CS534: EXPLORING THE AGRICULTURE IN INDIA USING SPARK AND MACHINE LEARNING

Omkar Asawale (osa20)

Pranita Eugena Burani (peb63)

Introduction

Agriculture plays a vital role in Indian Economy. It contributes about 18% towards Annual GDP, 70% of jobs in rural areas are agriculture dependend. Crop Yield in any country mainly depends on technological advancements and weather variability. Technological advancements do not involve uncertainities, they only contibute towards the increase in production, therefore some parameter of time can be used used to study the effect of technology on yield, whereas the weather conditions have always been an uncontrolled source for the prediction of yield. But due to the latest advancements, weather forecast has been reliable and accurate. This makes the whole idea of predicting agricultural yield acceptable.

Data Collection/ Source of the Data

We wanted to know much India has progressed since its independence and the formation of Republic of India in 1950. The Indian Agriclture and Climate Data set we are using is from the dataset compiled by Duke university. "Information about data for development research". The database provides district level data on agriculture and climate in India from 1957/58 through 1986/87. The dataset includes information on

1. Area planted, production and farm harvest prices for five major and fifteen minor crops.
2. Areas under irrigated and high-yielding varieties (HYV) for major crops.
3. Data on agricultural inputs, such as, fertilizers, bullocks and tractors - in both quantity and price terms
4. Agricultural labor, cultivators, wages and factory earnings, rural population and literacy proportion.
5. Meteorological station level climate data (average climate over 30 year period)
6. Soil data

Objectives

1. Predict the crop yield for major crops in one state based on weather conditions.
2. Recommend high yield variety for every district in the state.

Data Description

Apart from basic ID's the important features in the dataset we used in this analysis are

1. STATENAM - Name of the state
2. DISTNAME - Name of the district
3. ACROP - Area of the crop planted(number*1000 ha)

4. QCROP - Production of the crop(number*1000 tonnes)
5. PCROP - Price of the crop(Rupees/quintal)
6. RNMONTH - Rainfall in mm for every month
7. TNMONTH - Temperature in degree celcius for every month
8. YEAR - 1956-1987

For more detailed description of the data
[https://ipl.econ.duke.edu/dthomas/dev_data/datafiles/india_agric_climate.htm (https://ipl.econ.duke.edu/dthomas/dev_data/datafiles/india_agric_climate.htm)](https://ipl.econ.duke.edu/dthomas/dev_data/datafiles/india_agric_climate.htm)

Preprocessing

The dataset provided was in cfm format(Coldfusion Markup Language) and each cfm file is a year's worth of data. Each file contains a contains a continuous list of space-separated values; these are observations for 271 districts and 227 variables per year.

cfm files were converted to text files. With the help of STATA software application, we converted .txt files to .dta files, finally used pandas library to convert .dta to .csv files.

STATA produced csv files based on the year of data. With the help of a small python script, we were able to merge all the files into a single csv file. This file was loaded into spark for analysis.

We have used Pyspark to remove unwanted data from the csv file. Due to inconsitencies in data from 1950 to 1955. We decided not use it for analysis in this report.

Data Analysis

Analysis 1:

Which States to Consider for the chosen Crops?

Basic objective is to maximize the yield production. In our dataset we have data from 1956-87 So we chose to plot the above graphs for 1956, 1966,1976 and 1986 In 1956 - Top 3 highest Rice Yield is from the states - West Bengal,Tamil Nadu and Bihar Lowest 3 Yield from Punjab, Haryana, Rajastan

Similarly for the years 1966, 1976,1986 The top and lowest yields are as follows 1966- High - West Bengal, Tamil Nadu, Orissa and Low - Punjab, Gujarat, Rajasthan 1976- High - West Bengal, Tamil Nadu, Bihar and Low - Madhya Pradesh, Gujarat, Rajasthan 1988- High - West Bengal, Punjab, Tamil Nadu and Low - Maharashtra ,Gujarat, Rajasthan

Not only should we cold we consider the Production but also the should consider the Area the crops were planted in.

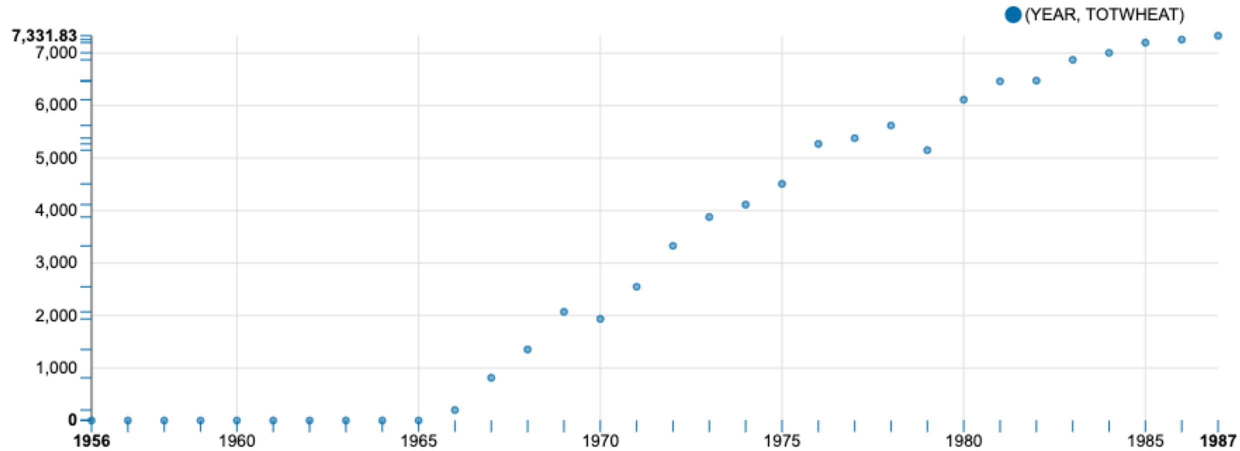
For this the results for the above years is as follows. 1956 - High - Orissa, West Bengal, Bihar and Low - Punjab, Haryana, Rajasthan 1966 - High - Orissa, West Benagl, Bihar and Low - Gujarat, Punjab, Rajasthan 1976 - High - Orissa, West Bengal, Bihar and Low - MadhyaPradesh, Gujarat, Rajasthan 1986 - High - West Bengal, Orissa, Bihar and Low - Maharashtra, Gujarat, Rajasthan.

From the above analysis it is clear that West Bengal, Orissa, Bihar have highest area allotted for Rice plantation and also had highest yield among others states.

Rice plantation and also had highest yield among others states.

