CS6301 – Machine Learning Laboratory

Rain prediction and crop plantation for maximum yield

Team Members:

NAME	REG.NO
PRAVIN K	2018103050
VIGNESH P	2018103080

Table of Contents

1. ABSTRACT:

Agriculture is one of the major sectors in Indian economy. Two-third of population is dependent on agriculture directly or indirectly. Annual rainfall plays a major role in cultivation of crops. Due to the unpredictable climatic changes, farmers are struggling to obtain a good amount of yield from the crops. Thus predicting rainfall could be really useful in case of getting maximum yield. All parts of India receive rainfall in different time of the year. Mainly the monsoon in India is classified into south-west and north east monsoon. Our task is to predict rainfall in a particular area in upcoming years. Then with the rainfall suggest better crop for plantation to get maximum yield.

The rainfall prediction is a time series prediction. So a model named SARIMAX is used to predict rainfall by reading the previously recorded rainfall at a particular place. The forecast of rainfall will be really helpful in predicting the crop production since the yield of crop is dependent on rainfall for some extent. SVR (support vector regression) model is used to predict the production of the crop based on the previously predicted monsoon rainfall. This work will ensure that the farmers get the maximum profit by selecting the right choice of crop for cultivation.

2. INTRODUCTION:

Agriculture is said to be the backbone of Indian economy. Over70% of the population is involved in agriculture and related fields. The forecast for agriculture yield is an important but also a tedious job in many countries. Due to changing climatic conditions the prediction of rainfall seems to be really difficult. Farmers are struggling to get a profitable production these days due to the unpredictable weather. Another major problem is diminishing cultivable land due to the raising population. The traditional techniques and level of infrastructure in the county also affect the crop production rate.

India has three cropping seasons — Rabi, Kharif and Zaid. We mainly focus on the kharif crops. Kharif crops are grown with the onset of monsoon in different parts of the country and these are harvested in September-October. Important crops grown during this season are paddy, maize, Jowar, bajra, tur (arhar), moong, urad, cotton, jute, groundnut and soyabean. Some of the most important rice-growing regions are Assam, West Bengal, coastal regions of Odisha, Andhra Pradesh, Telangana, Tamil Nadu, Kerala and Maharashtra, particularly the (Konkan coast) along with Uttar Pradesh and Bihar. The southwest monsoon from June to July plays an important role in crop production rate.

Since monsoon plays a vital role in Kharif crop cultivation we can use machine learning techniques to forecast monsoon rainfall. By predicting the monsoon rainfall we can also predict the crop production rate based on the monsoon rainfall for that year. We use a model named SARIMAX (Seasonal Auto Regressive Integrated Moving Average Exogenous model). The model predicts the future rainfall on analyzing the previous history of rainfall. Then the forecasted rainfall is used to predict the crop production. We use SVR (support vector regression) for predicting the crop production.

2.1. Models

SARIMAX MODEL:

SARIMAX stands for seasonal auto regressive integrated moving average with exogenous factors. In an auto regressive (AR) model the model predicts the next data point by looking at previous data points and using a mathematical formula similar to linear regression. There is something called an order (represented by "p") that determines how many previous data points will be used. When the p value is higher the model takes into account more data points that occurred a longer time ago.

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

The "c" is a constant and the "e" stands for error or noise.

SVR MODEL:

SVR (support vector regression) is a powerful algorithm that allows us to choose how tolerant we are of errors, both through an acceptable error margin(ϵ) and through tuning our tolerance of falling outside that acceptable error rate. In most linear regression models, the objective is to minimize the sum of squared errors. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.

3. LITERATURE SURVEY:

1. **Heuristic Prediction of Rainfall Using Machine Learning Techniques**. Chandrasegar Thirumalai, M Lakshmi Deepak, K Sri Harsha, K Chaitanya Krishna- published on 02 June 2017.
   Features: Linear regression analysis is used for predicting the unknown value of a season from the known value of another season.
   Advantages: Using the linear regression method which suggests the lower correlation between the various crop seasons and data results which we were are predicted are solely done based on the previous year's data.
   Disadvantage: prediction of particular season is depend on other seasons, so the data are limited, which makes it less accurate , it can be further enhanced .

2. **Rainfall Prediction: A Deep Learning Approach.**
   Emilcy Hern´andez, Victor Sanchez-Anguix, Vicente Julian, Javier Palanca, and N´estor Duque -published 02 June 2016.
   Features : This paper has presented a deep learning approach based on the use of auto-encoders and neural networks to predict the accumulated precipitation for the next day.
   Advantages: The results suggest that our proposed architecture outperform other approaches in terms of the MSE and the RMSE**.**
   Disadvantage: In this paper, they predict the rainfall for heavy rain scenario, this is not well modeled for heavy rain scenario, and also not well for seasonal rainfall prediction.

3. **Prediction Of Rainfall Using Machine Learning Techniques.** Moulana Mohammed, Roshitha Kolapalli, Niharika Golla, Siva Sai Maturi – published on 01, January 2020.
   Feature: The predictive model is used to prediction of the precipitation. The first step is converting data in to the correct format to conduct experiments then make a good analysis of data and observe variation in the patterns of rainfall.
   Advantage: Estimation of rainfall and it is estimated that SVR is a valuable and adaptable strategy, helping the client to manage the impediments relating to distributional properties of fundamental factors, geometry of the information and the normal issue of model over fitting.
   Disadvantage: they concentrated more in short term rainfall, low accuracy in long term rainfall falls in wrong prediction of crop production.

4. **A Study on Crop Yield Forecasting Using Classification Techniques:** R.Sujatha, Dr.P.Isakki- published on 9 Jan. 2016.
   Features: This paper has presented lots of machine learning methods like Rule Based Classifiers, Bayesian Networks, Nearest Neighbor, Support Vector Machine, Decision Tree, Artificial Neural Network, Rough Sets, Fuzzy Logic, and Genetic Algorithms
   Advantages: This paper presents new research possibilities for the application of new classification methodologies to the problem of yield prediction. Using these techniques, the crop yield can be improvised and increase the income level of the farmer, will be increased.

4. PROBLEM STATEMENT

The crop yield prediction is one of the most desirable yet challenging tasks for every nation. Nowadays, due to the unpredictable climatic changes, farmers are struggling to obtain a good amount of yield from the crops. To feed the increasing population of India, there is a need to incorporate the latest technology and tools in the agricultural sector. Rain is a major factor affecting crop yield. Since many factors contribute to rainfall it is a difficult task to predict rain. If we predict rain then it will be really useful to predict the crop yield since both are related to each other to some extent.

5. PROBLEM SOLUTION:

5.1 Dataset:

IMD dataset (rain fall dataset):

IMD New High Spatial Resolution (0.25X0.25 degree) Long Period (1901-2018) Daily Gridded Rainfall Data Set over India. This data product is a very high spatial resolution daily gridded rainfall data (0.25 x 0.25 degree). The unit of rainfall is in millimeter (mm). Data available for 118 years, from 1901 to 2018. Data is arranged in 135x129 grid points. The first data in the record is at 6.5N & 66.5E, the second is at 6.5N & 66.75E, and so on. The last data record corresponds to 38.5N & 100.0E. The yearly data file consists of 365/366 records corresponding to non-leap/ leap years.

Link: https://www.imdpune.gov.in/Clim_Pred_LRF_New/Grided_Data_Download.html

Crop production dataset:

The dataset consist of crop production in each state for variety of crops. The year wise data is available for crop production. The dataset consist of production in tones, area in hectares crop variety and place were crops are cultivated. The dataset is downloaded from an official government website.

Link: https://community.data.gov.in/average-monthly-wholesale-price-of-sesamum-kharif-crop-from-october-2017-to-november-2018/

5.2. APPROACH:

Machine learning approach is followed to predict the rain and crop yield. At first the rainfall data is collected from IMD (Indian meteorological department). The dataset consist of information regarding the amount of rainfall. Using the data we follow Seasonal Autoregressive Integrated Moving Average (SARIMA) method to predict rain fall. Then we use another dataset which has information on crop production. Then appropriate features are selected from the dataset. Finally we use a support vector regression model to predict the production of crops in future with the help of predicted rainfall.

6. Novelty:

Our novelty lies on predicting crop yield in a particular place with higher accuracy and faster model compared to linear regression methods. In previous works they concentrated on predicting rainfall in particular states or in whole India. This is unhelpful method to predict crop yield in a particular place. Our work is concentrated in prediction of rainfall in particular place which is not done in previous methods. The present study considers predicting and forecasting rainfall for any small meteorological sub-divisions of India. We try to achieve higher accuracy than what we referred for our literary survey (Heuristic Prediction of Rainfall Using Machine Learning Techniques & Prediction Of Rainfall Using Machine Learning Techniques) etc.

7. ARCHITECTURE:

7.1 ARCHITECTURE DIAGRAM

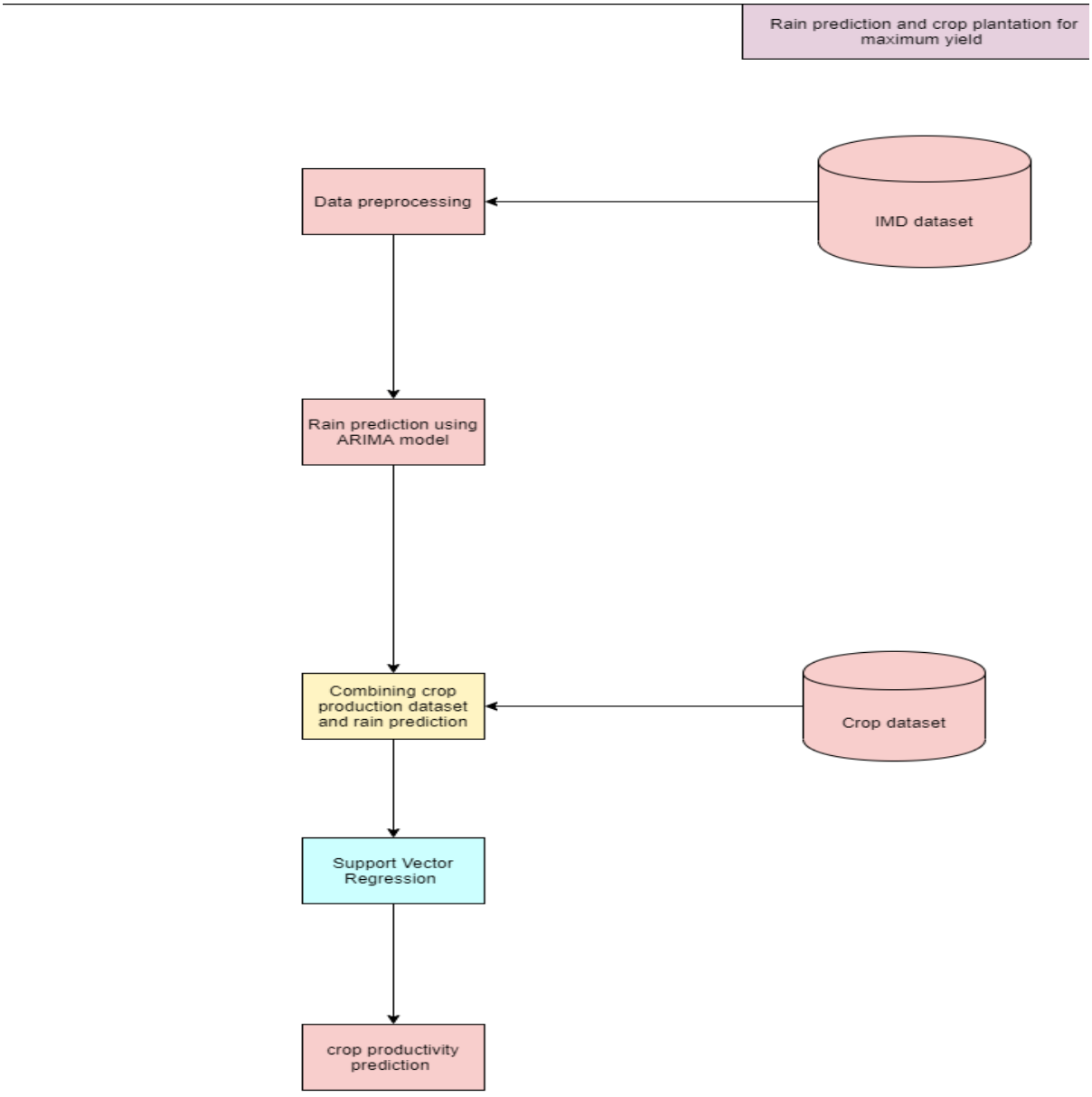


Fig: 7.1 architecture diagram

7.2 EXPLANATION OF ARCHITECTURE DIAGRAM

The fig 7.1 shows the entire architecture diagram of the system. First the IMD rainfall dataset is collected from official IMD website. Then the data undergoes a preprocessing stage where the data in gridded format is converted into CSV format. The data for the required place is collected with the help of matching the coordinates. On the next step the preprocessed data is fed into the SARIMAX model where the forecast of rainfall is done. At this stage we are able to predict the rainfall for the particular region. Then the previously preprocessed rainfall data is added to the crop dataset. The features in the combined dataset is altered so that the predictions could become more accurate. Then a support vector regression is performed over the crop production dataset. With the help of previously predicted monsoon rainfall the production rate of crops can be calculated for the upcoming years.

8. DETAILED MODULE DESIGN

8.1 Module1:

Data preprocessing

Input: IMD Dataset

Process:

The data is collected from the official IMD dataset. The data consist of rainfall information measured in mm for the past 100 years in grid format. From grid format the data is converted to CSV format. The unrecorded data are neglected. Then the rain details for particular place is gathered over a period. Then our main aim is to pair the data according to the period they belong to. (Eg: the rainfall data of June month is grouped together).

Output: processed data

8.2 Module 2:

Rain prediction using SARIMAX model:

Input: preprocessed data

Process:

Autoregressive Integrated Moving Average, or ARIMA, is one of the most widely used forecasting methods for univariate time series data forecasting. Although the method can handle data with a trend, it does not support time series with a seasonal component. An extension to ARIMA that supports the direct modeling of the seasonal component of the series is called SARIMA. The pre processesd data is sent to the SARIMAX model which is a model for time series prediction. The model gets trained with the input data and able to predict annual rainfall for the upcoming years.

output: trained model for rainfall prediction

8.3 Module 3:
Combining crop production dataset and rain prediction:

Input: crop production data

Process:

In this module the data for crop production is collected. Then the crop productivity data is combined with the predicted rainfall. The crop productivity data for each year is combined with the rainfall data of the year. This combination of data helps in predicting the changes in crop production accordingly with respect to rainfall amount.

Output: processed data with rainfall data.

8.4 Module 4:
Crop productivity prediction using SVR:

Input: preprocessed data.

Process:

Since this problem falls under regression category we use Support vector regression (SVR). Support vector regression (SVR) and support vector machine (SVM) use the same concepts with a few dissimilarity. For solving regression problems, we can extend the methodology used by support vector classification. Constructing a support vector classification model depends only on a subset of training data because the cost function for constructing the support vector classification model does not consider training points beyond a certain boundary. Similarly, a subset of data is required for building a support vector regression model.

Support vector regression formulas:

$$y_i - wx - b \leq \varepsilon + \xi_i$$
$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

Where x is inputs, w – weights, ξ_i is the slack variable and epsilon is the distance between hyper planes.

$$y = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*)(x_n \cdot x) + b$$

where α_n, α_n^* are nonnegative multipliers with for each observation x_n and $x_n.x$ is the dot product.

Output: crop production predicting model.

9. IMPLEMENTATION:

9.1 Module 1:

Pre processing IMD dataset

Downloading gridded data

At first the dataset is downloaded from the official IMD data base by using an in build module name imdlib. The data is extracted in a 0.25*0.25 grid format. The grid data consist of daily rainfall measured in mm over particular coordinate location. We consider the dataset from 1901 to 2020.

In [ ]:
```
import imdlib as imd
import numpy as np
import pandas as pd

start_yr = 2016
end_yr = 2020
variable = 'rain'
file_format = 'yearwise'
imd.get_data(variable, start_yr, end_yr, fn_format='yearwise', file_dir='E:/data/')
file_dir = 'E:/data/'
data = imd.open_data(variable, start_yr, end_yr,'yearwise', file_dir)
```

Data format conversion

The data in the gridded format is converted to a CSV file where it consist of daily rainfall data measured in mm according to the coordinate location. The data is available for every 0.25 resolution. The unavailable data is marked as -9999. Likewise all the gridded files (from the year 1925 to 2020) are converted into CSV files for better understanding and accessibility.

In [ ]:
```
grid_size = 0.25
y_count = 129
x_count = 135
x = 66.5
y = 6.5

print(data)
print(data.shape)
np_array = data.data
years_no = (end_yr - start_yr) + 1
day = 0
for yr in range(0,years_no):
    f = open("E:/data/"+str(start_yr+yr)+"_"+str(variable)+".csv",'w')
    if ((start_yr+yr) % 4 == 0) and ((start_yr+yr) % 100 != 0):
        days = 366
        count = yr + days
    elif ((start_yr+yr) % 4 == 0) and ((start_yr+yr) % 100 == 0) and ((start_yr+yr) % 400 == 0):
        days = 366
        count = yr + days
    else:
        days = 365
        count = yr + days

    day = day + days

    f.write("X,Y,")
    for d in range(0, days):
        f.write(str(d+1))
        f.write(",")
    f.write("\n")

    for j in range(0, y_count):

        for i in range(0, x_count):

            f.write(str((i * grid_size) + x))
            f.write(",")
            f.write(str((j * grid_size) + y))
            f.write(",")
            time = 0
            for k in range(day-days, day):

                val = np_array[k,i,j]
                if val == 99.9000015258789 or val == -999:
                    f.write(str(-9999))
                    f.write(",")
                else:
                    f.write(str(val))
                    f.write(",")


            f.write("\n")
    print("File for " + str(start_yr + yr) + "_" + str(variable) + " is saved")
print("CSV conversion successful !")
```

Data represented in CSV format

The CSV file which has the information about the rainfall in 2015 is converted into a data frame using pandas. The X and Y labels represent the coordinates of the location for which the daily rainfall for 365 days is recorded in mm. Since the below mentioned coordinates doesn't have a weather stations, the value is marked as -9999 because of the unavailability of the data.

```
In [118]: data1 = pd.read_csv("2015_rain.csv")
          data1.head()
```

Out[118]:

	X	Y	1	2	3	4	5	6	7	8	...	357	358	359	360	361	362	363	364	3
0	66.50	6.5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	...	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-999
1	66.75	6.5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	...	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-999
2	67.00	6.5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	...	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-999
3	67.25	6.5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	...	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-999
4	67.50	6.5	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	...	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-999

5 rows × 368 columns

Finding coordinates for particular region

The coordinates of the particular region is gathered from the following code. An inbuild module named geopy is used to find the coordinates. If we enter a particular location name the program returns the coordinates. The coordinates are rounded off to quarter since the available data is in 0.25 *0.25 resolution and stored into variables.

```
In [102]: from geopy.geocoders import Nominatim
          address=input("enter address/area")
          geolocator = Nominatim(user_agent="Your_Name")
          location = geolocator.geocode(address)
          print(location.address)
          print((location.latitude, location.longitude))
          y=roundPartial(location.latitude,0.25)
          x=roundPartial(location.longitude,0.25)
          print(x,y)
```

```
enter address/areasankarapuram
Sankarapuram, Kallakurichi District, Tamil Nadu, India
(11.8555913, 78.98579670754442)
79.0 11.75
```

Round off function

```
In [100]: def roundPartial (value, resolution):
              return round (value / resolution) * resolution
```

Collecting data for particular coordinates

The yearly rainfall for the coordinates is collected from each year CSV file and stored in a 2D array.

```
In [109]: import numpy
          import numpy as np
          import pandas
          import pandas as pd
```

```
In [112]: loc_data = []

          for i in range(1925,2021):
              name=str(i)+"_rain.csv"
              result=f1(name)
              print(result.shape)
              #loc_data = numpy.concatenate([loc_data, result])
              loc_data.append(result)
          print(loc_data)
```

```
13.278441, 8.371855, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 12.527720, 0.000000, 14.229252, 3.807090, 1.568849,
0.368957, 0.000000, 3.350026, 1.612350, 8.695511, 0.344192,
2.694585, 17.835670, 0.242570, 6.085267, 2.074666, 5.586050,
0.000000, 1.648039, 5.061123, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 5.262785, 5.085267, 8.960577, 0.000000,
0.000000, 0.000000, 0.000000, 0.153668, 1.903241, 0.000000,
0.000000, 3.139331, 3.939596, 3.095710, 0.000000, 0.000000,
5.146143, 4.954653, 0.000000, 1.626799, 0.206238, 0.000000,
0.000000, 2.468961, 0.519781, 20.189331, 0.788751, 5.163855,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 1.848635, 6.157545, 0.000000, 0.000000,
17.919123, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
1.819234, 0.000000, 0.000000, 40.283371, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.554013, 9.078800, 8.039081, 11.237563, 6.883322, 7.693997,
0.519781, 0.214326, 10.966660, 39.195713, 9.628024, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 11.928061, 8.320742,
9.667926, 9.377707, 9.644723, 17.316721, 25.090126, 2.106149,
6.549122, 0.000000, 0.000000, 0.000000, 16.929470, 19.926537,
4.758866, 0.000000, 2.033499, 0.529942, 0.000000, 0.000000,
```

Function used to return data for particular year

```
In [ ]: def f1(name):
            data = pd.read_csv(name)
            arr = numpy.array(data)
            temp=-1
            for i in range(1,len(data)):
                if arr[i][0]==x and arr[i][1]==y:
                    temp=i

            return arr[temp]
```

Daily data is converted to monthly data

The conversion of data into monthly data will better the performance of the upcoming model.

```
In [116]: monthwise = np.zeros((len(loc_data), 12))
          monthwise

Out[116]: array([[0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000],
                 [0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000],
                 [0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000],
                 ...,
                 [0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000],
                 [0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000],
                 [0.000000, 0.000000, 0.000000, ..., 0.000000, 0.000000, 0.000000]])

In [117]: normal=[31,59,90,120,151,181,212,243,273,304,334,365]
          temp = 0
          for i in range(len(loc_data)):
              for j in range(12):
                  m = normal[j]
                  for k in range(temp,m):
                      monthwise[i][j]+=loc_data[i][k]
                  temp = k
              temp = 0
          np.set_printoptions(formatter={'float_kind':'{:f}'.format})
          print(monthwise)

          [[18.288161 0.000000 23.240082 ... 162.411164 246.001328 241.771671]
           [267.169352 0.000000 17.341101 ... 161.956287 138.496477 22.577448]
           [3.498812 6.029912 0.000000 ... 68.237540 154.705995 54.810256]
           ...
           [0.235357 4.880067 1.216100 ... 167.264812 122.903497 30.386879]
           [0.000000 0.000000 0.000000 ... 166.112482 112.617024 160.119071]
           [16.065397 0.000000 0.000000 ... 176.244304 229.837529 358.965525]]
```

Final output of module 1

N * M dimension array where N represents the no of year (from 1925 to 2020) and M represents the months (from Jan to Dec).

9.2 Module 2:

Building SARIMAX (seasonal auto regression integrated moving model)

Seasonal rainfall conversion

The conversion of monthly data into seasonal data since the model works on seasonal model

```
In [47]: import numpy as np
         monsoon = np.zeros((len(loc_data), 2))
         year = 1925
         for i in range(len(a)):
             for k in range(4):
                 monsoon[i][1] = monsoon[i][1]+a[i][k]
             monsoon[i][0] = year
             year+=1
         np.set_printoptions(formatter={'float_kind':'{:f}'.format})
         print(monsoon)

         [[1925.000000 383.063445]
          [1926.000000 490.050053]
          [1927.000000 609.868194]
          [1928.000000 459.301587]
          [1929.000000 331.977953]
          [1930.000000 362.162289]
          [1931.000000 598.875689]
          [1932.000000 257.974751]
          [1933.000000 377.614581]
          [1934.000000 391.576580]
          [1935.000000 497.844003]
          [1936.000000 364.453524]
          [1937.000000 446.746322]
          [1938.000000 718.607205]
          [1939.000000 282.379276]
          [1940.000000 361.794965]
          [1941.000000 287.189242]
          [1942.000000 452.032996]
          [1943.000000 346.187229]
```

The first column refers to the year and the second column refers to the annual rainfall.

Importing required modules to build the model

```
In [12]: %matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
from dateutil.relativedelta import relativedelta
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
In [13]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
from dateutil.relativedelta import relativedelta
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.seasonal import seasonal_decompose
```

Importing the data using pandas and describing the data

```
In [14]: rain = pd.read_csv('pravin.csv',parse_dates = ['YEAR'])

rain['YEAR'] = pd.date_range(start='1925', end='2020',freq = 'AS')
rain = rain.set_index(['YEAR'])
```

```
In [15]: rain.describe()
```

Out[15]:

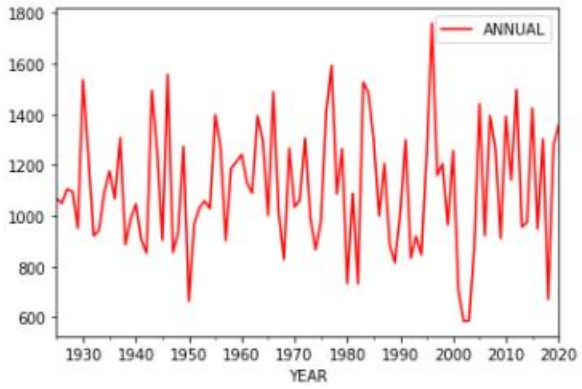
	ANNUAL
count	96.000000
mean	1109.104927
std	243.131380
min	585.152747
25%	930.552916
50%	1076.987850
75%	1274.183201
max	1758.393105

Plotting (seasonal rainfall in mm)

X axis represent the year (1925 – 2020)

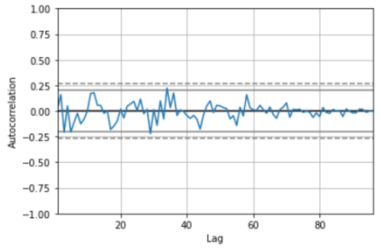
Y axis represents seasonal rainfall in mm (Vellore district)

```
In [21]: rain.plot(color="red")
Out[21]: <AxesSubplot:xlabel='YEAR'>
```



Auto correlation of data

```
In [60]: from pandas.plotting import autocorrelation_plot
         autocorrelation_plot(rain['monsoon'])
         plt.show()
```



Building SARIMAX model

Inputs: seasonal rainfall

SARIMAX is a time series model. The warning occurs due to the unspecified frequency. It can be ignored since the required frequency AS-JAN is set as default.

```
In [67]: model1=sm.tsa.statespace.SARIMAX(rain['monsoon'],order=(0,1,1),seasonal_order=(1,1,0,12),trend='n')
         result=model1.fit()
         result.summary()
```

```
C:\Users\Public\anaconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency i
nformation was provided, so inferred frequency AS-JAN will be used.
  warnings.warn('No frequency information was'
C:\Users\Public\anaconda3\envs\tensorflow\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency i
nformation was provided, so inferred frequency AS-JAN will be used.
  warnings.warn('No frequency information was'
```

Out[67]: SARIMAX Results

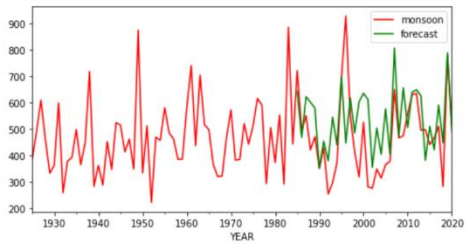
Dep. Variable:	monsoon	**No. Observations:**	96
Model:	SARIMAX(0, 1, 1)x(1, 1, [], 12)	**Log Likelihood**	-545.869
Date:	Fri, 14 May 2021	**AIC**	1097.738
Time:	19:25:57	**BIC**	1104.994
Sample:	01-01-1925	**HQIC**	1100.653
	- 01-01-2020		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.9998	6.559	-0.152	0.879	-13.855	11.856
ar.S.L12	-0.4280	0.121	-3.527	0.000	-0.666	-0.190

Training data

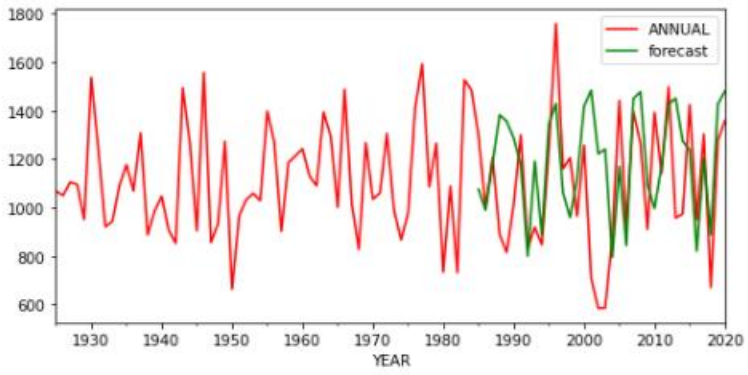
The red line represents the actual rainfall and the green line represents the predicted Output from the model.

```
In [68]: rain['forecast']=result.predict(start=60,end=96,dynamic=True)
         rain[['monsoon','forecast']].plot(figsize=(8,4),color=("red","green"))
Out[68]: <AxesSubplot:xlabel='YEAR'>
```




```
In [69]: from pandas.tseries.offsets import DateOffset
         future_dates=[rain.index[-1]+ DateOffset(years=x)for x in range(0,31)]
```

```
In [30]: df['forecast']=result.predict(start=60,end=96,dynamic=True)
         df[['ANNUAL','forecast']].plot(figsize=(8,4),color=("red","green"))
Out[30]: <AxesSubplot:xlabel='YEAR'>
```



Creating data frame to store the value of forecasted and previous seasonal rainfalls

```
In [69]: from pandas.tseries.offsets import DateOffset
         future_dates=[rain.index[-1]+ DateOffset(years=x)for x in range(0,31)]

In [70]: y_a = rain['forecast']
         y_b= rain['monsoon']

In [71]: y_pred=y_a.tail()
         y_test=y_b.tail()

In [72]: y_test

Out[72]: YEAR
         2016-01-01    465.994810
         2017-01-01    510.218608
         2018-01-01    281.190408
         2019-01-01    778.448768
         2020-01-01    512.833111
         Name: monsoon, dtype: float64

In [73]: future_datest_df=pd.DataFrame(index=future_dates[1:],columns=rain.columns)

In [74]: future_datest_df.tail()

Out[74]:
```

	monsoon	forecast
2046-01-01	NaN	NaN
2047-01-01	NaN	NaN

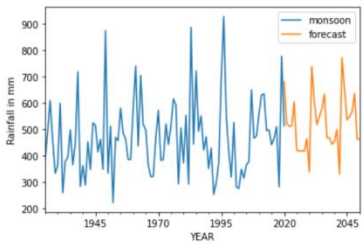
Plotting results

The data frame consists of actual and predicted data.

Here in the graph the blue line refers to the actual rainfall data from the year(1925-2020)

The orange line refers to the forecasted rainfall data from the year (2021-2050)

```
In [76]: future_df['forecast'] = result.predict(start = 95, end = 150, dynamic= True)
         ax =future_df['1925':].plot(label='observed')
         ax.set_xlabel('YEAR')
         ax.set_ylabel('Rainfall in mm')

Out[76]: Text(0, 0.5, 'Rainfall in mm')
```




```
In [77]: from sklearn import metrics
```

Parameter of analysing model

A basic program for calculating mean absolute error since it is a time series regression model. The mean absolute error is 65.20 mm rainfall annually.

```
In [77]: from sklearn import metrics

In [78]: print(metrics.mean_absolute_error(y_test, y_pred))

         65.20856480797116
```

Predicting rainfall

The forecasted data is stored in a data frame. Here is the predicted rainfall of Vellore measured in mm for the year 2022.

	monsoon	forecast
1925-01-01	383.063445	NaN
1926-01-01	490.050053	NaN
1927-01-01	609.868194	NaN
1928-01-01	459.301587	NaN
1929-01-01	331.977953	NaN
...	...	...
2046-01-01	NaN	547.809080
2047-01-01	NaN	566.133905
2048-01-01	NaN	636.634798
2049-01-01	NaN	462.746650
2050-01-01	NaN	462.195417

126 rows × 2 columns

```
In [80]: print('predicted rainfall for 2022:')
         mon = future_df['forecast'][96]
         print(mon)

         predicted rainfall for 2022:
         522.597061235427

In [81]: crop = pd.read_csv('Tamilnadu agriculture yield data.csv')
```

Predicted rainfall: 522.59 mm rainfall in Vellore (2022)

Final output of module

Rainfall forecast for the upcoming years(2022 to 2050) for a particular region.

Crop production dataset:

The dataset consist of crop production in each state for variety of crops. The year wise data is available for crop production. The dataset consist of production in tones, area in hectares crop variety and place were crops are cultivated. The dataset is downloaded from an official government website.

Link: https://community.data.gov.in/average-monthly-wholesale-price-of-sesamum-kharif-crop-from-october-2017-to-november-2018/

9.3 Module 3: Combining crop production dataset and rain prediction:

Data are read from dataset and extracted particular region's information.

```
In [81]: crop = pd.read_csv('Tamilnadu agriculture yield data.csv')
         crop.head()
```

Out[81]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Tamil Nadu	ARIYALUR	2008	Kharif	Rice	24574	NaN
1	Tamil Nadu	ARIYALUR	2008	Whole Year	Arhar/Tur	209	NaN
2	Tamil Nadu	ARIYALUR	2008	Whole Year	Bajra	565	NaN
3	Tamil Nadu	ARIYALUR	2008	Whole Year	Banana	190	NaN
4	Tamil Nadu	ARIYALUR	2008	Whole Year	Cashewnut	31113	NaN

```
In [82]: dv = crop[crop["District_Name"] == 'VELLORE']
         dv = dv[dv["Crop"]=='Rice']
         dv.head()
```

Out[82]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
12061	Tamil Nadu	VELLORE	1997	Whole Year	Rice	71607	236910.0
12080	Tamil Nadu	VELLORE	1998	Kharif	Rice	77109	263941.0
12105	Tamil Nadu	VELLORE	1999	Kharif	Rice	49868	190899.0
12131	Tamil Nadu	VELLORE	2000	Kharif	Rice	52332	163141.0
12150	Tamil Nadu	VELLORE	2001	Kharif	Rice	60113	234130.0

Rainfall data for that particular region is combined with that region's crop production data.

Out[107]:

	Crop_Year	ANNUAL	JUN	JUL	AUG	SEP	Monsoon
0	1925.0	1067.694015	42.459496	36.307954	170.673479	133.622517	383.063445
1	1926.0	1049.847534	55.883556	112.533191	125.942346	195.690960	490.050053
2	1927.0	1104.860059	105.889871	75.334561	110.369337	318.274425	609.868194
3	1928.0	1094.719877	50.459520	118.923222	148.439960	141.478886	459.301587
4	1929.0	951.141881	51.495967	85.379786	120.170266	74.931934	331.977953

```
In [108]: temp2 = pd.merge(dv, temp1, on='Crop_Year')
          temp2.head()
```

Out[108]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	ANNUAL	JUN	JUL	AUG	SEP	Monsoon
0	Tamil Nadu	VELLORE	1997	Whole Year	Groundnut	62236	109770.0	1159.910138	193.669942	78.710622	115.520254	162.004163	549.904981
1	Tamil Nadu	VELLORE	1998	Kharif	Groundnut	81520	155776.0	1204.956848	44.240445	65.324264	163.383730	140.210223	413.158662
2	Tamil Nadu	VELLORE	1999	Kharif	Groundnut	57658	114006.0	965.091221	75.129185	27.869697	161.238775	53.424634	317.662292
3	Tamil Nadu	VELLORE	2000	Kharif	Groundnut	57290	113913.0	1256.466951	85.893764	81.797634	144.624388	213.615412	525.931199
4	Tamil Nadu	VELLORE	2001	Kharif	Groundnut	57373	101310.0	708.281719	38.399017	71.720598	32.170898	138.094689	280.385202

```
In [109]: temp2['PA'] = temp2.apply(lambda row: row.Production / row.Area , axis = 1)
          temp2.head()
```

Calculate production per area using total production and area:

This simple calculation reduces the importance of area since it is directly proportional to the production. Then monsoon is used as a major feature to predict the crop production.

```
In [109]: temp2['PA'] = temp2.apply(lambda row: row.Production / row.Area , axis = 1)
          temp2.head()
```

Out[109]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	ANNUAL	JUN	JUL	AUG	SEP	Monsoon	
0	Tamil Nadu	VELLORE	1997	Whole Year	Groundnut	62236	109770.0	1159.910138	193.669942	78.710622	115.520254	162.004163	549.904981	1.7
1	Tamil Nadu	VELLORE	1998	Kharif	Groundnut	81520	155776.0	1204.956848	44.240445	65.324264	163.383730	140.210223	413.158662	1.9
2	Tamil Nadu	VELLORE	1999	Kharif	Groundnut	57658	114006.0	965.091221	75.129185	27.869697	161.238775	53.424634	317.662292	1.9
3	Tamil Nadu	VELLORE	2000	Kharif	Groundnut	57290	113913.0	1256.466951	85.893764	81.797634	144.624388	213.615412	525.931199	1.9
4	Tamil Nadu	VELLORE	2001	Kharif	Groundnut	57373	101310.0	708.281719	38.399017	71.720598	32.170898	138.094689	280.385202	1.7

```
In [110]: df = temp2.reindex(columns=['Crop_Year','Area','ANNUAL','JUN','JUL','AUG','SEP','Monsoon','Production','PA'])
          df.head()
```

Out[110]:

	Crop_Year	Area	ANNUAL	JUN	JUL	AUG	SEP	Monsoon	Production	PA
0	1997	62236	1159.910138	193.669942	78.710622	115.520254	162.004163	549.904981	109770.0	1.763770
1	1998	81520	1204.956848	44.240445	65.324264	163.383730	140.210223	413.158662	155776.0	1.910893
2	1999	57658	965.091221	75.129185	27.869697	161.238775	53.424634	317.662292	114006.0	1.977280
3	2000	57290	1256.466951	85.893764	81.797634	144.624388	213.615412	525.931199	113913.0	1.988357
4	2001	57373	708.281719	38.399017	71.720598	32.170898	138.094689	280.385202	101310.0	1.765813

Final output for module 3:

Pre-processed dataset with N rows where N represents no of years, with year, seasonal rainfall, area, production, production per area.

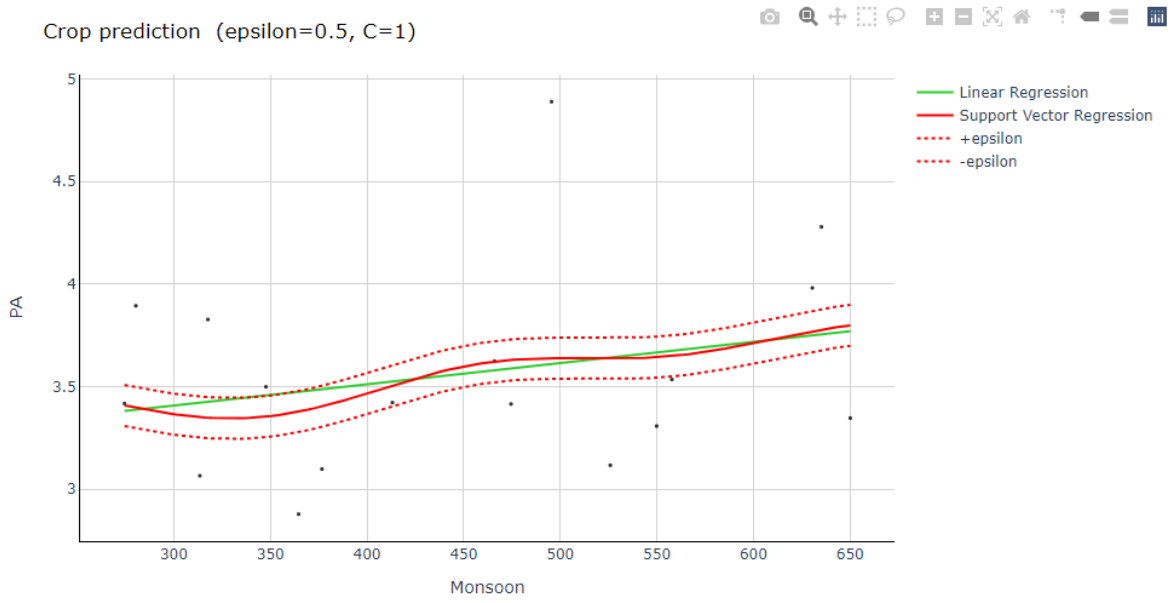
9.4. Module 4:
Crop productivity prediction using SVR:

Data pre-processed for different crops, and fit the support vector regression model and predicted the production (tone per hectare) for each crop. The prediction is for the rice crop production. The model produces a mean squared error of 0.2294.

```
In [150]: from sklearn.metrics import mean_squared_error
          X=df['Monsoon'].values.reshape(-1,1)
          y=df['PA'].values

          model1 = LinearRegression()
          lr = model1.fit(X, y)

          model2 = SVR(kernel='rbf', C=1, epsilon=0.5)
          svr = model2.fit(X, y)

          x_range = np.linspace(X.min(), X.max(), 100)
          y_lr = model1.predict(x_range.reshape(-1, 1))
          y_svr = model2.predict(x_range.reshape(-1, 1))
          y_svr1 = model2.predict(X)
          mean_squared_error(y, y_svr1)
```

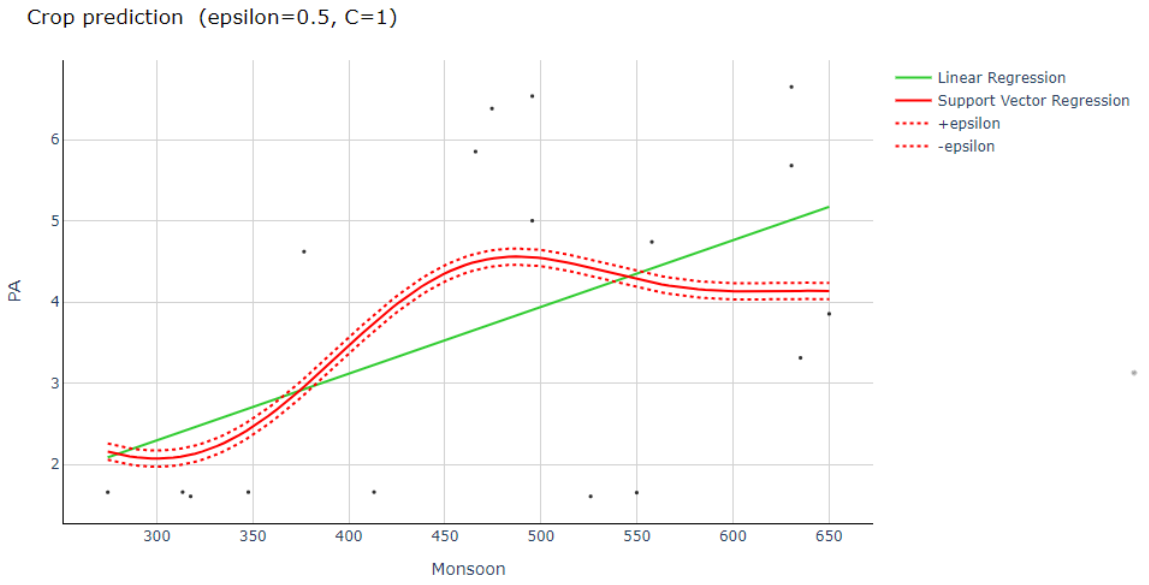
```
Out[150]: 0.2294395304742347
```

The prediction is plotted using a graph with the linear model prediction to compare the effectiveness of the SVR model.

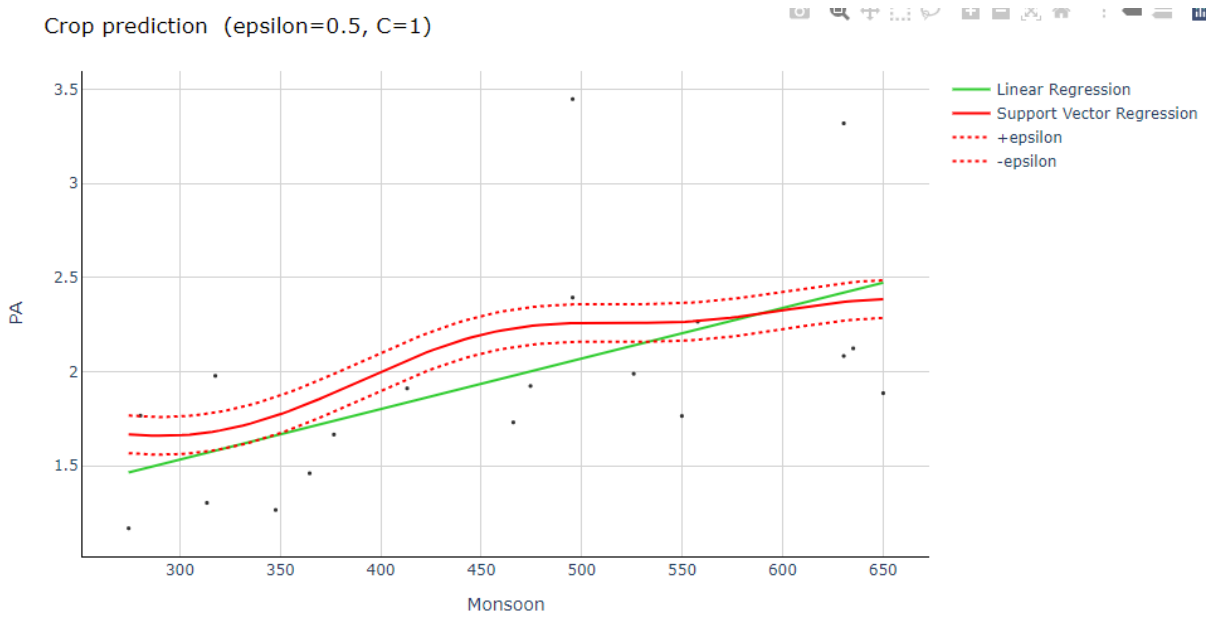


Prediction for rice production

Same pre-processing , model building, and result prediction is done for different crops.



Prediction for maize production



Prediction for groundnut production

Final output for module 4:

From the predictions of each crop we can get the production rate for particular amount of monsoon rainfall. Since we previously predicted the monsoon rainfall of the next year 2022, we could get the production rate for the crops in 2022. The below diagram shows the production of each crop for the next year. The best crop is chosen accordingly based on their market price. For example if rice is sold at a rate of 35000 per Tone then cost price of the crop can be calculated as (3.6380*35000 = 127330) per hectare of cultivation.

```
In [211]: print('rice(tone/hectare):')
          print(rice)
          print('Maize(tone/hectare):')
          print(maize)
          print('groundnut(tone/hectare):')
          print(gn)
```

```
rice(tone/hectare):
3.638082450574147
Maize(tone/hectare):
4.439634543137882
groundnut(tone/hectare):
2.2587454066051627
```

10. RESULT ANALYSIS:

SVR have hyper parameters C, and epsilon. C is Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty. Epsilon specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. Rice crop is taken for analyzing the results.

10.1 Analysis by changing parameters:

Changing values for C

C	epsilon	Mean Squared Error
0.5	0.5	0.33189
1	0.5	0.33435
5	0.5	0.31940
10	0.5	0.32270

Table 10.1

Changing values for epsilon

C	Epsilon	Mean Squared Error
5	0.25	0.31432
5	0.5	0.31940
5	1	0.45828
5	0.15	0.29792

Table 10.2

By analyzing the tables 10.1 and 10.2 the best value for parameters C, epsilon is 5 and 0.15 respectively. For these values of parameters the model produces least mean squared error.

10.2 Comparing with other models:

In this section the SVR model's performance is compared with the performance of

Linear regression and Random forest Regression.

CROP	Support vector Regression (Mean square error)	Linear regression (Mean square error)	Random forest regression (Mean square error)
Rice	0.19443	0.21554	0.25204
Maize	2.56997	3.68447	2.78506
Groundnut	0.22268	0.22770	0.46365

Table 10.3

Table 10.3 clearly shows that the SVR model has less mean squared error when compared with other models. Therefore SVR model predicts with more accuracy when compared with other models.

Fig 10.1 shows that SVR provides a better flexibility in prediction than the linear regression model.

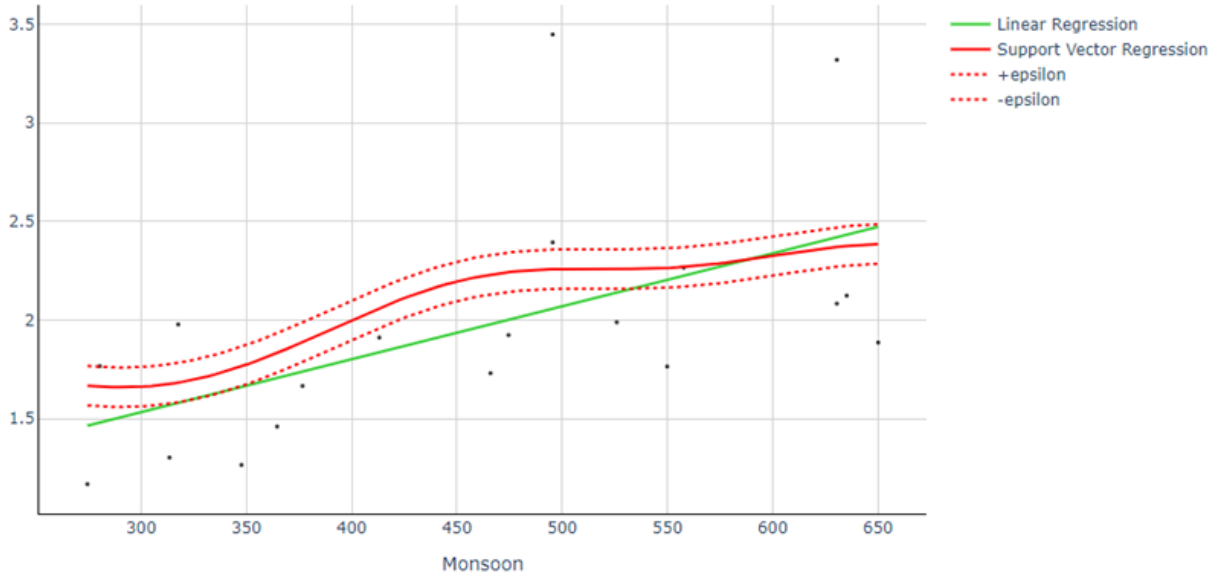


Fig 10.1: Graph of SVR VS Linear regression

11. CONCLUSION:

Our project aims to maximize crop production level based on seasonal rainfall. It helps the farmers or agriculture workers such that they can do agriculture more smartly in a much better calculated way. By predicting rainfall, we did feature selection in order to select only important features in predicting various crops. Since the model is build using the climatic parameters, only the values of weather attributes like rainfall need to be passed, where the model can predict the correct output. The decision of bit capacity is basic for SVR displaying. We see that SVR is better than other regression methods as an expectation strategy. Other regression methods like MLR can't catch the non-linearity in a data set and SVR winds up helpful in such circumstances. SVR proves to be the best model for predicting crop production.

12. LIST OF REFERENCES

1. Heuristic Prediction of Rainfall Using Machine Learning Techniques – by Chandrasegar Thirumalai, M Lakshmi Deepak, K Sri Harsha, K Chaitanya Krishna
2. Rainfall Prediction: A Deep Learning Approach – by Emilcy Hern´andez, Victor Sanchez-Anguix, Vicente Julian, Javier Palanca, and N´estor Duque.
3. Prediction Of Rainfall Using Machine Learning Techniques – by Moulana Mohammed, Roshitha Kolapalli, Niharika Golla, Siva Sai Maturi.
4. A Study on Crop Yield Forecasting Using Classification Techniques by: R.Sujatha, Dr.P.Isakki.
5. Data mining for meteorological applications: Decision trees for modeling rainfall prediction – by Geetha, A., and G. M. Nasira.
6. Machine learning techniques for rainfall prediction: A review – by Parmar, Aakash, Kinjal Mistree, and Mithila Sompura.
7. Rainfall prediction for the Kerala state of India using artificial intelligence approaches – by Dash, Yajnaseni, Saroj K. Mishra, and Bijaya K. Panigrahi.
8. Application of vegetation indices for agricultural crop yield prediction using neural network techniques – by Panda, S. S., Ames, D. P., & Panigrahi, S.
9. Climate change impact assessment: The role of climate extremes in crop yield simulation – by Moriondo, M., Giannakopoulos, C., & Bindi, M.
10. Prediction of crop yield, water consumption and water use efficiency with a SVAT-crop growth model using remotely sensed data on the North China Plain – by Mo, X., Liu, S., Lin, Z., Xu, Y., Xiang, Y., & McVicar, T. R.

13. APPENDIX A:

The undersigned acknowledge they have completed implementing the project “Rain prediction and crop plantation for maximum yield” and agree with the approach it presents.

Signature: _____ Date: 24/05/2021

Name: PRAVIN K

Signature: _____ Date: 24/05/2021

Name: VIGNESH P

14. APPENDIX B: REFERENCES

The following table summarizes the research papers referenced in this document.

Document name and version	Description	Location
Heuristic Prediction of Rainfall Using Machine Learning Techniques.	Linear regression analysis is used for predicting the unknown value of a season from the known value of another season	https://ieeexplore.ieee.org/document/8300884
Rainfall Prediction: A Deep Learning Approach	This paper has presented a deep learning approach based on the use of auto-encoders and neural networks to predict the accumulated precipitation for the next day	https://www.researchgate.net/publication/301320757_Rainfall_Prediction_A_Deep_Learning_Approach
Prediction Of Rainfall Using Machine Learning Techniques	The predictive model is used to prediction of the precipitation. The first step is converting data in to the correct format to conduct experiments then make a good analysis of data and observe variation in the patterns of rainfall	http://www.ijstr.org/final-print/jan2020/Prediction-Of-Rainfall-Using-Machine-Learning-Techniques.pdf

15. Appendix C: key term

Term	Definition
SVR	**SVR** uses the same basic idea as Support Vector Machine (SVM), a classification algorithm, but applies it to predict real values rather than a class. **SVR** acknowledges the presence of non-linearity in the data and provides a proficient prediction **model**
ARIMA	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. A statistical **model** is autoregressive if it predicts future values based on past values.
SARIMAX	**SARIMAX** is similar and stands for seasonal auto regressive integrated moving average with exogenous factors. It uses differencing at a lag equal to the number of seasons to remove additive **seasonal** effects. As with lag 1 differencing to remove a trend, the lag s differencing introduces a moving average term. The **seasonal ARIMA model** includes autoregressive and moving average terms at lag s.
IMD	The **India Meteorological Department** (**IMD**) is an agency of the Ministry of Earth Sciences of the Government of India. It is the principal agency responsible for meteorological observations, weather forecasting and seismology.
grid format	Grid is a raster file format developed by ESRI to contain information about geographic space in a grid. A grid defines geographic space as an array of equally sized square grid points arranged in rows and columns.