A

Minor Project Report

On

Crop Yield Prediction & Comparison using Machine Learning & Deep Learning Techniques.

Submitted for partial fulfillment for the degree of

Bachelor of Technology

(Information Technology)

in

Department of Information Technology

by

Sanchit Jain
189302008
&
Sukriti Nijhawan
189303114

Under the Guidance of
Ms. Kavita Jhajharia

(July-2021)

SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

MANIPAL UNIVERSITY JAIPUR

CERTIFICATE

Date: 17th June, 2021

This is to certify that the project titled **CROP YIELD PREDICTION AND COMPARISON USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES** is a record of the bonafide work done by **SANCHIT JAIN** (189302008) and **SUKRITI NIJHAWAN** (189303114) submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech) in **Information Technology** of Manipal University Jaipur, during the academic year 2020-21.

Ms. Kavita Jhajharia

Project Guide, Dept of Information Technology

Manipal University Jaipur

Dr. Pankaj Vyas

HOD, Dept of Information Technology

Manipal University Jaipur

ABSTRACT

Agriculture is India's backbone; more than half of the country's population relies on agricultural activities for a living. Agriculture is a sector that contributes significantly to the economic development of our country, it was the catalyst for the development of humanity. Crop production and food security are both affected by rising populations and changing climatic conditions. Crop selection is influenced by a variety of factors, including market price, production rate, soil type, rainfall, temperature, government policies, etc. To improve our Indian economy, many changes are needed in the agricultural sector. We can boost agriculture by employing machine learning techniques that are simple to implement in the agricultural sector. Along with all of the advancements in computers and technology, valuable and detailed knowledge on a variety of topics is also critical. The aim of this project is to put the crop selection method into practice so that it can help farmers solve problems related to crop yield. This will boost the yield rate of crop production, benefitting the Indian economy. In this project Random forest, SVM, Gradient Descent, RNN LSTM, Lasso regression techniques were applied to predict the yield of five crops (wheat, barley, jowar, rapeseed & mustard, and bajra) in Rajasthan (district-wise). Cross-validation techniques like r2 score, RMSE, MAE were used to validate the results. Random Forest outperforms all other techniques.

Keywords – crop yield prediction, machine learning, deep learning, area, yield, production, soil, rainfall, encoding, random forest, svm, lasso regression, gradient descent, lstm, r2 score, root mean square error (rmse), mean absolute error (mae).

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

1.1. MOTIVATION

It's not for nothing that the science of teaching machines to learn and generate models for future predictions is commonly used. Agriculture is extremely important to the global economy. Understanding global crop yield is critical for resolving food security issues and mitigating the effects of climate change as the human population continues to grow.

Crop yield forecasting is a significant agricultural problem. Weather conditions (rain, temperature, etc.) and pesticides have a great impact on agricultural yield. It is important to have accurate knowledge about crop yield history while making decisions about agricultural risk management and forecasting the future.

Crop yield prediction is a challenge for decision-makers at all levels, including national and regional levels. Farmers may use an effective crop yield prediction model to help them determine what to grow and when to grow it. Crop yield prediction can be done in a variety of ways.

1.2. MACHINE LEARNING IN CROP YIELD PREDICTION

Machine learning is a realistic method that can provide better yield prediction based on many attributes. It is a subdivision of Artificial Intelligence (AI) that focuses on learning. Machine learning (ML) can discover information from datasets by identifying patterns and correlations. The models must be trained with datasets that reflect the outcomes based on previous experience. The predictive model is developed using a variety of features, and the models' parameters are determined during the training process using historical data. During the testing process, some of the historical data that was not used for preparation is used to measure results.

An ML model can be descriptive or predictive, depending on the research topic and questions. Predictive models allow us to take what we know about the past and apply it to forecasting what will happen in the future. Descriptive templates, on the other hand, assist us in describing how things are now or what has occurred in the past.

Machine learning is a useful decision-making method for predicting crop yields, as well as for deciding which crops to plant and what to do during the crop's growing season. To aid crop yield prediction study, a number of machine learning models have been used.

Machine learning techniques such as multivariate regression, decision trees, association rule mining, and artificial neural networks have recently been used to predict crop yields.

Machine learning models consider the output (crop yield) to be an implicit function of the input variables (area and environmental factors), which can be a non-linear relationship.

1.3. DEEP LEARNING IN CROP YIELD PREDICTION

Deep learning models have recently been used for crop yield prediction.

Deep learning is a form of machine learning that can predict outcomes from a variety of raw data arrangements. Deep learning algorithms, for example, can build a probability model from ten years of field data and provide insights into crop output under various climatic conditions.

Artificial neural networks (ANNs) are used in deep learning to mimic how humans think and learn. Deep learning is driven by artificial neural networks with several layers. Deep Neural Networks (DNNs) are networks of multiple layers that can perform complex operations including representation and abstraction to understand images, sound, and text.

In the same way, as the human brain is made up of neurons, neural networks are made up of layers of nodes. Individual layer nodes are connected to nodes in neighbouring layers. The number of layers in the network indicates that it is deeper.

Signals move between nodes in an artificial neural network and allocate weights to them. A node with a higher weight would have a greater impact on the nodes below it. The weighted inputs are compiled in the final layer to generate an output.

1.4. PROJECT AIM

In this project, we compare and predict the yield of five crops (wheat, barley, jowar, rapeseed & mustard, and bajra) in Rajasthan (district-wise) using lasso regression, two machine learning techniques: random forest and SVM, and two deep learning techniques: gradient descent and RNN LSTM.

To apply the models to our data, we divide it into training and testing datasets. Each model is tested twice: once with only "area" and "production" in mind, and then again with additional factors (rainfall and soil type) in mind to predict crop yield.

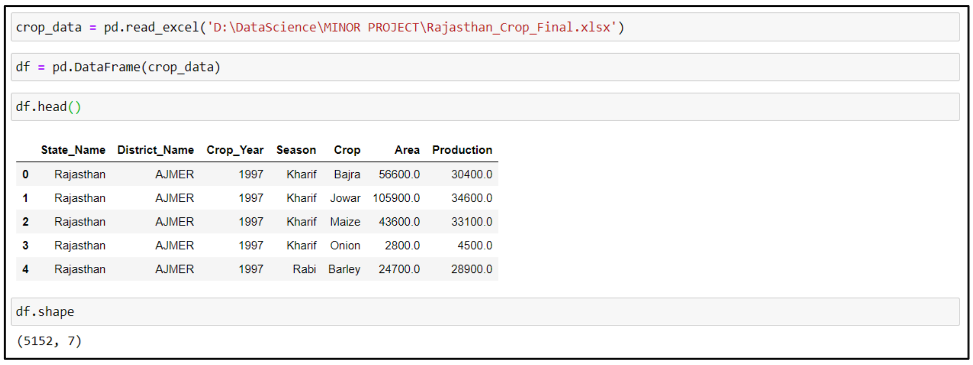
To find the model that most accurately predicts the yield, the R2 score, Root Mean Squared Error (RMSE) and Mean Average Error (MAE) are calculated for each model.

2. DATASET

2.1. OVERVIEW

The data collection procedure refers to the programmer acquiring and quantifying information based on variables relevant to the project. Sources like Kaggle supply us with a repository of a wide range of datasets from which we may extract the essential information, and thus for this project as well we have extracted some of our dataset from Kaggle. The remaining data we have obtained from a variety of sources, which include the Rajasthan Government's official website.

We gathered and recorded data (figure 1) from the 33 districts of Rajasthan (table 2) based on the state's most harvested crops, which include wheat, rapeseed & mustard, barley, bajra, jowar onion, and maize (table 1). The data we obtained from Kaggle was from the years 1997 to 2010, and the rest, viz., the data from 2011 to 2019 from the official Rajasthan Government website.



```
crop_data = pd.read_excel('D:\DataScience\MINOR PROJECT\Rajasthan_Crop_Final.xlsx')
```

```
df = pd.DataFrame(crop_data)
```

```
df.head()
```

```
   State_Name  District_Name  Crop_Year  Season    Crop      Area  Production
0   Rajasthan          AJMER       1997  Kharif   Bajra   56600.0     30400.0
1   Rajasthan          AJMER       1997  Kharif   Jowar  105900.0     34600.0
2   Rajasthan          AJMER       1997  Kharif   Maize   43600.0     33100.0
3   Rajasthan          AJMER       1997  Kharif   Onion    2800.0      4500.0
4   Rajasthan          AJMER       1997    Rabi  Barley   24700.0     28900.0
```

```
df.shape
```

```
(5152, 7)
```

Figure 1: *Raw Dataset*

Table 1: *Crop data in the dataset*

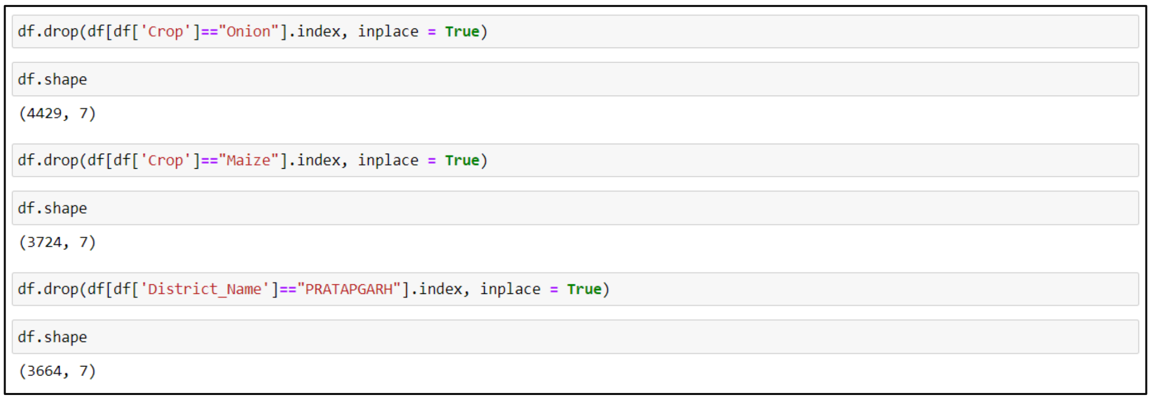
CROP	DATA AVAILABLE IN DATASET
Rapeseed and Mustard	748
Wheat	748
Barley	747
Bajra	742
Jowar	739
Onion	723
Maize	705

Table 2: *District data in the dataset*

DISTRICT	DATA AVAILABLE IN DATASET
Tonk	161
Jalore	161
Jhalawar	161
Jaipur	161
Alwar	161
Bharatpur	161
Sawai Madhaopur	161
Ajmer	161
Pali	161
Nagaur	161
Sikar	160
Chittorgarh	160
Baran	160
Banswara	160
Sirohi	160
Dausa	160
Rajsamand	160
Jodhpur	160
Bhilwara	160
Kota	159
Barmer	159
Udaipur	159
Dungarpur	159
Ganganagar	159
Bundi	157
Dholpur	157
Karauli	155
Jhunjhunu	155
Hanumangarh	154
Bikaner	151
Churu	148
Jaisalmer	146
Pratapgarh	84

2.2. DATA CLEANING

Based on preliminary investigation, we can observe that there is insufficient data for the crops - onion and maize – as well as for the district of Pratapgarh. As a consequence of this study, we can conclude that there is no relevant data for us to forecast for these specific entities, which will result in lower accuracy when we apply the relevant models to this dataset. So, to overcome these discrepancies we decided to drop these particular entities from our dataset (figure 2).



```
df.drop(df[df['Crop']=="Onion"].index, inplace = True)
```

```
df.shape
```

```
(4429, 7)
```

```
df.drop(df[df['Crop']=="Maize"].index, inplace = True)
```

```
df.shape
```

```
(3724, 7)
```

```
df.drop(df[df['District_Name']=="PRATAPGARH"].index, inplace = True)
```