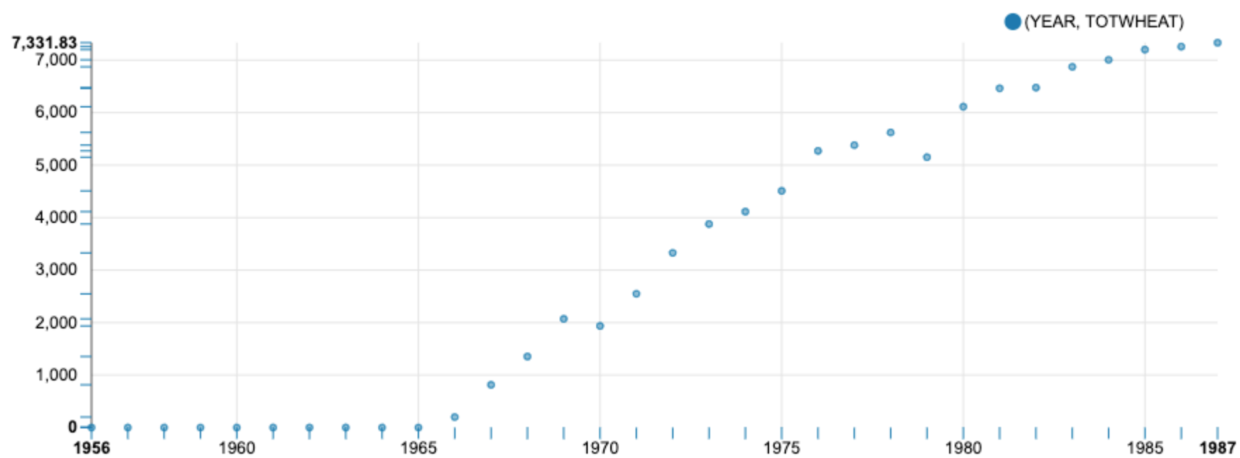
Similar Analysis was done for other crops and decided to use. West_Bengal - For Rice Uttar_Pradesh - For Sugar Punjab - For Cotton Punjab - For Wheat

Plots for the above analysis are in the folder with file names as crop names



Analysis 2

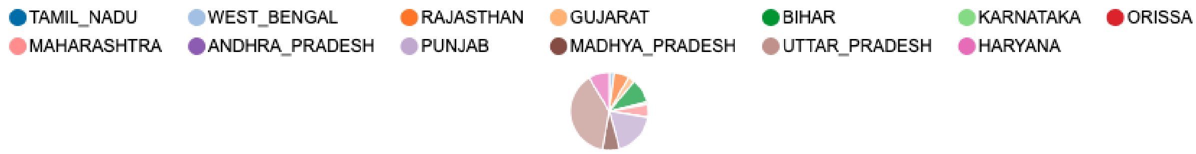
The cost per quintal values are plotted against the years for crops corresponding to the states of their high production and low production, no big difference in the price of the crop for that particular year, but with the increase in years the price increased(with some dips, because of few nulls in the data).



Analysis 3

Comparing production of wheat:

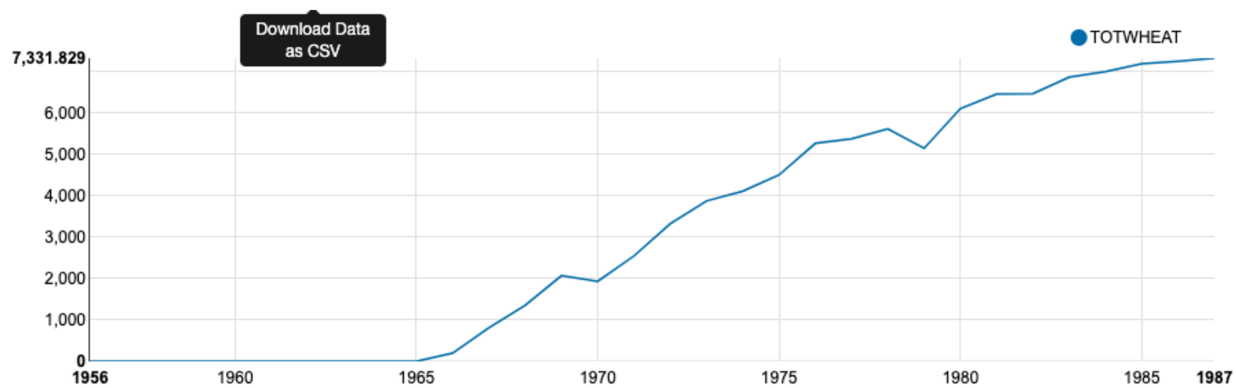
We find that Uttar Pradesh has the largest wheat production amongst all the other states. It has a very large lead over other states.



Analysis 4

Exploring the growth of wheat in Uttar Pradesh:

We find that there was very little production of wheat in the starting year of this dataset but after 1965 there was a growth spurt. The growth is almost linear except for the year 1979. We initally anticipated that rainfall might be a key factor in this. But soon after doing some research, we found that India had expreinced a slight recession. Two of the prime ministers had resigned in a quick succession.



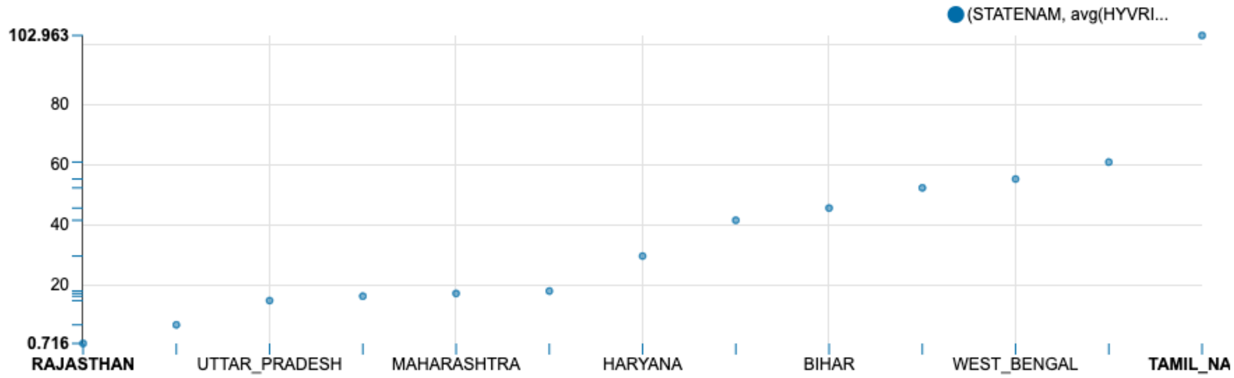
Analysis 5

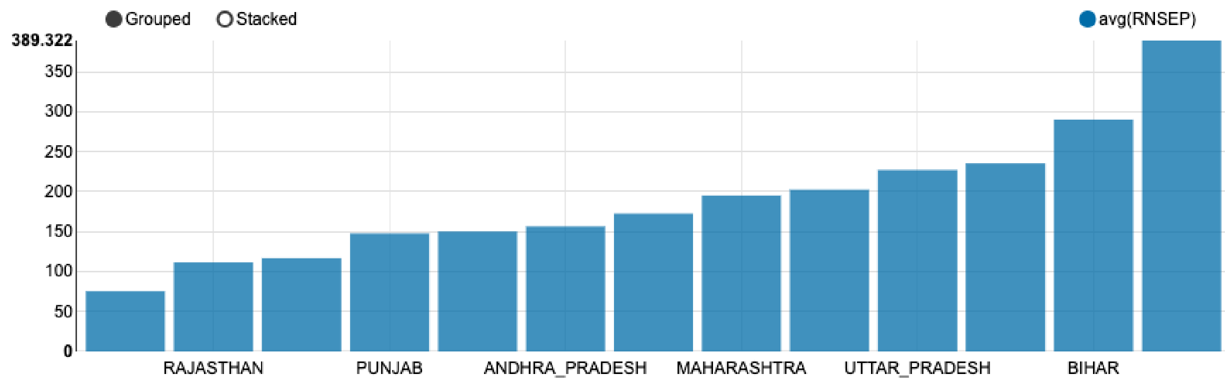
Mansoon season in India:

India experiences the most rainfall in the month of September. We have plotted the rainfall data for all the states. West Bengal experienes the most rainfall. It is located in Nort-West region of the country. As per common knowlegde, rice usually grows in regions where there is a lot of rainfall.

