SIX MONTHS INDUSTRIALTRAINING REPORT

Rainfall Prediction

Submitted in partial fulfillment of the

Requirements for the award of

Degree of Bachelor of Technology in Computer

Submitted By:

Name of Team Members

TCIL-IT CHANDIGARH (ICS GROUP)

SUBMITTED TO:

Your College Name, place

Your Degree Tenure/ Batch

January 2021

ACKNOWLEDGEMENT

I would like to avail this opportunity to thank all of the people who have stood by me, encouraged me, inspired me and have contribute greatly in providing the joy of achievement and thrill of creative effort experienced all the way through the accomplishment of this project.

I would like to place on record my deep sense of gratitude to Er. TCIL-IT Chandigarh (ICS Group), his generous guidance, help and useful suggestions

I express my sincere gratitude to Er. TCIL-IT Chandigarh (ICS Group), Head of department (HOD), Computer Science for his stimulating guidance, and continuous encouragement.

I am extremely thankful to Prof TCIL-IT Chandigarh (ICS Group), College_ name*** for valuable suggestions and encouragement.

I am also thankful to Mr. ******, Training and placement officer for providing the opportunity to get the knowledge.

I also wish to extend my thanks to Mr. Manoj, Training Incharge for guiding and providing the knowledge related to my work.

I also wish to extend my thanks to Mr. **, Training Incharge for guiding and providing the knowledge related to my work.

TCIL-IT Chandigarh (ICS GROUP)
Signature of Student

TCIL-IT Chandigarh (ICS GROUP)

CERTIFICATE

I hereby certify that I have completed the Six months Training in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering.

I did my training in TCIL-IT, Chandigarh from 14-08-2020 to 30-02 -2020.

The matter presented in this report has not been submitted by me for the award of any other degree elsewhere.

TCIL-IT Chandigarh (ICS GROUP)
Signature of Student

TCIL-IT Chandigarh (ICS GROUP)

TABLE OF CONTENTS

TOOLS USED

Python
Numpy and Pandas

Scikit Learn
Matplotlib
Seaborn
Keras,

Other Tools and Environments like Anaconda, Jupyter Notebooks.
System Study

PROJECT WORK

Ipython Notebook

ReadMe File

DataSets

TABLE OF FIGURES

SkyMetWeather	https://www.skymetweather.com/
Output Screen Shots	Kept in 'Plots' Folder of the project
Ipython Notebook	Kept in the 'Src' Folder of the Project
DataSets	India Rainfall Excel, District wise Rainfall Excel File

INTRODUCTION

Rainfall prediction is used to predict the rainfall on the basis of given dataset as State , District , Month , Annual Rainfall Level and the Grouped Months Precipitation.

We develop this project based on linear regression algorithm using normal equation.

Linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

We have a data set for predict the coming precipitation. We predict the price with normal equation, gradient descent, classification, and logistic regression etc.

But we are use normal equation with linear regression.
We use multiple linear regression algorithm for prediction and when we run the model on jupyter notebook then we display the results on Ipython Notebook Cells.

PROJECT PROPOSAL

To Predict the value of the precipitation according to features.

To understand how each feature increase/ decrease the value of Rainfall

ABOUT ALGORITHM

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task.

Regression models a target prediction value based on independent variables.

It is mostly used for finding out the relationship between variables and forecasting.

Different regression models differ based on – the kind of relationship between

dependent and independent variables, they are considering and the

number of independent variables being used.

In statistics, linear regression is a linear approach to modeling the relationship

between a scalar response (or dependent variable) and one or more

explanatory variables (or independent variables).

The case of one explanatory variable is called simple linear regression . For more

than one explanatory variable, the process is called multiple linear regression.

This term is distinct from multivariate linear regression, where multiple

correlated dependent variables are predicted, rather than a single scalar

variable

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used.

Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

--> If the goal is prediction, forecasting, or error reduction,linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.

--> If the goal is to explain variation in the response variable that can be attributed to variation in the explanatory variables, linear regression analysis can be applied to quantify the strength of the relationship between the response and the explanatory variables, and in particular to determine whether some explanatory variables may have no linear relationship with the response at all, or to identify which subsets of explanatory variables may contain redundant information about the response.

ABOUT DATA:

The first problem was where can I get the data to build a large enough dataset since I want to be able to predict the price for a given apartment according to the real estate agency chosen.

Firstly ,I tried to find a dataset already done on the web but unfortunately

nothing was available or corresponded to what I was looking for.

To address this problem, I decided to use the “web scraping“ which is a technique of extracting information from websites.

The train and test data will consist of various features. Each row contains fixed size object of features. There are eight features and each feature can be accessed by its name. And we take data from API.

We use kaggle library for API. And the data is of States and

in this there are many features and we have a target is to predict the fore coming rainfall levels.

Level of Precipitation is depends on the many features. And one feature will increase the possibility of rain and gives ascend to the graph.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares cost function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). The linear regression models the relationship between a dependent variable that we are trying to predict with a vector of independent variables. The vector of variables is also called regression algorithm.

Linear regression assumes that there is a linear relationship between the vector of independent variables and the dependent variable that we are trying to predict. Hence, linear regression models learn the unknown variable and constants of a linear function using the training data, such that the linear function best fit the training data. Linear regression can be applied in cases where the goals are to predict or forecast the dependent variable based on the regressor variables. We will use an example to explain how linear regression trains based on data.

As we know, a straight line can be represented by the following equation:

$$Y = \beta 0 + \beta 1x$$

Hence, the approximately linear relationship in the preceding

diagram can also be represented using the same formula, and the

task of the linear regression model is to learn the value of β0 and β1.

ABOUT DATA SCALING

Data scaling is the technique that is used for scale the data.

When we use dataset for prediction then firstly we scale the data means we remove some values that are null or none.

When null values are occurs then we have a problem to predict or

to use the model that is for prediction.

In data science, outliers are the null values.

According to different measures and charts of data, there was no doubt about the presence of outliers.

Taking abnormal value such as very large or very small, even zero, in

one or more of variables.

These outliers can affect greatly the results of my learning algorithm.

Here is the code for removing outliers

```
from pandas.api.types import is_numeric_dtype
def remove_outlier(df):
      low = .05
      high = .95
      quant_df = df.quantile([low, high])
      for area_type in list(df.columns):
        if is_numeric_dtype(df[area_type]):
         df = df[(df[area_type] > quant_df.loc[low, area_type])
               & (df[area_type] < quant_df.loc[high, area_type])]
      return df
remove_outlier(df)
```

TECHNCAL REVIEW:

INDIA METEOROLOGICAL DEPARTMENT
Ministry of Earth Sciences
Government of India

AccuWeather

SkymetWeather

INDIA METEOROLOGICAL DEPARTMENT

It is the National Meteorological Service of the country and the principal government agency in all matters relating to meteorology and allied subjects.

To take meteorological observations and to provide current and forecast meteorological information .

To warn against severe weather phenomena like tropical cyclones, norwesters,

duststorms, heavy rains and snow, cold and heat waves, etc.,

which cause destruction of life and property.

To provide meteorological statistics required for agriculture , water resource management , industries , oil exploration and other nation-building activities.

FEATURES

1.Agricultural Meteorology

2. Civil Aviation

3. Climatology

4. Hydrometeorology

5. Seismology

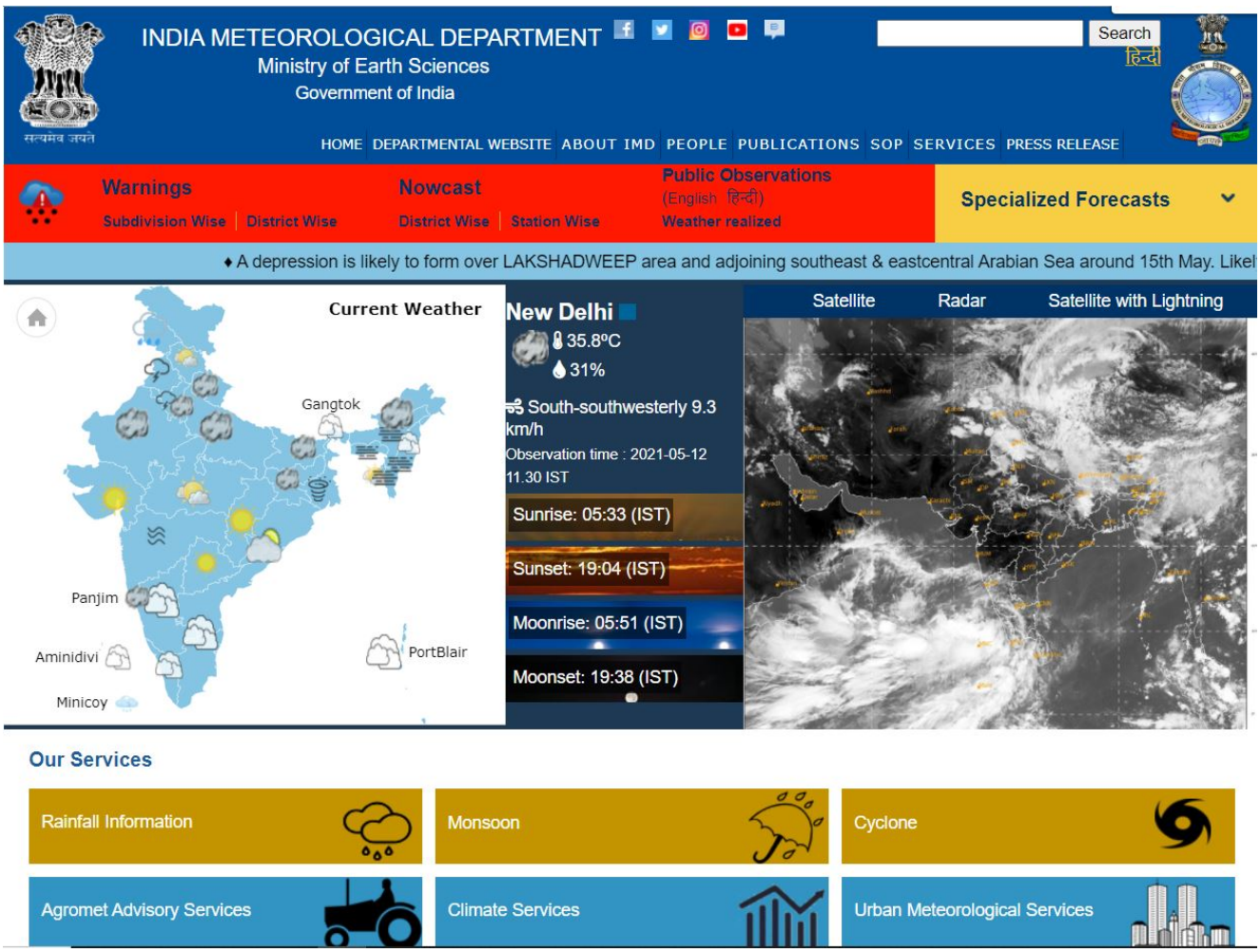
6. Training

7. Strumentation

8. Meteorological Telecommunication Regional

9. Specialised Meteorological Centre Positional Astronomy

10. Satellite Meteorology



ACCUWEATHER

In study after study AccuWeather has been proven as the most statistically accurate source of weather forecasts and warnings .

Our over 100 expert meteorologists plus designers, writers, developers all collaborate to bring the weather forecast to life for our users, partners and corporate clients.

AccuWeather's foundation of Superior Accuracy is further enhanced and expanded with the best communication, wording, detail and displays ensuring that important weather forecasts, warnings, news and information are more useful than other sources and easily understood.

AccuWeather 's forecasts and warnings focus on the impact to people and businesses , so they can make the best weather -impacted decisions . A storm producing 2-3 inches of snow may prove insignificant in one geography and bring another to a complete standstill.

FEATURES

1. Increase safety, reduce liability and losses, minimize risk and drive efficiency.

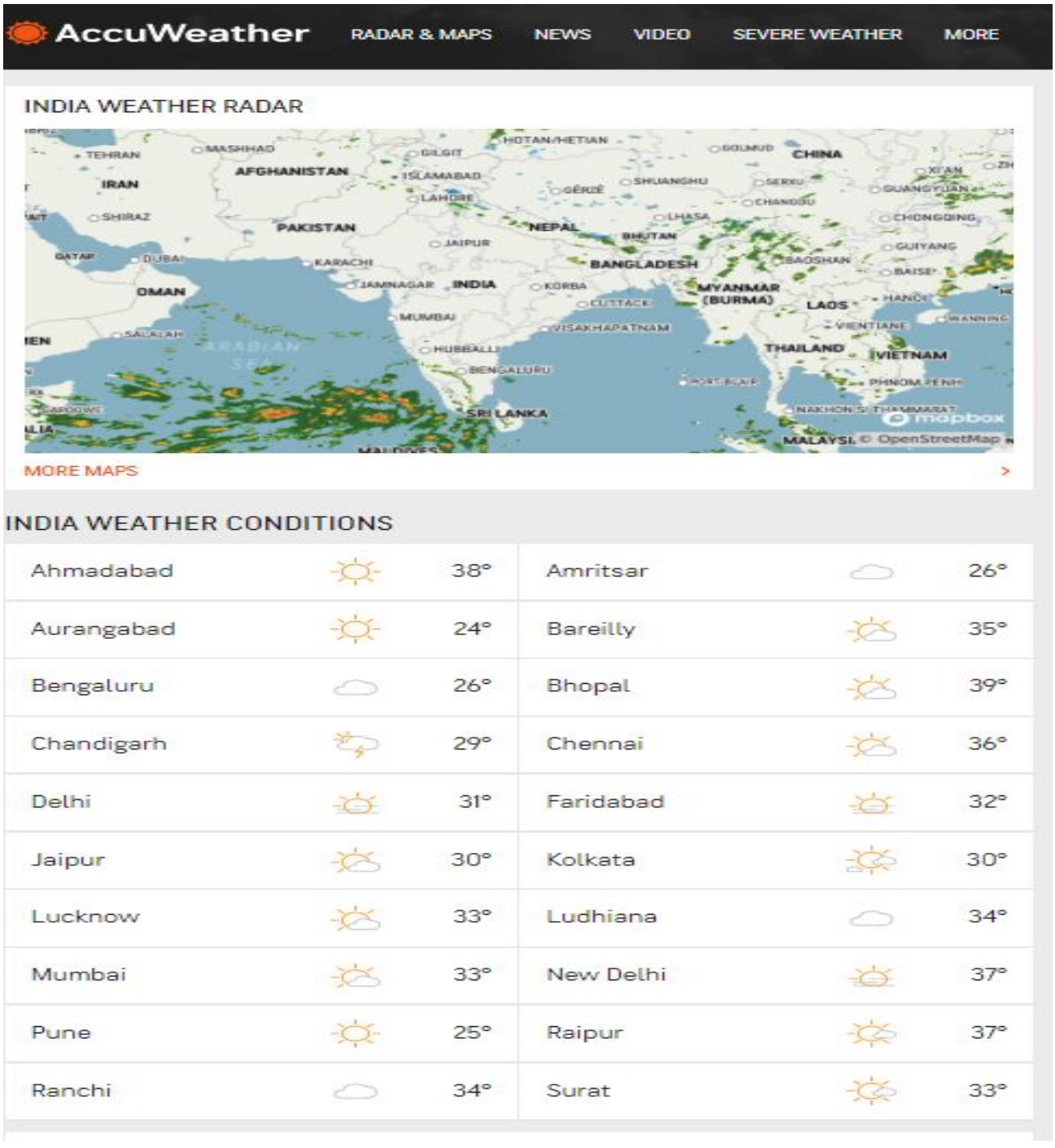
2. Building lasting relationships powered by innovation and the most accurate weather forecasts, warnings, data and insights.

3. Largest and best collection of real-time data .

4. Expert data analysis.

5. Forecasts with proven Superior Accuracy.

6. Partner , prepare and protect with greater confidence.



SKYMET

India's leading weather and agri-risk monitoring company Skymet aims at helping small marginal farmers by mitigating the risks associated with agriculture through the use of IoT, SaaSS (software as a smart solution) and DaaS (Data as a Services).

Climate change is preponderant and is going to affect India and the world . Technology is the only means left for us to feed many more with much less . Skymet 's technology will help in the fight against the following challenges -

Strong Climate change, overpopulated country, dwindling monsoons, five drought years, Stress on soil, the threat to food security of India.

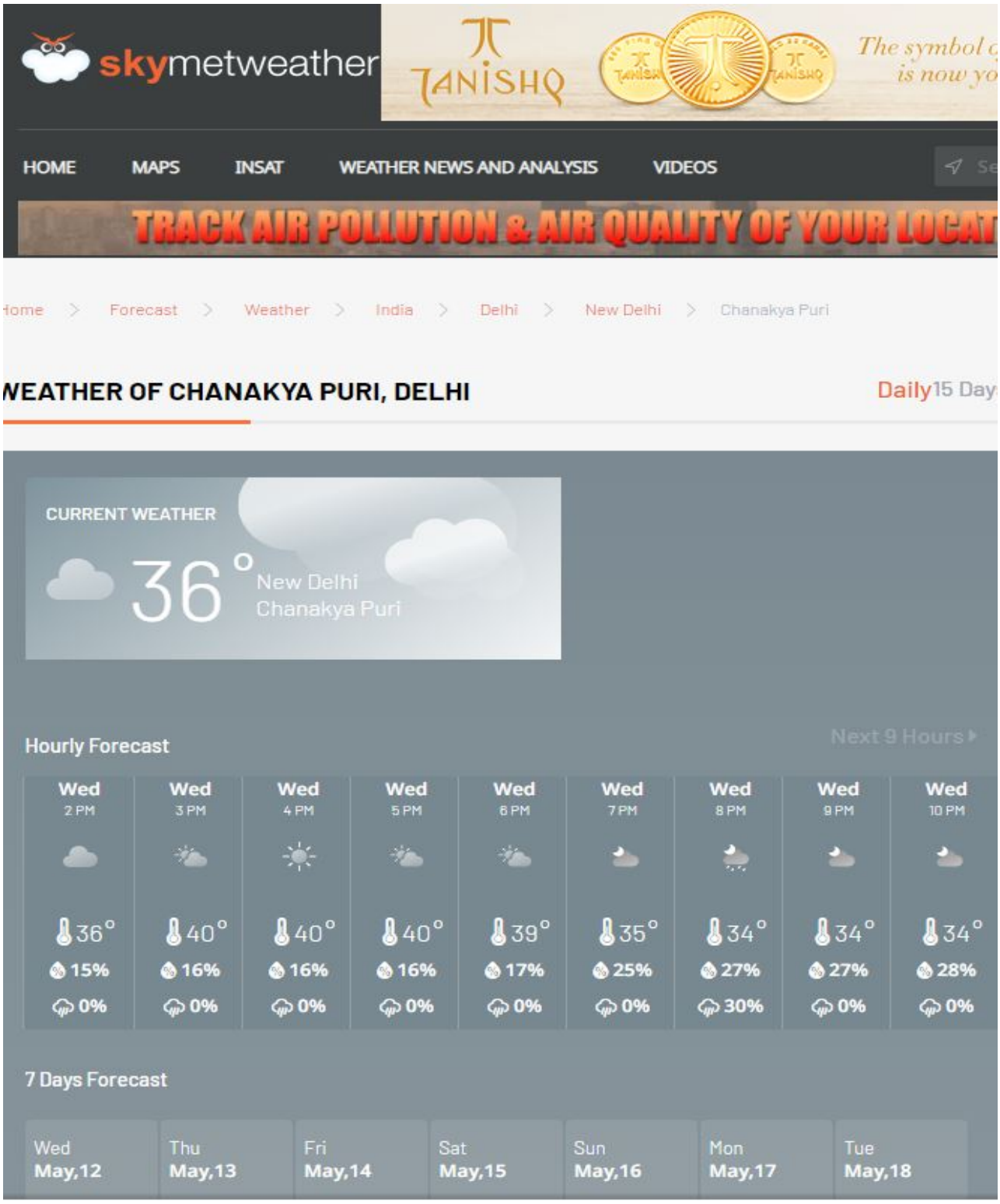
Skymet is the pioneer in IoT in the area of weather industry. Skymet from its very inception has been managing risk posed by climate change in agriculture . We have sensors for the weather , air-quality , crop , lightening , AWS, Drones, and Patented applications & Data.

