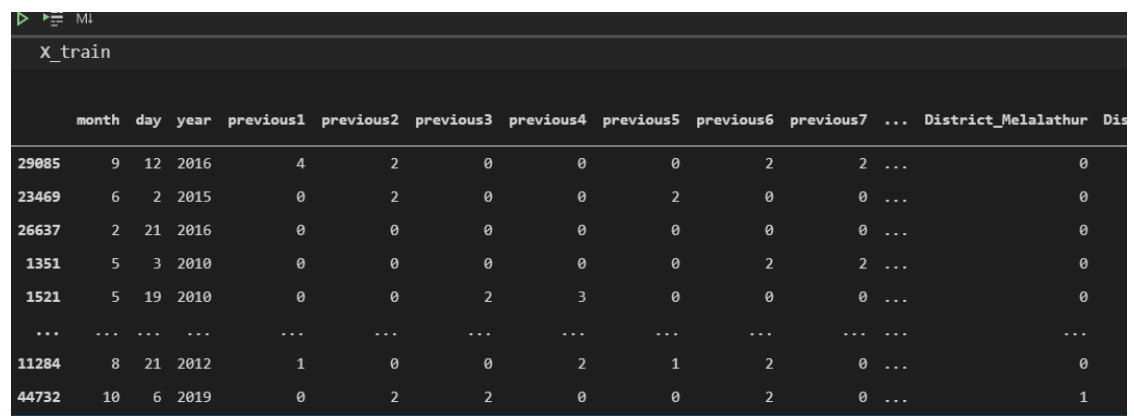
The value of this key is a list of all the rainfall measured in mm over the years for the particular district, month and day. We also have a function **RainCompress**, that takes the value of rainfall and converts it into a categorical value [0,1,2,3,4,5]

- >100 =>5
- >50=>4
- >25=>3
- >2=>2
- >0 =>1
- Otherwise 0

Baiscally the PreviousN column is summarized as follows:

What was the rainfall measured **N years** back for **District x, month y and day z(x_y_z)** and compress it using **CompressRainfall**, to get a categorical value.

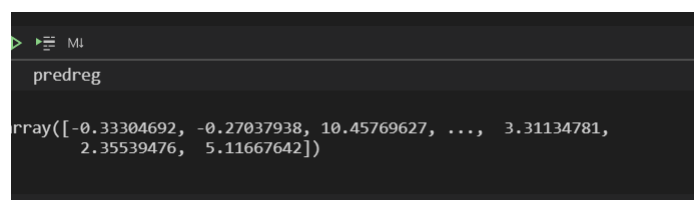
The advantage in recreating our dataset in order to fit regression models, is that a **single model** will be enough to forecast **any district**. It will consider district a separate feature and save us time and resources to dedicate distinct models for each district. We use **one-hot encoding** to convert the categorical district values to binary values of 0 and 1.



	month	day	year	previous1	previous2	previous3	previous4	previous5	previous6	previous7	...	District_Melalathur	District_Melalathur_1
29085	9	12	2016	4	2	0	0	0	2	2	...	0	0
23469	6	2	2015	0	2	0	0	2	0	0	...	0	0
26637	2	21	2016	0	0	0	0	0	0	0	...	0	0
1351	5	3	2010	0	0	0	0	0	2	2	...	0	0
1521	5	19	2010	0	0	2	3	0	0	0	...	0	0
...
11284	8	21	2012	1	0	0	2	1	2	0	...	0	0
44732	10	6	2019	0	2	2	0	0	2	0	...	1	1

10.3) Linear Regression Baseline

In order to apply various regression based techniques, we need to start with the basics. We apply Linear regression to create a baseline. **Mean Absolute Error on test dataset is 3.527** Although it seems an acceptable error for a baseline model, the predictions are not acceptable for our usecase, as the forecast contains **negative values!**



```

array([-0.33304692, -0.27037938, 10.45769627, ..., 3.31134781,
       2.35539476, 5.11667642])

```

11) Support vector Regression Model

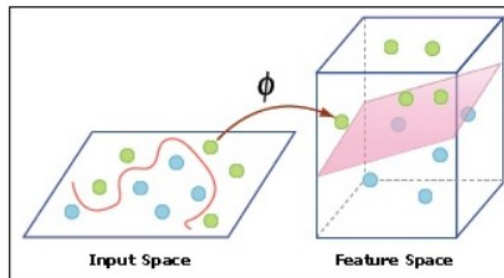
11.1) Understanding SVM first

Before understanding SVR, it is important to understand SVM, which is the predecessor of SVR. SVM(Support vector machines) is a great model used to solve **classification** problems. It is even able to classify data which is not linearly separable. The intuition behind the algorithm is to raise the dimensions of the data and find a hyperplane that separates the data into classes. Below are the key parameters used in SVM

11.2) Kernel: It is a function which helps save our computational resources in finding the right hyperplane. Normally transforming the data into higher dimensions becomes computationally intensive proportional to the dimension being raised.

11.3) Hyperplane: The line distinguishing between the classes in our model and aiding further classification of data being inputted. For SVR, this same hyperplane will be forecasting continuous values.

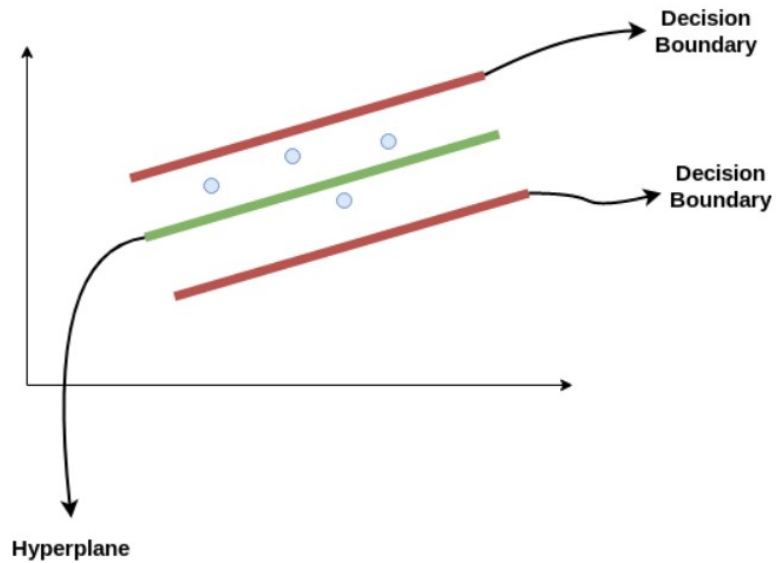
11.4) Decision Boundary: it is a line which has two sides, one being the positive side and other being the negative side. Data point within the positive side of a hyperplane belong to a specific class and on the negative side, the data points do not belong to that particular class.



SVR utilizes almost the same algorithm, except that instead of classifying data points, we use the decision boundary lines and hyperplane to find the best fit line for our regression usecase. We assume a distance “c”(epsilon) between the hyperplane and decision boundary line.

Hyperplane: $y=wx+b$

Decision boundary lines: $wx+b= c$, $wx+b= -c$



Now that we are clear in the theory behind SVR, it is quite simple to implement it using Scikit-learn library.

```
from sklearn.svm import LinearSVR
```

```
eps = 5
```

```
svr = LinearSVR(epsilon=eps, C=0.01, fit_intercept=True)
```

11.5) TEST ACCURACY

Mean Absolute Error for test data->6.92

```

> MI
from sklearn.svm import LinearSVR
eps = 5
np.random.seed(2)
svr = LinearSVR(epsilon=eps, C=0.1,)
svr.fit(X_train,y_train)
predsvr=svr.predict(X_test)
for i in range(len(predsvr)):
    if predsvr[i]<1:
        predsvr[i]=0
mean_absolute_error(y_test,predsvr)

C:\Users\om purohit\anaconda3\lib\site-packages\sklearn\svm\
warnings.warn("Liblinear failed to converge, increase "
6.92344100826768

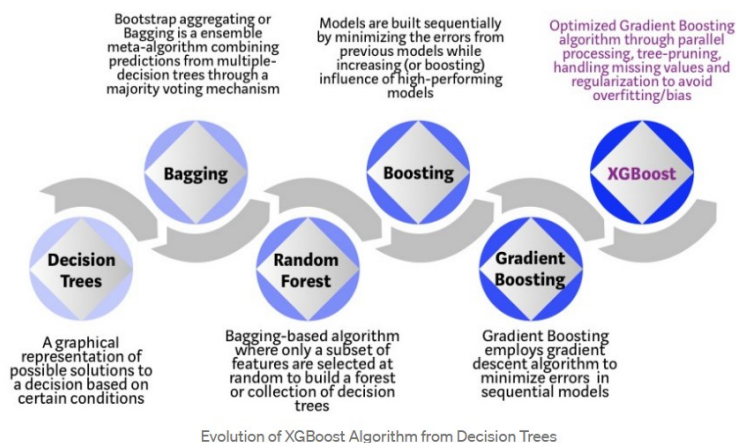
```

We are still not happy with our accuracy hence we move to a better model.