We were suprised to learn that Tamil Nadu is largest producer of rice. Although, it is located in the opposite part of the country where it does not rain as much as West Bengal.

While in the rainfall chart Tamil Nadu is at the bottom. This suggests that there are some descrepancies in the dataset.

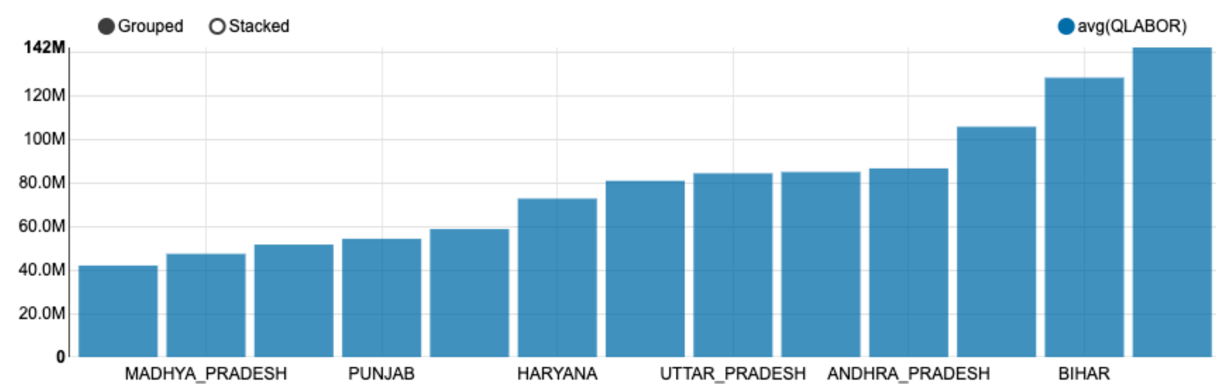


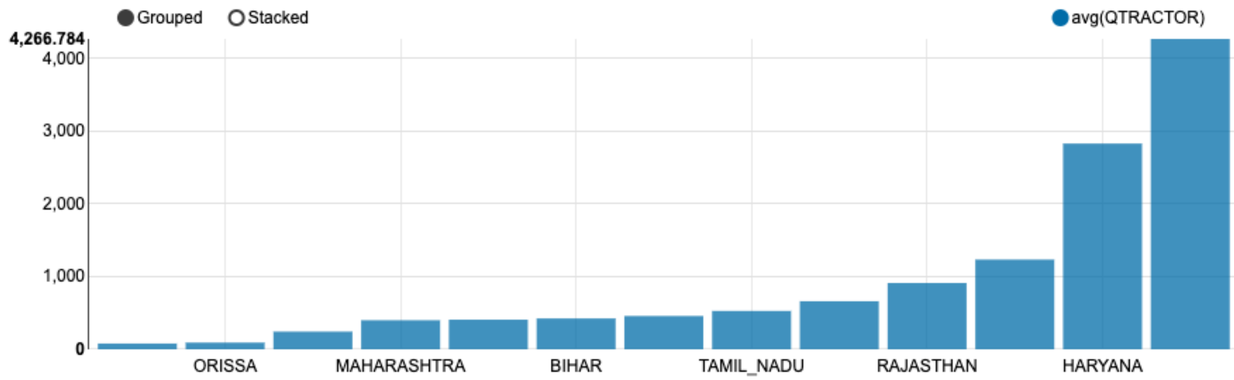


Analyis 6

Comparing mechanical labor and manual labor:

We find that Tamil Nadu has the most number of people working in fields as compared to Punjab where more people own tractors. West bengal is at the bottom beacuse there are a lot of paddy fields where the rice is cultivated so tractor does not improve efficency.





Type *Markdown* and LaTeX: α^2

Loading data into csv for crop production

In [20]:
```
rop_prod=pd.read_csv('agri_data_3/sum_prod.csv',delimiter=',')
rop_prod=crop_prod.fillna(0)
 print(crop_prod.dtypes)
rop_prod.sort_values(by=crop_prod.columns[0])
TATENAM = sorted(set(crop_prod.iloc[:,0].values))
rop_price=pd.read_csv("agri_data_3/price_avg.csv",delimiter=',')
or i in range(1,crop_price.shape[1]):
   crop_price.iloc[:,i]=pd.to_numeric(crop_price.iloc[:,i],errors='coerce')
   crop_price.iloc[:,i]=crop_price.iloc[:,i].fillna(0)
```

Area of cultivation

In [59]:
```
area_cult=pd.read_csv("agri_data_3/sum_area.csv",delimiter=',')
# print(sum(area_cult.isna()))
#area_cult.head()
```

Price of the crop

In [22]:
```
crop_price.head()
```

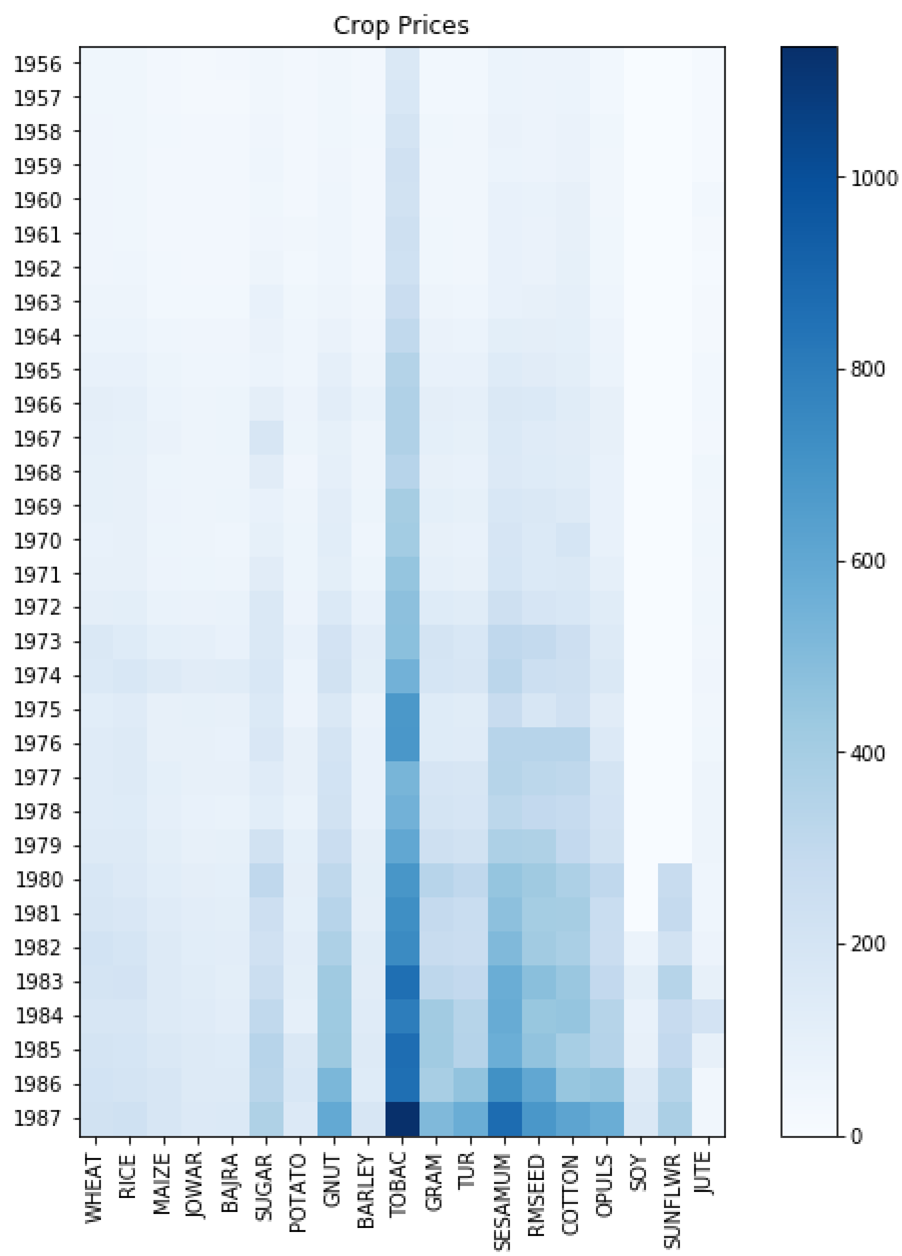
Out[22]:

