Assignment No-03 (CS 6301- Machine Learning)

Title : Crop Yield Prediction

Statement :

CYP is one of the challenging problems in precision agriculture. Crop yield depends on many factors such as climate, whether, soil, use of fertilizers and seed variety. To develop a model to predict the yield of a crop before cultivating into the agricultural field. A predictive model will be using the evaluation parameters like temperature, rail fall, area, season. The prediction is based on this past experience.

A. Identification of the Dataset :

I. Type of the Dataset (Description) : The data is collected from different sources.

1. Data.gov – state wise, district wise, year wise crop "production"

State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0

2. Indiawaterportal.org – State wise, district wise "rainfall" for all months and year wise "temperature".

State	District	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	vlookup
Andhra Pradesh	Adilabad	1901.0	6.725	10.488	23.288	35.560	23.119	115.546	294.119	276.865	181.615	47.310	1.339	0.000	Andhra PradeshAdilabad
Andhra Pradesh	Adilabad	1902.0	0.420	0.000	0.388	6.070	3.331	45.960	233.973	167.971	198.177	26.447	35.083	11.222	Andhra PradeshAdilabad
Andhra Pradesh	Adilabad	1903.0	6.643	1.956	0.173	4.551	33.348	132.078	436.611	334.544	226.037	138.818	14.095	8.823	Andhra PradeshAdilabad
Andhra Pradesh	Adilabad	1904.0	0.054	0.121	11.446	0.017	16.900	131.048	160.694	81.865	251.577	110.391	0.146	0.130	Andhra PradeshAdilabad
Andhra Pradesh	Adilabad	1905.0	0.589	2.293	8.252	35.020	17.569	79.937	96.331	313.522	361.697	4.950	0.146	0.000	Andhra PradeshAdilabad

YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1901	17.99	19.43	23.49	26.41	28.28	28.60	27.49	26.98	26.26	25.08	21.73	18.95
1902	19.00	20.39	24.10	26.54	28.68	28.44	27.29	27.05	25.95	24.37	21.33	18.78
1903	18.32	19.79	22.46	26.03	27.93	28.41	28.04	26.63	26.34	24.57	20.96	18.29
1904	17.77	19.39	22.95	26.73	27.83	27.85	26.84	26.73	25.84	24.36	21.07	18.84
1905	17.40	17.79	21.78	24.84	28.32	28.69	27.67	27.47	26.29	26.16	22.07	18.71

II. **Data Quality and Analysis** : The dataset needed a lot of processing before it could actually be used I the model. The dataset contained null values, they were not directly mergeable.

- First dataset(Production)
  Rows = 246091 & columns = 7
  Null values 3730
- Second dataset(rainfall)
  Rows = 55319 & columns = 16
  Null values 35
- Third dataset temperature) – rows = 117 and columns = 13
  Null values 0

III. **Features Pre-Processing** :

1. Production data:
   Checked the unique season.

```
crop_df['Season'].unique()
```

```
array(['Kharif     ', 'Whole Year ', 'Autumn     ', 'Rabi       ',
       'Summer     ', 'Winter     '], dtype=object)
```

From the dataset it is understood that some of the names has spaces after them and some attributes has same meaning but with different names. Thus renaming of the attributes is required. For renaming I used replace function of pandas.

Renaming the columns for consistancy in the dataset

```
#renaming the columns
crop_df['Season'] = crop_df['Season'].replace(['Winter     '],'Rabi')
crop_df['Season'] = crop_df['Season'].replace(['Autumn     '],'Rabi')
crop_df['Season'] = crop_df['Season'].replace(['Rabi       '],'Rabi')
crop_df['Season'] = crop_df['Season'].replace(['Kharif     '],'Kharif')
crop_df['Season'] = crop_df['Season'].replace(['Whole Year '],'whole_year')
crop_df['Season'] = crop_df['Season'].replace(['Summer     '],'Summer')
```

```
crop_df['Season'].unique()
```

```
array(['Kharif', 'whole_year', 'Rabi', 'Summer'], dtype=object)
```