Our key clients are Insurance companies, Banks, Agro companies, traders, e- commerce, retailers, logistics, and Govt. departments.

FEATURES

1. CROP INSURANCE
2. BANKING
3. ECOMM & RETAIL
4. TRANSPORTATION

Skymet is India's largest weather monitoring agri-risk solutions company. Experts in measuring, predicting. limiting climate risk to agriculture.



PROPOSED WORK

GENERAL

We develop a rainfall prediction web application for predicting the level and chances of precipitation based on previous data and other features . We develop a rainfall prediction using normal equation in linear regression algorithm.

There is lots of algorithm to develop a prediction algorithm but we use normal equation in linear regression algorithm because we think this is the basic algorithm for rainfall prediction.

We develop the rainfall prediction algorithm with numpy and pandas. And

when we develop with numpy and pandas then we also need to understand

the mathematicaL knowledge.

The main benefit of this normal equation method is, it can handle large training set

efficiently in terms of number of instances. The normal function reduces the

mean square error.

We are implementing with the help of numpy and pandas and both are third party

libraries which are extensively use in data analysis and scientific

computing in python programing language.

But because in these techniques we only need to give the input data set and then

we get the output of the particular algorithm In these tools we did not need

any mathematical background and run easily. But when someone give us

any other language then how we develop the algorithm

So we develop house price prediction algorithm using scratch . And

other algorithms are used for complex problems.

FEATURES OF PROPOSED PROJECT:

1. The Day and Time of Precipitation can be.

2. Predicted Weather Forecast can be done online.

3. The Intensity of rain and humidity can be measured.

4. Previous Rainfall data and graphs can be viewed.

5. The area wise prediction is possible.

6. Graphs and visualization for the previous days and months possible.

7. Variation of rainfall level and pattern in forms of bar graphs and line graphs.

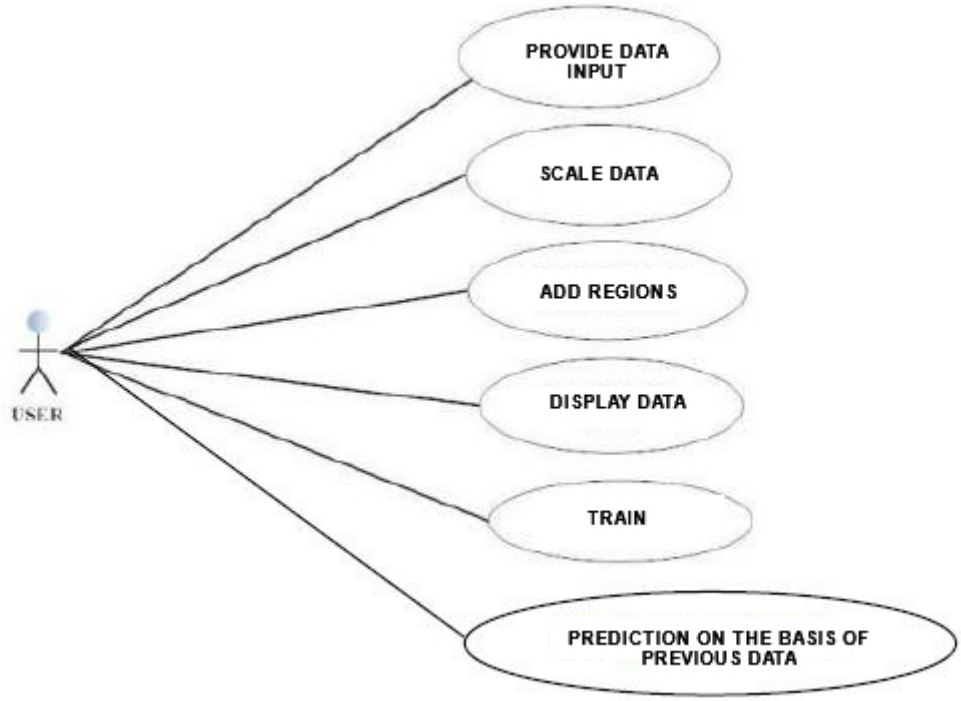
8. Datasets can be changed by replacing the original Data Sets.

9. Custom Visualization with different colors and different patterns.

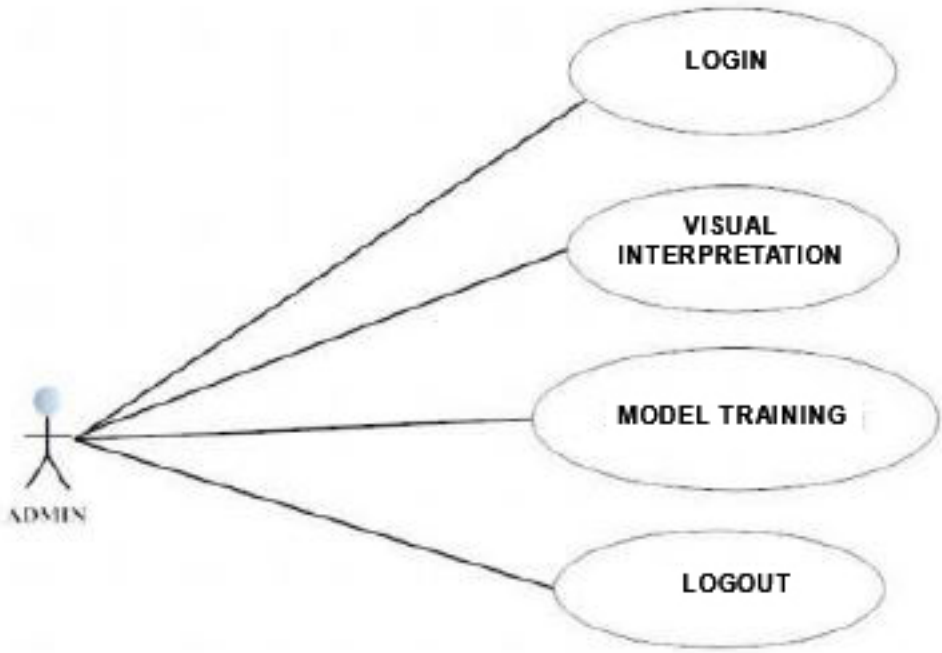
10. District wise and area wise prediction.

11. Implementing machine learning algorithms for the prediction of rainfall.

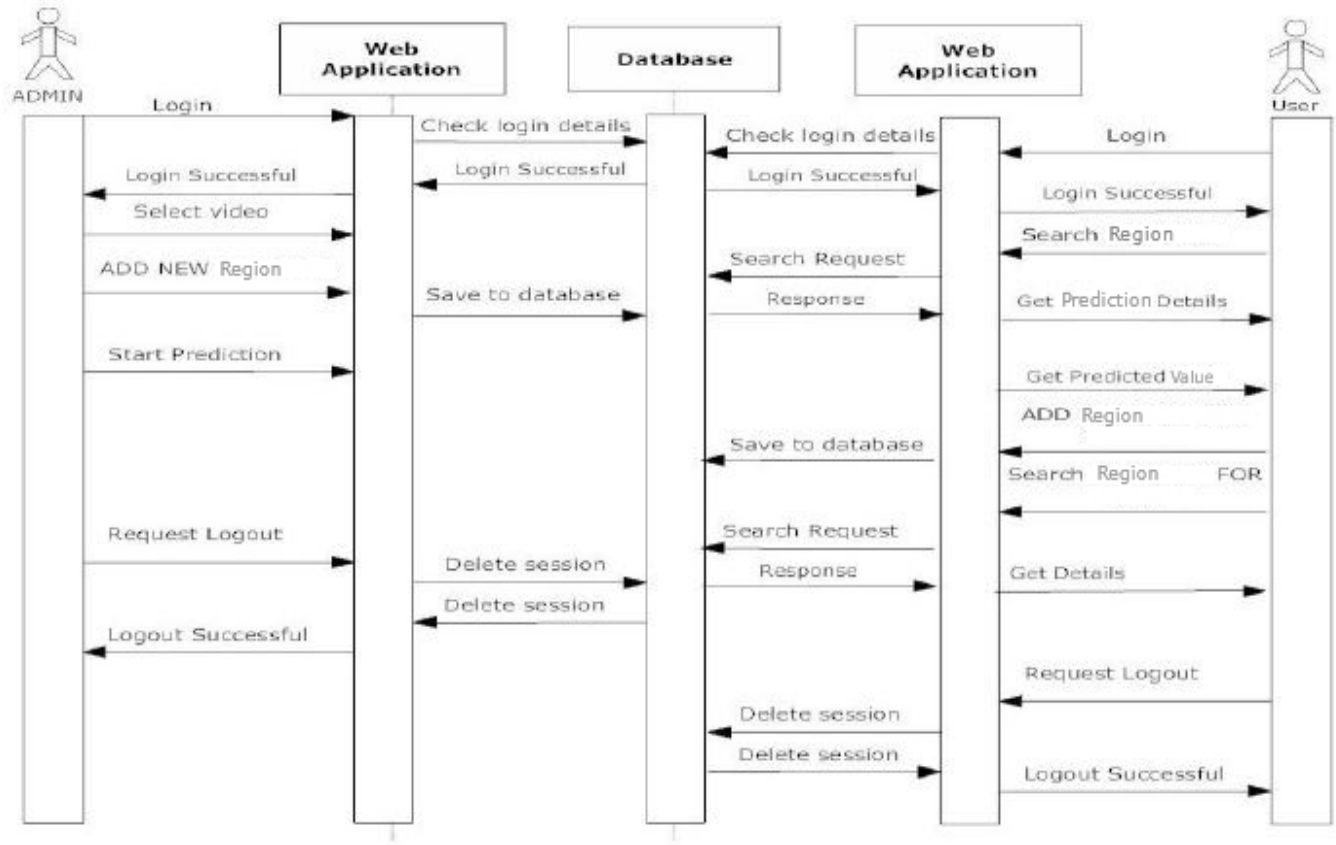
DIAGRAMS



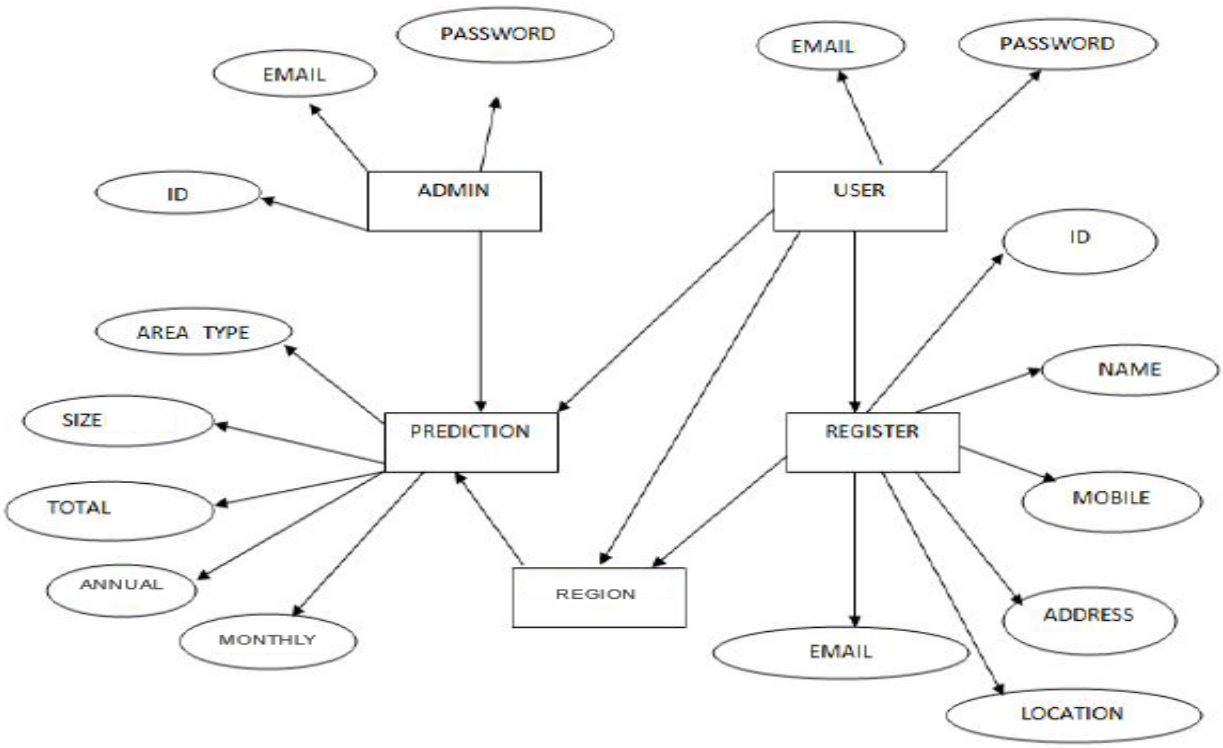
Use Case Diagram of User



Use Case Diagram of Admin



Sequence Diagram



ER DIAGRAM

TOOLS USED:

Python	(Programming Language)
Numpy and pandas	(Libraries for mathematical dealing)
Scikit Learn	(Library For Scientific Caluclations)
Matplotlib	(Library For Graph Plotting and Data Visualization)
Seaborn	(Advanced Graph Plotting and Data Visualization)
Keras	(Library for Machine Learning)

+other Machine Learning tools

PYTHON

Python is an interpreted, high level, general purpose programming language. Its language constructs and object–oriented approach aim to help programmers write ckear, logical code for small and large scale projects.

NUMPY AND PANDAS

Numpy is the core library for scientific computing in python . It provides a high-performance multidimensional array object and tools for working with these arrays . Numpy is a powerful N-dimensional array object which is linear algebra for python

Pandas

Pandas is an open-source library built on top of numpy providing high-performance , easy to use data structures and data analysis tools for the python programming language. it allows for fast analysis and data cleaning and preparation . It excels in performance and productivity

Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language

It features various classification , regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN , and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter , wxPython , Qt, or GTK. There is also a procedural " pylab " interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB , though its use is discouraged .[3] SciPy makes use of Matplotlib.

Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Keras

Keras Python library that provides a clean and convenient way to create a range of deep learning models on top of Theano or TensorFlow

Keras Machine Learning

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development

HARDWARE AND SOFTWAR E REQUIREMENTS :

OPERATING SYSTEM: We use Windows7 or newer

for RAINFALL prediction web application.
BROWSER: There is no single answer for this. We run it on any

browser. Use whichever browser works best in your computer.

However, I recommend the Chrome, Opera. PROCESSOR: 1.6

GHz CPU is minimal for web servers
RAM: 4GB RAM is recommended.

SYSTEM STUDY FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are
ECONOMICAL FEASIBILITY
TECHNICAL FEASIBILITY
SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization . The amount of fund that the company can pour into the research and development of the system is limited . The expenditures must be justified . Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available . Only the customized products had to be purchased.

TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement , as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

India Rainfall Analysis

Description

The value of monsoon forecasting for India cannot be overstated . understated Rainfall forecasting can be achieved in two ways: long-term forecasting (predicting rainfall for several weeks or months in advance) and short -term forecasting (predicting rainfall for a few weeks or months in ahead of time)

Forecasts : Forecast rainfall in specific areas a few days before the date.

The value of monsoon forecasting in India is unquestionable.
It is possible to make two kinds of rainfall forecasts.
Predictions over the long term Rainfall can be predicted a few weeks/months ahead of time. Predict rainfall in remote areas a few days ahead of time with short-term forecasts.

Converting data further into structured sequence for experimenting.
Conduct a comprehensive study of the information to look for variations in precipitation trends.
Ultimately , we differentiate the results into preparation and research in order to determine the annual rainfall.

We have used a combination of statistical and machine learning techniques (*SVM

*, for eg) to predict things and evaluate them . Using a variety of approaches, we were able to reduce the error.

The Indian meteorological department provides the project with the forecasting data it needs. We want to focus on long-term rainfall forecasting in this project. The project's key goal is to forecast rainfall amounts in a certain section or region ahead of time. Utilizing historical evidence , we can forecast the amount of rain that will happen.

```
Data columns (total 19 columns):
SUBDIVISION    4116 non-null object
YEAR           4116 non-null int64
JAN            4116 non-null float64
FEB            4116 non-null float64
MAR            4116 non-null float64
APR            4116 non-null float64
MAY            4116 non-null float64
JUN            4116 non-null float64
JUL            4116 non-null float64
AUG            4116 non-null float64
SEP            4116 non-null float64
OCT            4116 non-null float64
NOV            4116 non-null float64
DEC            4116 non-null float64
ANNUAL         4116 non-null float64
Jan-Feb        4116 non-null float64
Mar-May        4116 non-null float64
Jun-Sep        4116 non-null float64
Oct-Dec        4116 non-null float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.0+ KB
```

1.5 Dataset-1 Description

- Data has 36 sub divisions and 19 attributes (individual months, annual, combinations of 3 consecutive months).
- For some of the subdivisions data is from 1950 to 2015.
- All the attributes has the sum of amount of rainfall in mm.

```
In [3]: data.head()
```

Out[3]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7

	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6
3	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.0
4	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.8

```
In [4]: data.describe()
```

Out[4]:

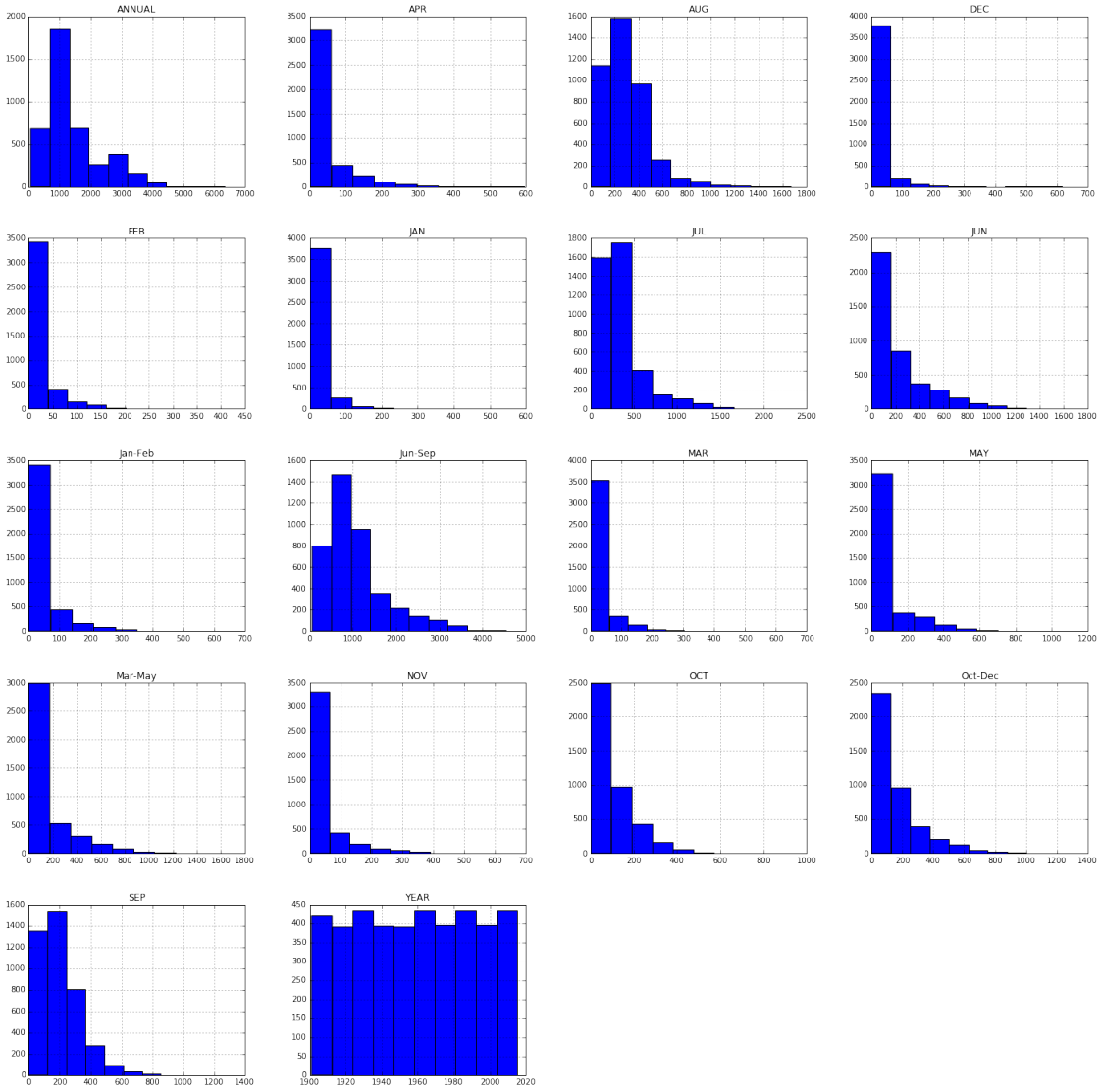
	YEAR	JAN	FEB	MAR	APR	\
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	
std	33.140898	33.569044	35.896396	46.925176	67.798192	
min	1901.000000	0.000000	0.000000	0.000000	0.000000	
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	
50%	1958.000000	6.000000	6.700000	7.900000	15.700000	
75%	1987.000000	22.125000	26.800000	31.225000	49.825000	
max	2015.000000	583.700000	403.500000	605.600000	595.100000	