	STATENAM	YEAR	PJOWAR	PMAIZE	PWHEAT	PSUGAR	PPOTATO	PGNUT	
0	BIHAR	1956	0.000000	29.885588	49.465294	41.099412	28.259765	36.620000	48.
1	HARYANA	1956	27.611667	30.828667	39.260000	29.198577	28.707825	40.249539	31.
2	WEST_BENGAL	1956	0.000000	27.967157	40.333238	41.578080	24.780750	0.000000	51.
3	MADHYA_PRADESH	1956	35.146512	24.500000	43.913953	36.691395	28.560710	34.370930	42.
4	KARNATAKA	1956	31.831579	27.757895	51.610526	32.278947	38.688244	40.758421	67.

5 rows × 22 columns

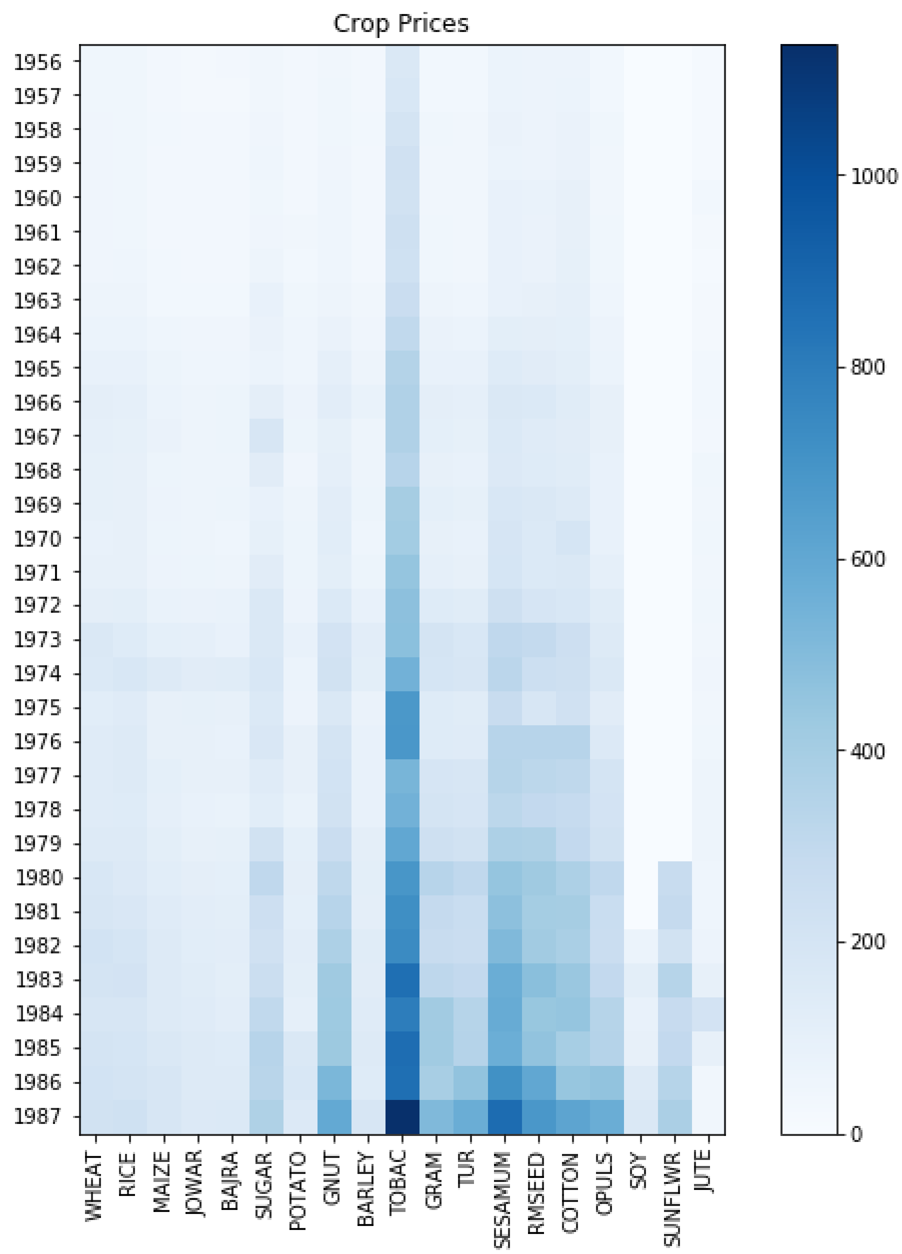
Visualization of the Crop Price data

In [29]:
```
 np.matrix(crop_price_year.values[:,1:],dtype='float')
t(mat2.shape)
p.delete(mat1,12,1)
t(mat2.shape)
plt.figure(num=1,figsize=(10,10))
ig.add_subplot(1,1,1)
et_aspect('equal')
_xticks(np.arange(len(np.delete(crop_price_year.columns[1:],12))))
_yticks(np.arange(len(crop_price_year.iloc[:,0])))
_xticklabels(np.delete(crop_price_year.columns[1:],12),rotation='vertical')
_yticklabels(area_cult_year.iloc[:,0])
show(mat1, cmap='Blues')
lorbar()
tle('Crop Prices')
ow()
```



Visualization of the Crop Price data

In [31]:
```
mat1 = np.matrix(crop_price_year.values[:,1:],dtype='float')
# print(mat2.shape)
mat1=np.delete(mat1,12,1)
# print(mat2.shape)
fig = plt.figure(num=1,figsize=(10,10))
ax = fig.add_subplot(1,1,1)
# ax.set_aspect('equal')
ax.set_xticks(np.arange(len(np.delete(crop_price_year.columns[1:],12))))
ax.set_yticks(np.arange(len(crop_price_year.iloc[:,0])))
ax.set_xticklabels(np.delete(crop_price_year.columns[1:],12),rotation='vert
ax.set_yticklabels(area_cult_year.iloc[:,0])
plt.imshow(mat1, cmap='Blues')
plt.colorbar()
plt.title('Crop Prices')
plt.show()
```



