A project report on

NLP Dashboard & Rainfall Forecasting

Submitted in partial fulfillment for the award of the degree of

Bachelor in Technology Information Technology

by

Om Hemantkumar Purohit (17BIT0368)

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

SITE

April 2021

SIGNATURE BY INTERNSHIP GUIDE

SIGNATURE BY PROJECT GUIDE

RAINFALL FORECASTING
ABSTRACT

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. 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. Rain Prediction will be done on the basis of a simple factor the date and district. 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, Exponentional Smoothening. 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 dashboard, which I plan to create after 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.

Literature Survey

Title	Dataset	Abstract	Parameters	Advantages of the model	Limitations of the model
Rain rate and rain attenuation prediction with experimental rain attenuation efforts in south-western Nigeria	This data is available at the Kitami Institute of Technology databank	Rain induced attenuation is a prominent loss factor for communication system design in the terrestrial and satellite- earth links. Its severity is more pronounced at frequencies above 10GHz [1]	The prominent ITU - R rain rate model as detailed in is based on the use of meteorological parameters available from ITU's 3M Group website. The Kitami rain rate Distribution model employs two regional	The ITU and RH models show good performance at low rain rates while Kitami model shows the worst result for the location. Fig . 3 shows the predicted rain attenuatio n at 12.736 GHz, 12.522GHz,	The model works at Certain frequency Only such as 31.4 GHZ
Novel integration- time conversion of rain-rate statistics for rain attenuation prediction models	A disdrometer has been used for rain accumulation measurements Thirty rain events during 2011-2012 are considered``	Rain attenuation prediction model is important for both satellite and terrestrial communication s.	The instability parameters are estimated from radiometric data to point the development of atmospheric	The nowcasting technique is, therefore, able to predict both rain occurrence and rain accumulatio n.	We get results at 22.24, 23.8, 26.4 and 31GHz but only optimal and consiferable is at 31.4Ghz
A model of rain attenuation in Ka band based on the Wiener prediction	DAH is a prediction model, which is proposed by Allnutt, Dissanayake and Haidara after analyzing the data that come from series of experiments based on INTEL SAT satellite system,	In this paper a new rain attenuation model based on Wiener prediction is established after analyzing the DAH model in Ka band.	Because of more parameters More complex process of calculation and the need of renewing all the parameters	In this paper, a new rain attenuation model is introduced based on the analysis of DAH model. Simulatio n results show that this new model can achieve the same effect with DAH model	Besides, this new model of 3th order has only 3 parameters, and there are no close relationshi p between the parameter s and frequency.

A New Rain Attenuation Prediction Model for the Earth-Space Links	Based on the measurement data by Meteorological radar,a rain attenuation prediction model was	Earth-space communicationOn systems arenow utilizing the Ku-and Ka- frequency	Twelve parameters sets, one for each month of the year, are available. The model	A new rain attenuation prediction model for the earth-space links is	over various Ranges of latitudes, frequencies, and

REFERENCES

[1] Nandi, D. (2018, November). Prediction of Rain Attenuation Statistics from Measured Rain Rate Statistics using Synthetic Storm Technique for Micro and Millimeter wave Communication Systems. In 2018 IEEE MTT-S International Microwave and RF Conference (IMaRC) (pp. 1-4). IEEE.

[2]Ibiyemi, T. S., Ajewole, M. O., Ojo, J. S., & Obiyemi, O. O. (2012, November). Rain rate and rain attenuation prediction with experimental rain attenuation efforts in south-western Nigeria. In 2012 20th Telecommunications Forum (TELFOR) (pp. 327-329). IEEE

[3]Thiennviboon, P., Intarawichian, S., Zhao, Z. W., Lin, L. K., & Lu, C. S. Novel integration-time conversion of rain-rate statistics for rain attenuation prediction models. In 2017 International Symposium on Antennas and Propagation (ISAP) (pp. 1-2). IEEE.

[4]Xinyu, D., Yong, S., & Yanling, W. (2010, November). A model of rain attenuation in Ka band based on the Wiener prediction. In Proceedings of Papers 5th European Conference on Circuits and Systems for Communications (ECCSC'10) (pp. 264-267). IEEE.

[5]Lu, C. S., Zhao, Z. W., Wu, Z. S., Lin, L. K., Thiennviboon, P., Zhang, X., & Lv, Z. F. (2018). A new rain attenuation prediction model for the earth-space links. IEEE Transactions on Antennas and Propagation, 66(10), 5432-5442.

Detailed Design

Modules:
- Data Aggregator(Excel files)
- Preprocessing data
- EDA(Exploratory Data Analysis)
- Detect and remove Trend, Seasonality as per Time Series standards
- Final Prediction using different algorithms

Database: Excel Files

Hardware requirements: NONE

Software requirements:
- Pandas
- Numpy and core preprocessing libraries

- Libraries such as Statistical models, Random Forest, Stacked ensemble, linear regression models, keras, Xgboost, scikit learn
- Dashboard (Dash/plotly or D3.js)

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.

Date	Alangayam	Ambur	Arakkonam	Arcot	Gudiyatham	Kaveripakkam	Melalathur	Sholingur	Tirupattur	Vaniyambad	Vellore	Wallajah	Total	Avg
VELLORE DISTRICT RAINFALL ANNUAL REPORT- 2010														
January	-	-	-	-	14.00	NF	9.20	20.00	3.50	-	44.80	-	91.50	7.63
Febraury	-	-	-	-	-	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	107.3
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	111.0
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	232.4
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	920.2

Season	Alangayam	Ambur	Arakkonam	Arcot	Gudiyatham	Kaveripakkam	Melalathur	Sholingur	Tirupattur	Vaniyambad	Vellore	Wallajah	Total	Avg
Season Report														
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	113.7
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	396.2
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	402.5

District Total	11042.78
District Avg	920.23

Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec | **Annual**

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

VELLORE DISTRICT RAINFALL REPORT JANUARY - 2010

Date	Alangaya m	Ambur	Arakkonam	Arcot	Gudiyathan	Melalathur	Sholingur	Tirupattur	Vaniyambad	Vellore	Wallajah	Total	Avg
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	2.60	2.00	-	2.00	-	-	-	6.60	0.60
########	-	-	-	-	11.40	7.20	-	1.50	-	44.80	-	64.90	5.90
########	-	-	-	-	-	-	20.00	-	-	-	-	20.00	1.82
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00
########	-	-	-	-	-	-	-	-	-	-	-	0.00	0.00

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

Preprocessing and Cleaning Data

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                District
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	-	

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

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.

```python
# %%
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_filt
er(select_district)


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


# %%
```
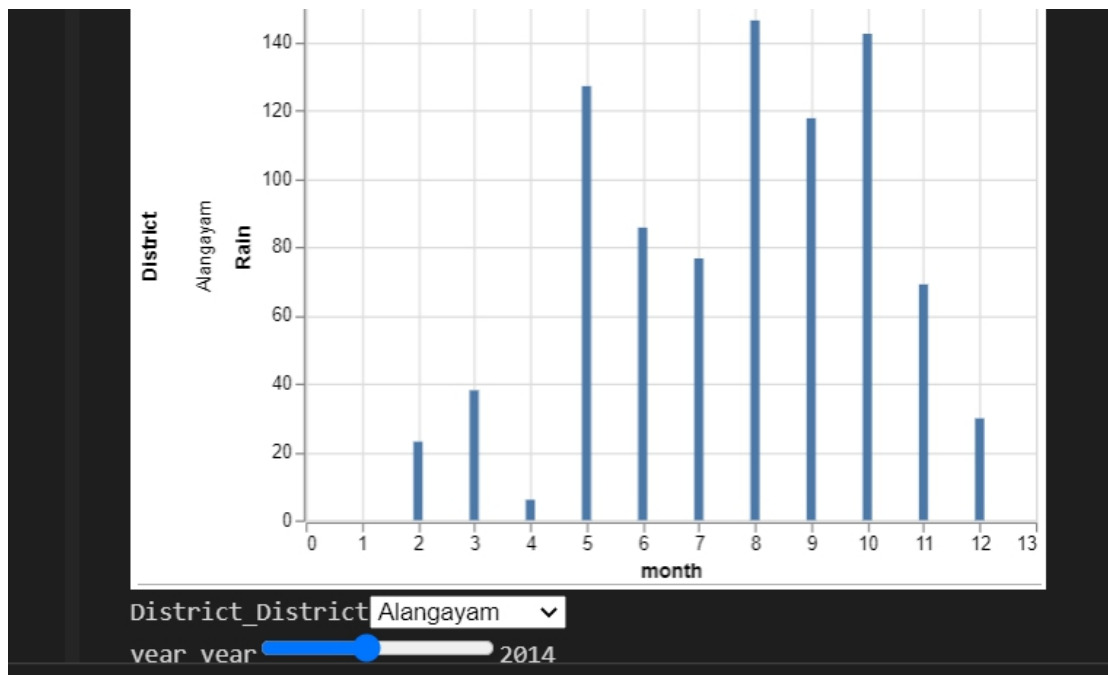
Detecting Trend and Seasonality in the Time-Series Data

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)

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.

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)
```
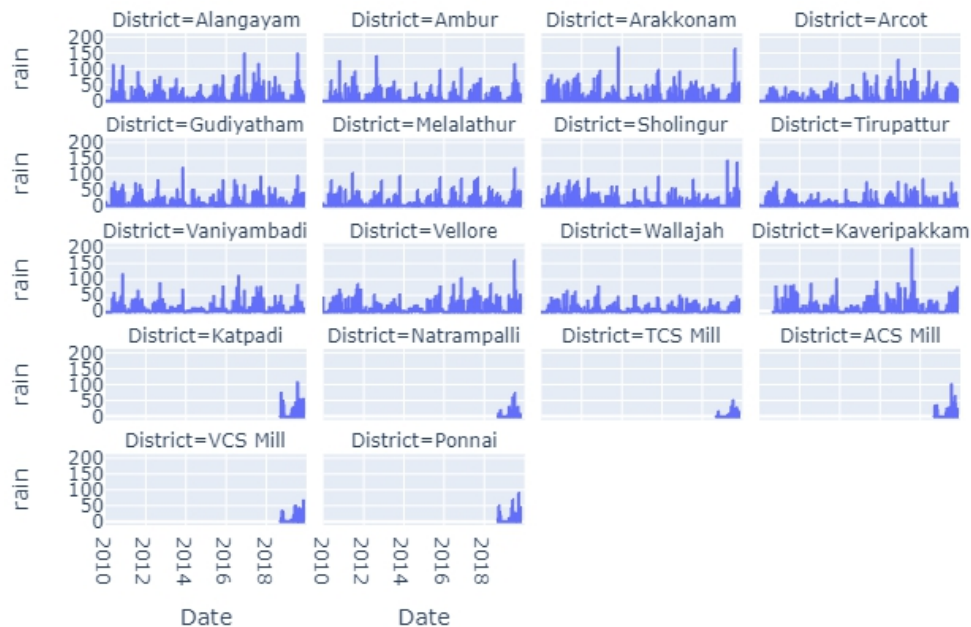

Although detecting seasonality from visualizations is not practical, it helps in detecting if there are any trends in the data. From the above visualization we can easily conclude that there is **no increasing or decreasing trend.**

Augmented Dicky 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 nonsta- tionary 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.**

<u>Using MaxLag=400 and AutoLag=None</u>

```python
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
```

```python
# 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
```

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.

Autoregressive

A statistical model is Autoregressive if it 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.

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 initalise this to 0.

ARIMA(7,0,1)

```python
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(disp=0)
#     output = model_fit.forecast()
#     yhat = output[0]
#     predictions.append(yhat)
#     obs = test_log_v[t]
#     train_log_v.append(obs)
```

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 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!

```python
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.

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.

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

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:

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

```python
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])
```

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

3.5206459947827082

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!

LSTM-Long-Short Term Memory

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.
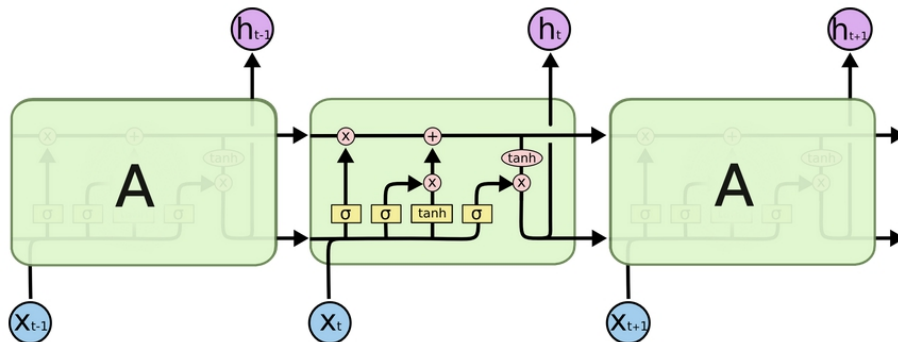
The repeating module in an LSTM contains four interacting layers.

Pink circles -> denote pointwise opeartion

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 happen such as a dot product.
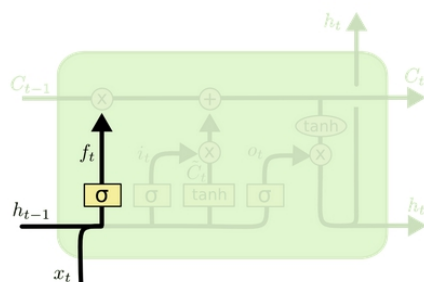
$$f_t = \sigma \left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f \right)$$

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.**
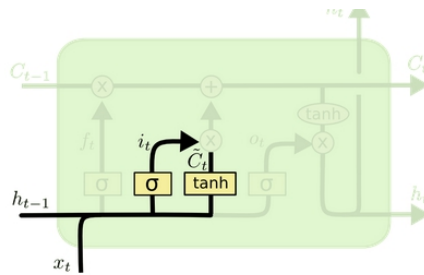
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

We go over our next gate, which involves using the previous value we have forgotten to create a new candidate for cell state Ct. Using the tanh activation aides in creating this new vector and updating our cell state.
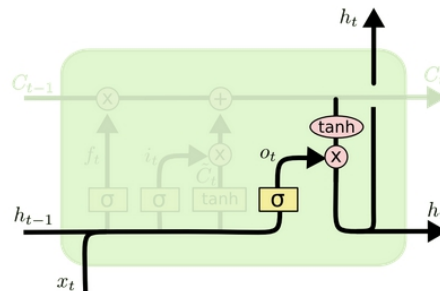
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

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.

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):

```python
        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

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!

<u>Feature Engineering and Xgboost gradient boosting</u>

What are the 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(Zt) 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**.

<u>Transforming our timeseries dataset to regular dataset</u>

We generate new features from out dataset using feature engineering. A total of **4 new features** are to be generated, the first 3 being easily obtained from the date feature.

- Month
- Year
- Day
- Is_rain

Month,Day and Year are obtained from stripping the date column using datetime library, whereas for Is_rain feature, we need to analyse our dataset.

```
count     46292.000000
mean          2.421041
std           8.384517
min           0.010000
25%           0.010000
50%           0.010000
75%           0.010000
max         201.400000
Name: rain, dtype: float64
```

- Max rainfall in mm is 201.40
- Min rainfall in mm is 0.01(I replaced all 0's to 0.01 to remove sparsity from our data)
- Average rainfall in mm is 2.42

My intuition is that the first decision our model should make is whether it has **rained or not** on that particular date, and if it has then **how much hard** did it rain? Hence we create **levels[0,1,2,3,4,5]**

- Level 0: **No Rainfall**
- Level 1: Has drizzled, less than average rainfall
- Level 2: Average rainfall not more than **3mm** has occurred
- Level 3: Has rained a good amount of more than **25mm**
- Level 4: Has rained heavily, more than **50mm**
- Level 5: Heavy rainfall **>100mm**

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.

Creating Linear Regression Baseline model

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 2.203

Although it seems an acceptable error for a baseline model, the predictions are not acceptable for our usecase, as the forecast contains **negative values!**

```
print(reg.predict(X_test))
print(min(reg.predict(X_test)))
print(max(reg.predict(X_test)))

[-0.57837406 -0.69231057 29.93379084 ... -1.09670694 -0.64170013
 14.06976681]
-1.404206625853364
38.66818389907459
```

Support vector Regression Model

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

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.

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.

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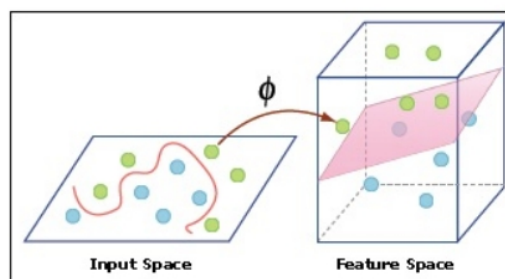
SVR utilizes almost the same algorithm, except that instead of classifying data points, we use the decision boundary lines and hyperplane to find the best fit line for our regression usecase. We assume a distance "c"(epsilon) between the hyperplane and decision boundary line.

Hyperplane: y=wx+b

Decision boundary lines: wx+b= c, wx+b= -c
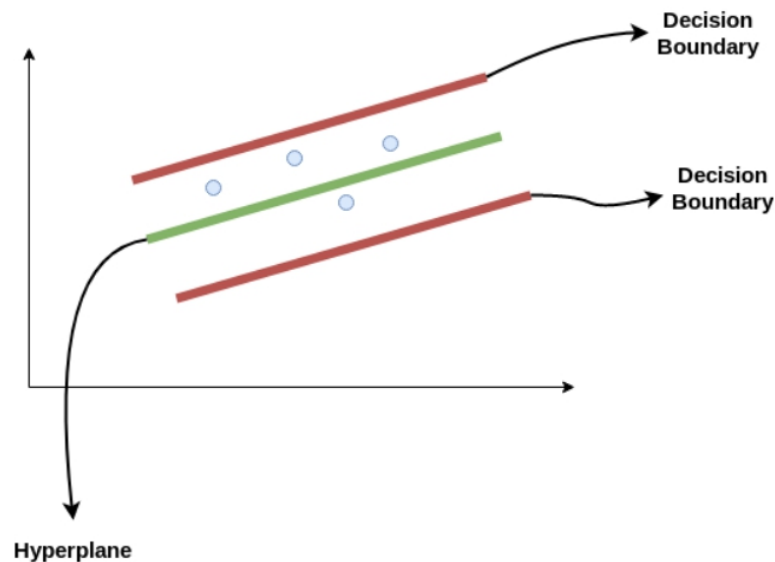
Now that we are clear in the theory behind SVR, it is quite simple to implement it using Scikit-learn library.

from sklearn.svm import LinearSVR

eps = 5

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

**Mean Absolute Error for test data->2.95**

```
[57]  ▷  ▶☰  M↓
        print(min(xt))
        print(max(xt))
        print(xt)

      -3.0449957832145667
      37.29391387465892
      [-1.84043347  0.13173469 31.2027917  ...  2.07642433  2.44550631
       17.27458292]
```

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

XGBOOST REGRESSOR

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.

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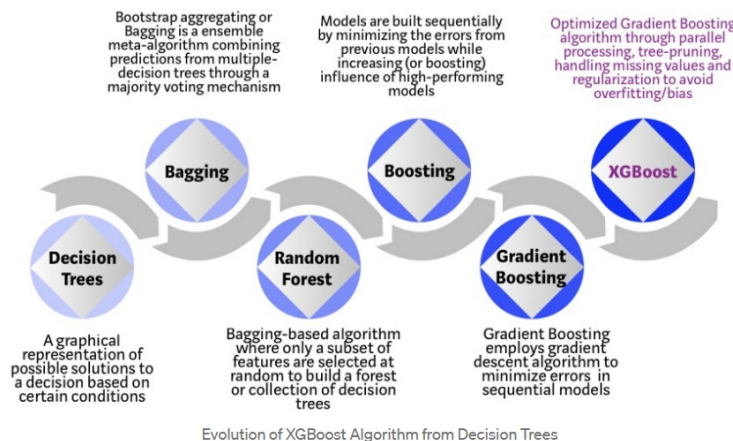
XGBoost enhances Gradient boosting algorithm by reducing the training time significantly via hardware optimizations. XGBoost also has **sparse data** handling capabilities, which is perfect for our dataset, it does not simply replace zeroes with the averages, but by **"learning"** best fit values for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

We implement XGBoost using its open-source library, with hyperparameters n_esitmators 50(the number of trees) and max_depth=20 to see how much depth a tree is allowed to go.

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

        n_estimators = 50, seed = 123,max_depth=20,eta=0.7)

Mean Absolute Error of test data -> 0.899

The model has learned to predict that when it is not raining it will be return 0.01001 to make sure the predictions are not sparse.

We will utilize this model along with other models in upcoming dashboard.

Model	Test Accuracy(MAE)
ARIMA model	4.143
Exponential Smoothing	3.52
LSTM	2.955
Support Vector Regression	2.203 2.95
Linear Regression	2.203
XGBoost Regression	0.899

Internship Project Abstract(TRIMMED DOWN AS PER TWIMBIT PRIVACY POLICY)

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.

Literature Review

Author	Title	Technology/Techniques used	Abstract	Dataset

A.C. Santha Sheela ,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 ,Isakh 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 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	Academic abstracts, Wan Fang med Database

			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	
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)	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.	Web pages, MIT articles, Financial pages, clothing brands

REFERENCES

[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).

Detailed Design

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

Database: Raw File Structure in AWS using CSVs

Hardware requirements: NONE

Software requirements:
- Chrome
- Flask
- Postman
- Dash/Plotly
- NLP Libraries(NlTK,BERT..etc)
- Selenium and Beautifulsoup