	MAY	JUN	JUL	AUG	SEP	\
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	
mean	85.745417	230.234444	347.214334	290.263497	197.361922	
std	123.189974	234.568120	269.310313	188.678707	135.309591	
min	0.000000	0.400000	0.000000	0.000000	0.100000	
25%	8.600000	70.475000	175.900000	156.150000	100.600000	
50%	36.700000	138.900000	284.900000	259.500000	174.100000	
75%	96.825000	304.950000	418.225000	377.725000	265.725000	
max	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	

	OCT	NOV	DEC	ANNUAL	Jan-Feb	\
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	
mean	95.507009	39.866163	18.870580	1411.008900	40.747786	
std	99.434452	68.593545	42.318098	900.986632	59.265023	
min	0.000000	0.000000	0.000000	62.300000	0.000000	
25%	14.600000	0.700000	0.100000	806.450000	4.100000	
50%	65.750000	9.700000	3.100000	1125.450000	19.300000	
75%	148.300000	45.825000	17.700000	1635.100000	50.300000	
max	948.300000	648.900000	617.500000	6331.100000	699.500000	

	Mar-May	Jun-Sep	Oct-Dec
count	4116.000000	4116.000000	4116.000000
mean	155.901753	1064.724769	154.100487
std	201.096692	706.881054	166.678751
min	0.000000	57.400000	0.000000
25%	24.200000	574.375000	34.200000
50%	75.200000	882.250000	98.800000
75%	196.900000	1287.550000	212.600000
max	1745.800000	4536.900000	1252.500000

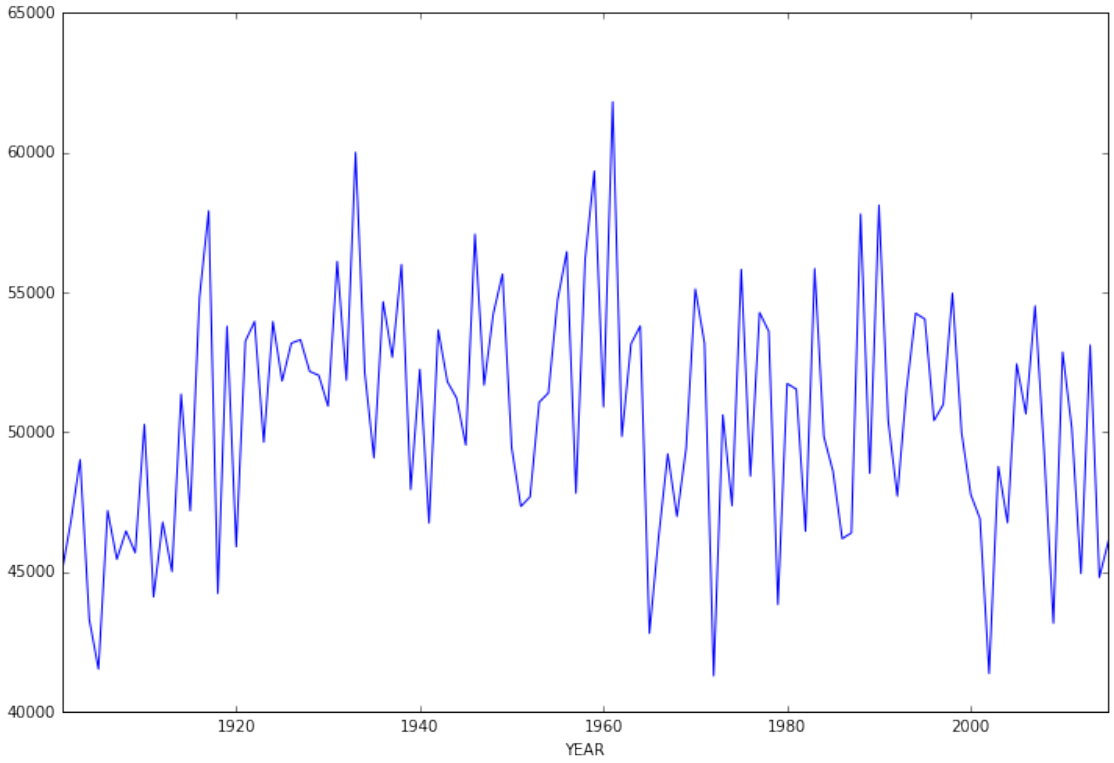
In [5]:
```
data.hist(figsize=(24,24));
```



1.6 Observations

- Above histograms show the distribution of rainfall over months.
- Observed increase in amount of rainfall over months July, August, September.

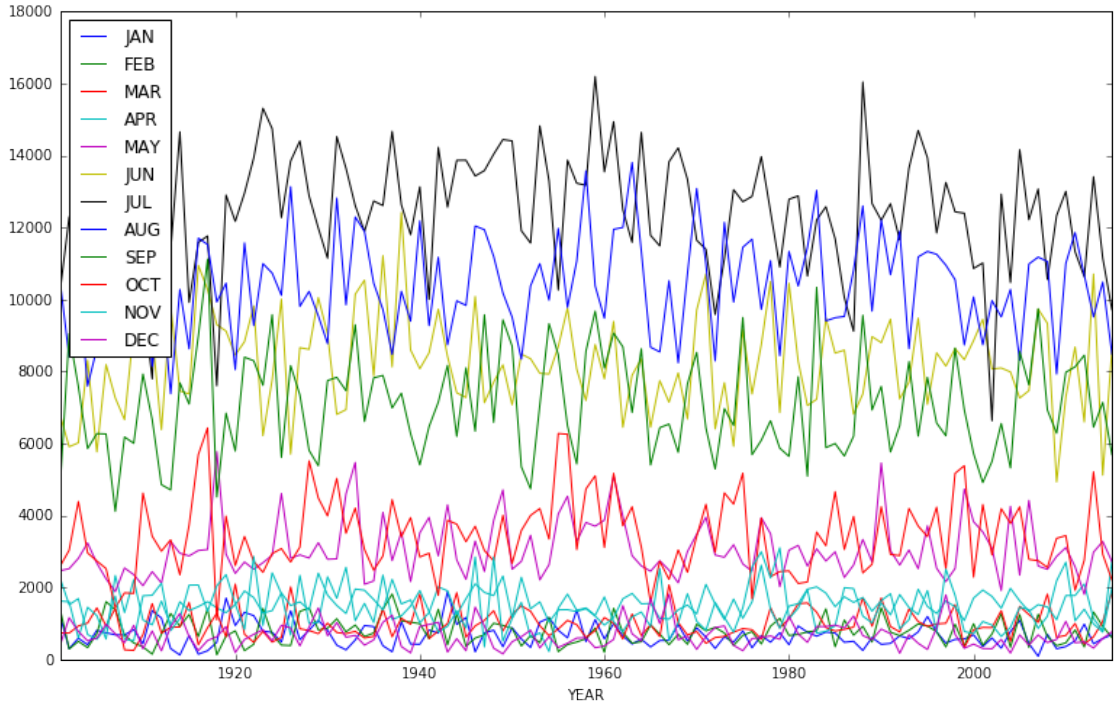
```
In [6]: data.groupby("YEAR").sum()['ANNUAL'].plot(figsize=(12,8));
```



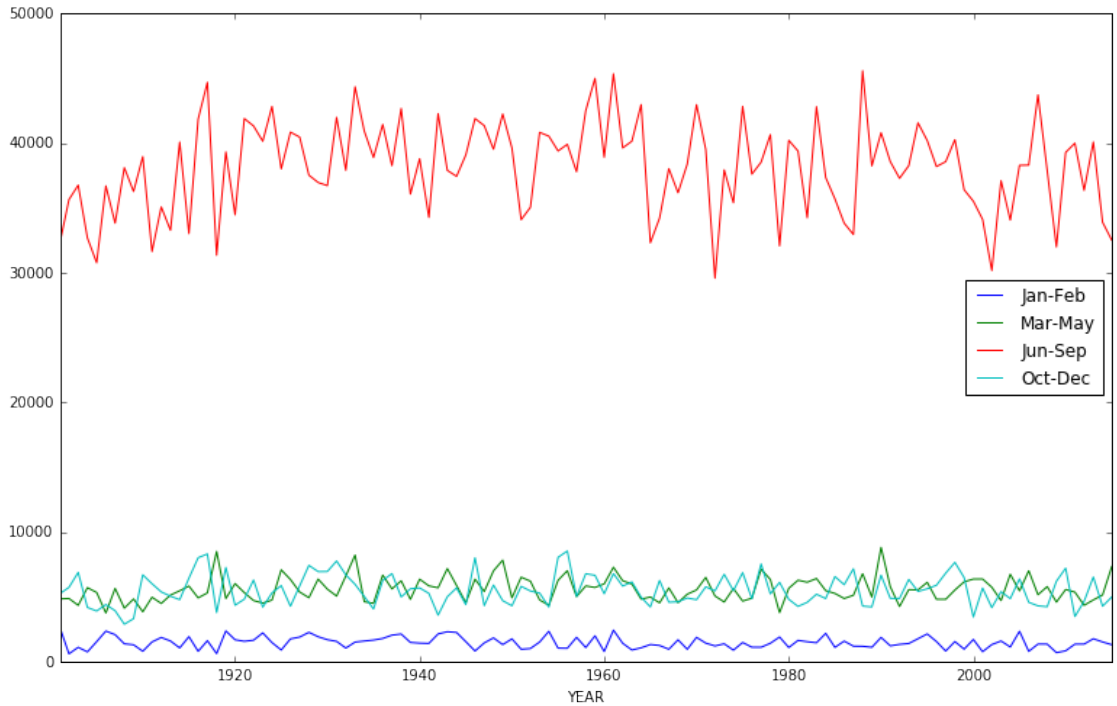
1.7 Observations

- Shows distribution of rainfall over years.
- Observed high amount of rainfall in 1950s.

```
In [7]: data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
               'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(figsize=(13,8));
```



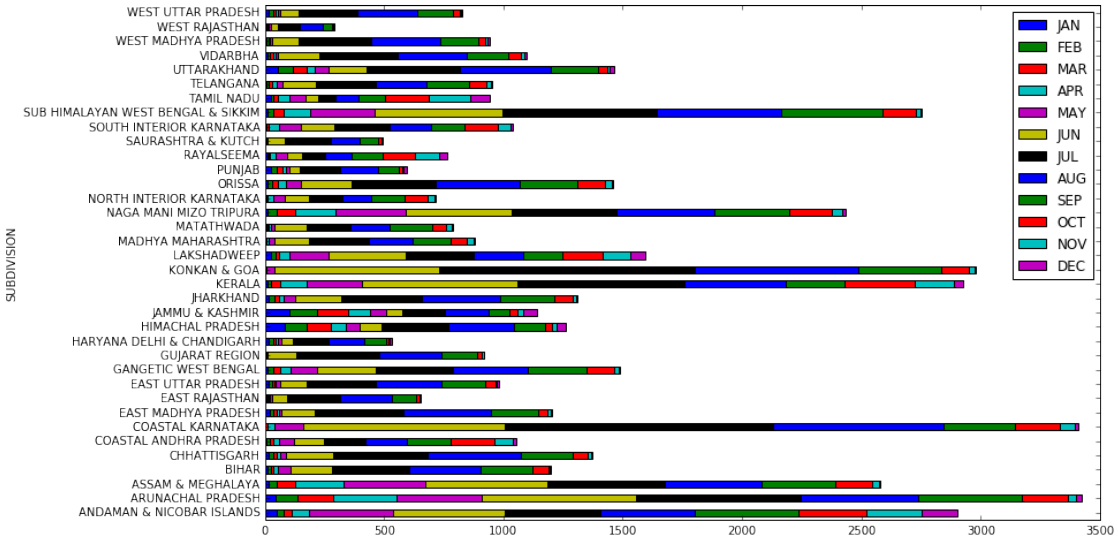

```
In [8]: data[['YEAR','Jan-Feb', 'Mar-May',
              'Jun-Sep', 'Oct-Dec']].groupby("YEAR").sum().plot(figsize=(13,8));
```



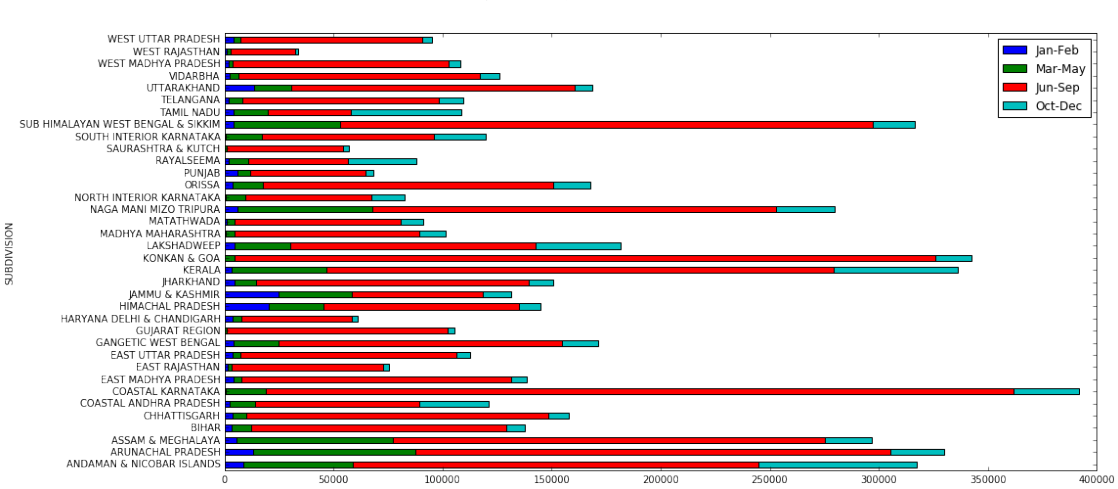
1.8 Observations

- The above two graphs show the distribution of rainfall over months.
- The graphs clearly shows that amount of rainfall in high in the months july, aug, sep which is monsoon season in India.

```
In [9]: data[['SUBDIVISION', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
               'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("SUBDIVISION").mean().plot.barh(stacked=True
```




```
In [10]: data[['SUBDIVISION', 'Jan-Feb', 'Mar-May',
                'Jun-Sep', 'Oct-Dec']].groupby("SUBDIVISION").sum().plot.barh(stacked=True,figsize=(16,8
```

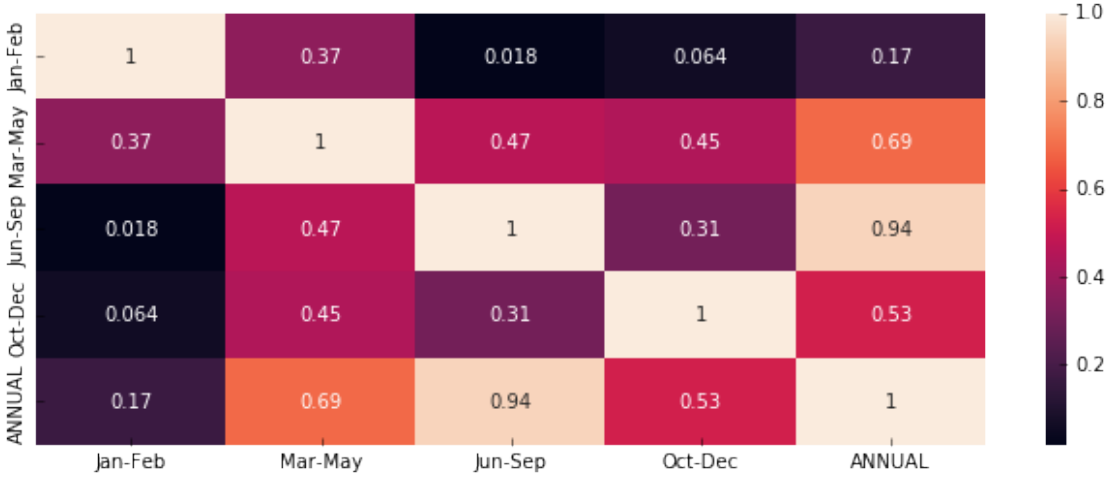


1.9 Observations

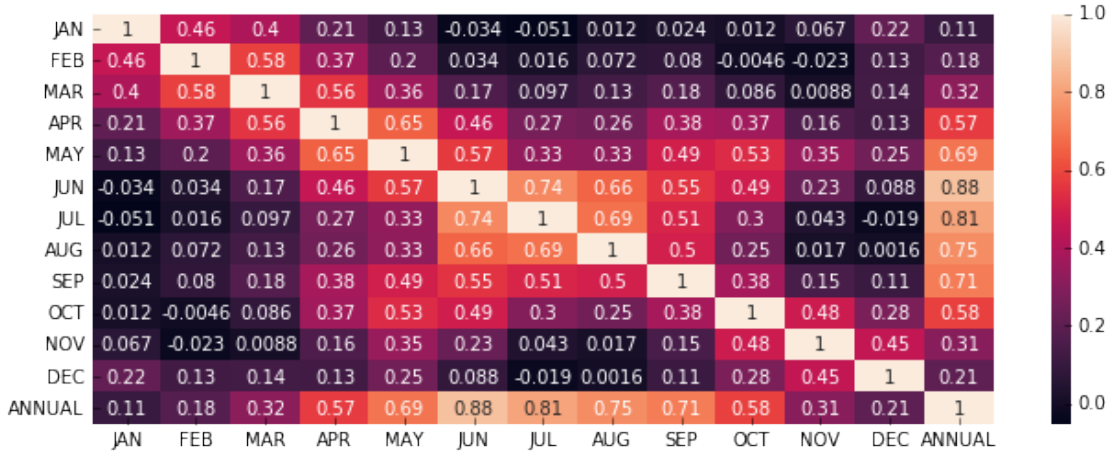
- Above two graphs shows that the amount of rainfall is reasonably good in the months of march, april, may in eastern India.

In [11]:
```
plt.figure(figsize=(11,4))
sns.heatmap(data[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec','ANNUAL']].corr(),annot=True)
plt.show()
```

```
/home/.local/lib/python2.7/site-packages/pandas/core/computation/check.py:17: UserWarning: The
installed version of numexpr 2.4.3 is not supported in pandas and will be not be usedThe minimum
supported version is 2.4.6

  ver=ver, min_ver=_MIN_NUMEXPR_VERSION), UserWarning)
```



In [12]:
```
plt.figure(figsize=(11,4))
sns.heatmap(data[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC','ANN
plt.show()
```



1.10 Observations

- **Heat Map** shows the co-relation(dependency) betwenn the amounts of rainfall over months.

- From above it is clear that if amount of rainfall is high in the months of july, august, september then the amount of rainfall will be high annually.
- It is also obwserved that if amount of rainfall in good in the months of october, november, december then the rainfall is going to b good in the overall year.