Area under Cultivation Yearly

In [32]:
```
area_cult_year=pd.read_csv('agri_data_3/area_year.csv',delimiter=',')
area_cult_year=area_cult_year.fillna(0)
```

In [33]:
```
area_cult_year.head()
```

Out[33]:

	YEAR	WHEAT	RICE	MAIZE	JOWAR	BAJRA	SUGAR	POTATO	G
0	1956	12951.36897	28783.21868	3352.050	16496.839	11386.333	2021.141931	231.402049	5674
1	1957	11124.86045	28693.36952	3679.540	17247.415	11107.748	2021.018879	258.429187	6414
2	1958	11964.39013	29432.29205	3782.756	17895.009	11357.569	1907.863530	271.796749	6240
3	1959	12724.82640	30077.45306	3881.671	17645.206	10645.070	2078.814530	290.731068	6449
4	1960	12259.01078	30315.49543	3925.277	18411.868	11482.312	2353.335932	291.139436	6449

5 rows × 21 columns

Production of the crop

In [34]:
```
crop_prod_year=pd.read_csv('agri_data_3/prod_year.csv',delimiter=',')
crop_prod_year=crop_prod_year.fillna(0)
```

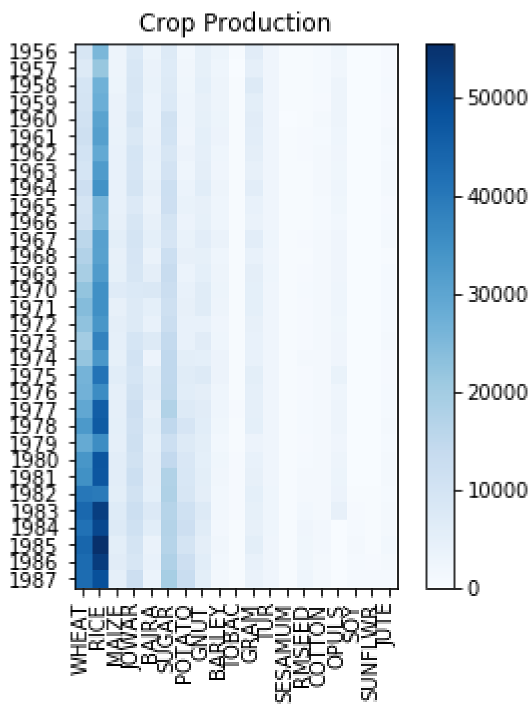
In [35]:
```
crop_prod_year.head()
```

Out[35]:

	YEAR	WHEAT	RICE	MAIZE	JOWAR	BAJRA	SUGAR	POTATO	(
0	1956	8994.794262	25509.26915	2710.378	7631.323	2886.312	7013.055172	1672.284000	456
1	1957	7343.911870	21604.97523	2802.579	8597.357	3616.600	6901.704800	1875.045115	467
2	1958	9413.368621	27008.66144	2992.087	8984.954	3838.561	7096.408200	2150.681824	514
3	1959	9789.582790	27687.38011	3602.979	8516.071	3465.496	7670.902600	2516.961578	452
4	1960	10528.610090	30394.82344	3594.961	9834.699	3254.185	10798.259200	2462.428590	472

5 rows × 21 columns

In [63]:
```
= np.matrix(crop_prod_year.values[:,1:],dtype='float')
nt(mat3.shape)
np.delete(mat3,12,1)
nt(mat3.shape)
 plt.figure(num=3,figsize=(5,5))
fig.add_subplot(1,1,1)
set_aspect('equal')
t_xticks(np.arange(len(np.delete(crop_prod_year.columns[1:],12))))
t_yticks(np.arange(len(crop_prod_year.iloc[:,0])))
t_xticklabels(np.delete(crop_prod_year.columns[1:],12),rotation='vertical')
t_yticklabels(crop_prod_year.iloc[:,0])
mshow(mat3, cmap='Blues')
olorbar()
itle('Crop Production')
how()
```



In [37]:
```
crop_yield_year=pd.read_csv('agri_data_3/yield_year.csv',delimiter=',')
crop_yield_year=crop_yield_year.fillna(0)
crop_yield_year.head()
```

Out[37]:

	YEAR	WHEAT	RICE	MAIZE	JOWAR	BAJRA	SUGAR	POTATO	
0	1956	142.139663	218.379154	154.830737	96.688463	74.567994	932.039639	1277.152929	21
1	1957	139.862257	194.720165	142.682246	113.721145	82.210835	946.250554	1246.345073	21
2	1958	165.875684	230.494105	167.739962	123.213744	88.798229	976.526417	1320.211813	24
3	1959	162.037833	229.635559	168.982956	117.295837	85.409247	979.687464	1325.044479	25
4	1960	177.034342	232.637265	176.061540	117.248091	86.372075	1089.761573	1342.382784	18

5 rows × 21 columns

Annual Rainfall

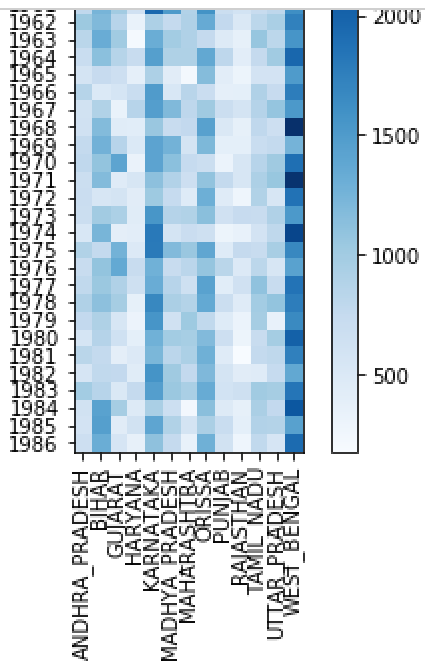
In [18]:
```
ainfall =  pd.read_csv('agri_data_3/rainfall_no_missing.csv',delimiter=',')
ainfall.head()
```

Out[18]:

	State	YEAR	RNJAN	RNFEB	RNMAR	RNAPR	RNMAY	RNJUN	RNJUL	RNAUG
0	PUNJAB	1956	21.60	6.80	45.30	5.10	0.70	53.5	236.50	28.70
1	BIHAR	1956	19.20	4.60	9.60	5.30	74.90	319.0	2.70	286.40
2	HARYANA	1956	16.80	3.00	24.80	0.80	2.20	5.7	23.30	161.60
3	MADHYA_PRADESH	1956	8.00	3.15	5.05	1.05	37.50	150.0	468.35	325.35
4	WEST_BENGAL	1956	9.65	17.65	24.15	95.25	92.25	545.6	362.65	391.10

In [19]:
```
rainfall1=rainfall.copy()
rainfall1 = rainfall1.groupby(['State','YEAR'],as_index=True).mean()
```

In [52]:
```
mat5 = np.matrix(rainfall1.iloc[:,12].unstack(level=-1).reset_index().value
fig = plt.figure(num=5,figsize=(5,5))
ax = fig.add_subplot(1,1,1)
ax.set_yticks(np.arange(len(rainfall1.iloc[:,12].unstack(level=-1).reset_in
ax.set_xticks(np.arange(len(rainfall1.iloc[:,12].unstack(level=-1).reset_in
ax.set_yticklabels(rainfall1.iloc[:,12].unstack(level=-1).reset_index().col
ax.set_xticklabels(rainfall1.iloc[:,12].unstack(level=-1).reset_index().val
plt.imshow(mat5, cmap='Blues')
plt.colorbar()
plt.title('Annual Rainfall in mm/sqmt')
plt.show()
```