12) XGBOOST REGRESSOR

12.1) About Gradient Boosting

Gradient boosting refers to an ensemble technique, where boosting implies the models are added **sequentially** in the ensemble. It uses multiple decision trees and weights are assigned to these trees to quantify their say to the final output. The trainable parameters are learned through gradient descent, hence the name gradient boosting.



XGBoost enhances Gradient boosting algorithm by reducing the training time significantly via hardware optimizations. XGBoost also has **sparse data** handling capabilities, which is perfect for our dataset, it does not simply replace zeroes with the averages, but by “**learning**” best fit values for sparse data.

12.2) TEST ACCURACY

We implement XGBoost using its open-source library, with hyperparameters `n_estimators` 3(the number of trees) and `max_depth`=35 to see how much depth a tree is allowed to go.

```
xgb_r = xg.XGBRegressor(objective ='reg:linear',
```

```
    n_estimators = 3, seed = 123,max_depth=35,eta=0.7)
```

Mean Absolute Error of test data -> 1.57

```

d={"orig":y_test,"pred_test":pred}
pred_df=pd.DataFrame(d)

pred_df[10:40].values

array([[ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 1.2        ,  0.         ],
       [ 0.         ,  0.         ],
       [ 3.6        , 11.59256268],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [13.         ,  3.37072515],
       [ 3.2        ,  2.44795179],
       [45.4        ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [49.         , 22.95454025],
       [ 3.         ,  4.80077457],
       [14.8        ,  3.11541271],
       [ 2.9        ,  0.         ],
       [ 1.2        ,  0.         ],
       [ 0.         ,  0.         ],
       [40.2        , 19.51211929],
       [ 0.         ,  0.         ],
       [ 6.9        ,  3.74354482],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ],
       [ 0.         ,  0.         ]

```

The model has learned to predict better than any other models and even for larger predictions it comes closer. We will utilize this model along with other models in upcoming website

12.3) Final Table

Model	MAE	RMSE	Training Time in Seconds
ARIMA model	4.14	12.02	1.11
Exponential Smoothing	3.52	9.59	0.02
LSTM	2.955	12.07	380.89
Support Vector Regression	6.92	9.33	2.55
Linear Regression	3.52	8.21	0.02
XGBoost Regression	1.57	5.57	0.43

13) WEBSITE DEVELOPMENT

13.1) Tech Stack

- Flask
- Pandas
- HTML, CSS, Javascript
- D3.js
- Deployment Heroku(Free tier)
- Website Link-> <https://rain-forecast.herokuapp.com/>

13.2) HTML

<html >

<head><title>Vellore RainFall Dashboard and Forecast by Om</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="/static/css/mycss.css">

<link

rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"

/>

</head>

<body class="white-bkg">

<div class="attribute">

Attribution 🙌 :

Icons made by smalllikeart , Freepik from www.flaticon.com & DotFix , <a


```

href="https://freeicons.io/profile/3">freeicons</a>,<a
href="https://freeicons.io/profile/3335">visuallanguage</a>on<a
href="https://freeicons.io">freeicons.io</a></div>

```

```
<div class="Navbar">
```

```
<div class="wand-icon">
```

```

<span >Forecast</span> 

```

```
</div>
```

```
<div class="graph-icon">
```

```

<span >Visualizations</span> 

```

```
</div>
```

```
<div class="notebook-icon">
```

```

<span >Home</span> 

```

```
</div>
```

```
</div>
```

```
<div class="tab1">
```

```
<div class="small-flex">
```

```

```

<div id="forecastOutput">? mm</div>

</div>

<form class="form-flex" name="singleform" onsubmit="LoadData();return false"
method="POST">

<input type="date" placeholder="Input Date" id="DateInput">

<select id="DistrictInput">

<option value=0>ACS Mill</option>

<option value=1>Alangayam</option>

<option value=2 selected>Ambur</option>

<option value=3>Arakkonam</option>

<option value=4>Arcot</option>

<option value=5>Gudiyatham</option>

<option value=6>Katpadi</option>

<option value=7>Kaveripakkam</option>

<option value=8>Melalathur</option>

<option value=9>Natrampalli</option>

<option value=10>Ponnai</option>

<option value=11>Sholingur</option>

<option value=12>TCS Mill</option>

<option value=13>Tirupattur</option>

<option value=14>VCS Mill</option>

<option value=15>Vaniyambadi</option>

<option value=16>Vellore</option>

<option value=17>Wallajah</option>

</select>

<button id="DateSubmit" class="buttonclass">submit</button>

</form>

</div>

<div class="tab2 tabinvisible">

<div class="row">

<div>

<select name="" id="graph_district">

<option value=0>ACS Mill</option>

<option value=1>Alangayam</option>

<option value=2>Ambur</option>

<option value=3>Arakkonam</option>

<option value=4 selected>Arcot</option>

<option value=5>Gudiyatham</option>

<option value=6>Katpadi</option>

<option value=7>Kaveripakkam</option>

<option value=8>Melalathur</option>

<option value=9>Natrampalli</option>

<option value=10>Ponnai</option>

<option value=11>Sholingur</option>

<option value=12>TCS Mill</option>

```
<option value=13>Tirupattur</option>
<option value=14>VCS Mill</option>
<option value=15>Vaniyambadi</option>
<option value=16>Vellore</option>
<option value=17>Wallajah</option>
</select>
```

```
<select name="" id="graph_year">
  <option value=2010>2010</option>
  <option value=2011>2011</option>
  <option value=2012>2012</option>
  <option value=2013>2013</option>
  <option value=2014>2014</option>
  <option value=2015>2015</option>
  <option value=2016>2016</option>
  <option value=2017>2017</option>
  <option value=2018>2018</option>
  <option value=2019>2019</option>
```

```
</select>
```

```
<button class="buttonclass" id="graph_1_submit">submit</button>
```

```
</div>
```

```
<div class="rowsvg">
```

```
<svg height=300 width=300 id="BasicGraph"></svg>
```

```
<label> Monthly Average Rainfall</label>
```

</div>

<!--

#####

-->

<div>

<select name="" id="graph_district_2">

<option value=0>ACS Mill</option>

<option value=1>Alangayam</option>

<option value=2>Ambur</option>

<option value=3>Arakkonam</option>

<option value=4 selected>Arcot</option>

<option value=5>Gudiyatham</option>

<option value=6>Katpadi</option>

<option value=7>Kaveripakkam</option>

<option value=8>Melalathur</option>

<option value=9>Natrampalli</option>

<option value=10>Ponnai</option>

<option value=11>Sholingur</option>

<option value=12>TCS Mill</option>

<option value=13>Tirupattur</option>

<option value=14>VCS Mill</option>

<option value=15>Vaniyambadi</option>

<option value=16>Vellore</option>

<option value=17>Wallajah</option>

</select>

```

        <button class="buttonclass" id="graph_2_submit">submit</button>

</div>

<div class="rowsvg">

    <svg height=300 width=300 id="Graph2"></svg>

    <label> Yearly Average Rainfall</label>

</div>

</div>

<!-- row 2#####row 2 -->

<div class="row">

    <div>

        <select name="" id="graph_district_3">

            <option value=0>ACS Mill</option>

            <option value=1>Alangayam</option>

            <option value=2>Ambur</option>

            <option value=3>Arakkonam</option>

            <option value=4 selected>Arcot</option>

            <option value=5>Gudiyatham</option>

            <option value=6>Katpadi</option>

            <option value=7>Kaveripakkam</option>

            <option value=8>Melalathur</option>

            <option value=9>Natrampalli</option>

            <option value=10>Ponnai</option>

            <option value=11>Sholingur</option>