In [13]:
```
#Function to plot the graphs
def plot_graphs(groundtruth,prediction,title):
    N = 9
    ind = np.arange(N)  # the x locations for the groups
    width = 0.27       # the width of the bars

    fig = plt.figure()
    fig.suptitle(title, fontsize=12)
    ax = fig.add_subplot(111)
    rects1 = ax.bar(ind, groundtruth, width, color='r')
    rects2 = ax.bar(ind+width, prediction, width, color='g')

    ax.set_ylabel("Amount of rainfall")
    ax.set_xticks(ind+width)
    ax.set_xticklabels( ('APR', 'MAY', 'JUN', 'JUL','AUG', 'SEP', 'OCT', 'NOV', 'DEC') )
    ax.legend( (rects1[0], rects2[0]), ('Ground truth', 'Prediction') )

#     autolabel(rects1)
    for rect in rects1:
        h = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%d'%int(h),
                ha='center', va='bottom')
    for rect in rects2:
        h = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%d'%int(h),
                ha='center', va='bottom')
#     autolabel(rects2)

    plt.show()
```

1.11 Predictions

- For prediction we formatted data in the way, given the rainfall in the last three months we try to predict the rainfall in the next consecutive month.
- For all the experiments we used 80:20 training and test ratio.
  - Linear regression
  - SVR
  - Artificial neural nets
- Tersting metrics: We used Mean absolute error to train the models.
- We also shown the amount of rainfall actually and predicted with the histogram plots.
- We did two types of trainings once training on complete dataset and other with training with only telangana data
- All means are standard deviation observations are written, first one represents ground truth, second one represents predictions.

In [14]:
```
# seperation of training and testing data
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

```
division_data = np.asarray(data[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']])

X = None; y = None
for i in range(division_data.shape[1]-3):
    if X is None:
        X = division_data[:, i:i+3]
        y = division_data[:, i+3]
    else:
        X = np.concatenate((X, division_data[:, i:i+3]), axis=0)
        y = np.concatenate((y, division_data[:, i+3]), axis=0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

In [15]:
```
#test 2010
temp = data[['SUBDIVISION','JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[data['YEAR'] == 2010]

data_2010 = np.asarray(temp[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[temp['SUBDIVISION'] == 'TELANGANA'])

X_year_2010 = None; y_year_2010 = None
for i in range(data_2010.shape[1]-3):
    if X_year_2010 is None:
        X_year_2010 = data_2010[:, i:i+3]
        y_year_2010 = data_2010[:, i+3]
    else:
        X_year_2010 = np.concatenate((X_year_2010, data_2010[:, i:i+3]), axis=0)
        y_year_2010 = np.concatenate((y_year_2010, data_2010[:, i+3]), axis=0)
```

In [16]:
```
#test 2005
temp = data[['SUBDIVISION','JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[data['YEAR'] == 2005]

data_2005 = np.asarray(temp[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[temp['SUBDIVISION'] == 'TELANGANA'])

X_year_2005 = None; y_year_2005 = None
for i in range(data_2005.shape[1]-3):
    if X_year_2005 is None:
        X_year_2005 = data_2005[:, i:i+3]
        y_year_2005 = data_2005[:, i+3]
    else:
        X_year_2005 = np.concatenate((X_year_2005, data_2005[:, i:i+3]), axis=0)
        y_year_2005 = np.concatenate((y_year_2005, data_2005[:, i+3]), axis=0)
```

In [17]:
```
#terst 2015
temp = data[['SUBDIVISION','JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[data['YEAR'] == 2015]

data_2015 = np.asarray(temp[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[temp['SUBDIVISION'] == 'TELANGANA'])

X_year_2015 = None; y_year_2015 = None
for i in range(data_2015.shape[1]-3):
```

```
            if X_year_2015 is None:
                X_year_2015 = data_2015[:, i:i+3]
                y_year_2015 = data_2015[:, i+3]
            else:
                X_year_2015 = np.concatenate((X_year_2015, data_2015[:, i:i+3]), axis=0)
                y_year_2015 = np.concatenate((y_year_2015, data_2015[:, i+3]), axis=0)
```

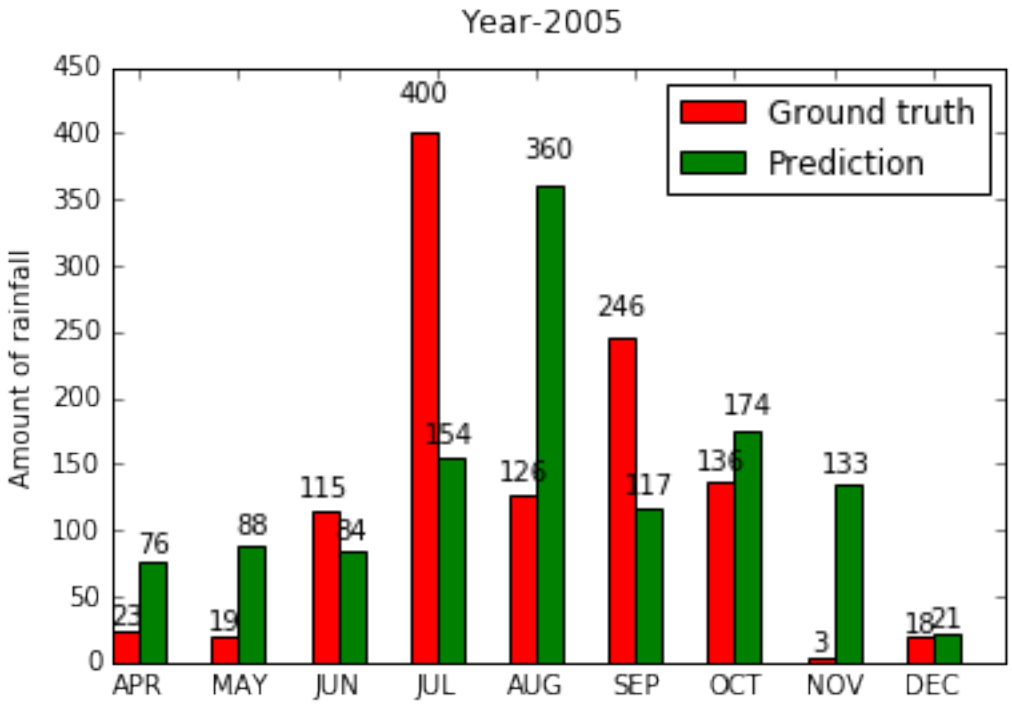
In [18]:
```
from sklearn import linear_model

# linear model
reg = linear_model.ElasticNet(alpha=0.5)
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

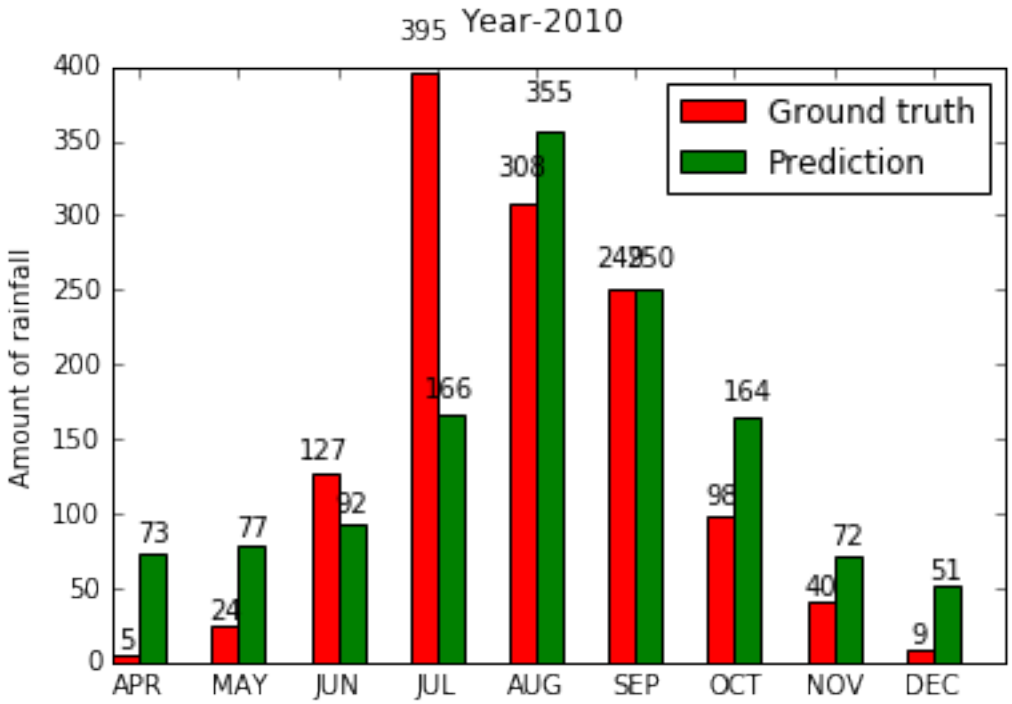
```
96.32435229744095
```

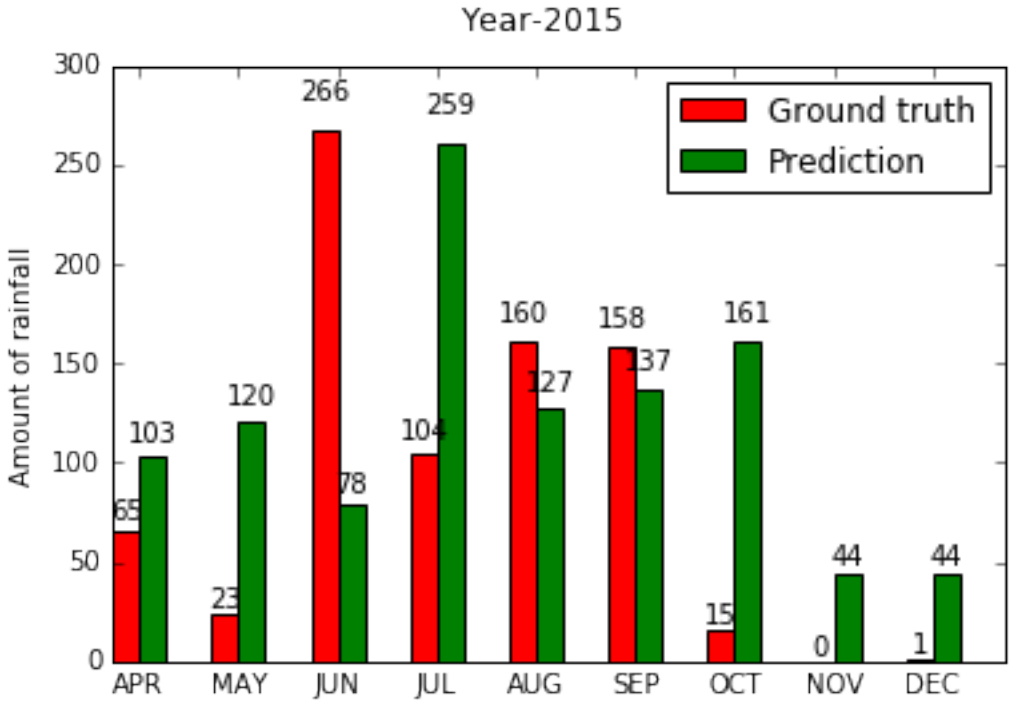
In [19]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)


#2010
y_year_pred_2010 = reg.predict(X_year_2010)

y_year_pred_2015 = reg.predict(X_year_2015)

print "MEAN 2005"
print np.mean(y_year_2005),np.mean(y_year_pred_2005)
print "Standard deviation 2005"
print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


print "MEAN 2010"
print np.mean(y_year_2010),np.mean(y_year_pred_2010)
print "Standard deviation 2010"
print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


print "MEAN 2015"
print np.mean(y_year_2015),np.mean(y_year_pred_2015)
print "Standard deviation 2015"
print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))


plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 134.68699821349824
Standard deviation 2005
123.77066107608005 90.86310230416397
MEAN 2010
139.93333333333334 144.8050132651592
```

```
Standard deviation 2010
135.71320250194282 95.94931363601675
MEAN 2015
88.52222222222223 119.64752006738864
Standard deviation 2015
86.62446123324875 62.36355370163346
```







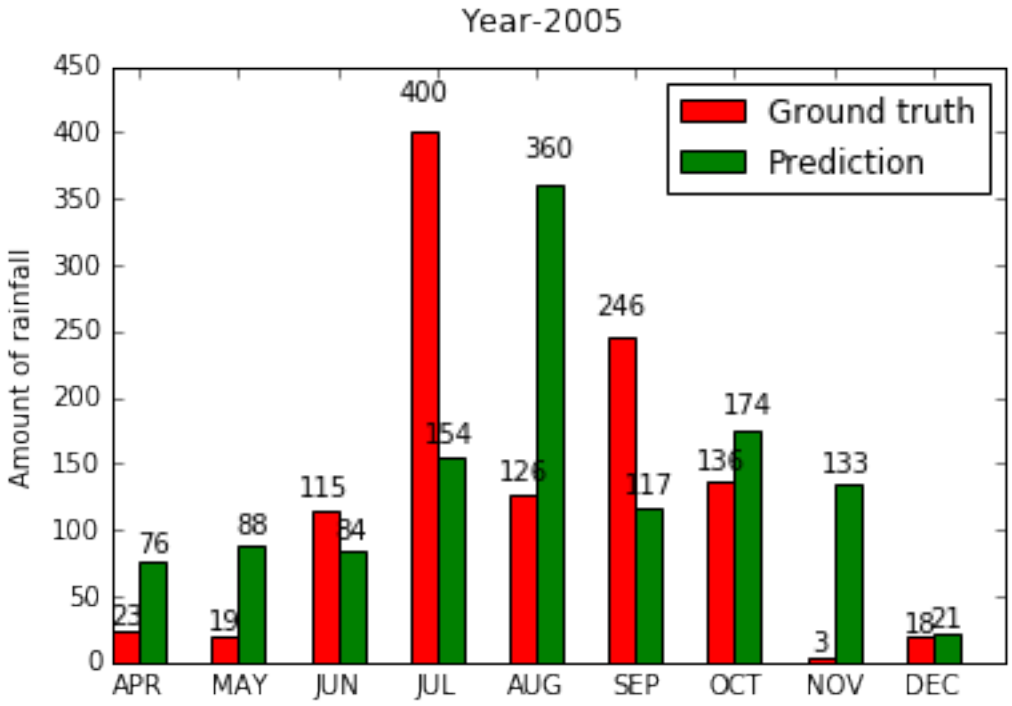

```
In [20]: from sklearn.svm import SVR
```

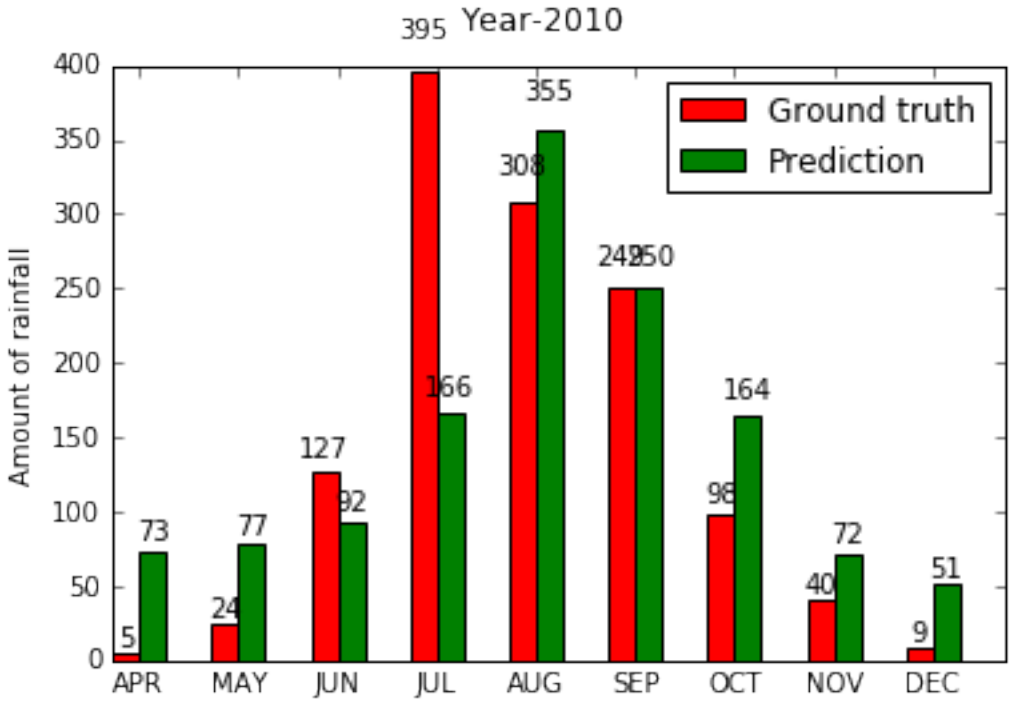
```
        # SVM model
        clf = SVR(gamma='auto', C=0.1, epsilon=0.2)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        print mean_absolute_error(y_test, y_pred)
```

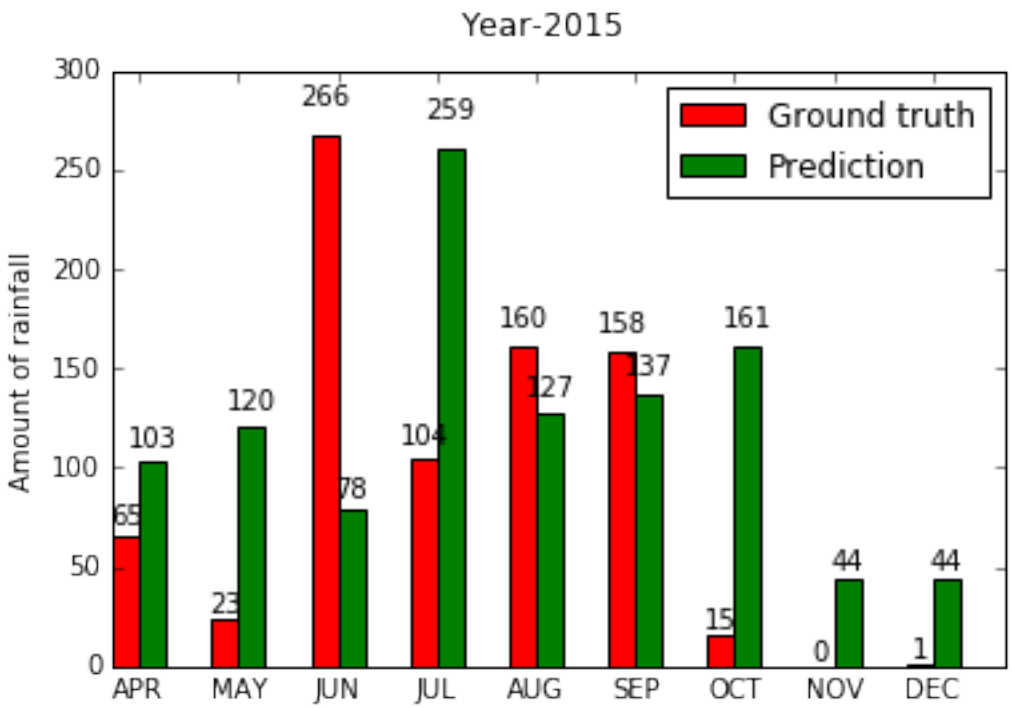
```
127.1600615632603
```

```
In [21]: #2005
        y_year_pred_2005 = reg.predict(X_year_2005)

        #2010
        y_year_pred_2010 = reg.predict(X_year_2010)

        #2015
        y_year_pred_2015 = reg.predict(X_year_2015)
        print "MEAN 2005"
        print np.mean(y_year_2005),np.mean(y_year_pred_2005)
        print "Standard deviation 2005"
        print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


        print "MEAN 2010"
        print np.mean(y_year_2010),np.mean(y_year_pred_2010)
        print "Standard deviation 2010"
        print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


        print "MEAN 2015"
        print np.mean(y_year_2015),np.mean(y_year_pred_2015)
        print "Standard deviation 2015"
        print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))

        plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
        plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
        plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 134.68699821349824
Standard deviation 2005
123.77066107608005 90.86310230416397
MEAN 2010
139.93333333333334 144.8050132651592
Standard deviation 2010
135.71320250194282 95.94931363601675
MEAN 2015
88.52222222222223 119.64752006738864
Standard deviation 2015
86.62446123324875 62.36355370163346
```