In [53]:
```
price=crop_price_year.transpose()
price.to_csv(r'price.csv')
crop_prod2= crop_prod.melt(id_vars=["STATENAM","YEAR"],var_name="CROP",valu
crop_prod2.to_csv(r'new_prod.csv')
area_cult2= area_cult.melt(id_vars=["STATENAM","YEAR"],var_name="CROP",valu
area_cult2.head()
```

In [41]:
```
area_cult2.to_csv(r'new_area.csv')
```

In [42]:
```
crop_yield=pd.read_csv('agri_data_3/sum_yield.csv',delimiter=',')
crop_yield=crop_yield.fillna(0)
# print(crop_prod.dtypes)
```

In [56]:
```
a_cult.melt(id_vars=["STATENAM","YEAR"],var_name="CROP",value_name="YIELD")
sv(r'new_yield.csv')
sv('data_set.csv')
```

Correlation Between All The Factors For Crop Production

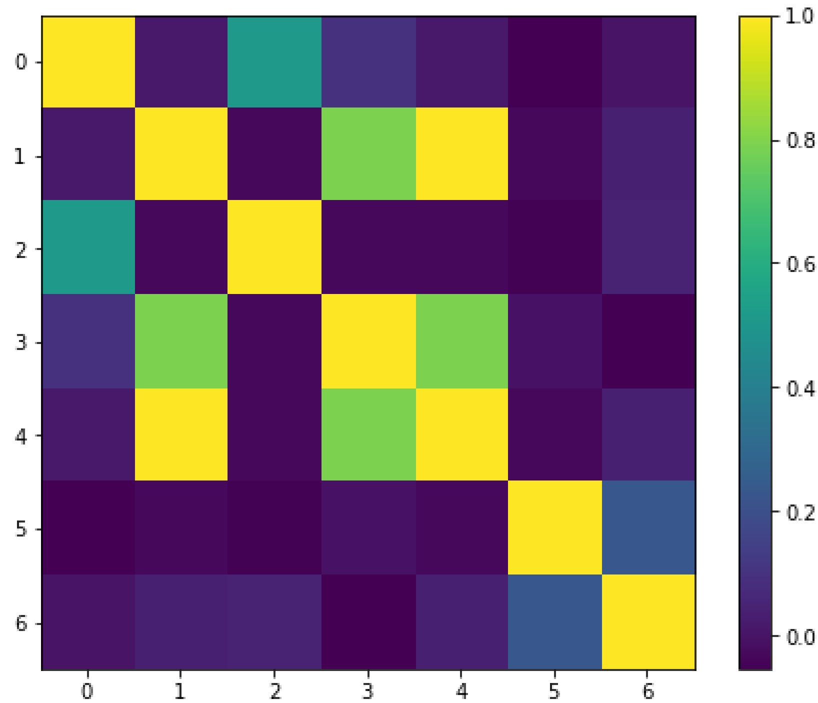
This heatmap shows the coorealtion between various factors that affect the crop production.

In [57]:
```
vals = data.corr()
vals
```

Out[57]:

	YEAR	AREA	PRICE	PROD	YIELD	ANN_RAIN	AVG_TEMP
YEAR	1.000000e+00	0.017297	0.510137	0.096909	0.017297	-0.049956	1.191578e-19
AREA	1.729740e-02	1.000000	-0.032887	0.790434	1.000000	-0.030226	3.708375e-02
PRICE	5.101370e-01	-0.032887	1.000000	-0.033362	-0.032887	-0.049312	4.810605e-02
PROD	9.690864e-02	0.790434	-0.033362	1.000000	0.790434	-0.003255	-5.404658e-02
YIELD	1.729740e-02	1.000000	-0.032887	0.790434	1.000000	-0.030226	3.708375e-02
ANN_RAIN	-4.995619e-02	-0.030226	-0.049312	-0.003255	-0.030226	1.000000	2.317707e-01
AVG_TEMP	1.191578e-19	0.037084	0.048106	-0.054047	0.037084	0.231771	1.000000e+00

In [58]:
```
fig=mplot.figure(figsize=(8,6))
mplot.imshow(vals.values)
mplot.colorbar()
mplot.show()
mplot.savefig('corr.png')
```




```
<Figure size 432x288 with 0 Axes>
```

Rainfall prediction

We have rainfall data on all the states for 12 months. This data is from 1956 to 1986. We are trying to predict the rainfall just by previous year data. Linear regression is the most basic form of regression which takes a single value and predicts next values. Since, our data is non-linear the regression model didn't work for us. The score for prediction was 0.0017.

We decided to do time-series analysis on our model using ARIMA model (Auto-regressive integrated moving average). Advatanges of using this model are that it takes into account the growth or decline in the data and the rate of change of data. It generates a sign wave form. $y(t)=A*\sin(2*\pi*f*t+\phi)$

As you can see from the superimposed plot that the predictions are closly related to the actual value of the plot. We get a standard error of 131.248.

In [126]:

```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(new_a2,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, res
        except:
            continue
```