```

```
<option value=12>TCS Mill</option>
<option value=13>Tirupattur</option>
<option value=14>VCS Mill</option>
<option value=15>Vaniyambadi</option>
<option value=16>Vellore</option>
<option value=17>Wallajah</option>
</select>
```

```
<select name="" id="graph_year_3">
  <option value=2010>2010</option>
  <option value=2011>2011</option>
  <option value=2012>2012</option>
  <option value=2013>2013</option>
  <option value=2014>2014</option>
  <option value=2015>2015</option>
  <option value=2016>2016</option>
  <option value=2017>2017</option>
  <option value=2018>2018</option>
  <option value=2019>2019</option>
</select>
```

```
<button class="buttonclass" id="graph_3_submit">submit</button>
```

```

</div>

<div class="rowsvg">

<svg height=300 width=300 id="Graph3"></svg>

<label> Monthly Total Rainfall</label>

</div>

<div>

    <select name="" id="graph_district_4">

        <option value=0>ACS Mill</option>

        <option value=1>Alangayam</option>

        <option value=2>Ambur</option>

        <option value=3>Arakkonam</option>

        <option value=4 selected>Arcot</option>

        <option value=5>Gudiyatham</option>

        <option value=6>Katpadi</option>

        <option value=7>Kaveripakkam</option>

        <option value=8>Melalathur</option>

        <option value=9>Natrampalli</option>

        <option value=10>Ponnai</option>

        <option value=11>Sholingur</option>

        <option value=12>TCS Mill</option>

        <option value=13>Tirupattur</option>

        <option value=14>VCS Mill</option>

        <option value=15>Vaniyambadi</option>

        <option value=16>Vellore</option>

```



```

        <option value=17>Wallajah</option>

    </select>

    <button class="buttonclass" id="graph_4_submit">submit</button>

</div>

<div class="rowsvg">

    <svg height=300 width=300 id="Graph4"></svg>

    <label >Yearly Total Rainfall</label>

</div>

</div>

</div>

<div class="tab3 tabinvisible">

    <div class="column">

        <div>

            <h2>Which algorithms I tested?</h2>

            <div class="label-flex">

                <span>LSTM</span><span>Xgboost</span>
                <span>Arima</span><span>Regression</span><span>Exponential
smoothening</span><span>SVM</span>

            </div>

        </div>

        <div><p>I started with ARIMA and Exponential smoothening. However I did
not get a good accuracy with these old models as the time series data is sparse and has
more stochastic component than deterministic component. LSTM also failed as our
sparse data resulted in very small prediction values. Finally after applying regression
techniques &#x1F525; XGBoost &#x1F525;gave the best result</p></div>

    </div>

```

```

<div class="column">

  <div>

    <h2>How can you apply regression on time series?</h2>

    <div class="label-flex">

      <span>Feature    Engineering</span><span>    Pattern    Finding</span>
<span>One-Hot Encoding</span> <span>8 lags</span>

    </div>

  </div>

  <div><p> Knowing time seires are autocorrelated, I took the particular day of
the month and went back in time(8 times) to that particular day and got the value.
Example: 8th August 2019, 8th August 2018.. and so on to 8 lags! Moreover I applied
OH-Encoding to the district column to use a single model for forecast!</p></div>

</div>

<div class="column">

  <div>

    <h2>Technologies used in project?</h2>

    <div class="label-flex">

      <span>Flask</span><span>    Keras</span>    <span>d3.js</span>
<span>pandas</span>

      <span>Numpy</span>    <span>Scikit-Learn</span>    <span>HTML</span>
<span>CSS</span> <span> JS</span>

    </div>

  </div>

  <div><p> I love Data Science and Web Development! Hence I was able to
work on this project in both the backend and frontend. It was an awesome experience
learning new technologies and my frontend is always improving! Do follow me on

```

```
linkedin!          <a          href="https://www.linkedin.com/in/om-purohit-
957187175/">here</a></p></div>
```

```
</div>
```

```
</div>
```

```
<script src="/static/js/jquery-1.8.3.min.js"></script>
```

```
<script src="/static/js/semantic.js"></script>
```

```
<script src="/static/js/d3.min.js"></script>
```

```
<script src="/static/js/index.js"></script>
```

```
<footer><p>
```

```
    This website was &#x26A1; by om
```

```
</footer>
```

```
</body>
```

```
</html>
```

13.3) Css

```
body{
```

```
    display: flex;
```

```
    flex-direction: column;
```

```
}
```

```
.attribute{
```

```
    width: 95px;
```

```
color:white;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: burlywood;

border-radius: 4px;

padding: 10px;

word-wrap: normal;

overflow: hidden;

white-space: nowrap;

transition: width 2s;

}

.attribute:hover{

    width: 60%;

    max-height: 20px;

}

footer{

    position:fixed;

    bottom:0px;

    width: 100%;

    left: 0px;

    height: 5%;

    font-size: 14px;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    background-color: rgb(27, 26, 26);
```

```

}

footer>p{

    color:white;

    position: relative;

    left:45%;

}

.buttonclass{

    margin-top: 10px;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    font-size:12px;

    width: 100px;

    height: 30px;

    border-right: 4px;

    border-top: 4px;

    border-radius: 4px;

    border-color: black;}

.buttonclass:hover{

    background-color: rgba(243, 227, 164, 0.596);

}

#fimg{

    opacity: 1;

}

.fadeout{

    opacity: 0;

```

```

    transition: opacity 2s;
}

.sun{
    animation-name: spin;
    animation-duration: 5000ms;
    animation-iteration-count: infinite;
    animation-timing-function: linear;
}

@keyframes spin {
    from {
        transform: rotate(0deg);
    }
    to {
        transform: rotate(360deg);
    }
}

.fadein{
    opacity: 1;
    transition: opacity 2s;
}

a{
    text-decoration: none;
}

```

```

.tabvisible{
display: block;
}

.tabinvisible{
    display: none !important;

}

.Navbar{
position:absolute;
top:30%;
width: 20px;
height: 150px  ;
}

.graph-icon{
    padding:10px;
    position: relative;
    left:-114px;
    width: 130px;
    margin-top:10px;
    display: flex;
    flex-direction: row;
    justify-content: flex-start;
    align-items:flex-start;
    border-radius: 4px;
    background-color: wheat;

```

```
    transition: left 1s;

}

.graph-icon:hover{

    left: -15px;

    transition: left 1s;

}

.graph-icon>span{

    width: 125px;

    font-family: 'Segoe UI';

}
```

```
.wand-icon{

    padding:10px;

    position: relative;

    width: 130px;

    left:-114px;

    margin-top:10px;
```



```
display: flex;

flex-direction: row;

justify-content: flex-start;

align-items: flex-start;

border-radius: 4px;

background-color: wheat;

transition: left 1s;

}
```

```
.wand-icon:hover{

left: -15px;

transition: left 1s;

}
```

```
.wand-icon>span{

font-family: 'Segoe UI';

width: 125px;

}
```

```
.notebook-icon{

padding: 10px;
```

```
position: relative;

width: 130px;

left: -114px;

margin-top: 10px;

display: flex;

flex-direction: row;

justify-content: flex-start;

align-items: flex-start;

border-radius: 4px;

background-color: wheat;

transition: left 1s;

}
```

```
.notebook-icon:hover{

left: -15px;

transition: left 1s;

}
```

```
.notebook-icon>span{

font-family: 'Segoe UI';

width: 125px;

}
```

```

.tab2{

    margin-bottom: 3px;

}

.tab3{

    margin-bottom: 7%;

}

.tab1{

    transition: display 4 ease;

}

.tab2{

    display: flex;

    flex-direction: column;

    width: 80%;

    margin-left: 0%;

    transition: display 4 ease;

}

.row{

    display:flex;

    flex-direction: row;

    justify-content: center;

    align-items: center;

    margin-left: 30%;

}

```

```

.row>div{

    display: flex;

    flex-direction: column;

}

.tab3>.row{

    box-sizing: border-box;

    display:flex;

    flex-direction: row;

    width: 70%;

    margin-left: 15%;

    flex:1 1 auto;

}

.tab3>.column>div{


display: flex;

flex-direction: column;

width: 50%;

padding-left: 30%;


}

.column{

    display: flex;

    flex-direction: column;

}

.tab3>.column>div>h2{

```

```

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

/* margin-top: 20px; */

flex:5;

padding-top:30px ;

height: 20%;

/* align-self:baseline; */
}

```

```

.tab3>.column>div>p{

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;


flex:1 1 auto;


}

```

```

.row>.rowsvg{

display: flex;

flex-direction: column;

margin-left: 10px;

}

```

```

.rowsvg>label{

padding:2px;

width: 44%;


box-shadow: white 10px 10px 10px inset;

```

```

margin-bottom: 28px;

margin-left: 70px;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

font-size: 10px;

border-radius: 4px;

border-width: 2px 0px 0px 2px;

border-color: rgb(46, 44, 42);

border-style: outset;
}

.row>div>select{

margin-top: 8px;

padding: 5px;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

font-size: 10px;

border-radius: 4px;

border: solid rgb(27, 20, 20) 2px;
}

.label-flex{

display: flex;

flex-direction: row;

align-items: center;

flex-wrap: wrap;
}

.label-flex>span{

```

```
margin-left: 4px;

padding: 8px;

margin-top: 2px;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

border-left: 2px;

border-top: 3px;

border-radius: 0%;

background-color: wheat;

border-style: solid;

border-radius: 9px;

border-color: wheat;

}
```

```
.small-flex {

display: flex;

flex-direction: column;

position: absolute;

justify-content: center;

left: 25%;

top: 25%;

height: 30%;

width: 20%;

}
```

```

.rain-bkg{

    background-color: #03d3fc;

}

.sun-bkg{

    background-color: coral;

}

.white-bkg{

    background-color: cornsilk;

}

.small-flex>img,.small-flex>div{

    font-size: 4vw;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;


    width: 70%;

}

.form-flex {

    position:absolute;

    left:30%;

    display:flex;

    justify-content: center;

    align-items:center;

    flex-direction: column;

    width: 50%;

    top: 20%;

    height: 50%;

```



```

}

.form-flex>option{

    z-index:2;


    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    font-size: 40px;

}

.form-flex>input,.form-flex>select{

    width:40%;

    z-index:2;

    left:0%;

    margin-top: 20px;

    padding: 10px;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    font-size: 30px;

    border-radius: 20px;

    border:solid rgb(27, 20, 20) 2px;

}

/* .change_img{

    width: 30%;

    flex-direction: row;

} */

```

13.4) Index.js

```
function precise(x) {  
    return Number.parseFloat(x).toFixed(2);  
}  
  
function LoadData(){  
    console.log("OVER HERE")  
  
    let date=$("#DateInput").val()  
  
    let district=$("#DistrictInput").val()  
  
  
    let flaskdata={"date":date,"dis":district}  
  
  
  
    $.ajax({  
        type:"POST",  
        url:"/getData",  
        contentType: 'application/json;charset=UTF-8',  
        data:JSON.stringify(flaskdata),  
        success:function(data) {  
            d3.json("/getData").then((d)=>{  
                let x=precise( d)  
                let out=String(x)+"mm"  
                $("#forecastOutput").text(out)  
                if(parseFloat(out)<1){  
                    console.log("done")  
                    $("#fimg").addClass("fadeout")  
                }  
            })  
        }  
    })  
}
```

```

        $("#fimg").attr("src","/static/images/sunny.svg")

        $("#fimg").addClass("fadein")

        $("#fimg").addClass("sun")


        $("#fimg").removeClass("fadeout")


    }

    else{

        $("#fimg").addClass("fadeout")

        $("#fimg").attr("src","/static/images/rainy.svg")

        $("#fimg").addClass("fadein")

        $("#fimg").removeClass("sun")

        $("#fimg").removeClass("fadeout")


    }

    })

}

})

}

}

```

////////////////////////////////////////HERE ARE THE
 GRAPHS

```

function PlotYearly(data){

    d3.selectAll(".headerg").remove()

    d3.selectAll(".bar").remove()

    if(data.length==0){

    }

    var svg=d3.select("#BasicGraph")

    margin = 20,

    width = svg.attr("width") - margin,

    height = svg.attr("height") - margin;

    var xScale = d3.scaleBand().range([0, width]).padding(0.3),

    yScale = d3.scaleLinear().range([height, 0]);

    xScale.domain(data.map(function(d) { return d[2]; }));

    yScale.domain([0, d3.max(data, function(d) { return d[3]; })]);

    var g=svg.append("g").attr("class","headerg")

    g.append("g")

    .attr("transform", "translate(20," + height + ")")

    .call(d3.axisBottom(xScale))

    g.append("g")

    .attr("transform", "translate(24,10)")

    .call(d3.axisLeft(yScale))

    g.selectAll(".bar")

```

```

.data(data)

.enter().append("rect")

.attr("class", "bar")

.attr("x", function(d) { return xScale(d[2]); })

.attr("y", function(d) { return yScale(d[3]); })

.attr("width", xScale.bandwidth())

.attr("fill", "#f5ec42")

.attr("transform", "translate(20,0)")

.attr("height", function(d) {return height - yScale(d[3]); });
}

```

```

function PlotYearly3(data){

```

```

    d3.selectAll(".headerg3").remove()

```

```

    d3.selectAll(".bar3").remove()

```

```

    if(data.length==0){

```

```

    }

```

```

var svg=d3.select("#Graph3")

```

```

margin = 20,

```

```

    width = svg.attr("width") - margin,

```

```

    height = svg.attr("height") - margin;

```

```

var xScale = d3.scaleBand().range([0, width]).padding(0.3),

```

```

    yScale = d3.scaleLinear().range([height, 0]);

    xScale.domain(data.map(function(d) { return d[2]; }));

    yScale.domain([0, d3.max(data, function(d) { return d[3]; })]);

var g=svg.append("g").attr("class","headerg3")

g.append("g")

    .attr("transform", "translate(20," + height + ")")

    .call(d3.axisBottom(xScale))

g.append("g")

    .attr("transform", "translate(24,10)")

    .call(d3.axisLeft(yScale))


g.selectAll(".bar3")

    .data(data)

    .enter().append("rect")

    .attr("class", "bar3")

    .attr("x", function(d) { return xScale(d[2]); })

    .attr("y", function(d) { return yScale(d[3]); })

    .attr("width", xScale.bandwidth())

    .attr("fill", "#f5ec42")

    .attr("transform", "translate(20,0)")


    .attr("height", function(d) {return height - yScale(d[3]); });

}

function PlotDistrictYearly(data){

```

```

d3.selectAll(".headerg2").remove()

d3.selectAll(".bar2").remove()

if(data.length==0){

}

var svg=d3.select("#Graph2")

    margin = 20,

    width = svg.attr("width") - margin,

    height = svg.attr("height") - margin;

var xScale = d3.scaleBand().range([0, width]).padding(0.3),

    yScale = d3.scaleLinear().range([height, 0]);

    xScale.domain(data.map(function(d) { return d[1]; }));

    yScale.domain([0, d3.max(data, function(d) { return d[2]; })]);

var g=svg.append("g").attr("class","headerg2")

g.append("g")

    .attr("transform", "translate(20," + height + ")")

    .call(d3.axisBottom(xScale))

g.append("g")

    .attr("transform", "translate(25,10)")

    .call(d3.axisLeft(yScale))


g.selectAll(".bar2")

    .data(data)

    .enter().append("rect")

```

```

.attr("class", "bar2")

.attr("x", function(d) { return xScale(d[1]); })

.attr("y", function(d) { return yScale(d[2]); })

.attr("width", xScale.bandwidth())

.attr("fill", "#f5ec42")

.attr("transform", "translate(20,0)")

.attr("height", function(d) {return height - yScale(d[2]); });

}

```

```

function PlotDistrictYearly4(data){

  d3.selectAll(".headerg4").remove()

  d3.selectAll(".bar4").remove()

  if(data.length==0){

  }

  var svg=d3.select("#Graph4")

  margin = 20,

  width = svg.attr("width") - margin,

  height = svg.attr("height") - margin;

  var xScale = d3.scaleBand().range([0, width]).padding(0.3),

  yScale = d3.scaleLinear().range([height, 0]);

  xScale.domain(data.map(function(d) { return d[1]; }));

```



```

    yScale.domain([0, d3.max(data, function(d) { return d[2]; })]);

var g=svg.append("g").attr("class","headerg4")

g.append("g")

    .attr("transform", "translate(20," + height + ")")

    .call(d3.axisBottom(xScale))

g.append("g")

    .attr("transform", "translate(25,10)")

    .call(d3.axisLeft(yScale))


g.selectAll(".bar4")

    .data(data)

    .enter().append("rect")

    .attr("class", "bar4")

    .attr("x", function(d) { return xScale(d[1]); })

    .attr("y", function(d) { return yScale(d[2]); })

    .attr("width", xScale.bandwidth())

    .attr("fill", "#f5ec42")

    .attr("transform", "translate(20,0)")


    .attr("height", function(d) {return height - yScale(d[2]); });

}


//#####HERE ARE
THE BUTTON EVENTS


$("#graph_1_submit").on("click",()=>{

```

```

let district=$("#graph_district option:selected").text()

let year=$("#graph_year option:selected").text()

$.ajax({

  type:"POST",

  url:"/GraphYearlyMonthly",

  contentType: 'application/json;charset=UTF-8',

  data:JSON.stringify({"dis":district,"year":year}),

  success:function(data) {

    d3.json("/GraphYearlyMonthly").then((d)=>{

      console.log(JSON.parse(d))

      PlotYearly(JSON.parse(d).data)

    })

  }

})

})

$("#graph_2_submit").on("click",()=>{

```

```

let district=$("#graph_district_2 option:selected").text()

$.ajax({
  type:"POST",
  url:"/GraphDistrictYearly",
  contentType: 'application/json;charset=UTF-8',
  data:JSON.stringify({"dis":district}),
  success:function(data) {

    d3.json("/GraphDistrictYearly").then((d)=>{

      console.log(JSON.parse(d))

      PlotDistrictYearly(JSON.parse(d).data)

    })

  }

})

})

```

```

$("#graph_3_submit").on("click",()=>{

  let district=$("#graph_district_3 option:selected").text()

  let year=$("#graph_year_3 option:selected").text()

```

```

$.ajax({
  type:"POST",
  url:"/GraphSumYearly",
  contentType: 'application/json;charset=UTF-8',
  data:JSON.stringify({"dis":district,"year":year}),
  success:function(data) {
    d3.json("/GraphSumYearly").then((d)=>{
      console.log(JSON.parse(d))

      PlotYearly3(JSON.parse(d).data)

    })

  }
})

})

$("#graph_4_submit").on("click",()=>{
  let district=$("#graph_district_4 option:selected").text()

  $.ajax({
    type:"POST",
    url:"/GraphSumDistrict",
    contentType: 'application/json;charset=UTF-8',
    data:JSON.stringify({"dis":district}),
    success:function(data) {

```

```

d3.json("/GraphSumDistrict").then((d)=>{

  console.log(JSON.parse(d))

  PlotDistrictYearly4(JSON.parse(d).data)

})

}

})

})

// #####
Some basic interactions

$(".wand-icon").on("click",()=>{

  console.log("its clicked ")

  $(".tab2").addClass("tabinvisible")

  $(".tab3").addClass("tabinvisible")

  $(".tab1").removeClass("tabinvisible")

  $(".tab1").addClass("tabvisible")

})

$(".graph-icon").on("click",()=>{

  console.log("its clicked ")

```

```

$(".tab1").addClass("tabinvisible")

$(".tab3").addClass("tabinvisible")


$(".tab2").removeClass("tabinvisible")


$(".tab2").addClass("tabvisible")


})


$(".notebook-icon").on("click",()=>{

    console.log("its clicked ")


$(".tab1").addClass("tabinvisible")

$(".tab2").addClass("tabinvisible")


$(".tab3").removeClass("tabinvisible")


$(".tab3").addClass("tabvisible")

})

// ##### Default selections
for the viz tab

d3.json("/GraphYearlyMonthly").then((d)=>{

```

```
console.log(JSON.parse(d))
```

```
PlotYearly(JSON.parse(d).data)
```

```
}}
```

```
d3.json("/GraphDistrictYearly").then((d)=>{
```

```
console.log(JSON.parse(d))
```

```
PlotDistrictYearly(JSON.parse(d).data)
```

```
}}
```

```
d3.json("/GraphSumYearly").then((d)=>{
```

```
console.log(JSON.parse(d))
```

```
PlotYearly3(JSON.parse(d).data)
```

```
}}
```

```
d3.json("/GraphSumDistrict").then((d)=>{
```

```
console.log(JSON.parse(d))
```

```
PlotDistrictYearly4(JSON.parse(d).data)
```

```
}}
```

13.5) some_func.py

```
import pandas as pd
```

```
import xgboost as xgb
```

```
import pickle
```

```
import numpy as np
```

```
def isRain(x):
```

```
    if x>100:
```

```
        return 5
```

```
    if x>50:
```

```
        return 4
```

```
    if x>25:
```

```
        return 3
```

```
    if x>2:
```

```
        return 2
```

```
    elif x!=0:
```

```
        return 1
```

```
    else:
```

```
        return 0
```

```
class Forecast():
```

```
    def __init__(self):
```

```
        self.forecast=0
```

```
        self.SparseArray=np.zeros(8)
```

```
        self.xgb=xgb.XGBRegressor(objective ='reg:squarederror',
```



```

        n_estimators=3, seed = 123,max_depth=35,eta=0.3)

self.xgb.load_model("models/modelsxgb.json")

with open("models/DayMon.json","rb") as fp:

    self.DayMonth=pickle.load(fp)

def GetYMD(self,date_val,District,District_index):

    self.District=District

    self.DistrictIndex=District_index

    self.year=date_val.split("-")[0]

    self.day=date_val.split("-")[2]

    self.month=date_val.split("-")[1]

    if self.month[0]=="0":

        self.month=self.month[1]

    if self.day[0]=="0":

        self.day=self.day[1]

def GetSparse(self):

    daymonth=str(self.District)+"_"+str(self.day)+"_"+str(self.month)

    for k in range(1,9):

        if k<len(self.DayMonth[daymonth]):

            val=self.DayMonth[daymonth][-k]

        else:

            val=0

        final=isRain(val)

        self.SparseArray[k-1]=final

```

```

def CreateNForecast(self):

    self.Input=[int(self.month),int(self.day),int(self.year)]

    for i in self.SparseArray:

        self.Input.append(int(i))

    templist=[0]*18

    templist[self.DistrictIndex]=1

    self.Input.extend(templist)

    print(np.array([np.array(self.Input)]))

    self.forecast=self.xgb.predict(np.array([np.array(self.Input)]))[0]

    if self.forecast<0.18:

        self.forecast=0


class Visualize():

    def __init__(self):

        self.district=""

        self.year=0

        self.month=0

        self.day=0

        self.df=pd.read_csv("static/DataNotebooks/converted.csv")

    def MonthlyRain(self,year,district):

        print(district)


self.df_grouped=self.df.groupby(["District","year","month"]).mean().reset_index()


self.df_grouped=self.df_grouped[(self.df_grouped["year"]==year)&(self.df_grouped[
"District"]==district)]

```

```

def DistrictRain(self,district):

    self.df_grouped2=self.df.groupby(["District","year"]).mean().reset_index()

    self.df_grouped2=self.df_grouped2[(self.df_grouped2["District"]==district)]

def SumMonthlyRain(self,year,district):

self.df_grouped3=self.df.groupby(["District","year","month"]).sum().reset_index()

self.df_grouped3=self.df_grouped3[(self.df_grouped3["year"]==year)&(self.df_group
ed3["District"]==district)]

def SumDistrictRain(self,district):

    self.df_grouped4=self.df.groupby(["District","year"]).sum().reset_index()

    self.df_grouped4=self.df_grouped4[(self.df_grouped4["District"]==district)]

```

13.6) app.py

```

from flask import Flask, jsonify, render_template,request

import csv

import pandas as pd

import numpy as np

import json

from some_funcs import *

app = Flask(__name__)

app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0

modelobj=Forecast()

graph=Visualize()

```

```

graph.MonthlyRain(2010, "Arcot")

graph.DistrictRain("Arcot")

graph.SumDistrictRain("Arcot")

graph.SumMonthlyRain(2010, "Arcot")

Districts=["ACS Mill", "Alangayam", "Ambur",
           "Arakkonam",
           "Arcot",
           "Gudiyatham",
           "Katpadi",
           "Kaveripakkam",
           "Melalathur",
           "Natrampalli",
           "Ponnai",
           "Sholingur",
           "TCS Mill",
           "Tirupattur",
           "VCS Mill",
           "Vaniyambadi",
           "Vellore",
           "Wallajah"]

@app.route('/')

def index():

    return render_template('home.html')

```

```

@app.route("/getData",methods = ['POST','GET'])

def getData():

    to_send=""

    if request.method=="POST":

        jsdata=request.json

        modelobj.GetYMD(jsdata["date"], Districts[int(jsdata["dis"])],int(jsdata["dis"]))

        modelobj.GetSparse()

        modelobj.CreateNForecast()

    else:

        to_send=float(modelobj.forecast)

    return jsonify(to_send)

@app.route("/GraphYearlyMonthly",methods = ['POST','GET'])

def graph_yearly_monthly():

    if request.method=="POST":

        jsdata=request.json

        graph.MonthlyRain(int(jsdata["year"]),jsdata["dis"])

        return jsonify("")

    if request.method=="GET":

        return jsonify(graph.df_grouped.to_json(orient="split"))

@app.route("/GraphDistrictYearly",methods = ['POST','GET'])

def graph_district_yearly():

    if request.method=="POST":

        jsdata=request.json

```

```

graph.DistrictRain(jsdata["dis"])

return jsonify("")

if request.method=="GET":

    return jsonify(graph.df_grouped2.to_json(orient="split"))

@app.route("/GraphSumYearly",methods = ['POST','GET'])
def graph_sum_yearly():

    if request.method=="POST":

        jsdata=request.json

        graph.SumMonthlyRain(int(jsdata["year"]), jsdata["dis"])

        return jsonify("")

    if request.method=="GET":

        return jsonify(graph.df_grouped3.to_json(orient="split"))

@app.route("/GraphSumDistrict",methods = ['POST','GET'])
def graph_sum_district():

    if request.method=="POST":

        jsdata=request.json

        graph.SumDistrictRain(jsdata["dis"])

        return jsonify("")

    if request.method=="GET":

        return jsonify(graph.df_grouped4.to_json(orient="split"))

if __name__ == '__main__':

    app.run(debug=True,port=5000)

```

13.7) Website Images and Important graphs

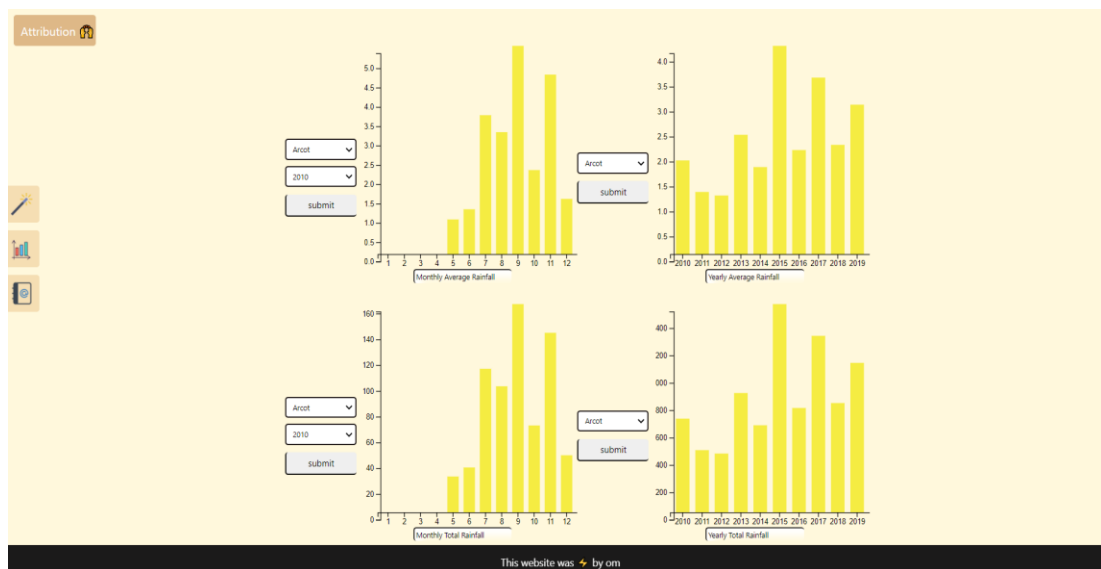
Attribution 






? mm



This website was  by om



Which algorithms I tested?

LSTM Xgboost Arima Regression Exponential smoothening SVM

I started with ARIMA and Exponential smoothening. However I did not get a good accuracy with these old models as the time series data is sparse and has more stochastic component than deterministic component. LSTM also failed as our sparse data resulted in very small prediction values. Finally after applying regression techniques 🏆 XGBoost 🏆 gave the best result

How can you apply regression on time series?

Feature Engineering Pattern Finding One-Hot Encoding 8 lags

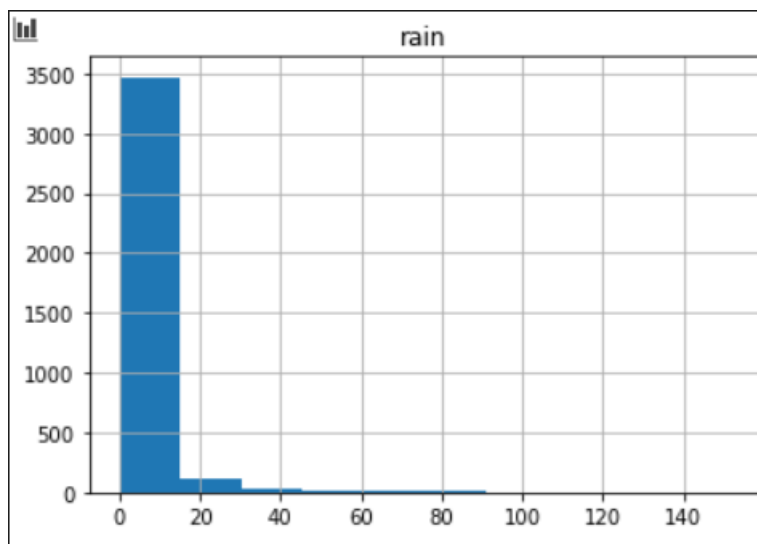
Knowing time series are autocorrelated, I took the particular day of the month and went back in time(8 times) to that particular day and got the value. Example: 8th August 2019, 8th August 2018, and so on to 8 lags! Moreover I applied OH-Encoding to the district column to use a single model for forecast!

Technologies used in project?

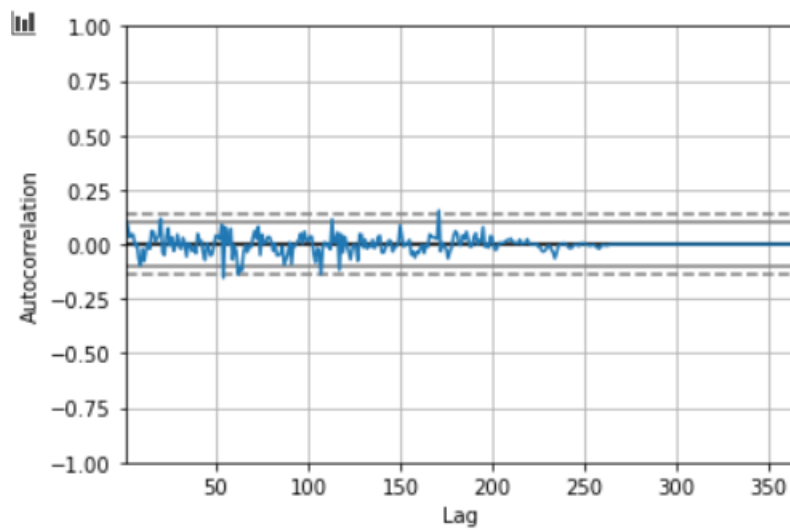
Flask Keras d3.js pandas Numpy Scikit-Learn HTML CSS JS

I love Data Science and Web Development! Hence I was able to work on this project in both the backend and frontend. It was an awesome experience learning new technologies and my frontend is always improving! Do follow me on linkedin! [here](#)

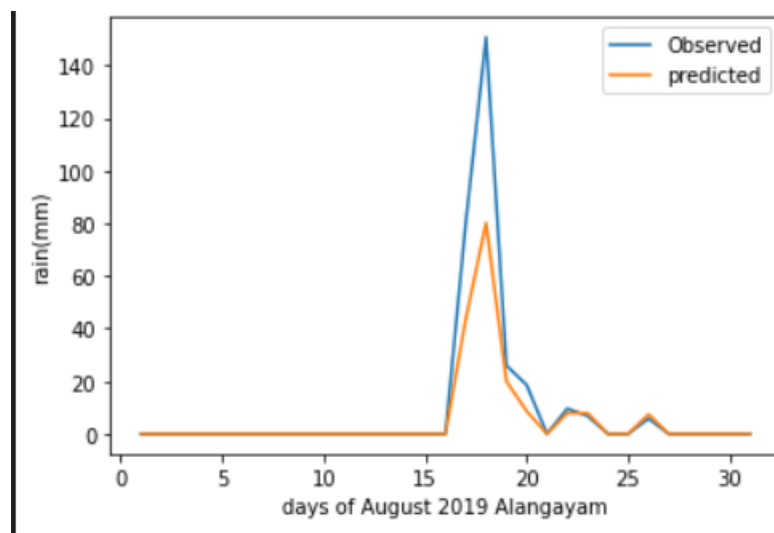
This website was ⚡ by om



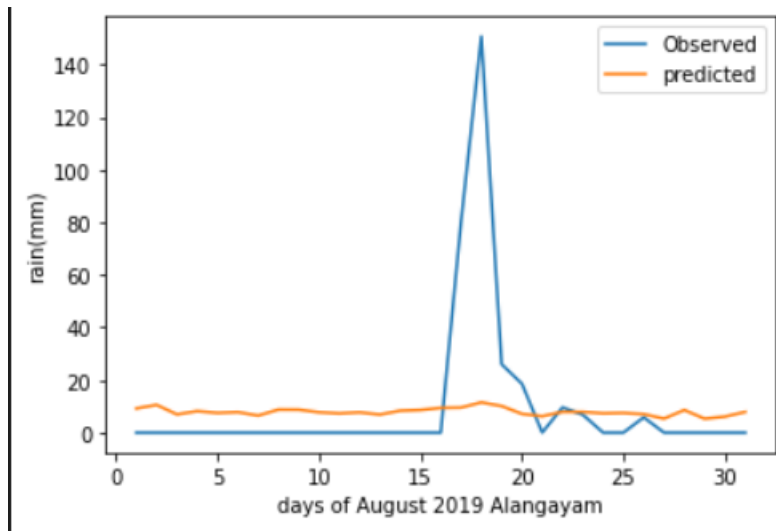
Histogram of Alangayam district as example of sparse data being 80% of dataset!



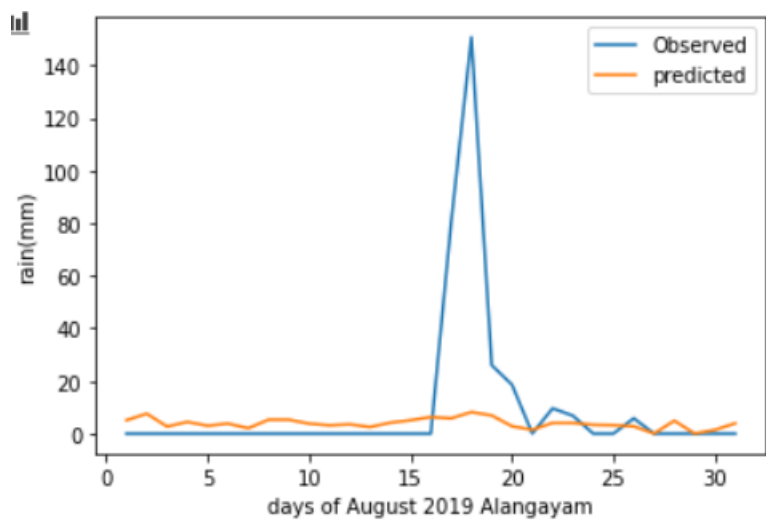
Autocorrelation plot with lags from 0 to 365(a year)



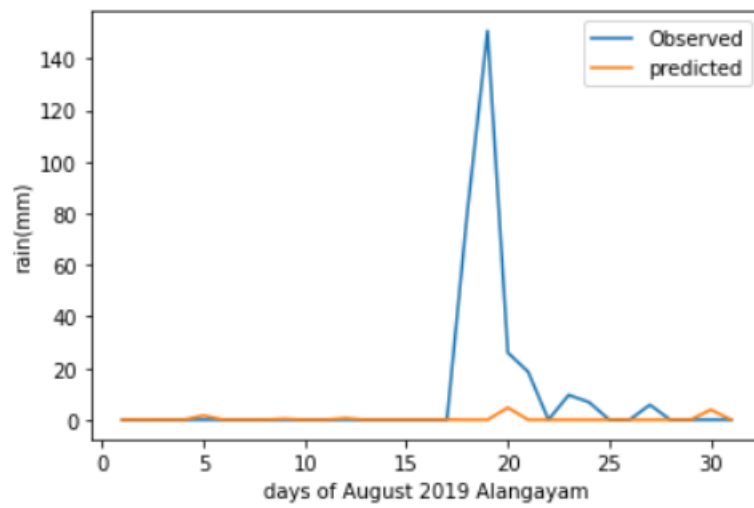
Xgboost(final model) forecast plot against original plot, on the year 2019 August Alangayam



SVM forecast plot against original plot, on the year 2019 August Alangayam



Linear Regression forecast plot against original plot, on the year 2019 january Alangayam



LSTM forecast plot against original plot, on year 2019 August Alangayam

14) Internship Project Abstract

The project aims to aid Research Analysts, who go through various articles on the web and is tedious to keep a track on all of them, especially chrome tabs which use a lot of memory. Furthermore, not only will this project shrink the chrome tabs needed for the research, it contains topic clustering, NLP base summarization, and context of information retrieval. In doing so the whole project eases the workflow of the analysts at Twimbit and augments their productivity. **(Trimmed down due to Twimbit Policy)**

15) Literature Review & Design

Author	Title	Technology/Techniques used	Abstract	Dataset
A.CSantha ,Dr.C.Jayakumar	Comparative Study of Syntactic Search Engine and Semantic Search Engine: A Survey	Semantic and syntactic search	Search engine symbolizes an extremely powerful and valuable tool for fetching any sort of information from Internet. There has been numerous researches carried on search engines techniques, the major ones are syntactic and semantic. Referring to the Syntactic web, the results obtained are purely as per the keyword match. That is the query outputs numerous web pages against the keyword match that may not even be relevant or meaningful. Whereas, unlike the syntactic web, the semantic web is a revised or upgraded version of the web which produces quiet meaningful and specific output as it has the potential to comprehend the query effectively. Few examples of Semantic based search engines include Kosmix, Hakia, Cognition, Swoogle and Lexxe. Whereas syntactic based search engines are Google, Yahoo, Ask. The work performs a comparison amidst the performance of semantic and syntactic based search engine and evaluates them by employing certain queries.	Kosmix, Hakia, Cognition, Swoogle and Lexxe. Whereas syntactic based search engines are Google, Yahoo, Ask.
Jianyou Lv, Yuqian Wang	Semantic Information Detection of Webpage Based on Word Vector and Infomap	Regular expression, viterbi algorithm, word segmentation,ngram2vec,infomap clustering, multi-layer neural network	For Chinese web pages, we use regular expression and Viterbi algorithm to realize Chinese filtering and word segmentation, then use ngram2vec algorithm to get the word vector set of web page and pre train the word vector set of Baidu Encyclopedia. Baidu Encyclopedia word vector set is based on Infomap clustering algorithm to realize word vectorClustering and tagging types, training neural network through training data set and Baidu Encyclopedia corpus to determine the type of unknown web pages through neural network, and achieve the purpose of detecting the semantic information of unknown web pages. This algorithm is has few super parameters and high calculation efficiency. Experiments show that the accuracy of the trained neural network model reaches 96.73%, which can quickly and accurately identify the type of web page	Chinese webpages and Baidu encyclopedia
Laureta Hajderanj Weheliye, Daqing Chen	ANew Supervised t-SNEwith dissimilarity Measure for Effective Data Visualization and Classification	S-tSNE ,t-SNE,KNN,dimensionality reduction	In this paper, a new version of the Supervised t- Stochastic Neighbor Embedding (S-tSNE) algorithm is proposed which introduces the use of a dissimilarity measure related to class information. The proposed S-tSNE can be applied in any high dimensional dataset for visualization or as a feature extraction for classification problems. In this study, the S-tSNE is applied to three datasets MNIST, Chest x-ray, and SEER Breast Cancer. The two-dimensional data generated by the S-tSNE showed better visualization and an improvement in terms of classification accuracy in comparison to the original t-Stochastic Neighbor Embedding(t-SNE) method. The results from k-nearest neighbors (k-NN) classification model which used the lower dimension space generated by the new S-tSNE method showed more than 20% improvement on average in accuracy in all the three datasets compared with the t-SNE method. Iaddition, the classification accuracy using the S-tSNE for feature extraction was even higher than classification accuracy obtained from the original high dimensional d	MNIST, chest x-rays and SEER Breast cancer dataset
Changzhou Li, Yao Lu, Junfeng Wu, Yongrui Zhang, Zhongzhou Xia, Tianchen Wang, Dantian Yu, Xurui Chen, Peidong Liu, Junyu Guo	LDA Meets Word2Vec: A Novel Model for Academic Abstract Clustering	Latent Dirchlet Allocation(LDA), Word2Vec, Document clustering, PW-LDA	Clustering narrow-domain short texts, such as academic abstracts, is an extremely difficult clustering problem. Firstly, short texts lead to low frequency and sparseness of words, making clustering results highly unstable and inaccurate; Secondly, narrow domain leads to great overlapping of insignificant words and	Academic abstracts, Wan Fang med Database

			<p>makes it hard to distinguish between sub-domains, or fine-grained clusters. The vocabulary size is also too small to construct a good word bag needed by traditional clustering algorithms like LDA to give a meaningful topic distribution. A novel clustering model, Partitioned Word2Vec-LDA (PW-LDA), is proposed in this paper to tackle the described problems. Since the purpose sentences of an abstract contain crucial information about the topic of the paper, we firstly implement a novel algorithm to extract them from the abstracts according to its structural features. Then high-frequency words are removed from those purpose sentences to get a purified-purpose corpus and LDA and Word2Vec models are trained. After combining the results of both models, we can cluster the abstracts more precisely. Our model uses abstract text instead of keywords to cluster because keywords may be ambiguous and cause unsatisfied clustering results shown by previous work. Experimental results show that the clustering results of PW-LDA are much more accurate and stable than state-of-the-art techniques</p>	
Jie Liu, Chun Yu ,Wenchang Xu, Yuanchun Shi	Clustering Web Pages to Facilitate Revisitation on Mobile Devices	VSM(vector space model), TF-IDF(term frequency inverse document frequency)	<p>Due to small screens, inaccuracy of input and other limitations of mobile devices, revisitation of Web pages in mobile browsers takes more time than that in desktopbrowsers. In this paper, we propose a novel approach to facilitate revisitation. We designed AutoWeb, a system that clusters opened Web pages into different topics based on their contents. Users can quickly find a desired opened Web page by narrowing down the searching scope to a group of Web pages that share the same topic. Clustering accuracy is evaluated to be 92.4%and computing resource consumption was proved to be acceptable. A user study wasconducted to explore user experience and how much AutoWeb facilitates revisitation. Results showed that AutoWeb could save up a significant time for revisitation and participants rated the system highly.</p>	Web pages, MIT articles, Financial pages, clothing brands

15.1) Modules:

- Universal Scraper
- Topic clustering,
- Chome Extension,
- Semantic Search,
- Visualization

15.2) Database: Raw File Structure in AWS using CSVs

15.3) Hardware requirements: NONE

15.4) Software requirements:

- Chrome
- Flask
- Postman
- Dash/Plotly
- NLP Libraries(NITK,BERT..etc)
- Selenium and Beautifulsoup

Conclusion

The various implementation of algorithms provided a range of different test scores using MAE. Moreover, each model provided a reason to why such low accuracy was obtained, such as our data being sparse, seasonality not in fixed pattern and different districts would presumably lead to different models. However, using feature engineering and converting it into a regression problem, aided in reducing the model to only one and also raised the accuracy score simultaneously. XGBoost was the final model selected for the website, as it had the best test score and unlike other models, it did not overfit the data. MAE of 1.57 is an acceptable score given the type of data on our hands. To further decrease the error substantially, climate indices data of Vellore would be required such as temperature and humidity. As the number of features increase, the data would become less time dependent and become more deterministic.

REFERENCES

- [1] Bari, S. H., Rahman, M. T., Hussain, M. M., & Ray, S. (2015). Forecasting monthly precipitation in Sylhet city using ARIMA model. *Civil and Environmental Research*, 7(1), 69-77.
- [2] Diodato, N., & Bellocchi, G. (2018). Using historical precipitation patterns to forecast daily extremes of rainfall for the coming decades in Naples (Italy). *Geosciences*, 8(8), 293.
- [3] Hung, N. Q., Babel, M. S., Weesakul, S., & Tripathi, N. K. (2009). An artificial neural network model for rainfall forecasting in Bangkok, Thailand. *Hydrology and Earth System Sciences*, 13(8), 1413-1425.
- [4] Yan, J., Xu, T., Yu, Y., & Xu, H. (2021). Rainfall Forecast Model Based on the TabNet Model. *Water*, 13(9), 1272.
- [5] Schepen, A., Wang, Q. J., & Robertson, D. (2012). Evidence for using lagged climate indices to forecast Australian seasonal rainfall. *Journal of Climate*, 25(4), 1230-1246.

REFERENCES(Internship)

- [1]Sheela, A. S., & Jayakumar, C. (2019, March). Comparative Study of Syntactic Search Engine and Semantic Search Engine: A Survey. In 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM) (Vol. 1, pp. 1-4). IEEE.
- [2]Wang, Y., & Lv, J. (2020, July). Semantic Information Detection of Webpage Based on Word Vector and Infomap. In 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS) (pp. 293-297). IEEE.
- [3]Hajderanj, L., Weheliye, I., & Chen, D. (2019, April). A new supervised t-SNE with dissimilarity measure for effective data visualization and classification. In Proceedings of the 2019 8th International Conference on Software and Information Engineering
- [4] Li, C., Lu, Y., Wu, J., Zhang, Y., Xia, Z., Wang, T., ... & Guo, J. (2018, April). LDA meets Word2Vec: a novel model for academic abstract clustering. In Companion proceedings of the the web conference 2018 (pp. 1699-1706).
- [5] Liu, J., Yu, C., Xu, W., & Shi, Y. (2012, February). Clustering web pages to facilitate revisitation on mobile devices. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (pp. 249-252).