Year-2005
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
23
76
19
88
115
84
400
154
126
360
246
117
136
174
3
133
18
21


Year-2010
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5
73
24
77
127
92
166
308
355
242
250
98
164
40
72
9
51




```
In [22]: from keras.models import Model
         from keras.layers import Dense, Input, Conv1D, Flatten

         # NN model
         inputs = Input(shape=(3,1))
         x = Conv1D(64, 2, padding='same', activation='elu')(inputs)
         x = Conv1D(128, 2, padding='same', activation='elu')(x)
         x = Flatten()(x)
         x = Dense(128, activation='elu')(x)
         x = Dense(64, activation='elu')(x)
         x = Dense(32, activation='elu')(x)
         x = Dense(1, activation='linear')(x)
         model = Model(inputs=[inputs], outputs=[x])
         model.compile(loss='mean_squared_error', optimizer='adamax', metrics
=['mae'])
         model.summary()

/home/.local/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning:
  Conversion of the second argument of issubdtype from `float` to `np.
  floating` is deprecated. In future, it will be treated as `np.float64 == np
  .dtype(float).type`.from ._conv import register_converters as _register_
  converters
Using TensorFlow backend.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 3, 1)              0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 3, 64)             192
_________________________________________________________________
```

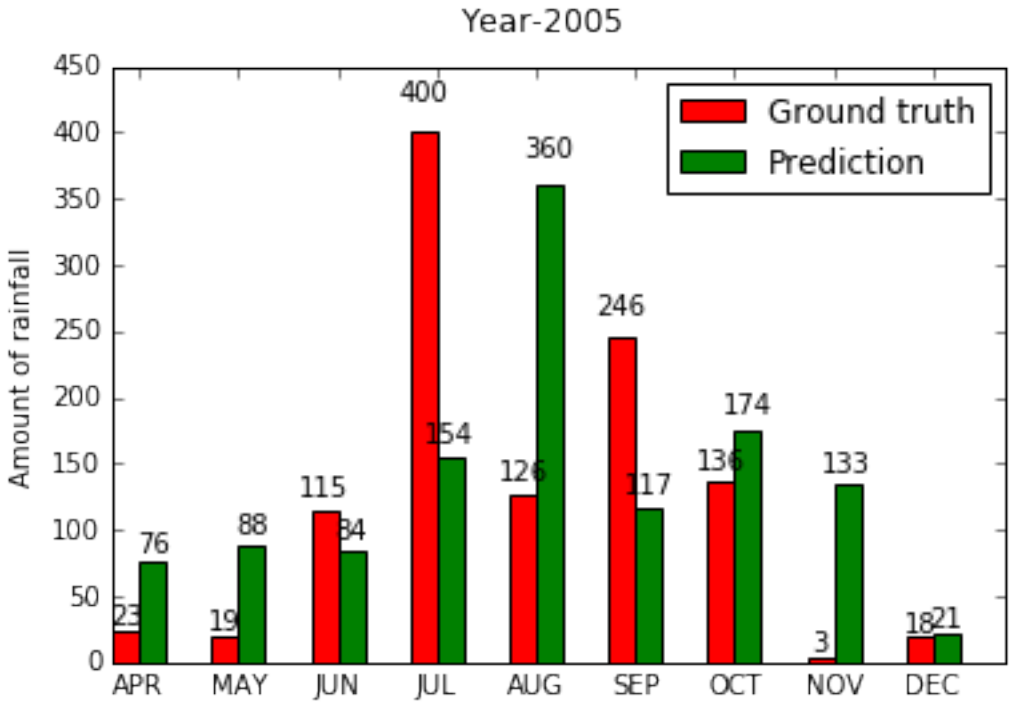
```
conv1d_2 (Conv1D)            (None, 3, 128)            16512
_________________________________________________________________
flatten_1 (Flatten)          (None, 384)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               49280
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 76,353
Trainable params: 76,353
Non-trainable params: 0
_________________________________________________________________
```

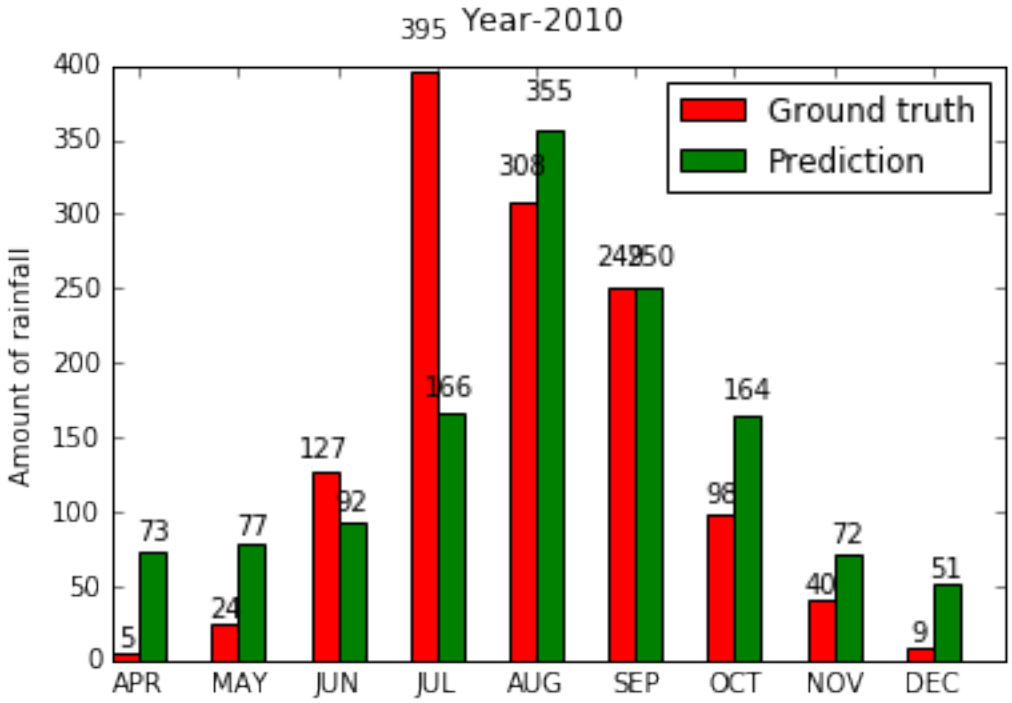
```
In [23]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=10,
verbose=1, validation_split=0.1, shuffle=True)
         y_pred = model.predict(np.expand_dims(X_test, axis=2))
         print mean_absolute_error(y_test, y_pred)

Train on 30005 samples, validate on 3334 samples
Epoch 1/10
30005/30005 [==============================] - 3s 111us/step - loss: 19589.7591 - mean_
absolute_error: 88.2724 - val_loss: 17504.6129 - val_mean_absolute_error: 84.5918Epoch 2/
10
30005/30005 [==============================] - 2s 53us/step - loss: 18582.2211 - mean_
absolute_error: 86.5600 - val_loss: 17480.7870 - val_mean_absolute_error: 84.0881Epoch 3/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18466.6604 - mean_
absolute_error: 86.3714 - val_loss: 17362.4027 - val_mean_absolute_error: 85.7931Epoch 4/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18482.5326 - mean_
absolute_error: 86.5104 - val_loss: 17358.8929 - val_mean_absolute_error: 86.6657Epoch 5/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18358.7726 - mean_
absolute_error: 86.0046 - val_loss: 17507.7787 - val_mean_absolute_error: 85.0990Epoch 6/
10
30005/30005 [==============================] - 2s 53us/step - loss: 18312.5666 - mean_
absolute_error: 85.8176 - val_loss: 17194.2290 - val_mean_absolute_error: 84.7509Epoch 7/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18236.9615 - mean_
absolute_error: 85.5843 - val_loss: 17125.2238 - val_mean_absolute_error: 84.4872Epoch 8/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18118.3601 - mean_
absolute_error: 85.2783 - val_loss: 17041.7574 - val_mean_absolute_error: 84.0770Epoch 9/
10
30005/30005 [==============================] - 2s 54us/step - loss: 18193.9362 - mean_
absolute_error: 85.4572 - val_loss: 17189.8372 - val_mean_absolute_error: 83.7721Epoch 10
/10
30005/30005 [==============================] - 2s 54us/step - loss: 18007.9055 - mean_
absolute_error: 84.9875 - val_loss: 18282.9227 - val_mean_absolute_error: 89.932092.
28250624049363
```

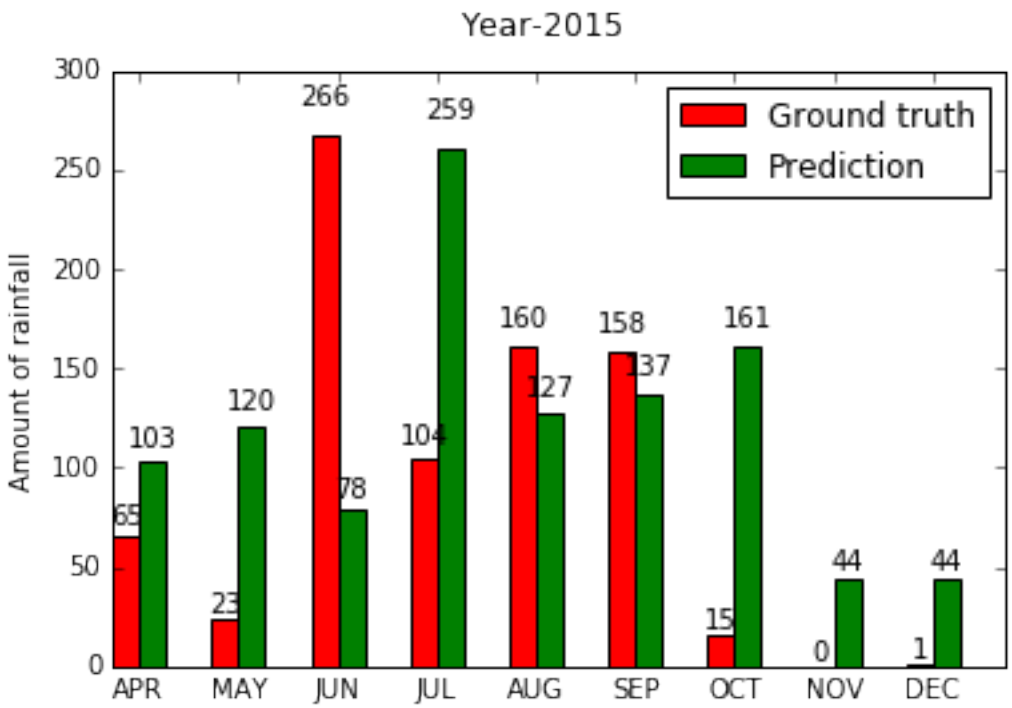
In [24]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)
#2010
y_year_pred_2010 = reg.predict(X_year_2010)
#2015
y_year_pred_2015 = reg.predict(X_year_2015)
```

```
print "MEAN 2005"
print np.mean(y_year_2005),np.mean(y_year_pred_2005)
print "Standard deviation 2005"
print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


print "MEAN 2010"
print np.mean(y_year_2010),np.mean(y_year_pred_2010)
print "Standard deviation 2010"
print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


print "MEAN 2015"
print np.mean(y_year_2015),np.mean(y_year_pred_2015)
print "Standard deviation 2015"
print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))

plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 134.68699821349824
Standard deviation 2005
123.77066107608005 90.86310230416397
MEAN 2010
139.93333333333334 144.8050132651592
Standard deviation 2010
135.71320250194282 95.94931363601675
MEAN 2015
88.52222222222223 119.64752006738864
Standard deviation 2015
86.62446123324875 62.36355370163346
```


Year-2005
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
23
76
19
88
115
84
400
154
126
360
246
117
136
174
3
133
18
21


Year-2010
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5
73
24
77
127
92
166
308
355
242
250
98
164
40
72
9
51



In [25]:
```
# spliting training and testing data only for telangana
telangana = np.asarray(data[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].loc[data['SUBDIVISION'] == 'TELANGANA'])

X = None; y = None
for i in range(telangana.shape[1]-3):
    if X is None:
        X = telangana[:, i:i+3]
        y = telangana[:, i+3]
    else:
        X = np.concatenate((X, telangana[:, i:i+3]), axis=0)
        y = np.concatenate((y, telangana[:, i+3]), axis=0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01, random_state=42)
```

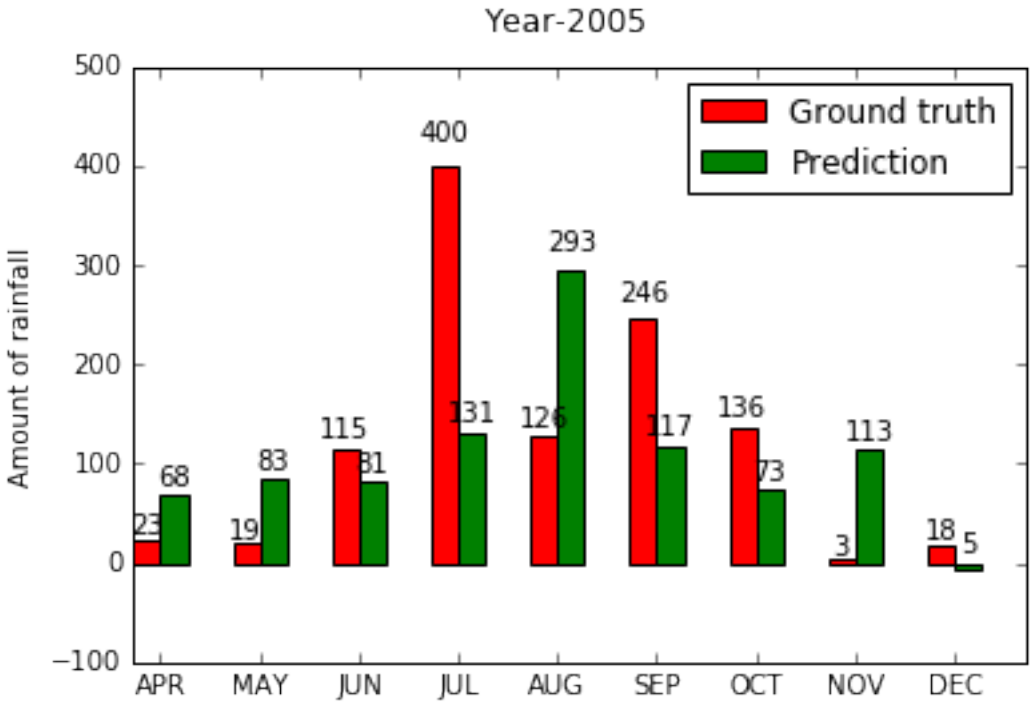
In [26]:
```
from sklearn import linear_model

# linear model
reg = linear_model.ElasticNet(alpha=0.5)
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

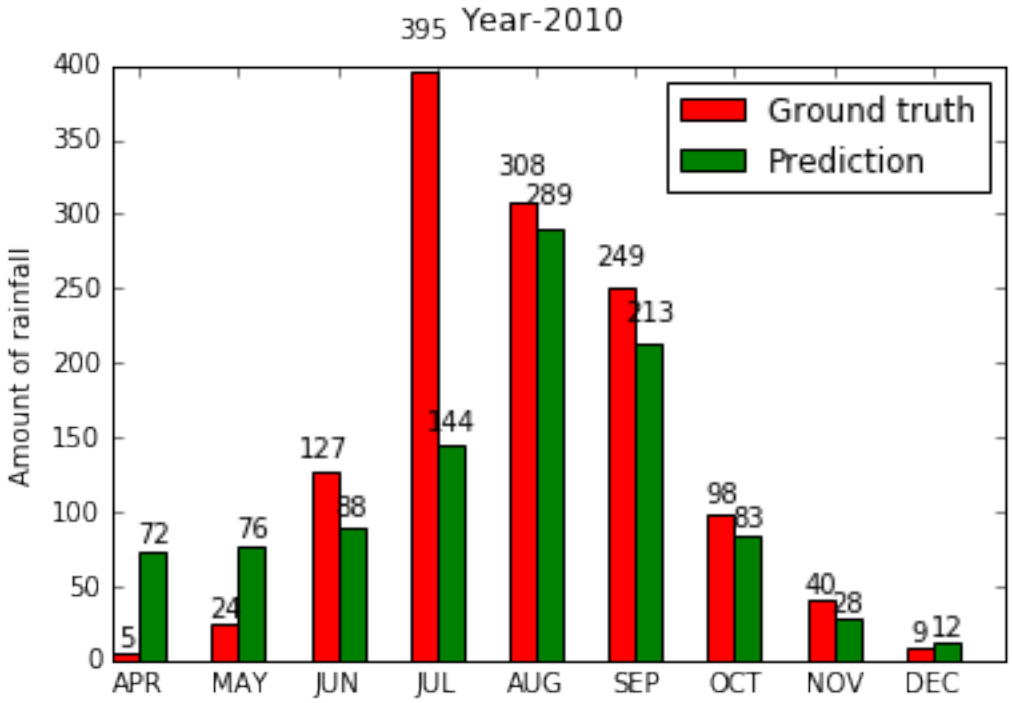
```
64.72601914484643
```

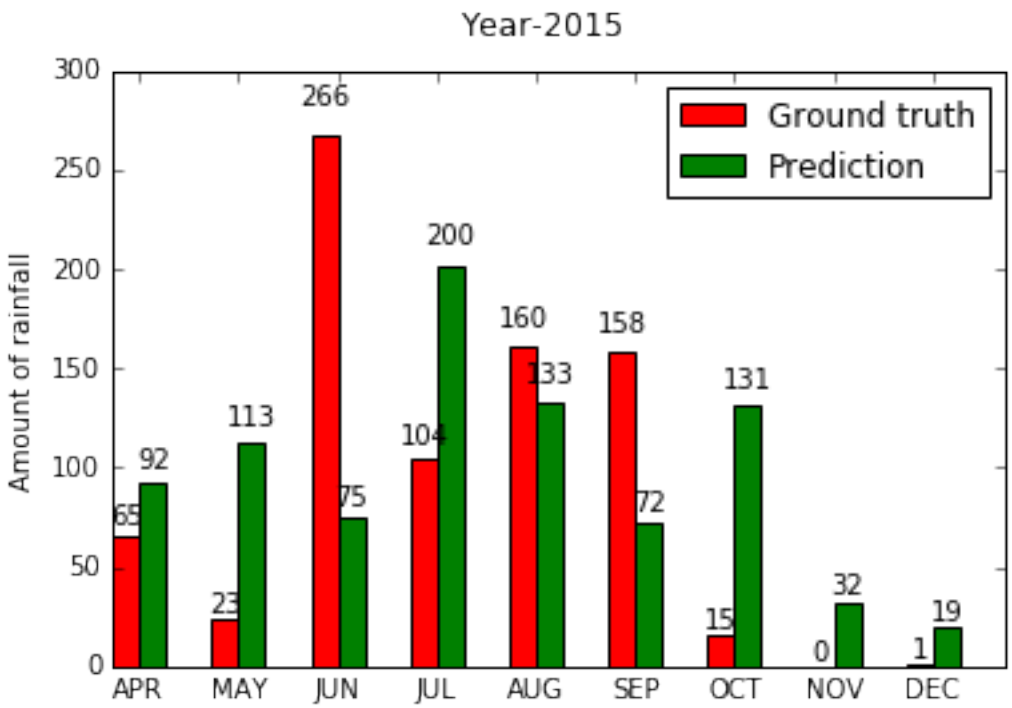
In [27]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)
```

```
        #2010
        y_year_pred_2010 = reg.predict(X_year_2010)

        #2015
        y_year_pred_2015 = reg.predict(X_year_2015)

        print "MEAN 2005"
        print np.mean(y_year_2005),np.mean(y_year_pred_2005)
        print "Standard deviation 2005"
        print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


        print "MEAN 2010"
        print np.mean(y_year_2010),np.mean(y_year_pred_2010)
        print "Standard deviation 2010"
        print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


        print "MEAN 2015"
        print np.mean(y_year_2015),np.mean(y_year_pred_2015)
        print "Standard deviation 2015"
        print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))

        plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
        plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
        plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 106.49798150231584
Standard deviation 2005
123.77066107608005 76.08558540019227
MEAN 2010
139.93333333333334 112.18662987131034
Standard deviation 2010
135.71320250194282 84.35813629737324
MEAN 2015
88.52222222222223 96.76817006572782
Standard deviation 2015
86.62446123324875 52.45304841713261
```


Year-2005
Ground truth
Prediction
Amount of rainfall
500
400
300
200
100
0
−100
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
23
68
19
83
115
81
400
131
126
293
246
117
136
73
3
113
18
5


395
Year-2010
Ground truth
Prediction
Amount of rainfall
400
350
300
250
200
150
100
50
0
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5
72
24
76
127
88
144
308
289
249
213
98
83
40
28
9
12



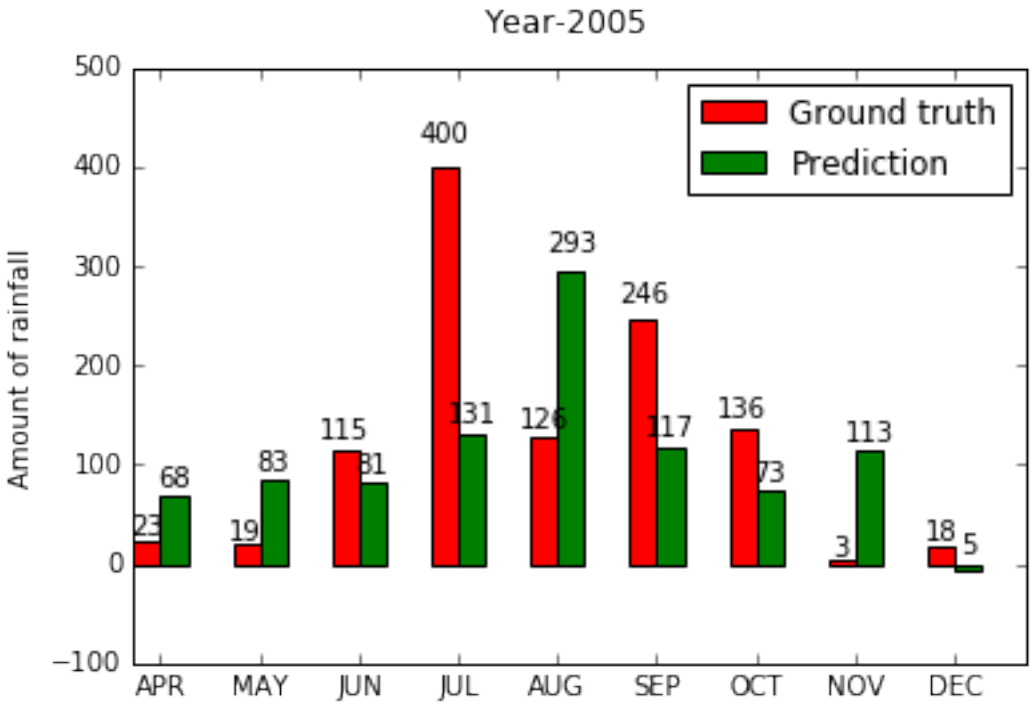
In [28]:
```
from sklearn.svm import SVR

# SVM model
clf = SVR(kernel='rbf', gamma='auto', C=0.5, epsilon=0.2)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

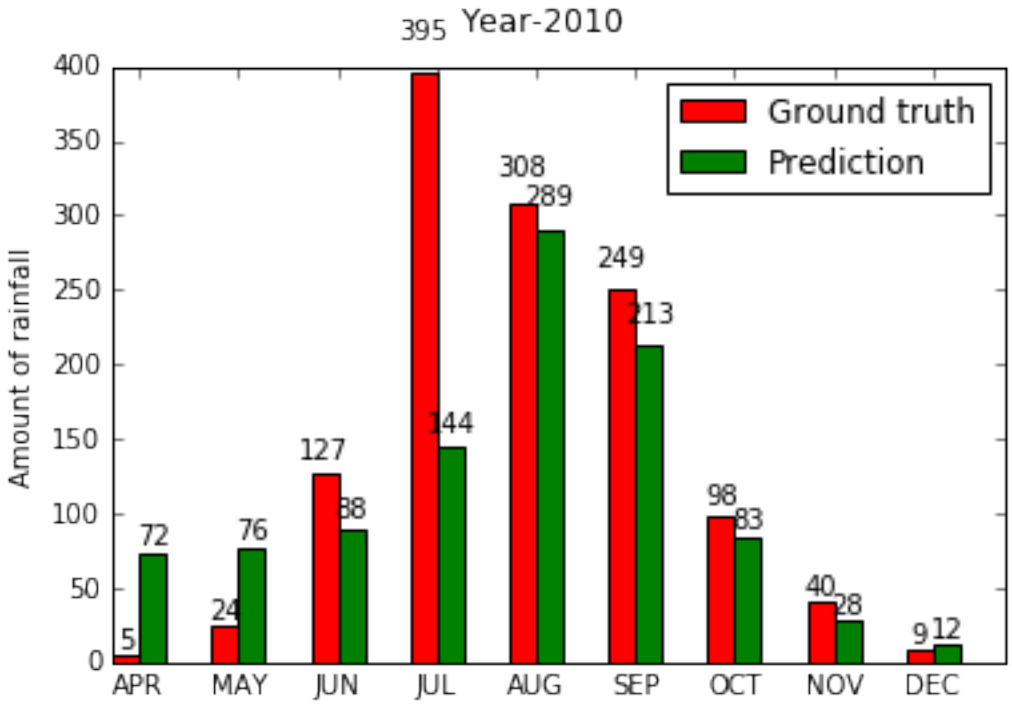
```
115.32415990638656
```

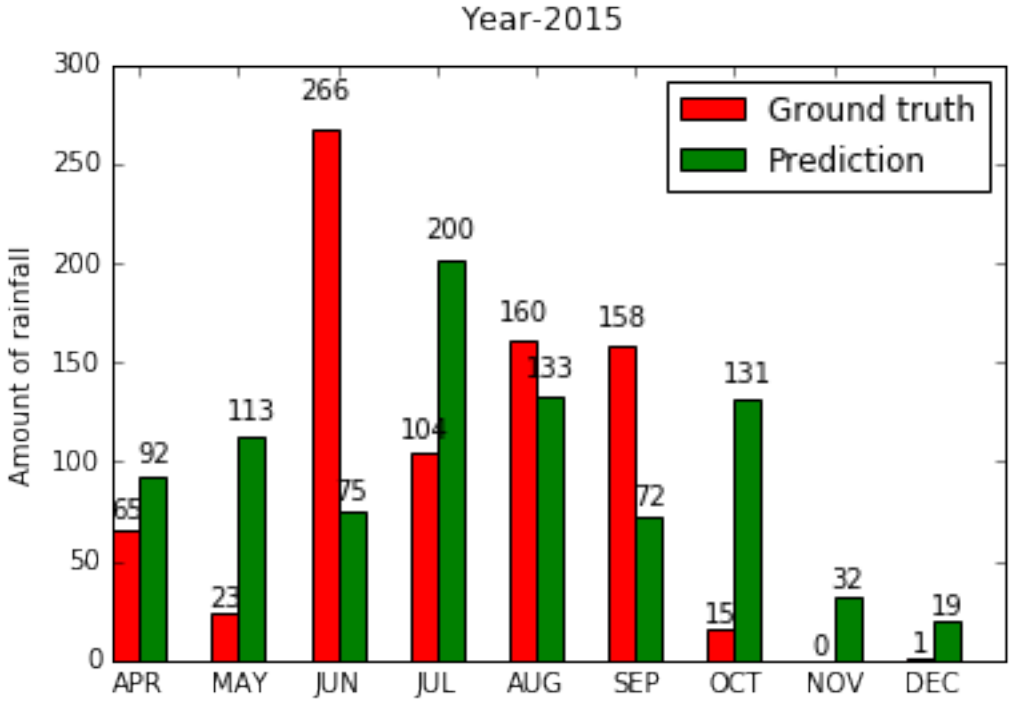
In [29]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)

#2010
y_year_pred_2010 = reg.predict(X_year_2010)

#2015
y_year_pred_2015 = reg.predict(X_year_2015)

print "MEAN 2005"
print np.mean(y_year_2005),np.mean(y_year_pred_2005)
print "Standard deviation 2005"
print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


print "MEAN 2010"
print np.mean(y_year_2010),np.mean(y_year_pred_2010)
```

```
print "Standard deviation 2010"
print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


print "MEAN 2015"
print np.mean(y_year_2015),np.mean(y_year_pred_2015)
print "Standard deviation 2015"
print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))

plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 106.49798150231584
Standard deviation 2005
123.77066107608005 76.08558540019227
MEAN 2010
139.93333333333334 112.18662987131034
Standard deviation 2010
135.71320250194282 84.35813629737324
MEAN 2015
88.52222222222223 96.76817006572782
Standard deviation 2015
86.62446123324875 52.45304841713261
```








```
In [30]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size
=64, epochs=10, verbose=1, validation_split=0.1, shuffle=True)
```

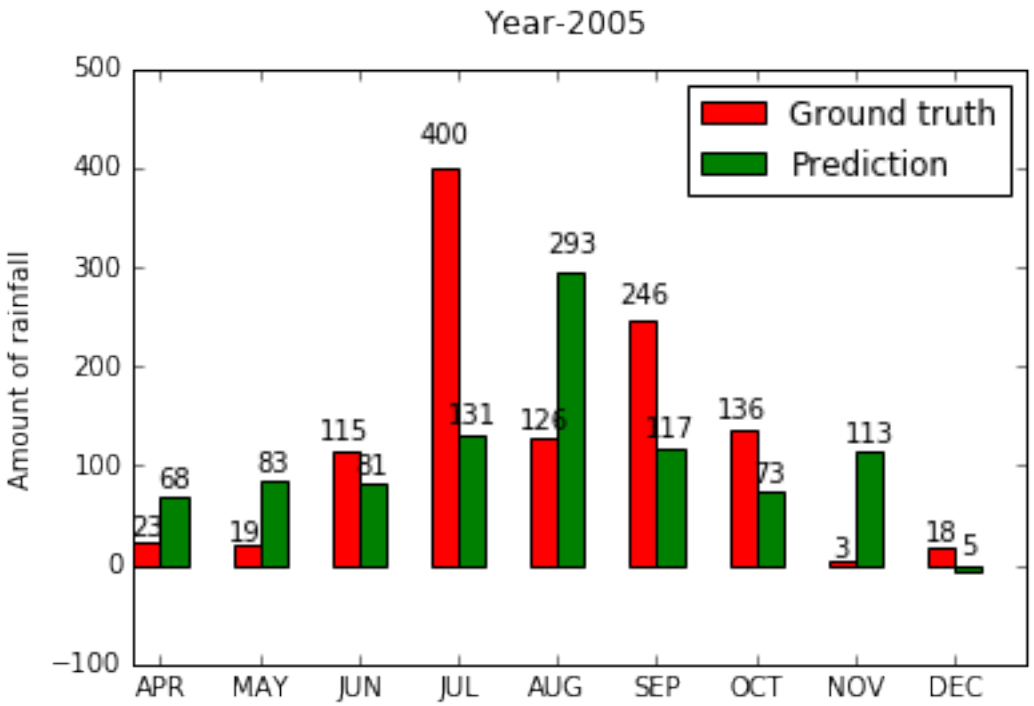
```
        y_pred = model.predict(np.expand_dims(X_test, axis=2))
        print mean_absolute_error(y_test, y_pred)
```

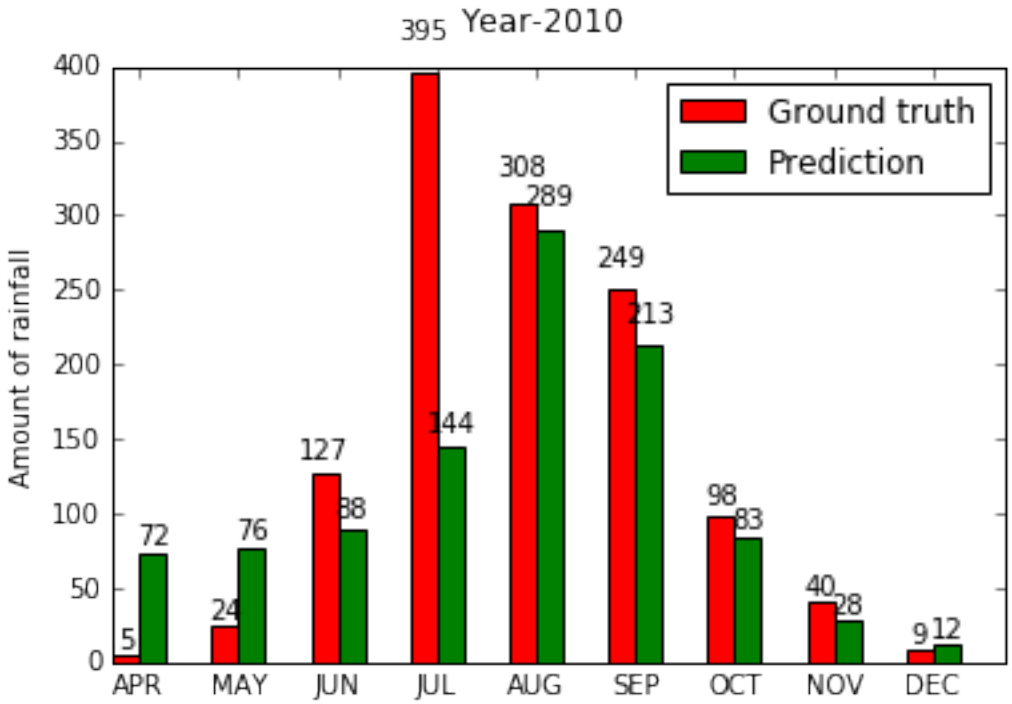
```
Train on 921 samples, validate on 103 samples
Epoch 1/10
921/921 [==============================] - 0s 66us/step - loss: 7274.9487 - mean_absolute_error: 63.5027 -
val_loss: 4981.0237 - val_mean_absolute_error: 51.2358Epoch 2/10
921/921 [==============================] - 0s 56us/step - loss: 6431.8426 - mean_absolute_error: 56.7670 -
val_loss: 4651.5261 - val_mean_absolute_error: 50.8381Epoch 3/10
921/921 [==============================] - 0s 56us/step - loss: 6046.0127 - mean_absolute_error: 58.4860 -
val_loss: 4725.7208 - val_mean_absolute_error: 52.5220Epoch 4/10
921/921 [==============================] - 0s 56us/step - loss: 5883.5181 - mean_absolute_error: 56.4383 -
val_loss: 4462.1687 - val_mean_absolute_error: 49.4729Epoch 5/10
921/921 [==============================] - 0s 57us/step - loss: 5764.2698 - mean_absolute_error: 55.1780 -
val_loss: 4419.6625 - val_mean_absolute_error: 49.3678Epoch 6/10
921/921 [==============================] - 0s 55us/step - loss: 5706.7510 - mean_absolute_error: 55.3464 -
val_loss: 4298.1492 - val_mean_absolute_error: 48.7378Epoch 7/10
921/921 [==============================] - 0s 56us/step - loss: 5636.2414 - mean_absolute_error: 54.4525 -
val_loss: 4264.9195 - val_mean_absolute_error: 48.5872Epoch 8/10
921/921 [==============================] - 0s 57us/step - loss: 5564.0726 - mean_absolute_error: 54.5664 -
val_loss: 4211.3810 - val_mean_absolute_error: 48.6857Epoch 9/10
921/921 [==============================] - 0s 57us/step - loss: 5529.6288 - mean_absolute_error: 54.0025 -
val_loss: 4201.4989 - val_mean_absolute_error: 48.0793Epoch 10/10
921/921 [==============================] - 0s 57us/step - loss: 5478.9296 - mean_absolute_error: 53.5252 -
val_loss: 4220.2539 - val_mean_absolute_error: 47.855265.82400645938786
```

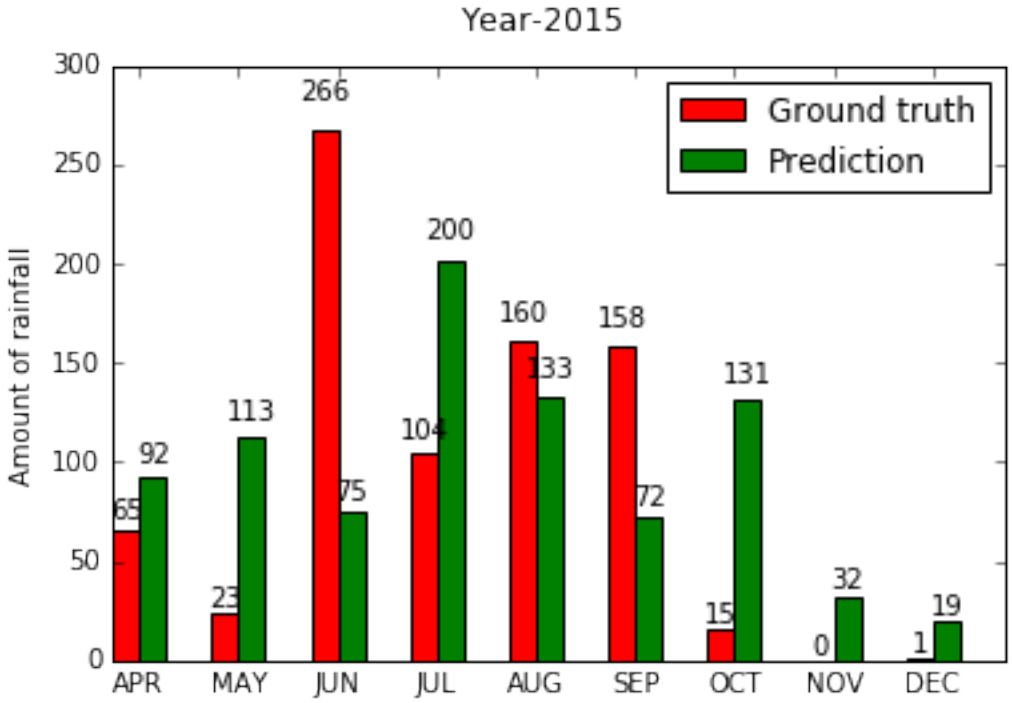
In [31]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)

#2010
y_year_pred_2010 = reg.predict(X_year_2010)

#2015
y_year_pred_2015 = reg.predict(X_year_2015)

print "MEAN 2005"
print np.mean(y_year_2005),np.mean(y_year_pred_2005)
print "Standard deviation 2005"
print np.sqrt(np.var(y_year_2005)),np.sqrt(np.var(y_year_pred_2005))


print "MEAN 2010"
print np.mean(y_year_2010),np.mean(y_year_pred_2010)
print "Standard deviation 2010"
print np.sqrt(np.var(y_year_2010)),np.sqrt(np.var(y_year_pred_2010))


print "MEAN 2015"
print np.mean(y_year_2015),np.mean(y_year_pred_2015)
print "Standard deviation 2015"
print np.sqrt(np.var(y_year_2015)),np.sqrt(np.var(y_year_pred_2015))

plot_graphs(y_year_2005,y_year_pred_2005,"Year-2005")
plot_graphs(y_year_2010,y_year_pred_2010,"Year-2010")
```

```
plot_graphs(y_year_2015,y_year_pred_2015,"Year-2015")
```

```
MEAN 2005
121.2111111111111 106.49798150231584
Standard deviation 2005
123.77066107608005 76.08558540019227
MEAN 2010
139.93333333333334 112.18662987131034
Standard deviation 2010
135.71320250194282 84.35813629737324
MEAN 2015
88.52222222222223 96.76817006572782
Standard deviation 2015
86.62446123324875 52.45304841713261
```




Year-2010
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5
72
24
76
127
88
395
144
308
289
249
213
98
83
40
28
9
12


Year-2015
Amount of rainfall
Ground truth
Prediction
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
65
92
23
113
266
75
104
200
160
133
158
72
15
131
0
32
1
19

1.12 Prediction Observations

1.12.1 Training on complete dataset

Algorithm	MAE
Linear Regression	94.94821727619338
SVR	127.74073860203839
Artificial neural nets	85.2648713528865

1.12.2 Training on telangana dataset

Algorithm	MAE
Linear Regression	70.61463829282977
SVR	90.30526775954294
Artificial neural nets	59.95190786532157

- Neural Networks performs better than SVR etc.
- Observed MAE is very high which indicates machine learning models won't work well for prediction of rainfall.
- Telangana data has a single pattern that can be learned by models, rather than learning different patterns of all states. so has high accuracy.
- Analysed individual year rainfall patterns for 2005, 2010, 2015.
- Approximately close means, noticed less standard deviations.

1.13 District wise details

- Similar to above the number of attributes is same, we don't have year in this.
- The amount of rainfall in mm for each district is added from 1950-2000.
- We analyse the data individually for the state **Andhra Pradesh**

```
In [32]: district = pd.read_csv("../data/district_wise_rainfall_normal.csv",sep=",")
         district = district.fillna(district.mean())
         district.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641 entries, 0 to 640
Data columns (total 19 columns):
STATE_UT_NAME    641 non-null object
DISTRICT         641 non-null object
JAN              641 non-null float64
FEB              641 non-null float64
MAR              641 non-null float64
APR              641 non-null float64
MAY              641 non-null float64
JUN              641 non-null float64
JUL              641 non-null float64
AUG              641 non-null float64
SEP              641 non-null float64
OCT              641 non-null float64
NOV              641 non-null float64
DEC              641 non-null float64
ANNUAL           641 non-null float64
```

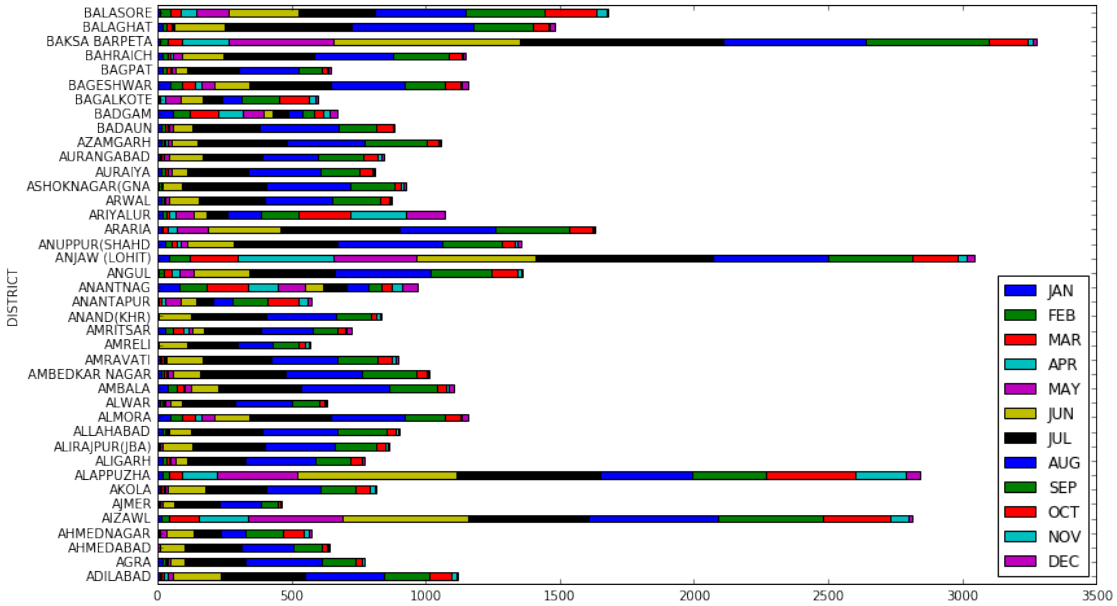
```
Jan-Feb          641 non-null float64
Mar-May          641 non-null float64
Jun-Sep          641 non-null float64
Oct-Dec          641 non-null float64
dtypes: float64(17), object(2)
memory usage: 95.2+ KB
```

In [33]: district.head()

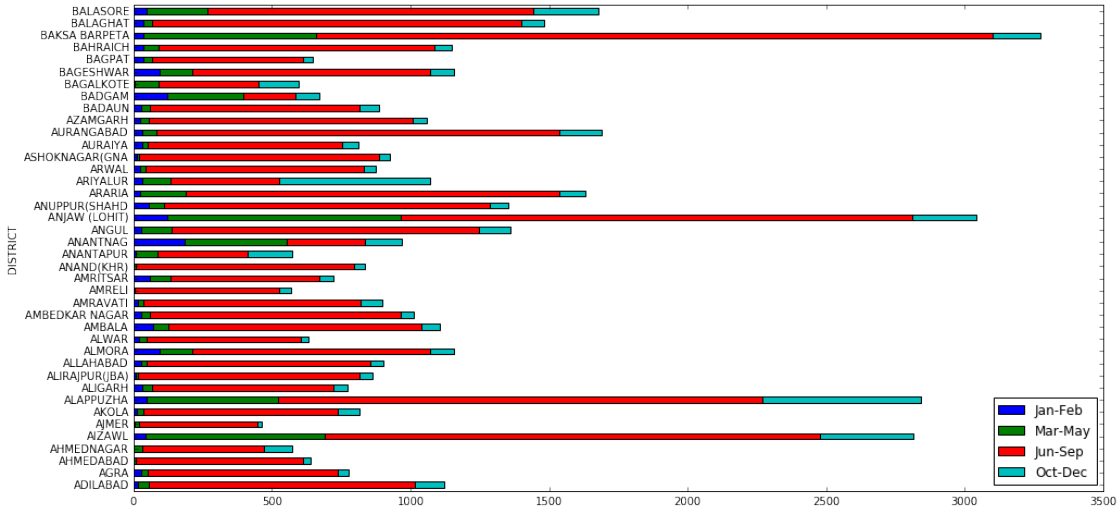
Out[33]:

	STATE_UT_NAME	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.0	315.2	250.9	2805.2	165.2	540.7	1207.2	892.1
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	301.2	275.8	128.3	3015.7	69.7	483.5	1757.2	705.3
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	276.1	198.6	100.0	2913.3	48.6	405.6	1884.4	574.7
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.1	34.1	29.8	3043.8	123.0	841.3	1848.5	231.0
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.9	29.5	31.7	4034.7	112.8	645.4	3008.4	268.1

In [34]:
```
district[['DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
          'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("DISTRICT").mean()[:40].plot.barh(stacked=Tr
```



```
In [35]: district[['DISTRICT', 'Jan-Feb', 'Mar-May',
    'Jun-Sep', 'Oct-Dec']].groupby("DISTRICT").sum()[:40].
                        plot.barh(stacked=True,figsize=(16,8));
```



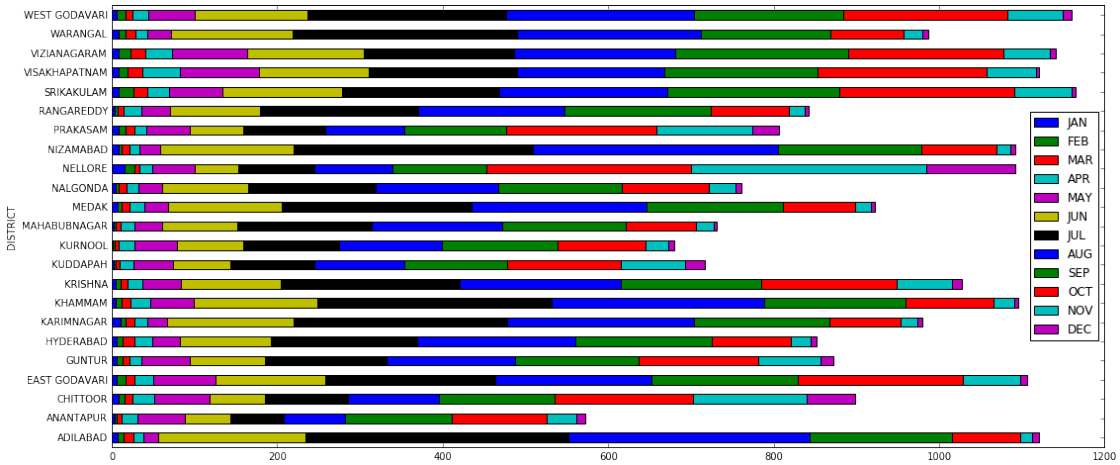
1.14 Observations

- The above two graphs shows the distribution of over each district.
- As there are large number of districts only 40 were shown in the graphs.

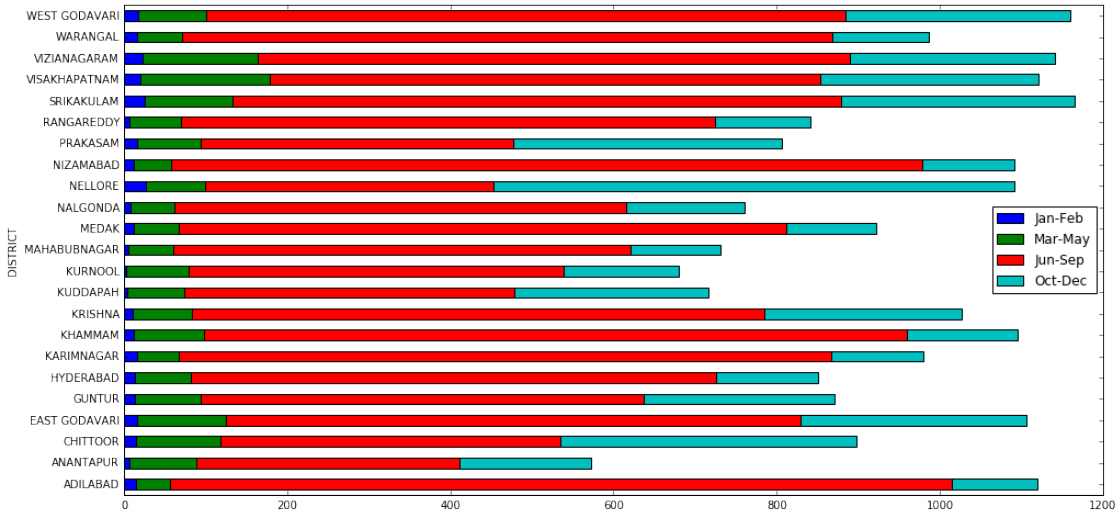
Andhra Pradesh Data

```
In [36]: ap_data = district[district['STATE_UT_NAME'] == 'ANDHRA PRADESH']
```

```
In [37]: ap_data[['DISTRICT', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
    'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("DISTRICT").mean()[:40].plot.barh(
                                        stacked=True,figsize=(18,8));
```



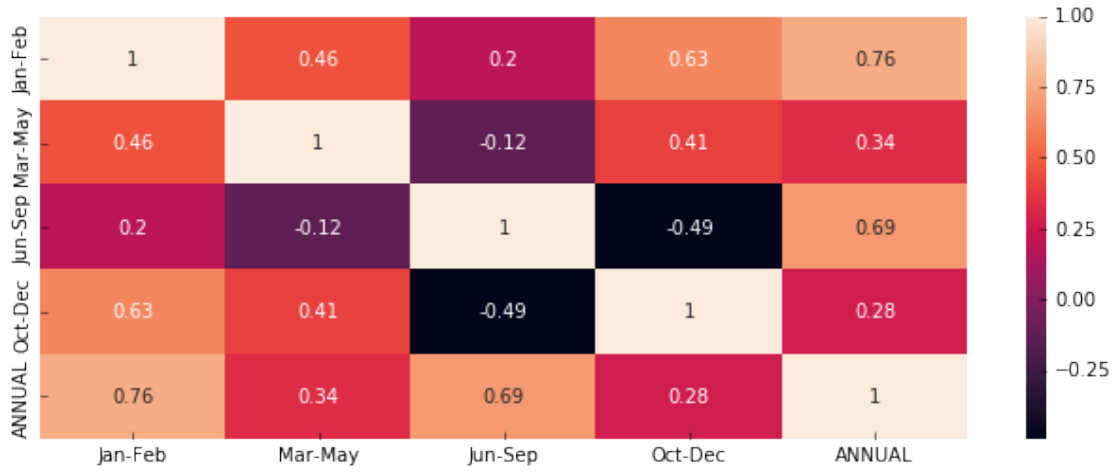

```
In [38]: ap_data[['DISTRICT', 'Jan-Feb', 'Mar-May',
    'Jun-Sep', 'Oct-Dec']].groupby("DISTRICT").sum()[:40].
                        plot.barh(stacked=True,figsize=(16,8));
```



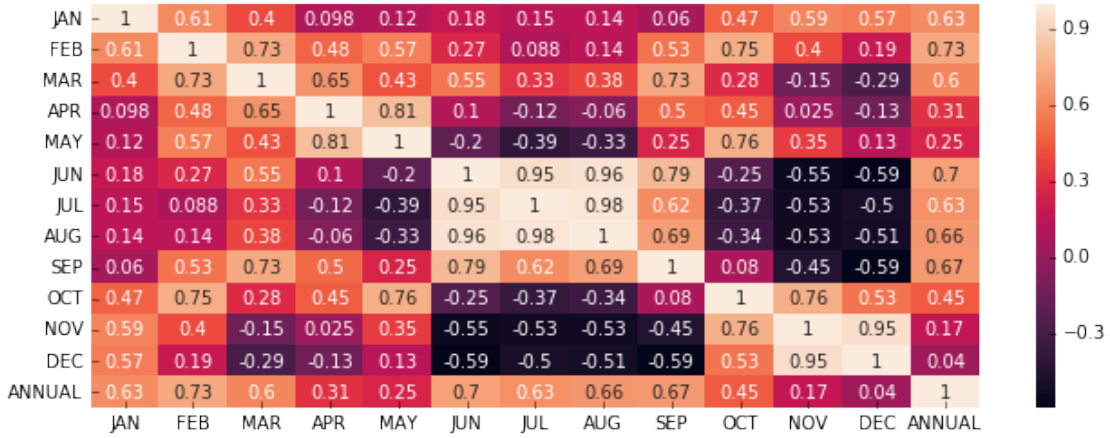
1.15 Observations

- The above two graphs shows the distribution of over each district in **Andhra Pradesh**.
- The above graphs show that more amount of rainfall is found in srikakulam district, least amount of rainfall is found in Anantapur district.
- It also shows that almost all states have more amount of rainfall have more amount of rainfall in the months june, july, september.

```
In [39]: plt.figure(figsize=(11,4))
         sns.heatmap(ap_data[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec','ANNUAL']].corr(),annot=True)
         plt.show()
```




```
In [40]: plt.figure(figsize=(11,4))
         sns.heatmap(ap_data[['JAN','FEB','
         MAR','APR','MAY','JUN','JUL','AUG','
         SEP','OCT','NOV','DEC','ANNUAL']].
         corr(),annot=True)plt.show()
```



1.16 Observations

- It is observed that in **Andhra Pradesh**, annual rainfall depends more in the months of january, febuary.
- It also shows that if there is rainfall in months march, april, may then there is less amount of rainfall
  in the months june, july, august, september.

1.17 Predictions

- We used the same types of models and evaluation metrics used for the above dataset.
- We also tested the amount of rainfall in hyderabad by models trained on complete dataset and andhra
  pradesh dataset.

```
In [41]: # testing and training for the complete data
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import mean_absolute_error

         division_data = np.asarray(district[['JAN', 'FEB', 'MAR', 'APR', 'MAY', '
JUN', 'JUL',
                'AUG', 'SEP', 'OCT', 'NOV', 'DEC']])

         X = None; y = None
         for i in range(division_data.shape[1]-3):
             if X is None:
                 X = division_data[:, i:i+3]
                 y = division_data[:, i+3]
             else:
                 X = np.concatenate((X, division_data[:, i:i+3]), axis=0)
                 y = np.concatenate((y, division_data[:, i+3]), axis=0)

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

In [42]:temp=district[['DISTRICT','JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG',
'SEP','OCT','NOV','DEC']].loc[district['STATE_UT_NAME']=='ANDHRAPRADESH']
 hyd = np.asarray(temp[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL','AUG', 'SEP
           ', 'OCT', 'NOV', 'DEC']].loc[temp['DISTRICT'] == 'HYDERABAD'])
         # print temp
         X_year = None; y_year = None
         for i in range(hyd.shape[1]-3):
             if X_year is None:
```

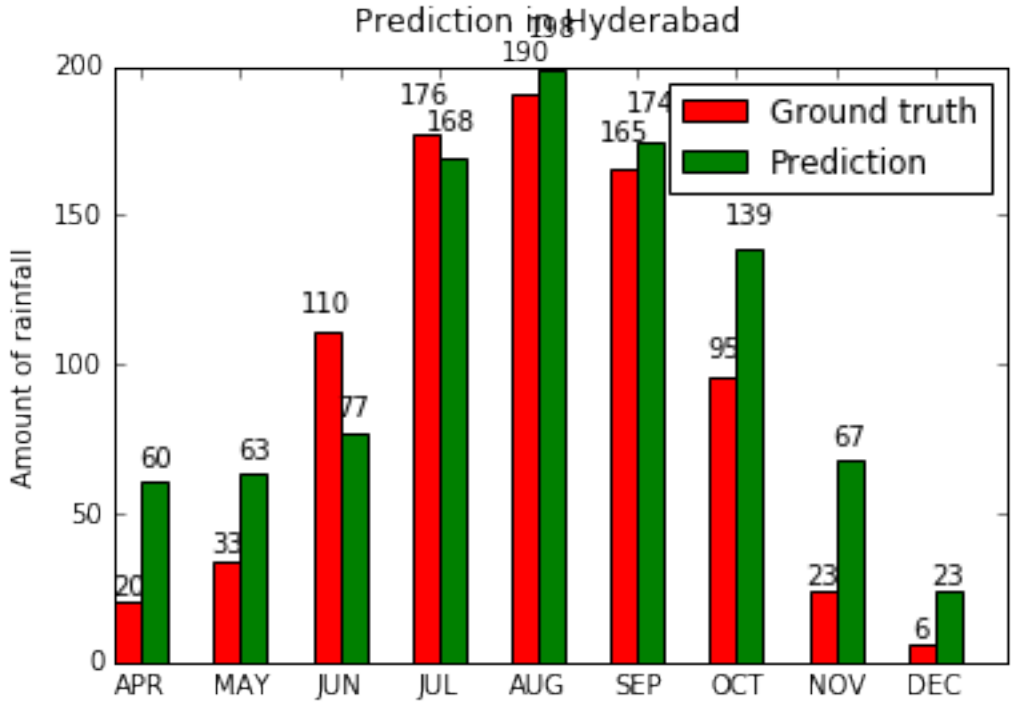
```
            X_year = hyd[:, i:i+3]
            y_year = hyd[:, i+3]
        else:
            X_year = np.concatenate((X_year, hyd[:, i:i+3]), axis=0)
            y_year = np.concatenate((y_year, hyd[:, i+3]), axis=0)
```

In [43]:
```
from sklearn import linear_model

# linear model
reg = linear_model.ElasticNet(alpha=0.5)
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

```
57.08862331011236
```

In [44]:
```
y_year_pred = reg.predict(X_year)
print "MEAN Hyderabad"
print np.mean(y_year),np.mean(y_year_pred)
print "Standard deviation hyderabad"
print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))

plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")
```

```
MEAN Hyderabad
91.48888888888888 108.20250522332894
Standard deviation hyderabad
69.2514651982091 58.90326979488754
```

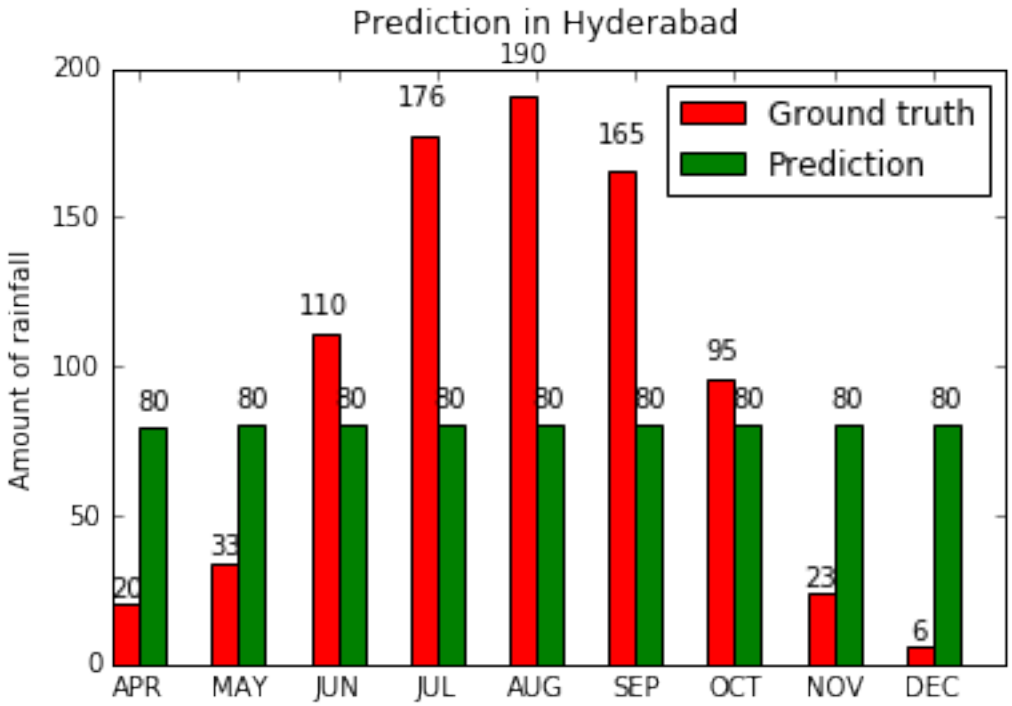


```
In [45]: from sklearn.svm import SVR

         # SVM model
         clf = SVR(gamma='auto', C=0.1, epsilon=0.2)
         clf.fit(X_train, y_train)
         y_pred = clf.predict(X_test)
         print mean_absolute_error(y_test, y_pred)
```

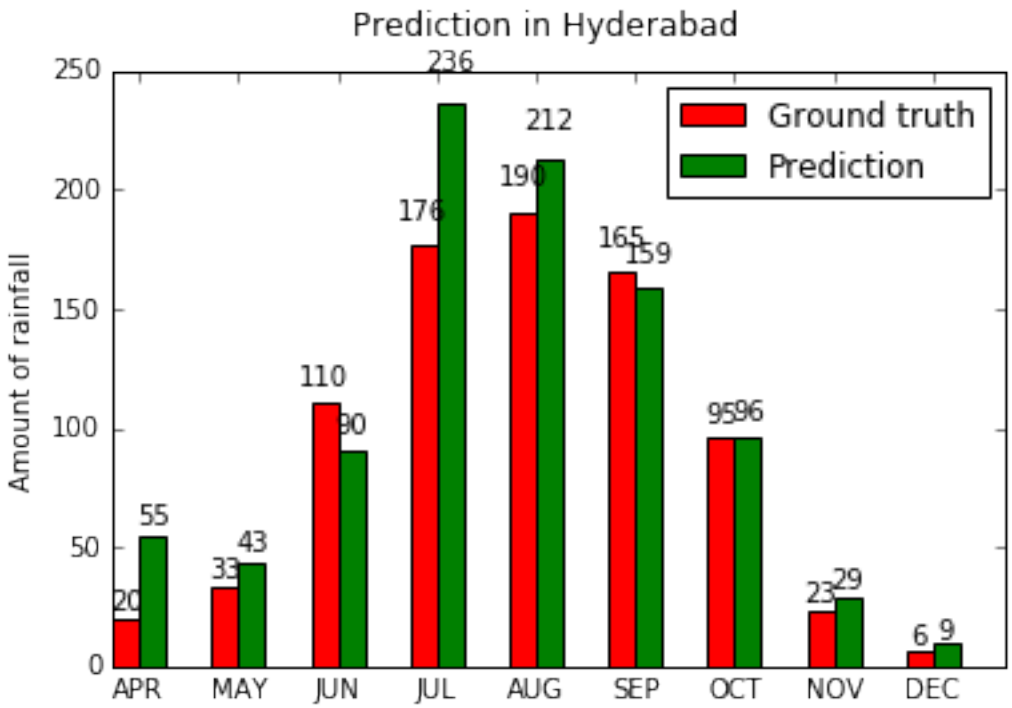
```
116.60671510825178
```

```
In [46]: y_year_pred = clf.predict(X_year)
         print "MEAN Hyderabad"
         print np.mean(y_year),np.mean(y_year_pred)
         print "Standard deviation hyderabad"
         print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))

         plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")
```

```
MEAN Hyderabad
91.48888888888888 80.34903236716154
Standard deviation hyderabad
69.2514651982091 0.14736007434982146
```



```
In [47]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=10, verbose=
1, validation_split=0.1, shuffle=True)
         y_pred = model.predict(np.expand_dims(X_test, axis=2))
         print mean_absolute_error(y_test, y_pred)

Train on 4153 samples, validate on 462 samples
Epoch 1/10
4153/4153 [==============================] - 0s 57us/step - loss: 6957.2028 - mean_absolute_error:
50.9468 - val_loss: 3784.2466 - val_mean_absolute_error: 38.6021Epoch 2/10
4153/4153 [==============================] - 0s 78us/step - loss: 5258.5688 - mean_absolute_error:
43.7378 - val_loss: 3618.7178 - val_mean_absolute_error: 37.8309Epoch 3/10
4153/4153 [==============================] - 0s 56us/step - loss: 5122.4481 - mean_absolute_error:
42.5880 - val_loss: 4047.8249 - val_mean_absolute_error: 37.8983Epoch 4/10
4153/4153 [==============================] - 0s 54us/step - loss: 5036.6459 - mean_absolute_error:
42.0330 - val_loss: 3693.8650 - val_mean_absolute_error: 37.4953Epoch 5/10
4153/4153 [==============================] - 0s 53us/step - loss: 4964.9985 - mean_absolute_error:
41.4354 - val_loss: 3532.0074 - val_mean_absolute_error: 35.9146Epoch 6/10
4153/4153 [==============================] - 0s 54us/step - loss: 5006.1478 - mean_absolute_error:
41.7652 - val_loss: 3535.7351 - val_mean_absolute_error: 36.3146Epoch 7/10
4153/4153 [==============================] - 0s 54us/step - loss: 4917.7239 - mean_absolute_error:
41.2176 - val_loss: 3659.4433 - val_mean_absolute_error: 37.9144Epoch 8/10
4153/4153 [==============================] - 0s 55us/step - loss: 4861.6742 - mean_absolute_error:
41.1526 - val_loss: 3432.5417 - val_mean_absolute_error: 35.8548Epoch 9/10
4153/4153 [==============================] - 0s 55us/step - loss: 4792.0339 - mean_absolute_error:
40.5132 - val_loss: 3964.3087 - val_mean_absolute_error: 37.4906Epoch 10/10
4153/4153 [==============================] - 0s 54us/step - loss: 4796.9758 - mean_absolute_error:
40.5613 - val_loss: 3627.6266 - val_mean_absolute_error: 35.768042.14205056426844


In [48]: y_year_pred = model.predict(np.expand_dims(X_year, axis=2))
         print "MEAN Hyderabad"
         print np.mean(y_year),np.mean(y_year_pred)
         print "Standard deviation hyderabad"
         print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))


         plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")

MEAN Hyderabad
91.48888888888888 103.67171
Standard deviation hyderabad
69.2514651982091 76.83028
```



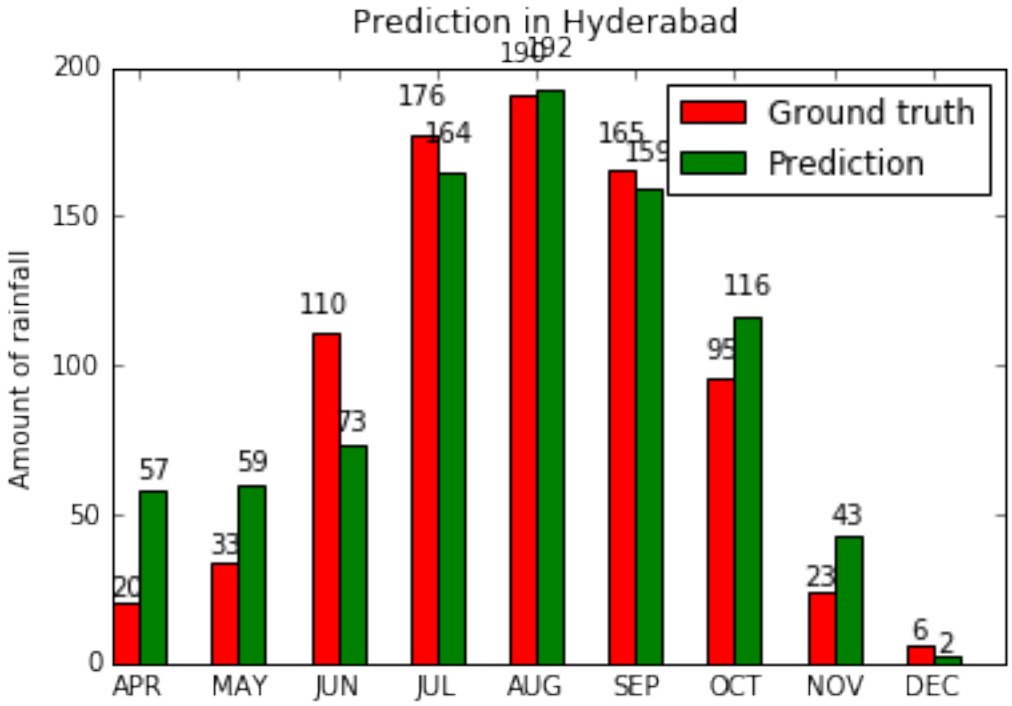
In [49]:
```
# training and testing sets for only andhra pradesh data
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

division_data = np.asarray(ap_data[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']])

X = None; y = None
for i in range(division_data.shape[1]-3):
    if X is None:
        X = division_data[:, i:i+3]
        y = division_data[:, i+3]
    else:
        X = np.concatenate((X, division_data[:, i:i+3]), axis=0)
        y = np.concatenate((y, division_data[:, i+3]), axis=0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [50]:
```
from sklearn import linear_model

# linear model
reg = linear_model.ElasticNet(alpha=0.5)
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

```
31.249748674622477
```

In [51]:
```
y_year_pred = reg.predict(X_year)
print "MEAN Hyderabad"
print np.mean(y_year),np.mean(y_year_pred)
print "Standard deviation hyderabad"
print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))
plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")
```

```
MEAN Hyderabad
91.48888888888888 96.54891993068443
Standard deviation hyderabad
69.2514651982091 60.819355195446896
```

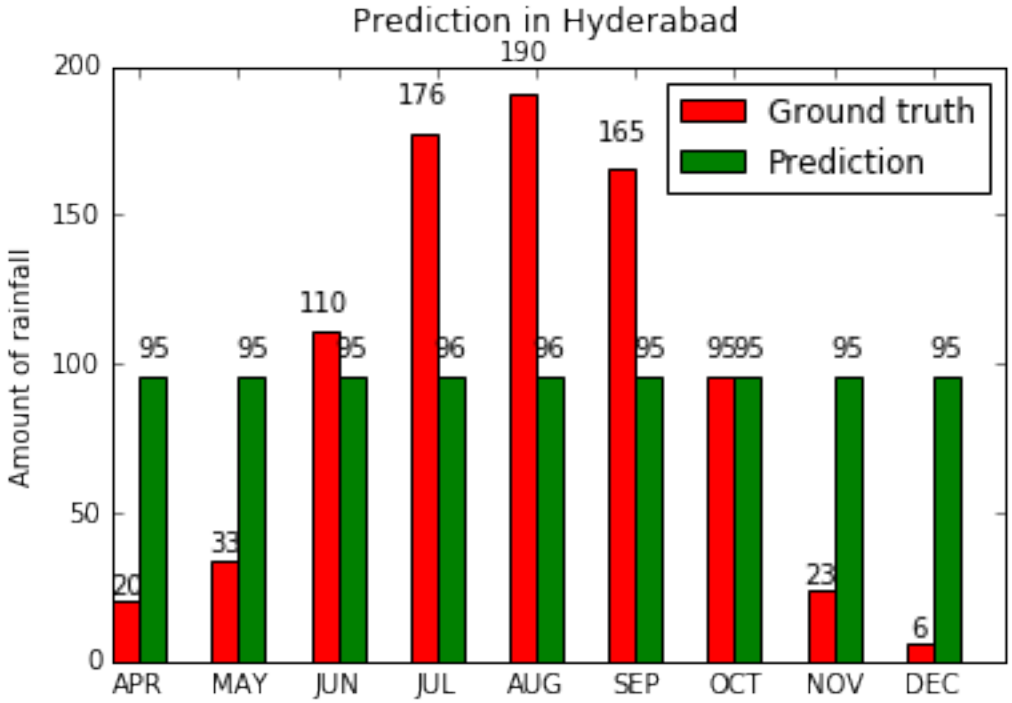


In [52]:
```
from sklearn.svm import SVR

# SVM model
clf = SVR(gamma='auto', C=0.1, epsilon=0.2)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

```
59.35057496896855
```

In [53]:
```
y_year_pred = clf.predict(X_year)
print "MEAN Hyderabad"
print np.mean(y_year),np.mean(y_year_pred)
print "Standard deviation hyderabad"
print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))
plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")
```

```
MEAN Hyderabad
91.48888888888888 95.89978206795146
Standard deviation hyderabad
69.2514651982091 0.09247315036320868
```

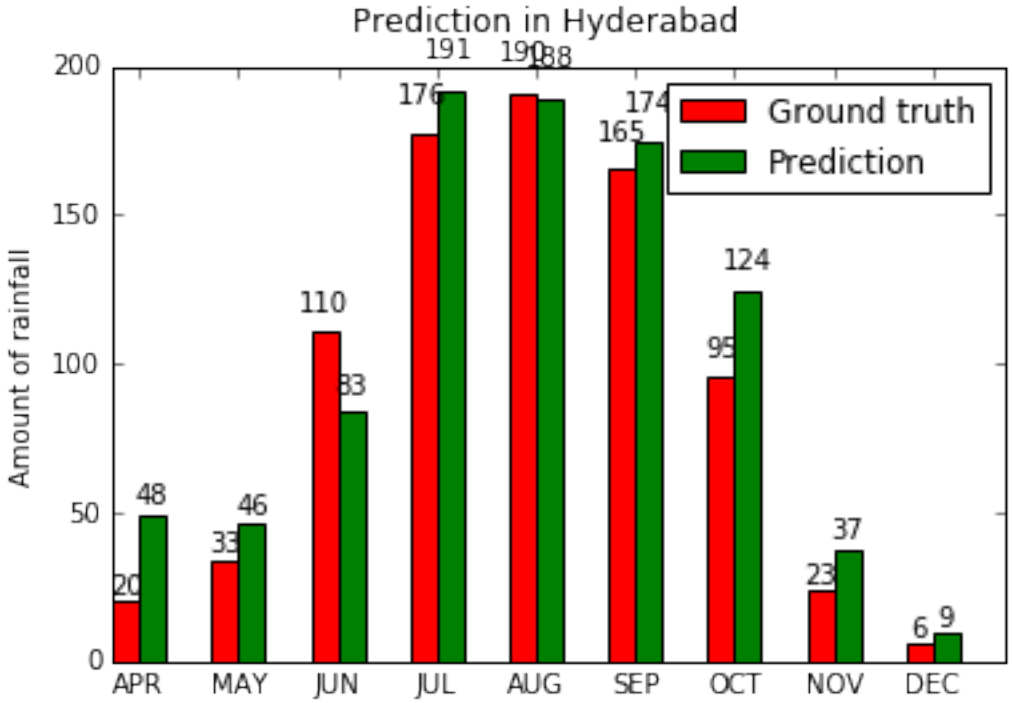



```
In [54]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_
size=64, epochs=10, verbose=1, validation_split=0.1, shuffle=True)
         y_pred = model.predict(np.expand_dims(X_test, axis=2))
         print mean_absolute_error(y_test, y_pred)

Train on 148 samples, validate on 17 samples
Epoch 1/10
148/148 [==============================] - 0s 120us/step - loss: 1778.6490
- mean_absolute_error: 30.9451 - val_loss: 1446.8748 - val_mean_absolute_
error: 27.5400Epoch 2/10
148/148 [==============================] - 0s 91us/step - loss: 1664.5006
- mean_absolute_error: 29.9809 - val_loss: 1278.2192 - val_mean_absolute_
error: 24.9887Epoch 3/10
148/148 [==============================] - 0s 92us/step - loss: 1504.4134
- mean_absolute_error: 28.0781 - val_loss: 1205.8508 - val_mean_absolute_
error: 24.3124Epoch 4/10
148/148 [==============================] - 0s 89us/step - loss: 1405.6685
- mean_absolute_error: 26.9898 - val_loss: 1207.5862 - val_mean_absolute_
error: 25.0976Epoch 5/10
148/148 [==============================] - 0s 84us/step - loss: 1399.6531
- mean_absolute_error: 26.8409 - val_loss: 1229.2842 - val_mean_absolute_
error: 26.1517Epoch 6/10
148/148 [==============================] - 0s 83us/step - loss: 1364.1109
- mean_absolute_error: 26.3740 - val_loss: 1223.0768 - val_mean_absolute_
error: 25.0201Epoch 7/10
148/148 [==============================] - 0s 79us/step - loss: 1321.5538
- mean_absolute_error: 26.1438 - val_loss: 1237.3645 - val_mean_absolute_
error: 23.8087Epoch 8/10
148/148 [==============================] - 0s 83us/step - loss: 1302.3495
- mean_absolute_error: 26.0448 - val_loss: 1252.1573 - val_mean_absolute_
error: 23.0894Epoch 9/10
```

```
148/148 [==============================] - 0s 81us/step - loss: 1290.8667
- mean_absolute_error: 25.8581 - val_loss: 1240.2283 - val_mean_absolute_
error: 22.2094Epoch 10/10
148/148 [==============================] - 0s 82us/step - loss: 1273.7824
- mean_absolute_error: 25.5667 - val_loss: 1211.7397 - val_mean_absolute_
error: 21.345532.25840494065058
```

```
In [55]: y_year_pred = model.predict(np.expand_dims(X_year, axis=2))
         print "MEAN Hyderabad"
         print np.mean(y_year),np.mean(y_year_pred)
         print "Standard deviation hyderabad"
         print np.sqrt(np.var(y_year)),np.sqrt(np.var(y_year_pred))
         plot_graphs(y_year,y_year_pred,"Prediction in Hyderabad")
```

```
MEAN Hyderabad
91.48888888888888 100.606964
Standard deviation hyderabad
69.2514651982091 66.957054
```



1.18 Prediction Observations

1.18.1 Training on complete dataset

Algorithm	MAE
Linear Regression	57.08862331011236
SVR	116.60671510825178
Artificial neural nets	44.329664907381066

1.18.2 Training on telangana dataset

Algorithm	MAE
Linear Regression	31.249748674622477
SVR	59.35057496896855
Artificial neural nets	31.0601823988415

- Neural Networks performs better than SVR etc.
- Bad performance by SVR model.
- Andhra Pradesh data has a single pattern that can be learned by models, rather than learning different patterns of all states. so has high accuracy.
- Analysed individual year rainfall patterns for Hyderabad district.
- Approximately close means, noticed close standard deviations.

1.19 Conclusions

- Various visualizations of data are observed which helps in implementing the approaches for prediction.
- Prediction of amount of rainfall for both the types of dataset.
- Observations indicates machine learning models won't work well for prediction of rainfall due to fluctutaions in rainfall.

1.20 Technologies

- Programming language : *Python*
- Libraries : *numpy*, *pandas*, *matplotlib*, *seaborn*, *keras*, *scipy*, *sklearn*