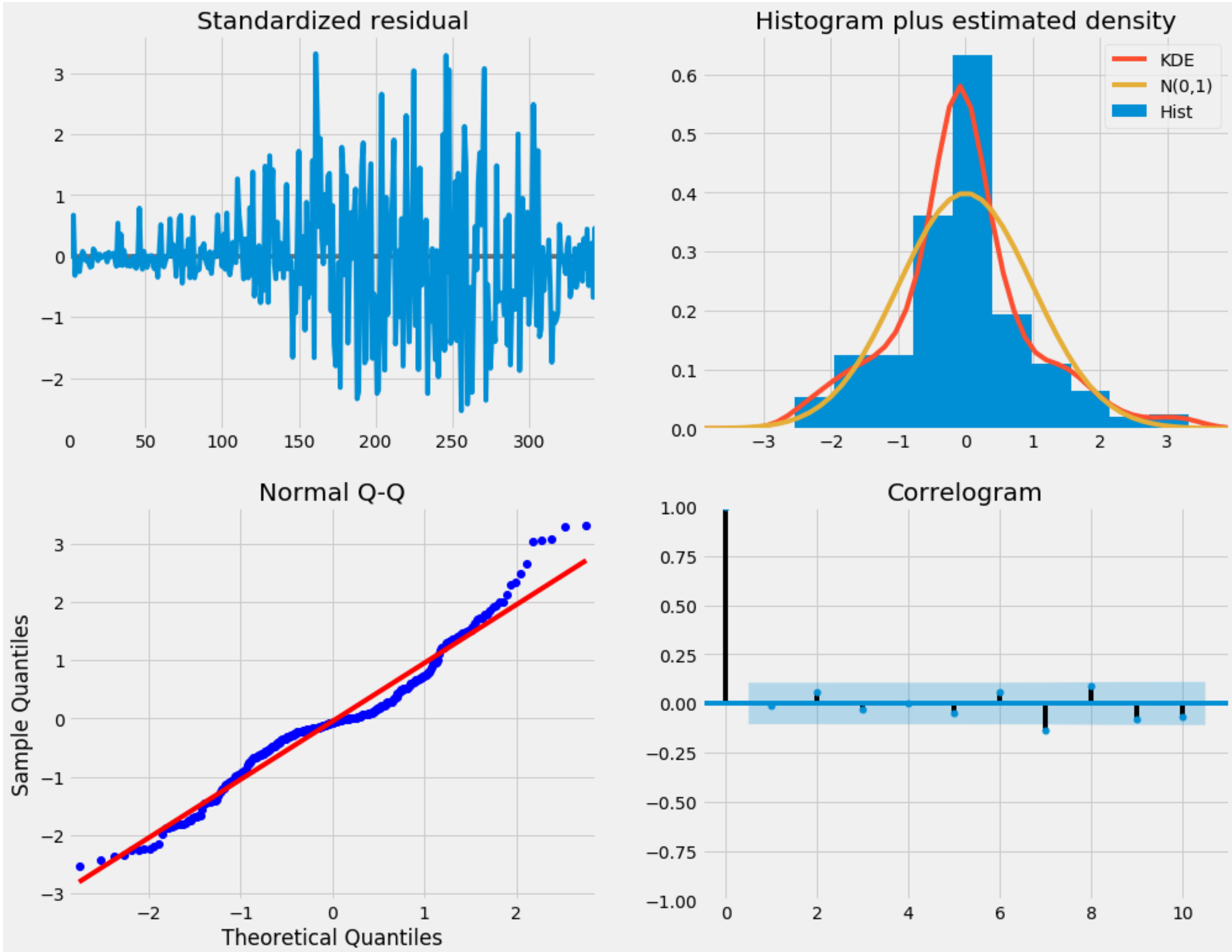
In [127]:
```
import statsmodels.api as sm
mod = sm.tsa.statespace.SARIMAX(new_a2,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
=============================================================================
=====
                 coef    std err          z      P>|z|      [0.025      
0.975]
-----------------------------------------------------------------------------
-----
ar.L1          0.1400      0.049      2.884      0.004       0.045      
0.235
ma.L1         -0.8868      0.027    -33.356      0.000      -0.939      -
0.835
ar.S.L12       0.0016      0.047      0.034      0.973      -0.091      
0.094
ma.S.L12      -0.9349      0.040    -23.143      0.000      -1.014      -
0.856
sigma2      1918.5184    131.248     14.617      0.000    1661.276    217
5.761
=============================================================================
=====
```

In [128]:
```
results.plot_diagnostics(figsize=(15, 12))
plt.show()
```



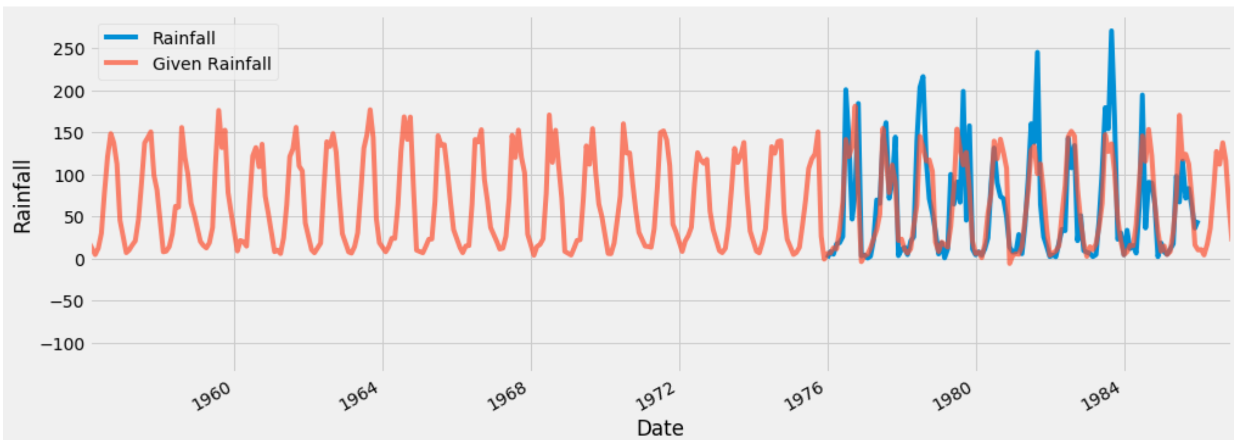
In [129]:
```
pred = results.get_prediction(start=pd.to_datetime('1976-01-01'), dynamic=F
pred_ci = pred.conf_int()
```

In [130]:
```
 = new_a2['1976-01-01':'1986-01-01'].plot(label='observed', figsize=(15,6))
ed.predicted_mean.plot(ax=ax, label='Given Rainfall', alpha=.7)

.fill_between(pred_ci.index,
              pred_ci.iloc[:, 0],
              pred_ci.iloc[:, 1], color='k', alpha = 0)

.set_xlabel('Date')
.set_ylabel('Rainfall')
t.legend()

t.show()
```



We created a dataframe called `wheat_df` which has the columns `YEAR` and `WHEAT` which signify the year and the wheat grown in that year respectively.

We tried to use various methods to predict the growth of wheat. The first method we tried was using Linear Regression. We used the first 20 data points ranging from year `1956` to `1977` as our training data and we used the remaining data to check whether the predictions are correct or not.

We splitted the `wheat_df` dataframe into `X_train`, `X_test`, `y_train` and `y_test`.

In [0]:
```
X = np.array(wheat_df["YEAR"])
y = np.array(wheat_df["WHEAT"])
X_train = X[:-10]
X_test = X[-10:]
y_train = y[:-10]
y_test = y[-10:]
```

In [217]:
```
X_train
```

Out[217]:
```
array([1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966,
       1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977])
```

In [105]:
```
X_test
```

Out[105]:
```
array([1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987])
```

We reshaped the data to arrange all the data points into a single row. We used reshape(-1, 1) to do so.

In [0]:
```
X = X.reshape(-1, 1)
X_train = X_train.reshape(-1, 1)
X_test = X_test.reshape(-1, 1)
```

We created a new Linear Regression using the `LR()` function found in `sklearn.linear_model` and stored it to `reg`. We then used the training datasets `X_train` and `y_train` to train our Linear Regression model. We then tried to predict the wheat grown in the year `1987` using the Linear Regression model and compared it with the real wheat growth value found in our `y_test` test data.

In [223]:
```
reg = LR()
reg.fit(X_train, y_train)
reg_1987 = reg.predict(np.array([1987]).reshape(1,-1))[0]
real_1987 = y_test[1987 - X_test[0]][0]
print(reg_1987)
print(real_1987)
```

```
198.81202099239817
215.17455619999998
```

The result was pretty satisfactory considering the training data. We can see the percentage difference in the cell below.

In [224]:
```
abs(real_1987 - reg_1987)/100
```

Out[224]:
```
0.16362535207601808
```

However, we realized that using a Linear Regression Model is not the best way to go. This is because Linear Regression does not take patterns into account and will give unidirectional results only. For example, if the model calculates that the training data's values are always going up on average, it will predict in a similar way, never dipping now.

Because of this, we tried using the decision trees as they might give more accurate results.

We use the `DecisionTreeRegressor` from `sklearn.tree`. We use the `fit()` function to train the Decision Tree model using the `X_train` and `y_train` data. Like before, we try to predict the wheat grown for the year `1987` but use the `dtree` instead of `reg`. We then print the predicted value as well as the real value.