Season column has string attributes, the algorithm takes only integer/float so I changed the season column attributes to integers using label encoder. Now the seasons like kharif is represented as 0, whole year as 3, etc. and stored that values into another column named Season_ID.

```
#Converting Season name to season id
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
seasonsI = crop_df['Season']
crop_df['Season_ID'] = label_encoder.fit_transform(seasonsI)
print(crop_df['Season_ID'])
```

```
0         0
1         0
2         0
3         3
4         3
         ..
246086    2
246087    2
246088    3
246089    1
246090    1
Name: Season_ID, Length: 246091, dtype: int32
```

I checked the data types of the dataframe. Here the state name and district names are object types and this two features will help in merging of the other datasets.

```
crop_df.dtypes
```

```
State_Name        object
District_Name     object
Crop_Year          int64
Season            object
Crop              object
Area             float64
Production       float64
dtype: object
```

As this features will be required for merging. For merging it is necessary that the column values match, so I decided that I will keep the strings in lower case. I converted the first two features into lowercase.

Converting the casing of strings to match the other datasets while merging

```
crop_df["State_Name"] = crop_df["State_Name"].str.lower()
crop_df["District_Name"] = crop_df["District_Name"].str.lower()
```

```
crop_df.head()
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Season_ID
0	andaman and nicobar islands	nicobars	2000	Kharif	Arecanut	1254.0	2000.0	0
1	andaman and nicobar islands	nicobars	2000	Kharif	Other Kharif pulses	2.0	1.0	0
2	andaman and nicobar islands	nicobars	2000	Kharif	Rice	102.0	321.0	0
3	andaman and nicobar islands	nicobars	2000	whole_year	Banana	176.0	641.0	3
4	andaman and nicobar islands	nicobars	2000	whole_year	Cashewnut	720.0	165.0	3

I then checked for null values. I found that there are 3730 null values in production column. For removing null values I used dropna() function of pandas to drop the rows having null values.

Removing Null values

```
#check if there are null values in the dataset
crop_df.isnull().sum()
```

```
State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       3730
Season_ID           0
dtype: int64
```

```
#Drop the rows with null values
crop_df = crop_df.dropna()
```

```
#check if there are null values in the dataset
crop_df.isnull().sum()
```

```
State_Name       0
District_Name    0
Crop_Year        0
Season           0
Crop             0
Area             0
Production       0
Season_ID        0
dtype: int64
```

Now it is required that this processed dataset should be stored somewhere for used when merging. I used to_csv() function to export the dataframe into csv file format.

converting final dataframe to csv

```
crop_df.to_csv('crop_yield.csv')
```

Final dataset

State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Season_ID
andaman and nicobar islands	nicobars	2000	Kharif	Arecanut	1254.0	2000.0	0
andaman and nicobar islands	nicobars	2000	Kharif	Other Kharif pulses	2.0	1.0	0
andaman and nicobar islands	nicobars	2000	Kharif	Rice	102.0	321.0	0
andaman and nicobar islands	nicobars	2000	whole_year	Banana	176.0	641.0	3
andaman and nicobar islands	nicobars	2000	whole_year	Cashewnut	720.0	165.0	3

2. Rainfall Data:

   Shape of the dataset.

```
rainfall_df.shape
```

```
(55319, 16)
```

I checked for null values and then removed them using dropna() function.

```
rainfall_df.isnull().sum()
```

```
State         0
District      0
Year         35
Jan          35
Feb          35
Mar          35
Apr          35
May          35
Jun          35
Jul          35
Aug          35
Sep          35
Oct          35
Nov          35
Dec          35
vlookup     102
dtype: int64
```

Their are null values in the datasets

```
rainfall_df = rainfall_df.dropna()
```

The rainfall given in the dataset has month wise data. But for prediction rainfall based on the seasons is required. In order to calculate the season wise rainfall I collected information about the cropping seasons. Here

Kharif = July – October

Rabi(winter) = October -March

Summer = March – June

Based on the months I calculated the mean of rainfall and added into a new column of that season for that year.

Computing the mean rainfall for kharif, rabi, whole year and summer

```
The kharif cropping season is from July –October
Rabi cropping season is from October-March (winter).
The crops grown between March and June are summer crops
```

```
kharif = rainfall_df.iloc[ : , 9:13]
rabi = rainfall_df.iloc[ : ,[13, 14, 3, 4, 5]]
whole_year = rainfall_df.iloc[:,3:15 ]
summer = rainfall_df.iloc[:, 6:9]
rainfall_df['kharif'] = kharif.mean(axis = 1)
rainfall_df['rabi'] = rabi.mean(axis = 1)
rainfall_df['whole_year'] = whole_year.mean(axis = 1)
rainfall_df['summer'] = summer.mean(axis = 1)
rainfall_df.head()
```

Here the “vlookup” feature is a concatenation of state and district column, thus it is redundant and it’s not required. I removed that using drop() function.

Dropping unnecessary column ¶

```
rainfall_df = rainfall_df.drop('vlookup', axis =1)
```

The data needed to be merged so I renamed the columns to match the production dataset.

Renaming of the features based on the features of the CYP dataset ¶

```
#renaming the columns
rainfall_df.rename(columns = {'State':'State_Name',
                              'District':'District_Name',
                              'Year':'Crop_Year'},inplace = True)
```

State_Name and District_Name has datatype as object. And the crop_year had float64.

```
rainfall_df.dtypes
```

```
State_Name        object
District_Name     object
Crop_Year        float64
Jan              float64
Feb              float64
Mar              float64
Apr              float64
May              float64
```

Crop_Year has a datatype as int in the production data, it needed to be changed. I did that using astype(int) function. And I lower cased the values of State_Name and District_Name.

Changing format of crop year to match the format of CYP dataset ¶

```
rainfall_df['Crop_Year'] = rainfall_df['Crop_Year'].astype(int)
```

Changing the casing of the strings to match the CYP dataset

```
rainfall_df["State_Name"] = rainfall_df["State_Name"].str.lower()
rainfall_df["District_Name"] = rainfall_df["District_Name"].str.lower()
```

Here I extracted the required features from the dataframe.

Extracting required feature from the dataframe

```
rain_df = pd.melt(rainfall_df.iloc[:,[0, 1, 2, 15,16, 17, 18]], id_vars=["State_Name", "District_Name","Crop_Year"],
                  var_name="Season", value_name="Rainfall")
```

```
rain_df['Season'].unique()
```

```
array(['kharif', 'rabi', 'whole_year', 'summer'], dtype=object)
```

Similar to the production data this data also needs to be stored. Used to_csv to export the data from dataframe to csv file.

Converting the final dataframe into a csv file

```
rain_df.to_csv('Rainfall.csv')
```

Final dataset

	State_Name	District_Name	Crop_Year	Season	Rainfall
0	andhra pradesh	adilabad	1901	kharif	199.97725
1	andhra pradesh	adilabad	1902	kharif	156.64200
2	andhra pradesh	adilabad	1903	kharif	284.00250
3	andhra pradesh	adilabad	1904	kharif	151.13175
4	andhra pradesh	adilabad	1905	kharif	194.12500

3. Temperature data:

   Similar steps has been taken for the temperature dataset. The final dataset is

	Crop_Year	Season	Temperature(M)
0	1901	kharif	26.4525
1	1902	kharif	26.1650
2	1903	kharif	26.3950
3	1904	kharif	25.9425
4	1905	kharif	26.8975

Final datasets

To create data for training and testing. I used merge() function to merge the three pre-processed datasets. I first merged production dataset and rainfall dataset.

Merging crop data and rainfall data

```
#Merging crop data and rainfall data based on'State_Name','District_Name','Crop_Year','Season' parameters
FCD_df = pd.merge(crop_df, rainfall_df, how='inner', left_on=['State_Name','District_Name','Crop_Year','Season'], right_on = ['St
```

```
FCD_df.head()
```

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Season_ID	Rainfall
0	andhra pradesh	anantapur	1998	whole_year	Sugarcane	700.0	65900.0	3	73.456083
1	andhra pradesh	anantapur	1998	whole_year	Tobacco	100.0	100.0	3	73.456083
2	andhra pradesh	anantapur	1999	whole_year	Dry chillies	5199.0	5759.0	3	53.643333
3	andhra pradesh	anantapur	1999	whole_year	Onion	1770.0	32364.0	3	53.643333
4	andhra pradesh	anantapur	1999	whole_year	Sugarcane	884.0	83126.0	3	53.643333

Merged the temperature dataset with the previously merged dataset.

```
#Merging crop data and temperature data based on 'Crop_Year','Season' parameters
CP_df = pd.merge(FCD_df, temp_df, how='inner', left_on=['Crop_Year','Season'], right_on = ['Crop_Year','Season'])
```

Final dataset

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Season_ID	Rainfall	Temperature(M)
0	andhra pradesh	anantapur	1998	whole_year	Sugarcane	700.0	65900.0	3	73.456083	24.695
1	andhra pradesh	anantapur	1998	whole_year	Tobacco	100.0	100.0	3	73.456083	24.695
2	andhra pradesh	chittoor	1998	whole_year	Sugarcane	34500.0	3377000.0	3	81.818750	24.695
3	andhra pradesh	east godavari	1998	whole_year	Sugarcane	16400.0	1254900.0	3	97.433667	24.695
4	andhra pradesh	east godavari	1998	whole_year	Tobacco	7000.0	15900.0	3	97.433667	24.695

IV. Format of the Dataset : .csv and .xlsx

B. Identification of Learning Model (Supervised Learning)

I. Algorithm used : Random Forest

I selected this algorithm because this algorithms has given the highest accuracy. I also used SVM and Linear Regression.

II. Methodology used : I used Regression analysis for prediction.

III. Model building, Training & Testing :

Separated the Dependent and Independent variables.

Dependent and Independent Variable sepration ¶

```
# Production/ yield of the crop depends on rainfall, temperature, area, crop, season
X = CP_df.iloc[:,[3,4,5,8,9]]
y = CP_df['Production']
```

```
X
```

	Season	Crop	Area	Rainfall	Temperature(M)
0	whole_year	Sugarcane	700.0	73.456083	24.695000
1	whole_year	Tobacco	100.0	73.456083	24.695000
2	whole_year	Sugarcane	34500.0	81.818750	24.695000
3	whole_year	Sugarcane	16400.0	97.433667	24.695000
4	whole_year	Tobacco	7000.0	97.433667	24.695000

Using label encoder I converted the season and crop column to inteegers.

Splitting the data into test and train

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
se = LabelEncoder()
X['Season'] = se.fit_transform(X['Season'])
X['Crop'] = se.fit_transform(X['Crop'])
```

Using train test and split function I sclised the data into 80/20 ratio with random_state as 42.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 42, test_size=0.2)
print('Train set', X_train.shape, y_train.shape)
print('Test set', X_test.shape, y_test.shape)
```

```
Train set (11252, 5) (11252,)
Test set (2814, 5) (2814,)
```

Used random forest regression to fit the model and train it.

Random Forest ¶

```
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor(n_estimators=150)
```

```
clf.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=150)
```

```
y_pred = clf.predict(X_test)
```

IV. Model Accuracy, Prediction & Precession :

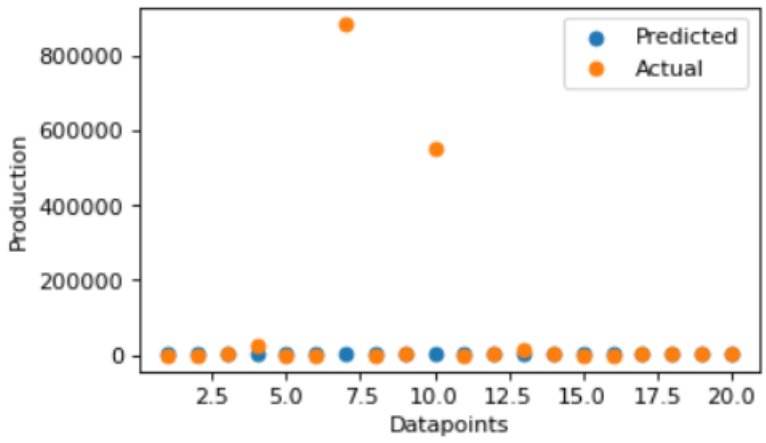
R^2 accuracy - 35.82%

```
from sklearn import metrics
```

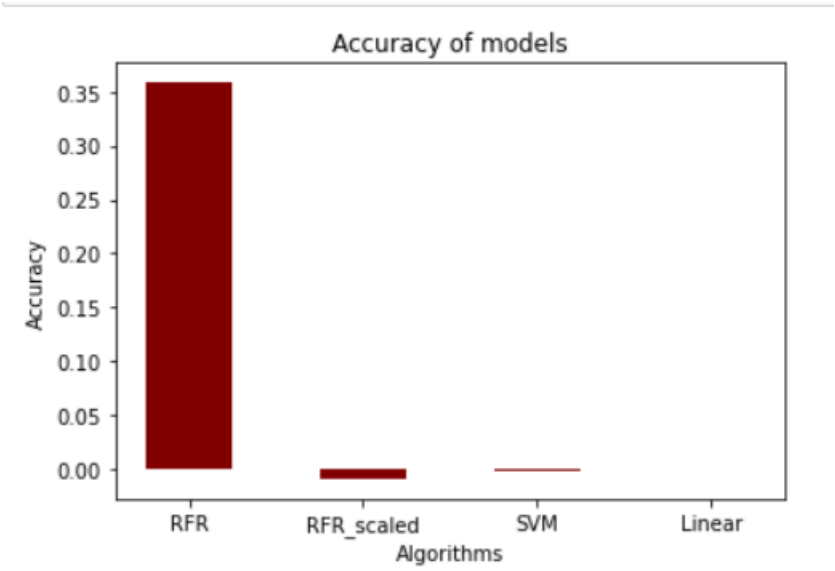
```
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
R^2: 0.35822084825800726
```

Graph between predicted and actual values.



The accuracy comparison between different algorithmic models.



C. Key Learning Outcomes :

Random forest as the most accuracy among the other algorithms. The highest accuracy achieved is 35.8%. The accuracy can be increased if temperature dataset also has state name and district name the accuracy would be increased by at least 10%. The datasets were not good enough for the prediction.

Production data pre-processing - https://drive.google.com/file/d/1VgW1yw36xSqXxm3fwC9YmPEEQR_DP5YN/view?usp=sharing

Rainfall data pre-processing - https://drive.google.com/file/d/1qWW9qr6-1cOd7zxNtXr5nTsqexkYoMI0/view?usp=sharing

Temperature data pre-processing – https://drive.google.com/file/d/1e3YAVMCe_Re_KeSVgaK1hl8NBuYFaS9N/view?usp=sharing

Prediction model - https://drive.google.com/file/d/1g6K2BfQuVsAfrTg4CXFEf_8T1cbnedi8/view?usp=sharing

Submitted By:

Jayant Choudhary

Roll No- AU18B1003