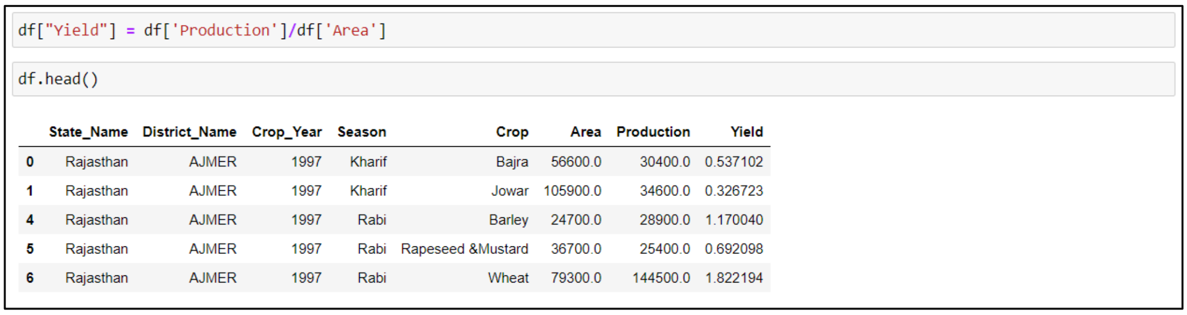
```
df.shape
```

```
(3664, 7)
```

Figure 2: *Dropping Crops – Onion and Maize, and District - Pratapgarh*

So, after cleaning all the invalid and null data the final shape of our dataset comes out to 3664 rows and 7 columns. Now, we add our independent variable – Yield, which we get from the following (figure 3):

$$Yield = Production(Tons)/Area(Ha)$$



```
df["Yield"] = df['Production']/df['Area']
```

```
df.head()
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield
0	Rajasthan	AJMER	1997	Kharif	Bajra	56600.0	30400.0	0.537102
1	Rajasthan	AJMER	1997	Kharif	Jowar	105900.0	34600.0	0.326723
4	Rajasthan	AJMER	1997	Rabi	Barley	24700.0	28900.0	1.170040
5	Rajasthan	AJMER	1997	Rabi	Rapeseed &Mustard	36700.0	25400.0	0.692098
6	Rajasthan	AJMER	1997	Rabi	Wheat	79300.0	144500.0	1.822194

Figure 3: *Adding our target feature – Yield*

2.3. FEATURE EXPANSION

Now in order for us to get a more scientific result, we add a few more independent features to our dataset, which are the soil type that is found in that particular district and the amount of rainfall that has happened over the years in these districts. The reason for us choosing these particular factors is that first of all, the soil is one of the most essential components of crop yield since it provides the required nutrients, water, and oxygen to the crop while rainfall patterns help in determining the amount of natural water that will be provided for crop growth, replenishment, and production.

2.3.1. ADDING SOIL DATA

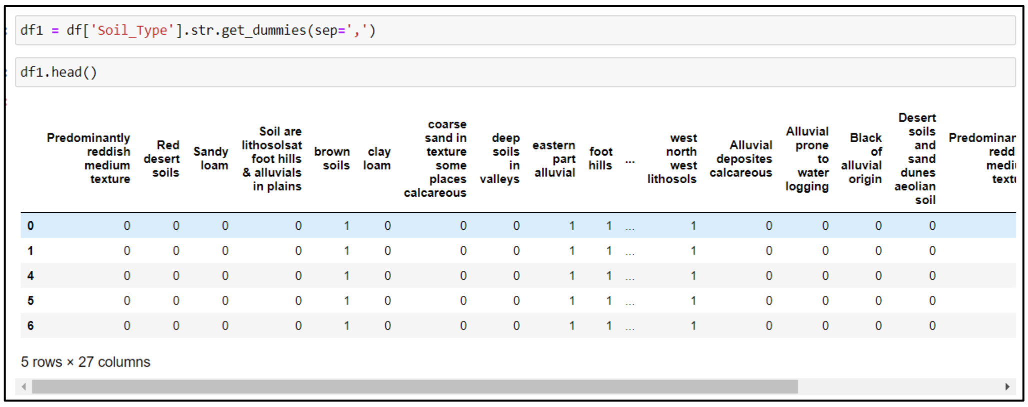
We start with the type of soil. For soil, we identified the type of soil that is regional to a particular district. We identified about 27 different types of soil are present in the region of Rajasthan, and in the present 32 districts, the soil in each district is a mixture of any of these 27 types of soils (table 3).

Table 3: *Soil type of each district*

DISTRICT	TYPE OF SOIL
Barmer	Desert soils and sand dunes aeolian soil, coarse sand in texture some places calcareous
Jodhpur	Desert soils and sand dunes aeolian soil, coarse sand in texture some places calcareous, Red desert soils
Ganganagar	Alluvial deposites calcareous, high soluble salts & exchangeable sodium
Hanumangarh	Alluvial deposites calcareous, high soluble salts & exchangeable sodium
Nagaur	Sandy loam, sallow depth red soils in depressions
Sikar	Sandy loam, sallow depth red soils in depressions
Jhunjhunu	Sandy loam, sallow depth red soils in depressions
Jalore	Red desert soils
Pali	Red desert soils
Jaipur	Sierozens, eastern part alluvial, west north west lithosols, foot hills, brown soils
Ajmer	Sierozens, eastern part alluvial, west north west lithosols, foot hills, brown soils
Dausa	Sierozens, eastern part alluvial, west north west lithosols, foot hills, brown soils
Tonk	Sierozens, eastern part alluvial, west north west lithosols, foot hills, brown soils
Alwar	Alluvial prone to water logging, nature of recently alluvial calcareous has been observed
Dholpur	Alluvial prone to water logging, nature of recently alluvial calcareous has been observed
Bharatpur	Alluvial prone to water logging, nature of recently alluvial calcareous has been observed
Karauli	Alluvial prone to water logging, nature of recently alluvial calcareous has been observed
Sawai Madhopur	Alluvial prone to water logging, nature of recently alluvial calcareous has been observed
Bhilwara	Soil are lithosolsat foot hills & alluvials in plains
Rajsamand	Soil are lithosolsat foot hills & alluvials in plains
Dungarpur	Predominantly reddish medium texture, well drained calcareous, shallow on hills, deep soils in valleys
Banswara	Predominantly reddish medium texture, well drained calcareous, shallow on hills, deep soils in valleys
Kota	Black of alluvial origin, clay loam, groundwater salinity
Jhalawar	Black of alluvial origin, clay loam, groundwater salinity
Bundi	Black of alluvial origin, clay loam, groundwater salinity
Baran	Black of alluvial origin, clay loam, groundwater salinity
Churu	Desert soils and sand dunes aeolian soil, loamycoarse in texture & calcareous, Sandy loam, sallow depth red soils in depressions
Sirohi	Red desert soils, Soil are lithosolsat foot hills & alluvials in plains
Udaipur	Soil are lithosolsat foot hills & alluvials in plains, Predominantly reddish medium texture, well drained calcareous, shallow on hills, deep soils in valleys
Chittorgarh	Soil are lithosolsat foot hills & alluvials in plains, Predominantly reddish medium texture, well drained calcareous, shallow on hills, deep soils in valleys

As one may notice that each district has its specific combination of soil, and it's in the form of categorical data. So, for us to perform prediction over the dataset, we need to encode it and hence create dummy variables (figure 4).

Then we combine the dataframes (figure 5) and drop the "Soil_Type" column from our actual dataframe (figure 7). Also, we simultaneously change the format in which the names of the districts have been represented, for a clear understanding of the dataset in hand (figure 6).



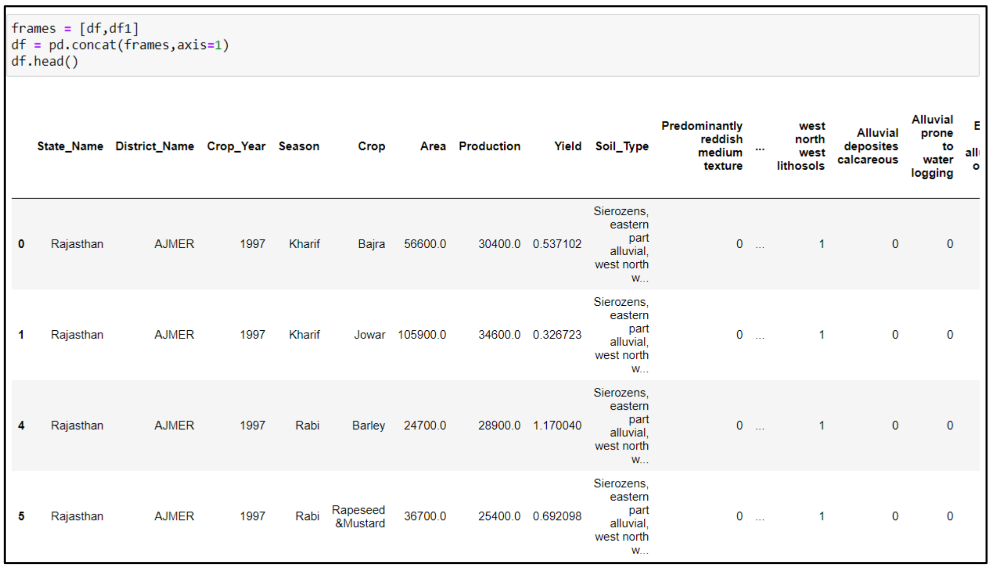
```
df1 = df['Soil_Type'].str.get_dummies(sep=',')
```

```
df1.head()
```

	Predominantly reddish medium texture	Red desert soils	Sandy loam	Soil are lithosolsat foot hills & alluvials in plains	brown soils	clay loam	coarse sand in texture some places calcareous	deep soils in valleys	eastern part alluvial	foot hills	...	west north west lithosols	Alluvial deposites calcareous	Alluvial prone to water logging	Black of alluvial origin	Desert soils and sand dunes aeolian soil	Predominar redd medi textu
0	0	0	0	0	1	0	0	0	1	1	...	1	0	0	0	0	
1	0	0	0	0	1	0	0	0	1	1	...	1	0	0	0	0	
4	0	0	0	0	1	0	0	0	1	1	...	1	0	0	0	0	
5	0	0	0	0	1	0	0	0	1	1	...	1	0	0	0	0	
6	0	0	0	0	1	0	0	0	1	1	...	1	0	0	0	0	

5 rows × 27 columns

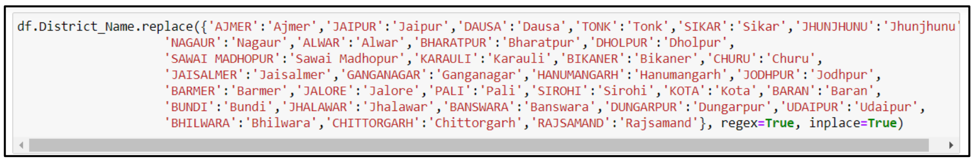
Figure 4: *Creating dummy variables of Soil_Type.*



```
frames = [df,df1]
df = pd.concat(frames,axis=1)
df.head()
```

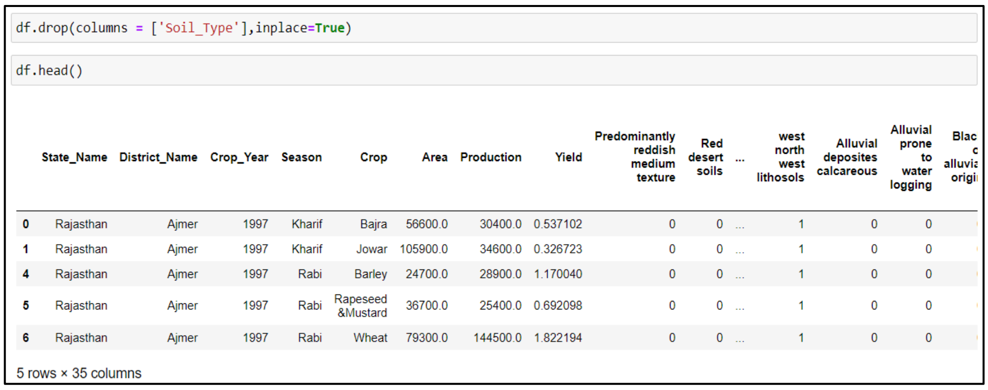
	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield	Soil_Type	Predominantly reddish medium texture	...	west north west lithosols	Alluvial deposites calcareous	Alluvial prone to water logging	E all o
0	Rajasthan	AJMER	1997	Kharif	Bajra	56600.0	30400.0	0.537102	Sierozens, eastern part alluvial, west north w...	0	...	1	0	0	
1	Rajasthan	AJMER	1997	Kharif	Jowar	105900.0	34600.0	0.326723	Sierozens, eastern part alluvial, west north w...	0	...	1	0	0	
4	Rajasthan	AJMER	1997	Rabi	Barley	24700.0	28900.0	1.170040	Sierozens, eastern part alluvial, west north w...	0	...	1	0	0	
5	Rajasthan	AJMER	1997	Rabi	Rapeseed &Mustard	36700.0	25400.0	0.692098	Sierozens, eastern part alluvial, west north w...	0	...	1	0	0	

Figure 5: *Merging the dataframe*



```
df.District_Name.replace({'AJMER':'Ajmer','JAIPUR':'Jaipur','DAUSA':'Dausa','TONK':'Tonk','SIKAR':'Sikar','JHUNJHUNU':'Jhunjhunu'
                          'NAGAUR':'Nagaur','ALWAR':'Alwar','BHARATPUR':'Bharatpur','DHOLPUR':'Dholpur',
                          'SAWAI MADHOPUR':'Sawai Madhopur','KARAULI':'Karauli','BIKANER':'Bikaner','CHURU':'Churu',
                          'JAISALMER':'Jaisalmer','GANGANAGAR':'Ganganagar','HANUMANGARH':'Hanumangarh','JODHPUR':'Jodhpur',
                          'BARMER':'Barmer','JALORE':'Jalore','PALI':'Pali','SIROHI':'Sirohi','KOTA':'Kota','BARAN':'Baran',
                          'BUNDI':'Bundi','JHALAWAR':'Jhalawar','BANSWARA':'Banswara','DUNGARPUR':'Dungarpur','UDAIPUR':'Udaipur',
                          'BHILWARA':'Bhilwara','CHITTORGARH':'Chittorgarh','RAJSAMAND':'Rajsamand'}, regex=True, inplace=True)
```

Figure 6: *Changing the names of the districts to improve visibility.*



```
df.drop(columns = ['Soil_Type'],inplace=True)
```

```
df.head()
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Yield	Predominantly reddish medium texture	Red desert soils	...	west north west lithosols	Alluvial deposites calcareous	Alluvial prone to water logging	Blac c alluvi origi
0	Rajasthan	Ajmer	1997	Kharif	Bajra	56600.0	30400.0	0.537102	0	0	...	1	0	0	
1	Rajasthan	Ajmer	1997	Kharif	Jowar	105900.0	34600.0	0.326723	0	0	...	1	0	0	
4	Rajasthan	Ajmer	1997	Rabi	Barley	24700.0	28900.0	1.170040	0	0	...	1	0	0	
5	Rajasthan	Ajmer	1997	Rabi	Rapeseed &Mustard	36700.0	25400.0	0.692098	0	0	...	1	0	0	
6	Rajasthan	Ajmer	1997	Rabi	Wheat	79300.0	144500.0	1.822194	0	0	...	1	0	0	

5 rows × 35 columns

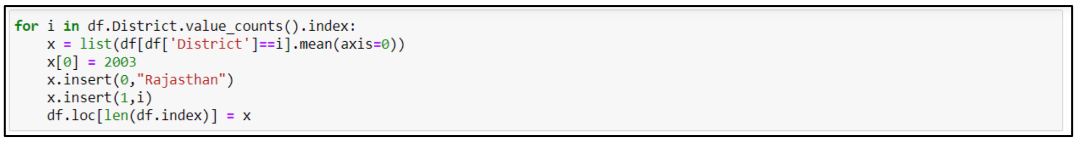
Figure 7: *Removing the soil type column.*

2.3.2. ADDING RAINFALL DATA

Next, we added rainfall to our dataset. The rainfall data available was from the year 1901 to 2002 and 2004 to 2010 corresponding to each district of Rajasthan. The rainfall data corresponding to 2003 was filled by using the mean from the data available, using the function in figure 8.

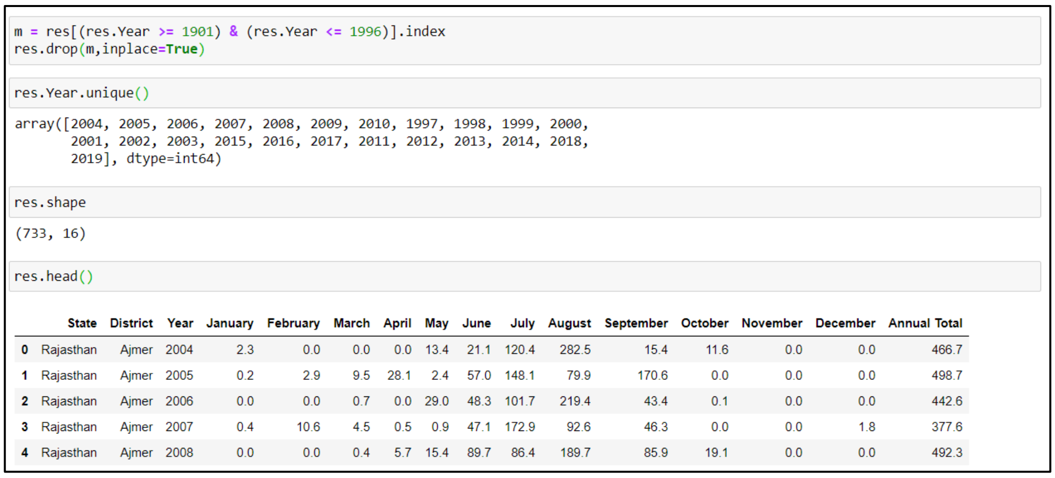
For data from 2011 to 2017, we extracted it, again, from the official documentation done by the Rajasthan Government. Next, we added the rainfall data corresponding to the years 2018 and 2019 by using the mean from the data from 1901 to 2017, using the same function as shown in figure 12. The final dataset contains the data of monthly rainfall of each district from 1997 to 2019 (figure 9) since our main dataset contains information on crop production during the same period.

Finally, we merge our rainfall dataset with our main dataset.



```
for i in df.District.value_counts().index:
    x = list(df[df['District']==i].mean(axis=0))
    x[0] = 2003
    x.insert(0,"Rajasthan")
    x.insert(1,i)
    df.loc[len(df.index)] = x
```

Figure 8: *Function to add 2003 rainfall data, later also used for adding data of 2018 and 2019.*



```
m = res[(res.Year >= 1901) & (res.Year <= 1996)].index
res.drop(m,inplace=True)
```

```
res.Year.unique()
```

```
array([2004, 2005, 2006, 2007, 2008, 2009, 2010, 1997, 1998, 1999, 2000,
       2001, 2002, 2003, 2015, 2016, 2017, 2011, 2012, 2013, 2014, 2018,
       2019], dtype=int64)
```

```
res.shape
```

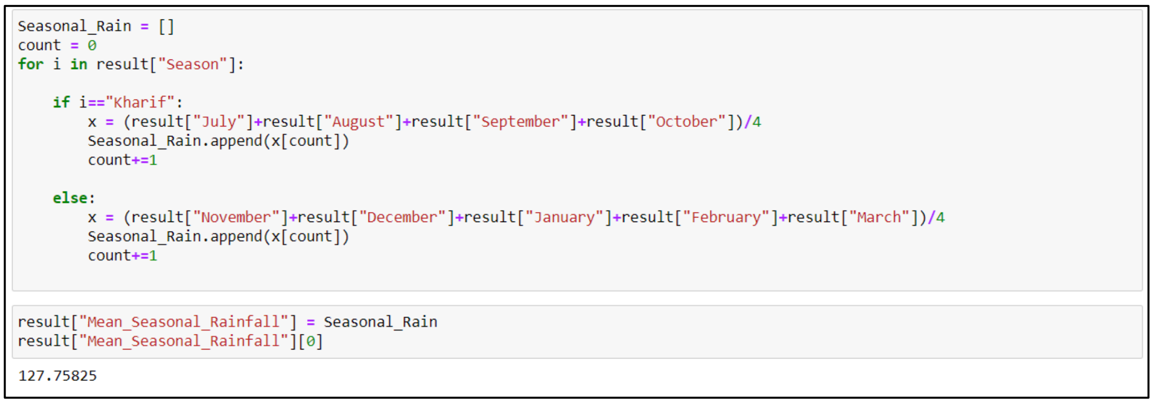
```
(733, 16)
```

```
res.head()
```

	State	District	Year	January	February	March	April	May	June	July	August	September	October	November	December	Annual Total
0	Rajasthan	Ajmer	2004	2.3	0.0	0.0	0.0	13.4	21.1	120.4	282.5	15.4	11.6	0.0	0.0	466.7
1	Rajasthan	Ajmer	2005	0.2	2.9	9.5	28.1	2.4	57.0	148.1	79.9	170.6	0.0	0.0	0.0	498.7
2	Rajasthan	Ajmer	2006	0.0	0.0	0.7	0.0	29.0	48.3	101.7	219.4	43.4	0.1	0.0	0.0	442.6
3	Rajasthan	Ajmer	2007	0.4	10.6	4.5	0.5	0.9	47.1	172.9	92.6	46.3	0.0	0.0	1.8	377.6
4	Rajasthan	Ajmer	2008	0.0	0.0	0.4	5.7	15.4	89.7	86.4	189.7	85.9	19.1	0.0	0.0	492.3

Figure 9: *Final Rainfall Dataset*

Now we must realize that each crop has its specific season in which it's produced. Thus, considering the rainfall pattern for the whole year for that crop doesn't make sense. So, to get the correct amount of rainfall that the crop would have received, we consider the mean value of the rainfall of the months corresponding to those seasons. Thus, for crops that are produced during the Kharif season, we consider the rainfall received during July, August, September, and October, while for Rabi we considered the mean of the rainfall during November, December, January, February, and March (figure 10).



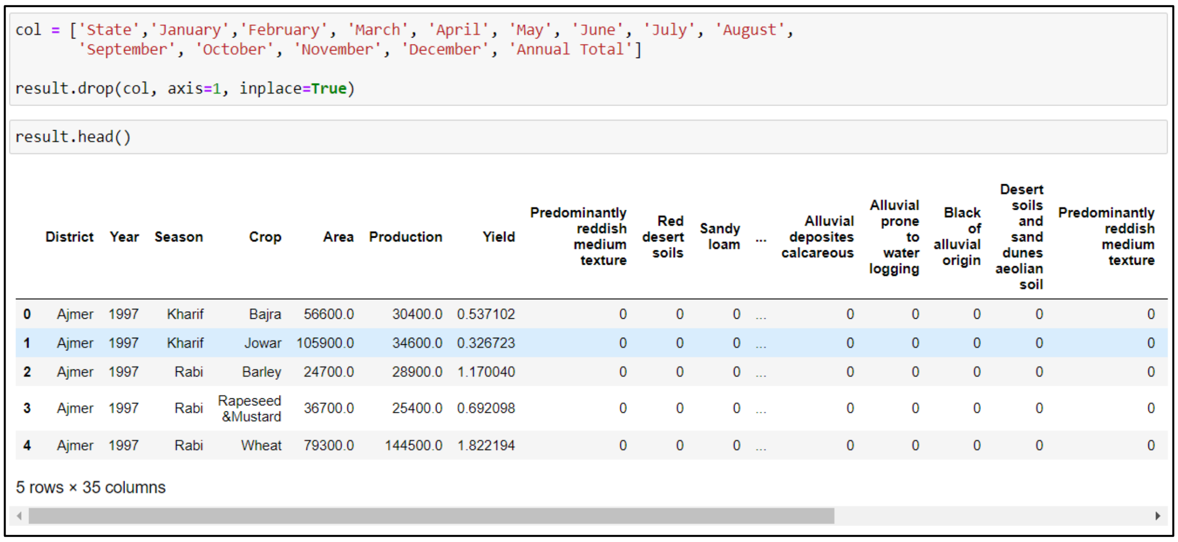
```
Seasonal_Rain = []
count = 0
for i in result["Season"]:

    if i=="Kharif":
        x = (result["July"]+result["August"]+result["September"]+result["October"])/4
        Seasonal_Rain.append(x[count])
        count+=1

    else:
        x = (result["November"]+result["December"]+result["January"]+result["February"]+result["March"])/4
        Seasonal_Rain.append(x[count])
        count+=1
```

```
result["Mean_Seasonal_Rainfall"] = Seasonal_Rain
result["Mean_Seasonal_Rainfall"][0]
```

```
127.75825
```

Figure 10: *Calculating and adding mean seasonal rainfall to our dataset.*

Then we remove the columns from January to Annual Total. Also, we remove the column "State" since we are considering only Rajasthan and the different districts act like distinguishing entities, the presence of this variable would not have any effect on the prediction of our data (figure 11).



```
col = ['State','January','February', 'March', 'April', 'May', 'June', 'July', 'August',
       'September', 'October', 'November', 'December', 'Annual Total']

result.drop(col, axis=1, inplace=True)
```

```
result.head()
```

	District	Year	Season	Crop	Area	Production	Yield	Predominantly reddish medium texture	Red desert soils	Sandy loam	...	Alluvial deposites calcareous	Alluvial prone to water logging	Black of alluvial origin	Desert soils and sand dunes aeolian soil	Predominantly reddish medium texture
0	Ajmer	1997	Kharif	Bajra	56600.0	30400.0	0.537102	0	0	0	...	0	0	0	0	0
1	Ajmer	1997	Kharif	Jowar	105900.0	34600.0	0.326723	0	0	0	...	0	0	0	0	0
2	Ajmer	1997	Rabi	Barley	24700.0	28900.0	1.170040	0	0	0	...	0	0	0	0	0
3	Ajmer	1997	Rabi	Rapeseed &Mustard	36700.0	25400.0	0.692098	0	0	0	...	0	0	0	0	0
4	Ajmer	1997	Rabi	Wheat	79300.0	144500.0	1.822194	0	0	0	...	0	0	0	0	0

5 rows × 35 columns

Figure 11: *Removing unnecessary columns.*

Table 4: *Data and their sources*

DATA	VARIABLE (S)	TIME COVERAGE	SOURCE
Rajasthan Crop data	State, District, Area, and Production	1997 to 2010	https://www.kaggle.com/abhinand05/crop-production-in-india
Rajasthan Crop data	State, District, Area, and Production	2011 to 2019	http://www.agriculture.rajasthan.gov.in/content/agriculture/en/ Agriculture-Department-dep/agriculture-statistics.html
Rainfall data	Jan, Feb, Mar, Apr, May, Jun, Jul, Sept, Oct, Nov, Dec, and Annual_Total	1901 to 2002	https://www.indiawaterportal.org/met_data
Rainfall data	Jan, Feb, Mar, Apr, May, Jun, Jul, Sept, Oct, Nov, Dec, and Annual_Total	2004 to 2010	https://water.rajasthan.gov.in/content/water/en/waterresourcesdepartment/ WaterManagement/IWRM/annualrainfall.html#
Rainfall data	Jan, Feb, Mar, Apr, May, Jun, Jul, Sept, Oct, Nov, Dec, and Annual_Total	2011 to 2019	http://www.agriculture.rajasthan.gov.in/content/agriculture/en/ Agriculture-Department-dep/agriculture-statistics.html
Soil data	District, and Soil_Type		http://www.agriculture.rajasthan.gov.in/content/agriculture/en/Agriculture-Department-dep/Departmental-Introduction/Agro-Climatic-Zones.html

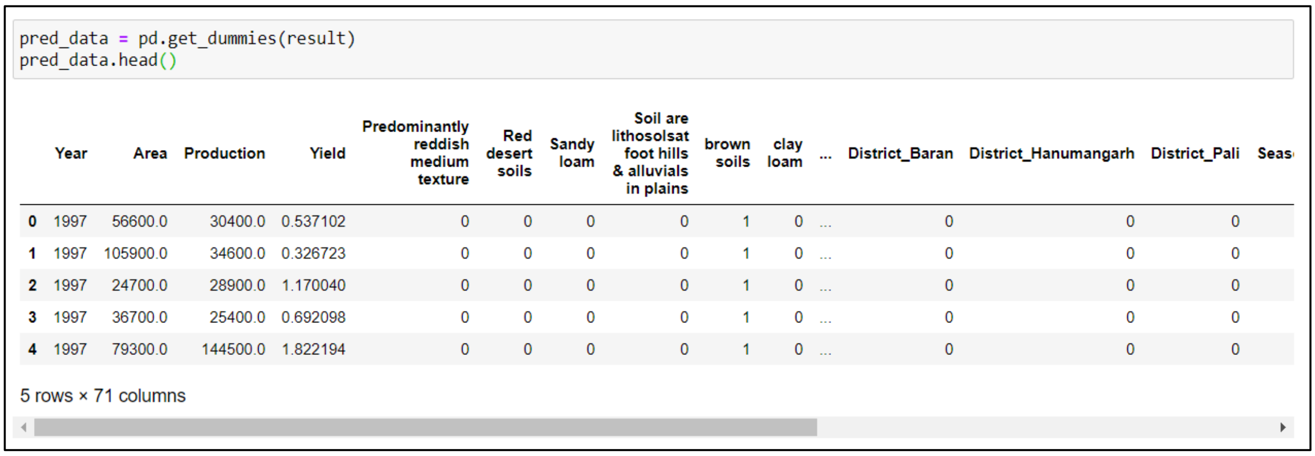
3. METHODOLOGY

3.1. DATA PREPROCESSING

Data preprocessing is a technique for transforming unprocessed data into a flawless data set. At the end of the day, whenever data is gathered from various sources, it is gathered in a raw or crude form that cannot be analyzed by machine learning or deep learning methodologies.

3.1.1 DATA ENCODING

In our final dataframe of ours there exist 6 categorical columns – State, District, Season, Crop, and Soil_Type. So, a dataset can be divided into 2 types of variables – Continuous and Categorical variables. A categorical variable is a variable that takes one value from a set of limited values and is not quantifiable. So, in order to apply any algorithm, it becomes imperative to encode such variables, since many machine learning algorithms cannot work on labeled data directly, thus, all input is required to be in the form of numeric values. There can be several methods that can be used to encode and handle such variables. These methods include LabelEncoder, OneHotEncoder, etc. Here for the given dataset, we chose to create dummy variables (figure 12) of all the categorical data present in our dataset. Creating dummy variables gives us flexibility while performing regression analysis and hence suits well for our given dataset.



```
pred_data = pd.get_dummies(result)
pred_data.head()
```

	Year	Area	Production	Yield	Predominantly reddish medium texture	Red desert soils	Sandy loam	Soil are lithosolsat foot hills & alluvials in plains	brown soils	clay loam	...	District_Baran	District_Hanumangarh	District_Pali	Seas
0	1997	56600.0	30400.0	0.537102	0	0	0	0	1	0	...	0	0	0	
1	1997	105900.0	34600.0	0.326723	0	0	0	0	1	0	...	0	0	0	
2	1997	24700.0	28900.0	1.170040	0	0	0	0	1	0	...	0	0	0	
3	1997	36700.0	25400.0	0.692098	0	0	0	0	1	0	...	0	0	0	
4	1997	79300.0	144500.0	1.822194	0	0	0	0	1	0	...	0	0	0	

5 rows × 71 columns

Figure 12: *The required dataset for predictive analysis.*

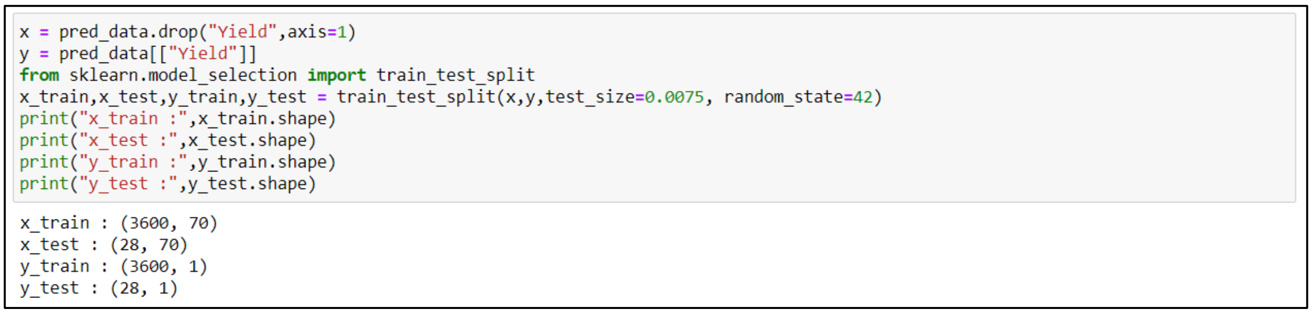
3.1.2. STANDARDIZATION OF FEATURES

The features of the dataframe will look like above with 71 columns. Then in order to standardize our data, we apply StandardScaler to our dataset. This helps us standardize features by removing mean and scaling to unit variance. Standardization of any dataset is common practice and a necessity at times for a lot of machine learning and deep learning methodologies.

3.1.3. SPLITTING DATASET INTO TESTING AND TRAINING SETS

The final step of data preprocessing is testing and training our data. The division between training and testing size of the data is not done equally. We require a larger training set as compared to the testing size, since, for prediction, the model needs to be trained over as many data points as possible. For this, we use the common ScikitLearn library and import train_test_split module (figure 13).

The variable "x" represents the dataframe of independent variables while stored in the "y" variable is the dependent or target variable, i.e., Yield. The division of data between testing and training size is then to be done by carefully examining the accuracy shown by the applied models.



```
x = pred_data.drop("Yield",axis=1)
y = pred_data[["Yield"]]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.0075, random_state=42)
print("x_train :",x_train.shape)
print("x_test :",x_test.shape)
print("y_train :",y_train.shape)
print("y_test :",y_test.shape)
```

```
x_train : (3600, 70)
x_test : (28, 70)
y_train : (3600, 1)
y_test : (28, 1)
```

Figure 13: *Dividing our dataset into dependent and independent features.*

3.2. MODELS

3.2.1. RANDOM FOREST

Random forest is a very famous machine learning algorithm that felicitates in cases of both classification and regression issues. This algorithm works on the notion of ensemble learning, which works on the principle of merging several classifiers to give the solution for any complex problem and improve the precision and performance of the applied model. Random Forest is a classifier that consists of multiple decision trees on several subsets of a given dataset and considers the average to improve the dataset's predictive accuracy. In other words, rather than relying on a single decision tree, this algorithm considers predictions from each tree and predicts the final output based on the majority votes of predictions. On observation we can say that:

$$No.of\ Trees\ in\ forest \propto\ Accuracy\ of\ the\ model$$

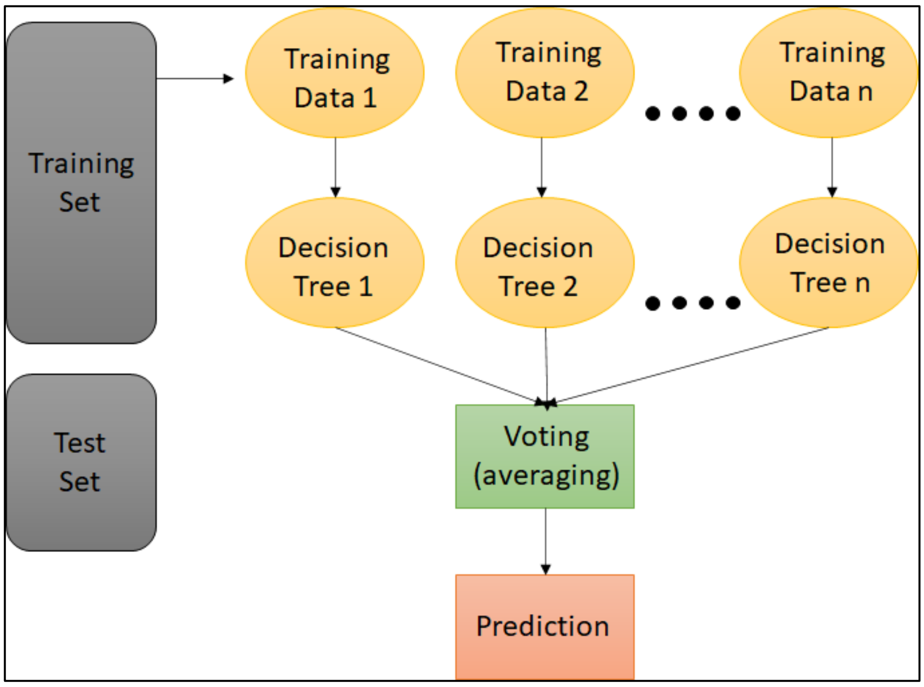


Figure 14: *Random forest model*

3.2.2. SUPPORT VECTOR MACHINE (SVM)

The goal of the support vector machine algorithm is to discover a hyperplane in N-dimensional space (N refers to the number of features present in the dataset) that classifies the data points very distinctly.

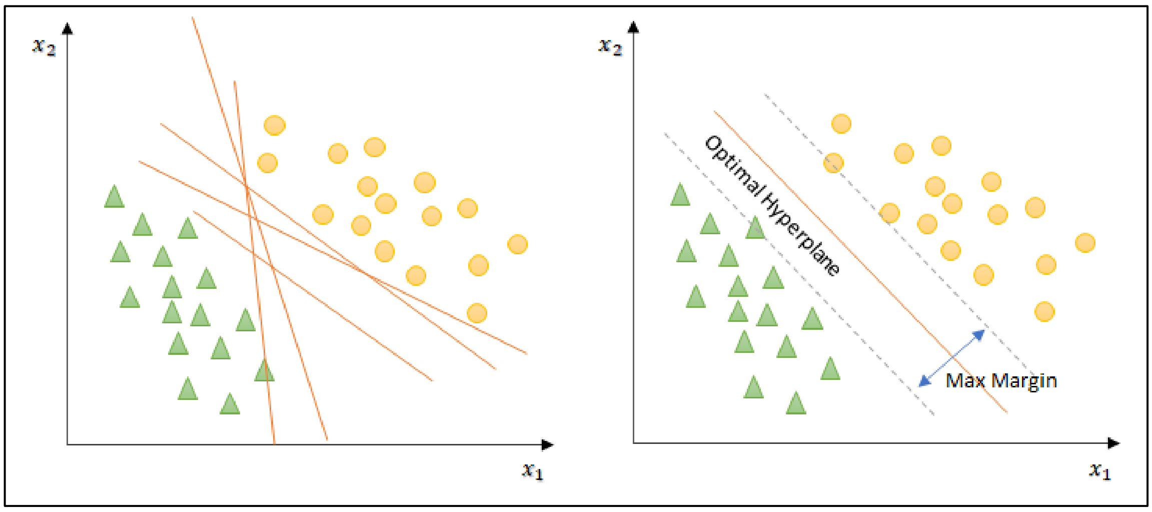


Figure 15: *SVM hyperplanes*

To detach the two classes of information points, there are various possible hyperplanes that could be chosen. We will likely find a plane that has the best edge, i.e., the highest distance between data points of the two classes we are dissecting. Extending the edge distance gives some help so future data points can be organized with more assurance. Since we are hoping to increase and maximize the edge between the data points and the hyperplane, we utilize the Hinge function to do as required:

$$c\big(x, y, f(x)\big) = \begin{Bmatrix} 0, & if\ y * f(x) \geq 1 \\ 1 - y * f(x), & else \end{Bmatrix}$$

Hinge loss function

The cost comes out to be 0 if the predicted value and the actual value are of the same sign, but if not, then we are able to calculate the loss value. In order for us to balance the regularized limits we add a regularization function to the Hinge loss function:

$$min_{\omega} \lambda \, ||\omega||^2 + \sum_{i=1}^{n} (1 - \ y_i(x_i, \omega))_+$$

Loss function for SVM

As the loss function has now been defined, we first need to partially differentiate with respect to weights in order to find the gradients, and by using this gradient we'll update the weights. After finding the gradients we add the regularization parameter so that there exists no misclassification. If there still exists any misclassification, our model will make mistakes when performing predictions of the class of our data points, thus we include the loss with the regularization parameter in order to perform the final gradient update.

$$\frac{\delta}{\delta \omega_k} \lambda \, ||\omega||^2 = 2\lambda\omega_k$$

$$\frac{\delta}{\delta \omega_k} (1 - \ y_i(x_i, \omega))_+ = \begin{Bmatrix} 0, & if\ y * f(x) \geq 1 \\ -y_i x_{ik}, & else \end{Bmatrix}$$

Gradients

$$\omega = \omega - \alpha 2\lambda\omega$$

Gradient Update – No misclassification

$$\omega = \omega + \alpha(y_i x_i - 2\lambda\omega)$$

Gradient Update – Misclassification

3.2.3. GRADIENT DESCENT

Gradient descent is a well-known optimization algorithm that is frequently used in machine learning and deep learning. It finds the coefficients that minimize the cost function as far as possible by identifying a local minimum of the differentiable function. Gradient descent begins by characterizing the initial parameter values and then uses analytics and calculus to iteratively change the values so that they limit the given cost function. Gradient descent can also be defined as the slope of any given function. The greater the gradient, the greater the slope, and thus the faster the model's learning rate. When the slope reaches zero, the model stops learning. As a result, we can argue that a gradient is simply the partial derivate with regard to the introduced values. The following equation describes what the given algorithm does:

$$b = a - \gamma \nabla f(a)$$

b = next position
a = current position

The negative sign denotes the minimization component of gradient descent, whereas the gamma denotes the waiting feature and the gradient term (Δf(a)) denotes the direction of the sharpest decrease.

In a machine or deep learning model on applying the algorithm to minimize the cost function J(ω, b) and reaching the local minimum by tweaking the parameter it is observed that the gradient descent function is convex, but there do exist non-convex examples as well, all depending upon the dataset.

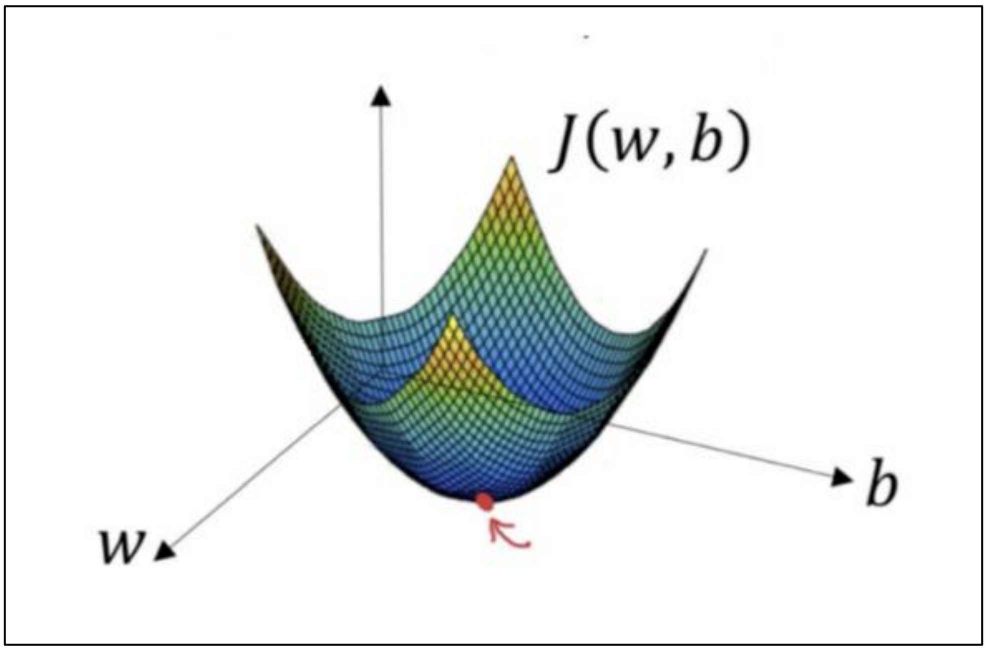


Figure 16: *Gradient descent curve*

3.2.4. LONG SHORT-TERM MEMORY (LSTM)

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that is capable of long-term dependence. Neural networks are a collection of algorithms designed to mimic the human brain and find patterns. RNN is simply a generalized feedforward neural network with internal memory. RNN is recurrent in nature, as the name implies, and so performs the exact function for all data inputs, although the output of the current input is significantly dependent on prior calculations.

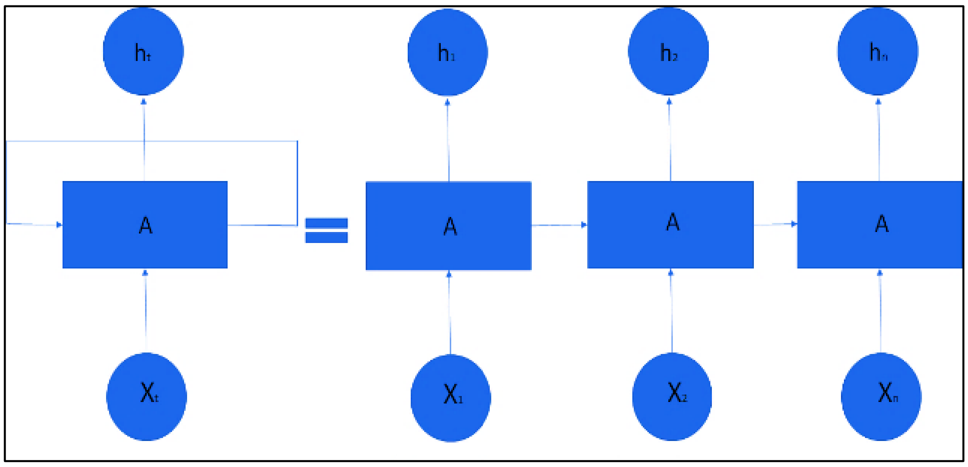


Figure 17: *RNN model*

Unlike other standard feedforward neural organizations, LSTM has feedback connections. It cannot only cycle single information points that are the particular data points, for example, pictures, but also the whole groupings of information, for example, discourse or video. Such a network is used to process and predict problems involving time series and other complex problems such as this one. LSTM, being a modification of RNN, helps resolve one of the major problems faced on any RNN network, which is the vanishing gradient problem. It trains any model by back-propagation. Thus, there are 3 gates that are present in any LSTM network:

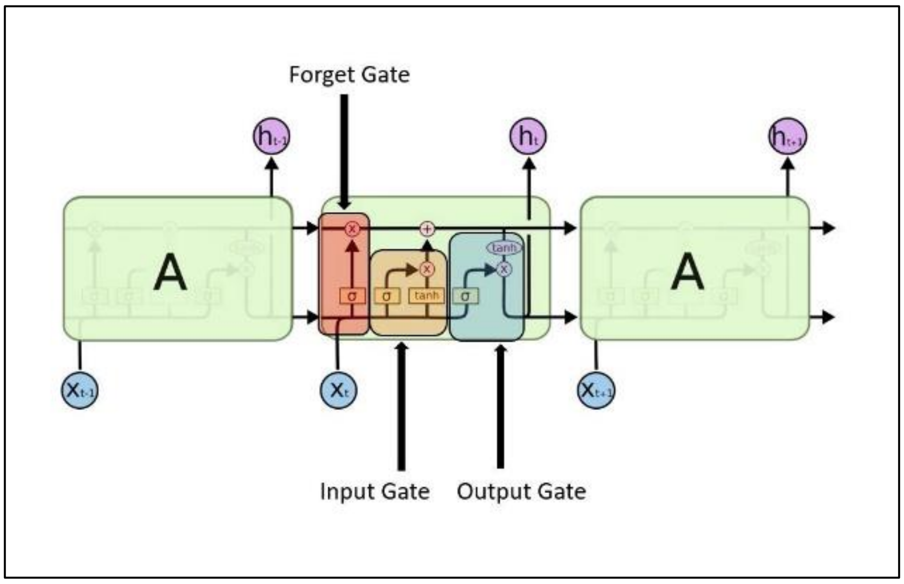


Figure 18: *LSTM model*

1. Input gate – this gate had the sigmoid function that decides which value to let in and the tanh function provides weightage to the values which pass through.
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i$$
$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
2. Forget data – this gate identifies the values that need to be discarded from the block. Another sigmoid function is used in this gate.
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
3. Output gate – the input with the memory present in the block concludes the output. Like the input gate, the sigmoid function decides the values that will go through and the tanh function gives weightage.
$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t * \tanh(C_t)$$

3.2.5. LASSO REGRESSION

Least Absolute Shrinkage and Selection Operator or in short Lasso regression is a kind of linear regression that utilizes shrinkage, i.e., data points are shrunk towards a mid-point, as done when finding the mean. This model is particularly useful and suits well for models that show high levels of multicollinearity as is the case in this particular dataset. This algorithm executes L1 regularization, which adds a penalty that is equivalent to the absolute value of the degree of the coefficients. Lasso solutions are quadratic programming problems, that use the following formula:

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p} |\beta_j|$$

The intensity of the L1 penalty is controlled by a tuning parameter. is essentially the amount of shrinking. If it is zero, we know that all features are taken into account, and it is equivalent to linear regression, in which just the residual sum of squares is used to form a predictive model. If it is infinity, it means that no features are taken into account. The bias increases while the variance decreases with an increase in λ, and vice-versa.

4. RESULTS

4.1 DATA ANALYSIS

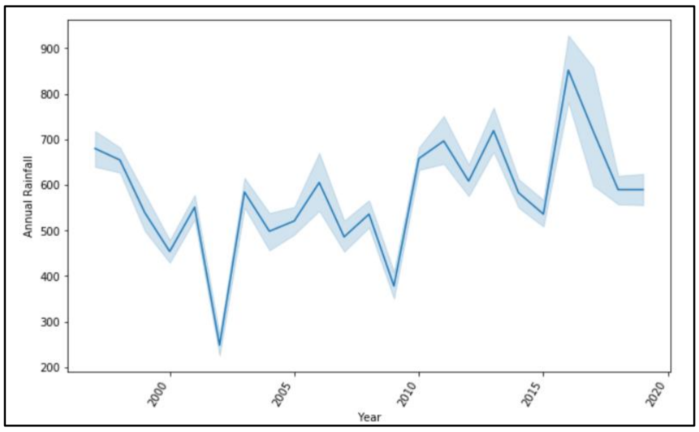


Figure 19: *Annual rainfall vs. Year*

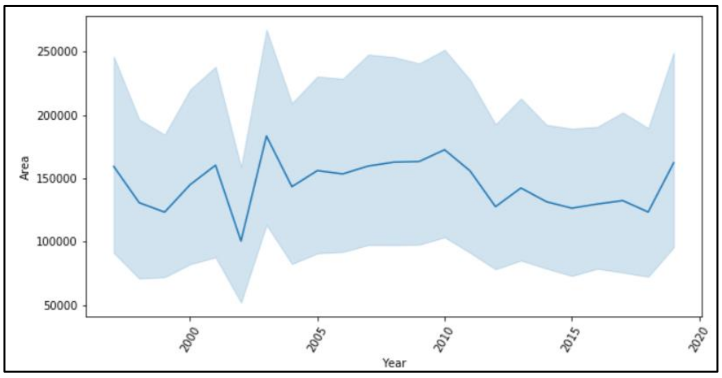


Figure 20: *Area under irrigation vs Year (Bajra)*

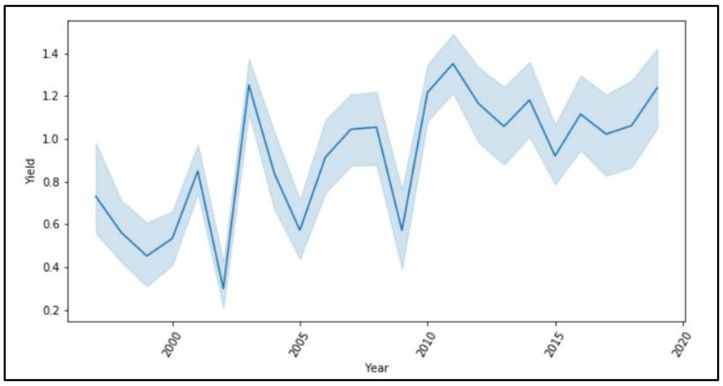


Figure 21: *Yield vs Year (Bajra)*

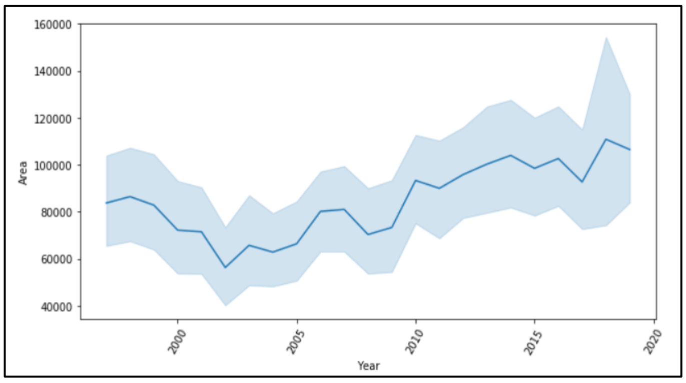


Figure 22: *Area under irrigation vs. Year (Wheat)*

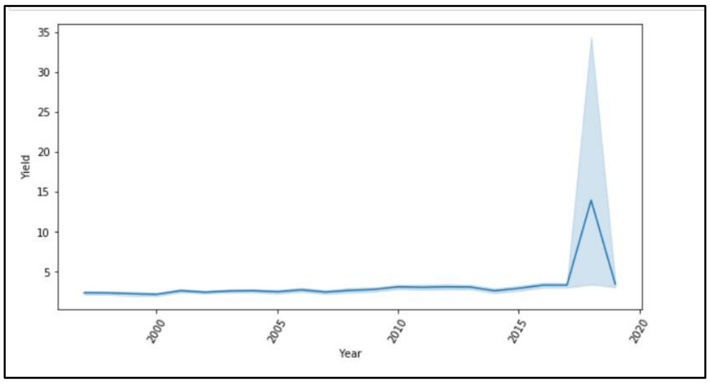


Figure 23: *Yield vs Year (Wheat)*

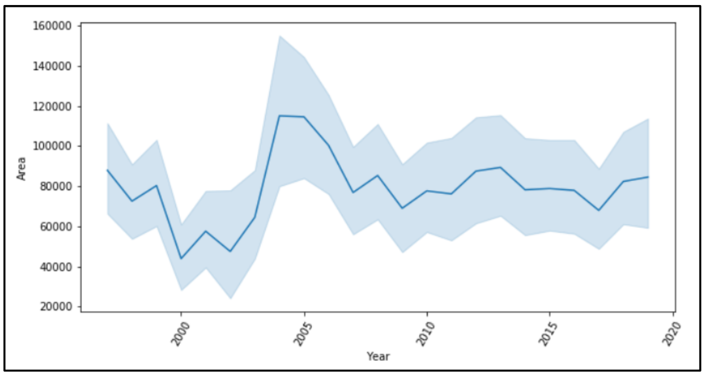


Figure 24: *Area under irrigation vs.Year (Rapeseed & Mustard)*

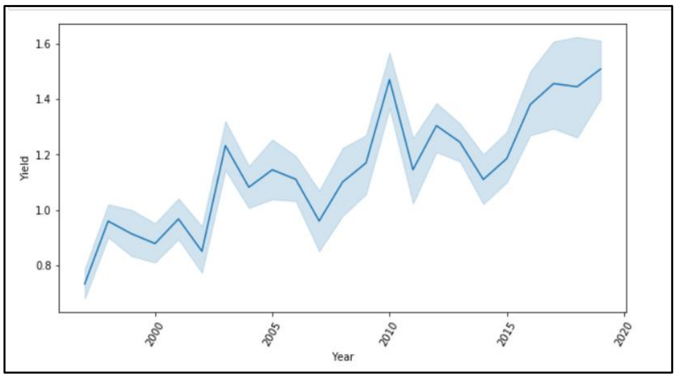


Figure 25: *Yield vs Year (Rapeseed & Mustard)*

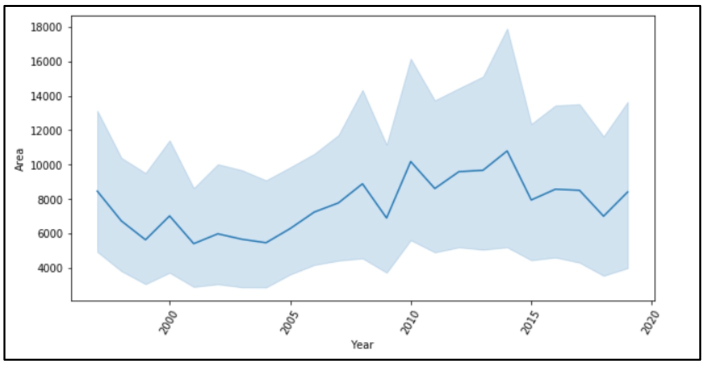


Figure 26: *Area under irrigation vs Year (Barley)*

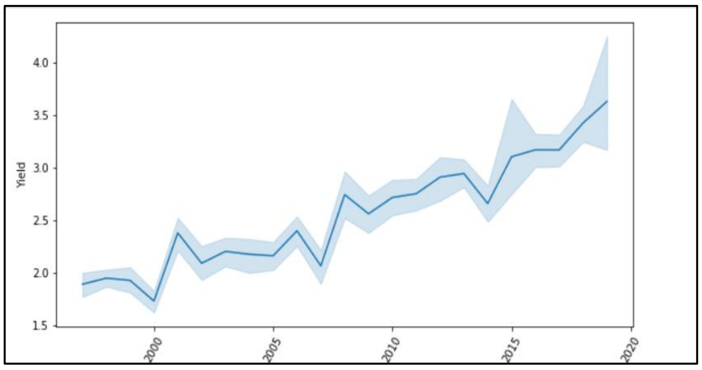


Figure 27: *Yield vs Year (Barley)*

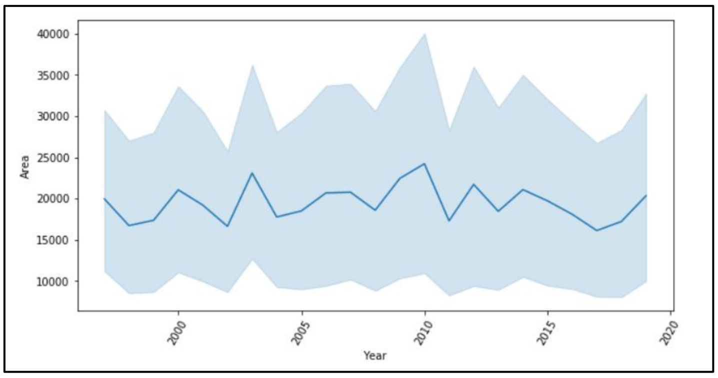


Figure 28: *Area under irrigation vs Year (Jowar)*

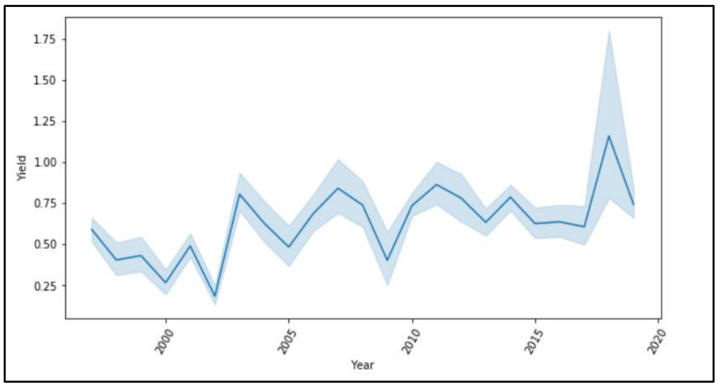


Figure 29: *Yield vs Year (Jowar)*

4.2. MODEL PERFORMANCE

4.2.1. INDIVIDUAL PERFORMANCE

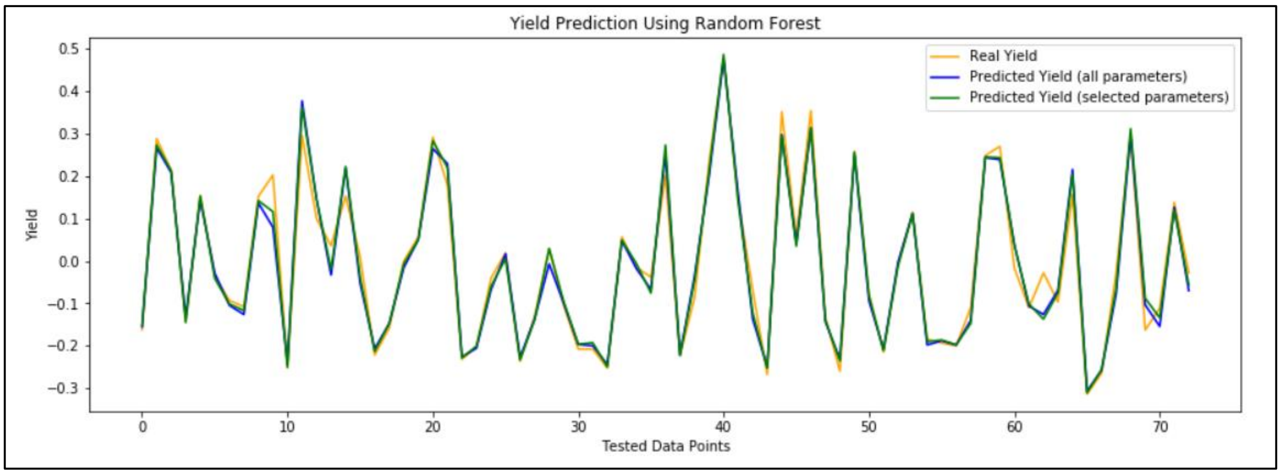


Figure 30: *Model performance of Random Forest*

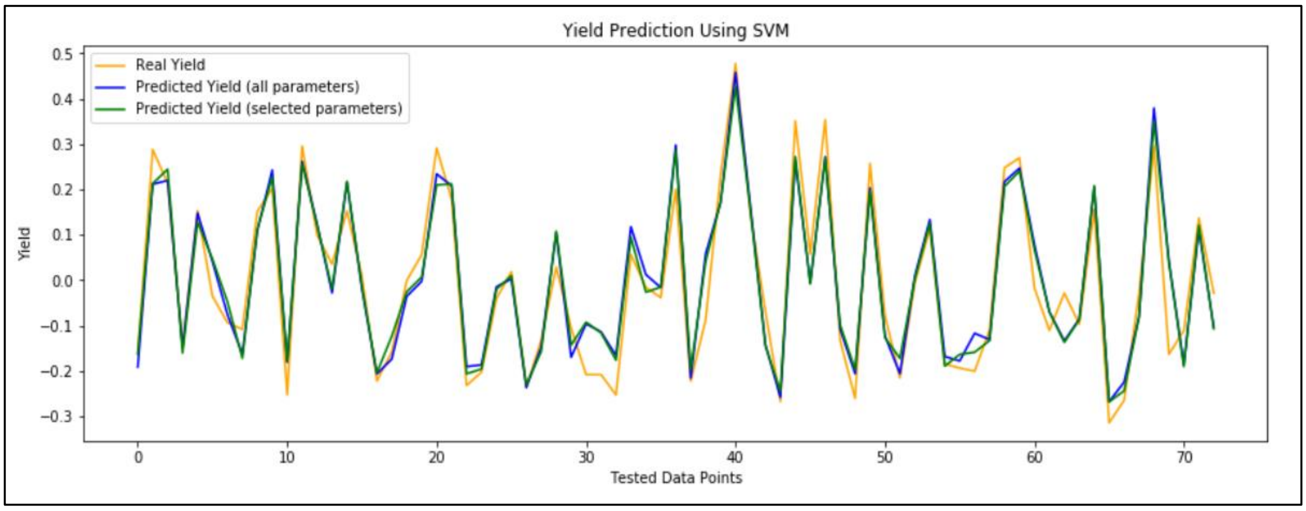


Figure 31: *Model performance of SVM*

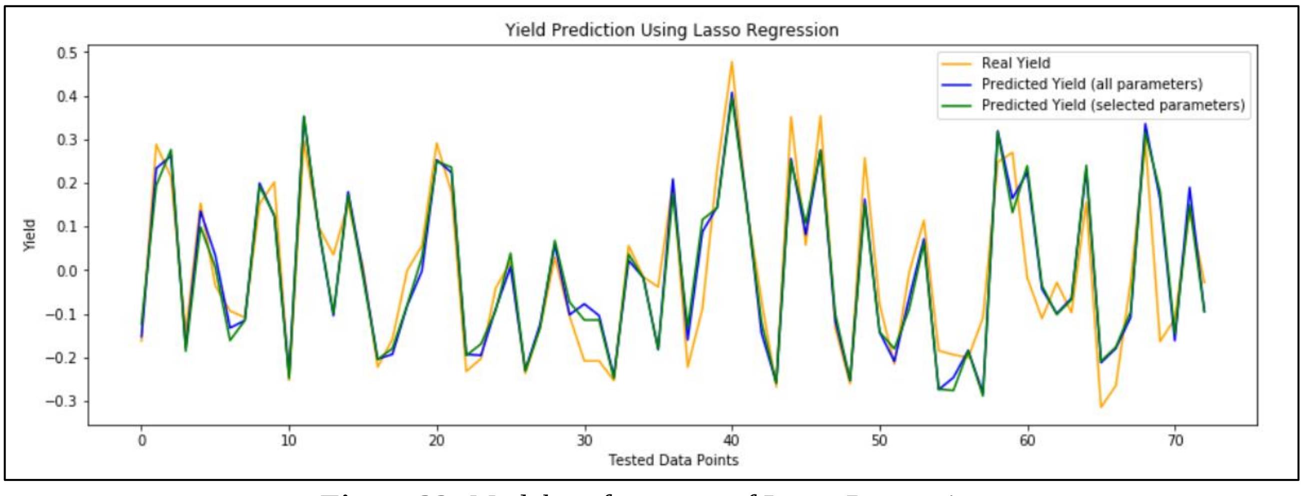


Figure 32: *Model performance of Lasso Regression*

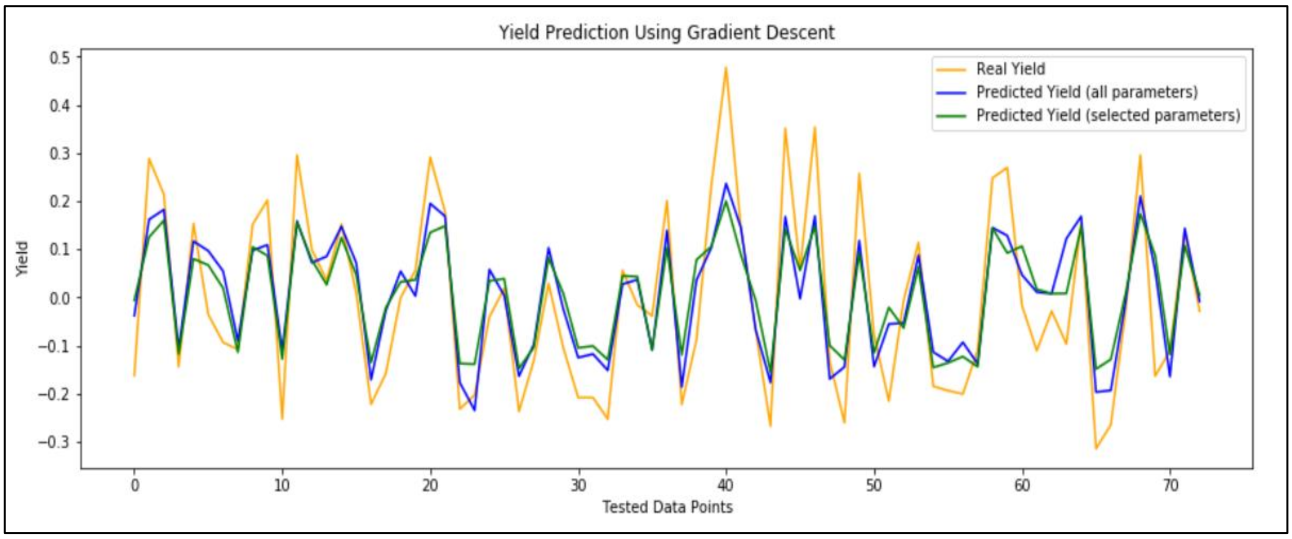


Figure 33: *Model performance of Gradient Descent*

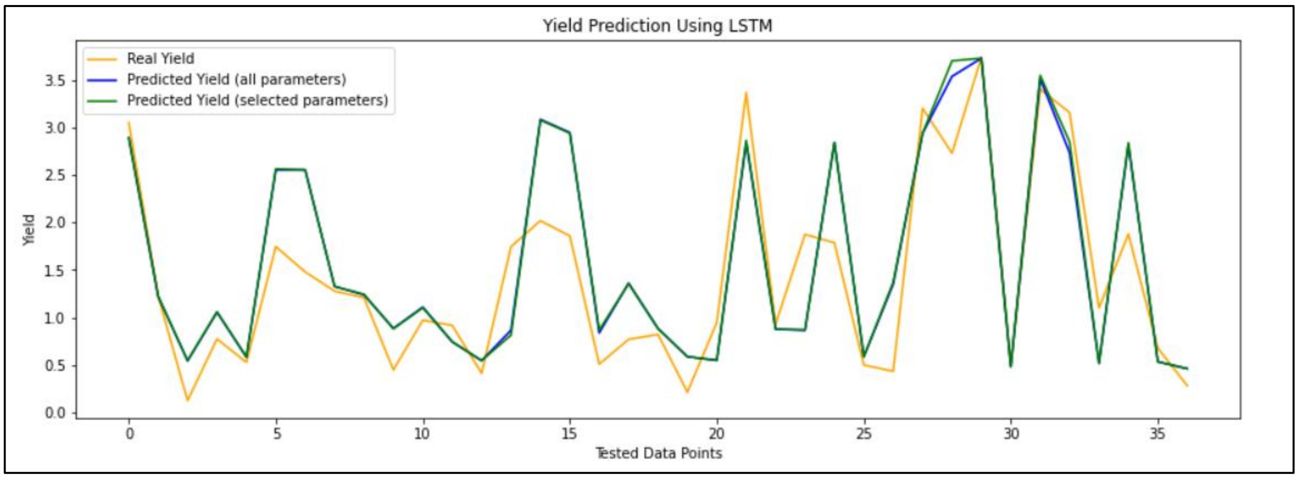


Figure 34: *Model performance of LSTM*

4.2.2. PERFORMANCE COMPARISON

Table 5*: Model performance using only area and production as parameters*

MODELS	R2 SCORE	RMSE	MAE
RANDOM FOREST	0.9706683562457551	0.03210438555877871	0.02171092742447838
SVM	0.9033044480088029	0.0582907187068775	0.04744273781182322
LASSO REGRESSION	0.7929212370102475	0.08530292881366584	0.06279053471454211
GRADIENT DESCENT	0.6721446803333357	0.10733399123627024	0.08833348922241419
LSTM	0.7605970571904346	0.4989149684766394	0.4128063470375437

***Table 6**: Model performance using all parameters*

MODELS	R2 SCORE	RMSE	MAE
RANDOM FOREST	0.9638436515081396	0.03564416484020771	0.025122868659073126
SVM	0.8982433161230687	0.059796757425198625	0.047852253033345137
LASSO REGRESSION	0.8146824212217434	0.08069645755381685	0.0588093331954562
GRADIENT DESCENT	0.737347767385024	0.09606976093705671	0.07906984158002965
LSTM	0.75781934584411	0.50180099201939	0.4164927767185484

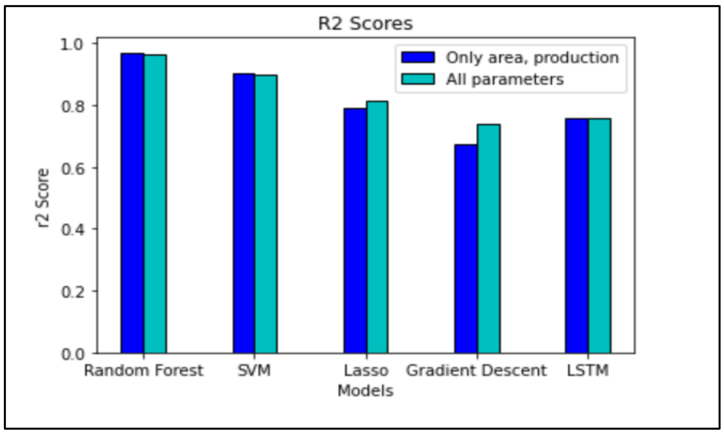


***Figure 35:** R2 Score values comparison*

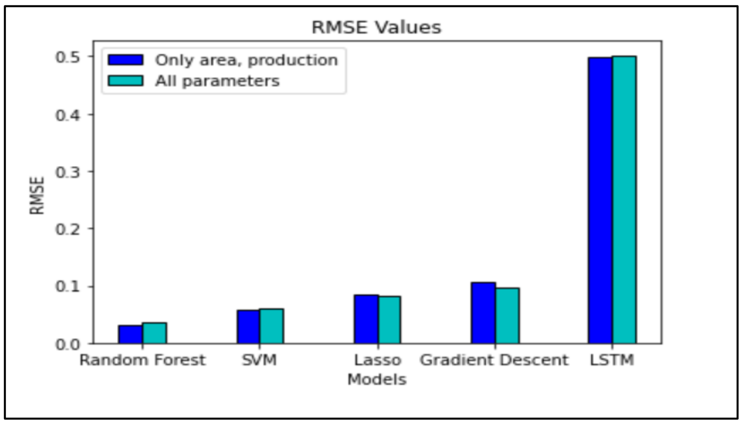


***Figure 36:** RMSE values comparison*

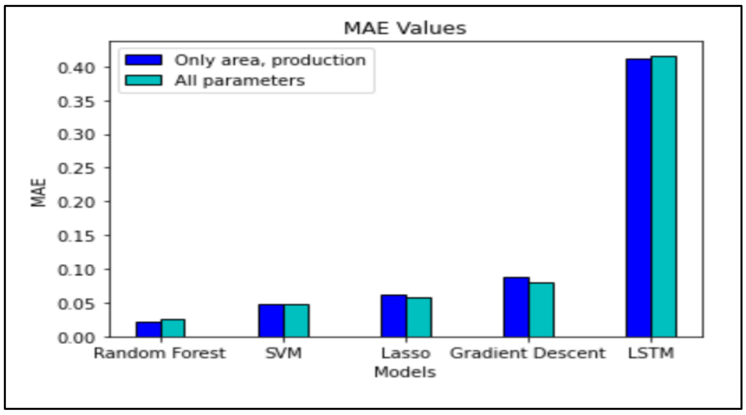


***Figure 37:** MAE values comparison*

5. DISCUSSION

5.1. DISCUSSION ON THE ANALYSIS OF OUR DATASET

From our analysis, we can infer that even though there hasn't been a set rainfall pattern over the years (figure 19), the production of each crop has steadily increased, barring Wheat and Jowar from the years 2016-2019 wherein we can definitely see a sharp decrease in the rainfall patterns. This indicates us towards the rise in dependency on modern irrigation techniques for most crops, while widely water-consuming crops such as Wheat and Jowar still need assistance from natural water sources, i.e., mainly the rain.

While analysing each crop separately, we can conclude that:

- For Bajra, the area under irrigation used for the production of this particular crop has more or less, hasn't changed erratically (figure 20), i.e., from 1997-2019, and has been constant, with a certain dip in between the years 2000-2003 due to reasons unknown, but the yield of this crop has seen a constant rise barring years 2003 and 2009 (figure 21). This can be factored due to better seasonal rainfall as well as the availability of modern and ever improving irrigation methods as well as better fertilizers, pesticides and production techniques.
- For Wheat, the area under irrigation and production when plotted against the years, one can observe that while there's a drop in the area used from 1997-2003, the area started increasing gradually from there on till 2019 (figure 22). Taking into account the yield, it has remained fairly consistent over the years, but we anticipate a certain boom in 2018 but a dip in 2019 for the state (figure 23).There can be several reasons for such an observation. One of them can be the growing need to produce for other parts of the country as well.
- For Rapeseed & Mustard, the area under irrigation used for the production of the given crop has remained pretty much constant if we observe the graph, with a dip during 2000-2003 and a high increase in from 2003-2005 and then becoming more or less constant (figure 24). In the case of yield, over the years there has been a gradual increase in the yield (figure 25) of this particular crop even though the area has been pretty much constant. This can be factored with the increase in demand and modernization of production and irrigation techniques of farming.
- For Barley, again as observed in the case of other crops, the area under production hasn't increased as such (figure 26) but when we observe the yield over the years, there has been a high rise in this particular factor over the years (figure 27).
- For Jowar, the area has remained almost the same over the years (figure 28), which has been the case for most of the crops, but the produce has yet again increased over the years (figure 29) as again observed is the case with all the crops that have been grown in the state of Rajasthan.

From these observations, the conclusion that we can draw is that area is something that has remained pretty much similar over the years, which tells us that area being a very limited resource can not be increased since land is required for several other purposes. But due to the increase in population, the demand has also been constantly increasing, so in order to meet these demands, production needs to be increased, which has been the case over the years. But, since the area hasn't altered much for most crops, the rise in the yield tells us that the modern methods of irrigation and production and the use of better fretilizers and other modernized methods has resulted in a better yield of these particular crops.

5.2. ANALYSING THE RESULTS

For all our models, except LSTM, we have split our dataset into a ratio - 98% training set and 2% testing set – and a random state set at 71. For LSTM we used the ratio – 99% training set and 1% test set – with the same value for random state. We have performed 2 separate analysiss, first using only area, production, year and district as the independent features for our prediction of yield, then adding on the mean seasonal rainfall as well as the soil type of the particular district in order to understand the effect of these variables on the yield. Now, when we analyse our results for each model individually, we observe that:

- In case of the Random Forest algorithm, we get the highest R^2 score, thus, it has higher accuracy when compared to other models for both cases, and that is about 97% in case of selected parameters and about 96.3% in case of all parameters. The RMSE for the selected parameters is about 3.2% and the MAE score is about 2.1%, in the case of all parameters, the RMSE is 3.5% and MAE is 2.5%.
- In case of the Support Vector Machine or commonly referred to as the SVM algorithm, we get a R^2 score of 90.3% when taken into account the selected factors, and for all factors considered we get the score of about 89.8%. For the selected parameters we get RMSE of 5.8% and MAE of 4.74%, in case of all parameters we get RMSE as 5.9% and MAE as 4.78%.
- In case of the Lasso Regression algorithm, we get the R^2 score of 79.2% in case of selected parameters and in case of all parameters we get 81.4%. We get RMSE as 8.5% and MAE as 6.2% in case of selected parameters while we get RMSE as 8.06% and MAE as 5.8% in case of all parameters.
- In case of the Gradient Descent algorithm, we get an R^2 score of 67.2% when we use if for the dataset with just the selected parameters, while on the dataset with all the parameters we get a score of 73.7%. For the RMSE and MAE values for the selected parameters dataset we get 10.7% and 8.8% respectively. In case of the dataset with all the parameters we get RMSE as 9.6% and MAE as 7.9%.
- In case of the Long Short Term Memory or commonly known as the LSTM algorithms, when this model is applied on both types of dataset of ours, we get an R^2 score of 76.05%, RMSE as 49.8% and MAE as 41.2% for dataset with selected parameters. While in the case of all parameters we get an R^2 score of 75.7%, RMSE as 50.1% and MAE as 41.6%. We have used the same parameters for all our models while splitting into training and testing sets to get the best possible accuracy, but in case of LSTM since the algorithm in general works best and requires a larger database than what is being provided, we decided to reduce the testing set by 1%. This gave us a higher accuracy when applied on our dataset.

On observation, we can see that most of our models have almost similar accuracy in the case of both types of our datasets. Mostly, the accuracy of most of our models decreases on the addition of the scientific parameters of soil and mean rainfall, but in models such as Lasso Regression and Gradient Descent we observe the contrary. Our LSTM model, though it shows a higher R^2 score when compared to the Gradient Descent algorithm, the RMSE and MAE values are also very high, which is not a very good sign for any predictive model.

6. CONCLUSION

From all the discussions and analysis we can understand that the machine learning models that we have applied, that are – Random Forest, Support Vector Machine (SVM) and Lasso Regression – when compared to the the deep learning models that we have applied – Gradient Descent and Long Short Term Memory (LSTM) – give us a better and more accurate result. This might be because models like LSTM require a bigger quantum of data for a better predictional analysis when compared to other models.

Also, based on our observations, we can deduce that most of our models perform better on the specified parameters, whereas models such as Gradient Descent and Lasso Regression perform better when applied to the dataset with all of the characteristics we wish to utilise for our predictional analysis. As we know, while soil and rainfall quantity do play a vital part in the production of any crop and in general farming, we can claim that we need a deeper investigation of these elements as well as a larger database for the research of such elements through our prediction models in real life.

In the end we can conclude that out of all our models, Random Forest algorithm outperforms them all when applied on any of the datasets that we have.

7. REFERENCES

[1] Rajasthan Crop Data (1997 – 2010), *www.kaggle.com*

[2] Rajasthan Crop Data (2011 – 2019), *www.agriculture.rajasthan.gov.in*

[3] Soil Type, *www.agriculture.rajasthan.gov.in*

[4] Rainfall data (1901 – 2002), *www.indiawaterportal.org*

[5] Rainfall data (2004 – 2010), *water.rajasthan.gov.in*

[6] Rainfall data (2011 – 2019), *www.agriculture.rajasthan.gov.in*

[7] Random Forest algorithm, *www.kaggle.com*

[8] Support Vector Machine algorithm, *www.kaggle.com*

[9] Lasso Regression algorithm, *www.pluralsight.com*

[10] Gradient Descent, *www.github.com*

[11] Data Pre-processing theory, *www. towardsdatascience.com*

[12] Long Short Term Memory (LSTM) algorithm, *www.kaggle.com*

[13] Saeed Khaki, Lizhi Wang and Sotirios V. Archontoulis, "A CNN-RNN Framework for Crop Yield Prediction", Frontiers in Plant Science, v. 10, 2019, 1750.

[14] Aditya Shastry, H.A. Sanjay and E. Bhanusree, "Prediction of Crop Yield using Regression Techniques", International Journal of Soft Computing, 12(2), 2017, 96 – 102.

[15] Anna X. Wang, Caelin Tran, Nikhil Desai, David Lobell, and Stefano Ermon. 2018, " Deep Transfer Learning for Crop Yield Prediction with Remote Sensing Data", In COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS), Menlo Park and San Jose, CA, USA. ACM, New York, NY, USA, June 20–22, 5 pages, 2018.

[16] Thomas van Klompenburg, Ayalew Kassahun and Cagatay Catal, "Crop yield prediction using machine learning: A systematic literature review", Computers and Electronics in Agriculture, Volume 177, October 2020, 105709.