In [230]:
```
dtree = scart()
dtree.fit(X_train, y_train)
dtree_1987 = dtree.predict(np.array([1987]).reshape(1,-1))[0]
print(dtree_1987)
print(real_1987)
```

```
135.39034750000002
215.17455619999998
```

We can see that the outputs are not as expected. We figured out that the samples, the training and the test data were major factors in giving accurate predictions using Decision trees. To get a fair analysis. We did this test numerous times with different samples and training:test ratios.

In [231]:
```
abs(real_1987 - dtree_1987)/100
```

Out[231]:
```
0.7978420869999997
```

The scores are shown below, the `dtree` score is significantly lesser than the `reg` score.

In [232]:
```
reg.score(X_test,y_test)
```

Out[232]:
```
0.4532576976736124
```

In [233]:
```
dtree.score(X_test, y_test)
```

Out[233]:
```
-3.9126333861424527
```

Our first prototype for improving the outputs is given below. Here we use the `train_test_split()` function found in `sklearn.model_selection` to get random samples to test and train our models.

In [0]:
```
#import sklearn.model_selection.train_test_split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size = 0.25, random_state = 100)
```

The output received below is much better. We then ran the procedure several times to be sure.

In [245]:
```
reg = LR()
reg.fit(X_train,y_train)
print(reg.predict(np.array([1987]).reshape(1,-1))[0])
#print(y_test[1987 - X_test[0]][0])
```

```
205.17241368856412
```

In [246]:
```
dtree = scart()
dtree.fit(X_train, y_train)
print(dtree.predict(np.array([1987]).reshape(1,-1))[0])
#print(y_test[1987 - X_test[0]][0])
```

```
215.17455619999998
```

In [247]:
```
wheat_df[wheat_df["YEAR"] == 1987]
```

Out[247]:

	YEAR	WHEAT
31	1987	215.174556

In [238]:
```
reg.score(X_test,y_test)
```

Out[238]:
```
0.87551060858414
```

In [239]:
```
dtree.score(X_test, y_test)
```

Out[239]:
```
0.8385247870034318
```

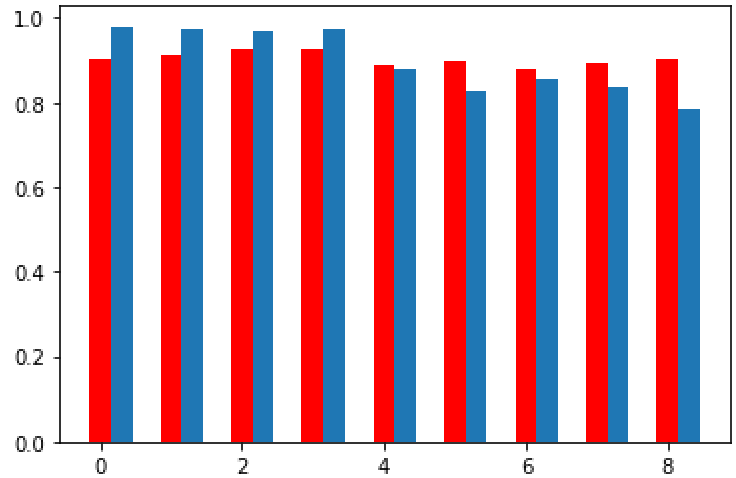
Given below is our code using 9 different variants of our samples `X` and `y`. We change the `test_size` (ratio between training and test data) in each iteration. We've used the `random_state` attribute for `train_test_split()` too. We append the scores to lists `reg_scores` and `dtree_scores` and plot them using `matplotlib.plt`.

In [0]:
```
from sklearn.model_selection import train_test_split

reg_scores = []
dtree_scores = []
for i in range(1, 10):
  X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size = i  * 0.09, random_state = 1000)
  reg = LR()
  reg.fit(X_train,y_train)
  dtree = scart()
  dtree.fit(X_train, y_train)
  reg_scores.append(reg.score(X_test, y_test))
  dtree_scores.append(dtree.score(X_test, y_test))
```

We can see that the scores from Linear Regression and Decision Tree are close to each other. We conclude that Decision Trees would've worked much better if we have more training data.

In [249]:
```
import matplotlib.pyplot as plt
plt.bar(np.arange(len(reg_scores)), reg_scores, width=0.3, color='red')
plt.bar(np.arange(len(dtree_scores))+ width, dtree_scores, width=0.3)
plt.show()
```



Conclusion:

In conclusion, we think that doing this project was a great experience. This is our first project related to data science and it offered various challenges which helped us get a grasp of the subject. Having said that, our project was far from perfect as we could have done many things better.

In the beginning of our project, we had issues while sanitizing our data to remove or parse information from the source. Eventually, we found a software called Stata which made the process of retreiving the data from our source much easier. If we had done that in a timely manner, we could've explored more into improving the predictioning mechanisms of the project. Both of us do not have a python background, which was frustrating as we knew what we had to do for the most part but were fighting the syntax of the language and the libraries that come with it. We were not well versed with numpy and pandas and spend a significant portion of our time in fixing issues which were thought to be trivial after finding the solution. We were not experienced with various models and technologies like RNN offered by the vast libraries like scikit and spent time in implementing things which were already done before and done much better. We did not explore enough into the world of machine learning algorithms until much later, making it difficult to implement said algorithms by scraping all our previous work.

While trying to understand the numbers obtained by our analysis, we had to take into account that the stats a result of numerous factors and reasons which where impossible for us to take into account. We needed to realize that Causation is not Correlation and that one thing may not necessarily be the reason for the occurrence of another thing. For example, the rainfall in June of 1986 in Bihar may not be have affected the growth of wheat in Assam in 1987, or it may have. It is very difficult for us decide which factors to take into account and which to ignore, especially as we are not experts in the field. All of these factors are why we can only give a 'prediction' of what 'could' happen. It was a blast doing this project and I have nothing but good feedback about the class. Thank you.

Acknowledgments:

Libraries:

Pyspark

Scikit-learn

Pandas

Numpy

STATA

These articles and Github repositories were invaluable resourse as they had good examples of all the above used libraries

https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ (https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/)

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html (https://spark.apache.org/docs/latest/api/python/pyspark.sql.html)

https://ademos.people.uic.edu/Chapter23.html (https://ademos.people.uic.edu/Chapter23.html)

https://machinelearningmastery.com/tune-arima-parameters-python/ (https://machinelearningmastery.com/tune-arima-parameters-python/)

https://machinelearningmastery.com/tune-arima-parameters-python/ (https://machinelearningmastery.com/tune-arima-parameters-python/)

https://github.com/nitinvbharti/Agriculture_Analysis/blob/master/Agro_analysis.ipynb (https://github.com/nitinvbharti/Agriculture_Analysis/blob/master/Agro_analysis.ipynb)

https://medium.com/hub-konam-foundation/predicting-crop-yield-and-profit-with-machine-learning-1d9c3216faf7 (https://medium.com/hub-konam-foundation/predicting-crop-yield-and-profit-with-machine-learning-1d9c3216faf7)

All the files and notebooks can be found on our Github repository:

https://github.com/omkarsgit/Agriculture-project (https://github.com/omkarsgit/Agriculture-project)

In [ ]: