kernel

Akhil Ralla, Sudheer Achary

November 18, 2018

1 India Rainfall Analysis

1.1 Motivation and Description

Monsoon prediction is clearly of great importance for India.Two types of rainfall predictions can be done, They are - Long term predictions: Predict rainfall over few weeks/months in advance. - Short term predictions: Predict rainfall a few days in advance in specific locations.

Indian meteorological department provides forecasting data required for project. In this project we are planning to work on long term predictions of rainfall. The main motive of the project is to predict the amount of rainfall in a particular division or state well in advance. We predict the amount of rainfall using past data.

1.2 Dataset

- Dataset1(dataset1) This dataset has average rainfall from 1951-2000 for each district, for every month.
- Dataset2(dataset2) This dataset has average rainfall for every year from 1901-2015 for each state.

1.3 Methodology

- Converting data in to the correct format to conduct experiments.
- Make a good analysis of data and observe variation in the patterns of rainfall.
- Finally, we try to predict the average rainfall by separating data into training and testing. We apply various statistical and machine learning approaches(*SVM*, etc) in prediction and make analysis over various approaches. By using various approaches we try to minimize the error.

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import matplotlib.pyplot as plt
        import seaborn as sns
```

1.4 Types of graphs

- Bar graphs showing distribution of amount of rainfall.
- Distribution of amount of rainfall yearly, monthly, groups of months.
- Distribution of rainfall in subdivisions, districts form each month, groups of months.
- Heat maps showing correlation between amount of rainfall between months.

```
In [2]: data = pd.read_csv("../data/rainfall_in_india_1901-2015.csv",sep=",")
        data = data.fillna(data.mean())
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
```

```
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  \
        0  365.1  481.1  332.6  388.5  558.2   33.6  3373.2    136.3    560.3
        1  228.9  753.7  666.2  197.2  359.0  160.5  3520.7    159.8    458.3
        2  728.4  326.7  339.0  181.2  284.4  225.0  2957.4    156.7    236.1
        3  502.0  160.1  820.4  222.2  308.7   40.1  3079.6     24.1    506.9
        4  368.7  330.5  297.0  260.7   25.4  344.7  2566.7      1.3    309.7

           Jun-Sep  Oct-Dec
        0   1696.3    980.3
        1   2185.9    716.7
        2   1874.0    690.6
        3   1977.6    571.0
        4   1624.9    630.8
```

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

Out[4]:

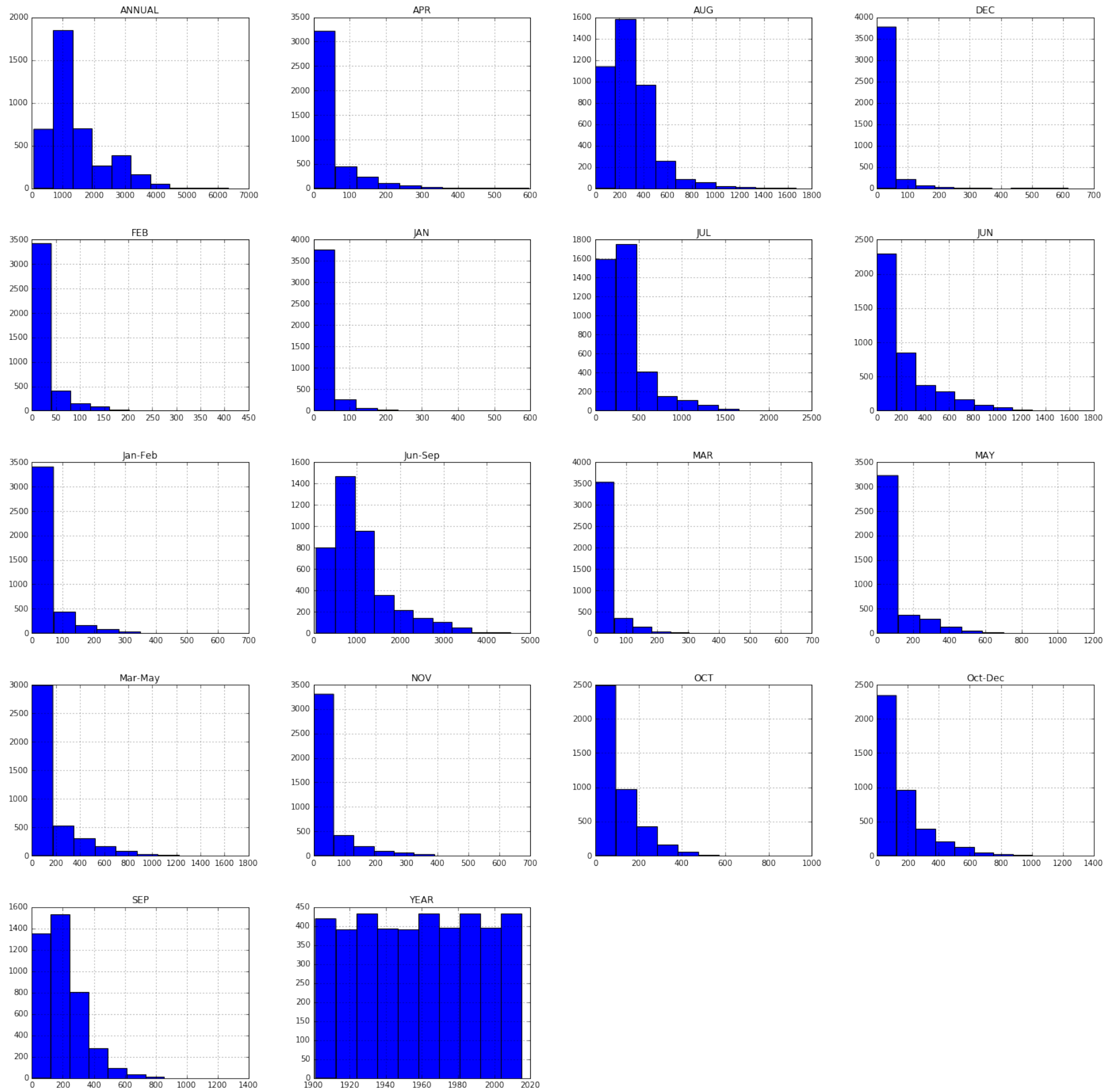
	YEAR	JAN	FEB	MAR	APR
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000
mean	1958.218659	18.957320	21.805325	27.359197	43.127432
std	33.140898	33.569044	35.896396	46.925176	67.798192
min	1901.000000	0.000000	0.000000	0.000000	0.000000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000
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

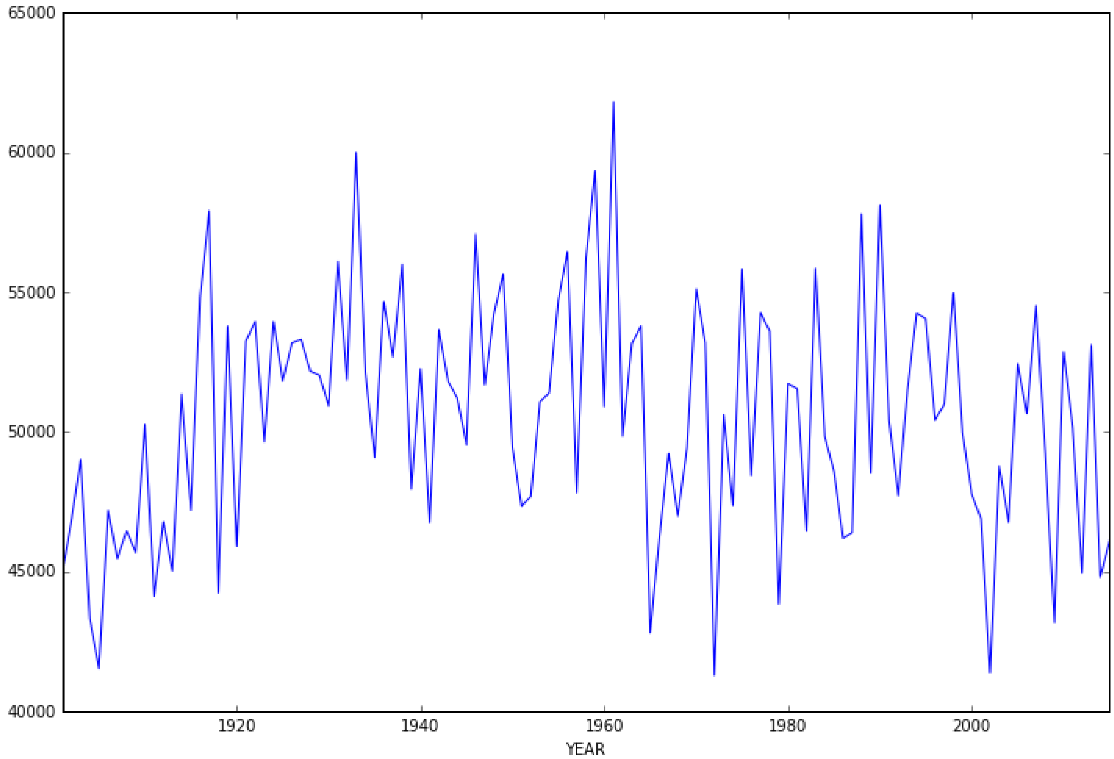
In [5]:
```
data.hist(figsize=(24,24));
```



1.6 Observations

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

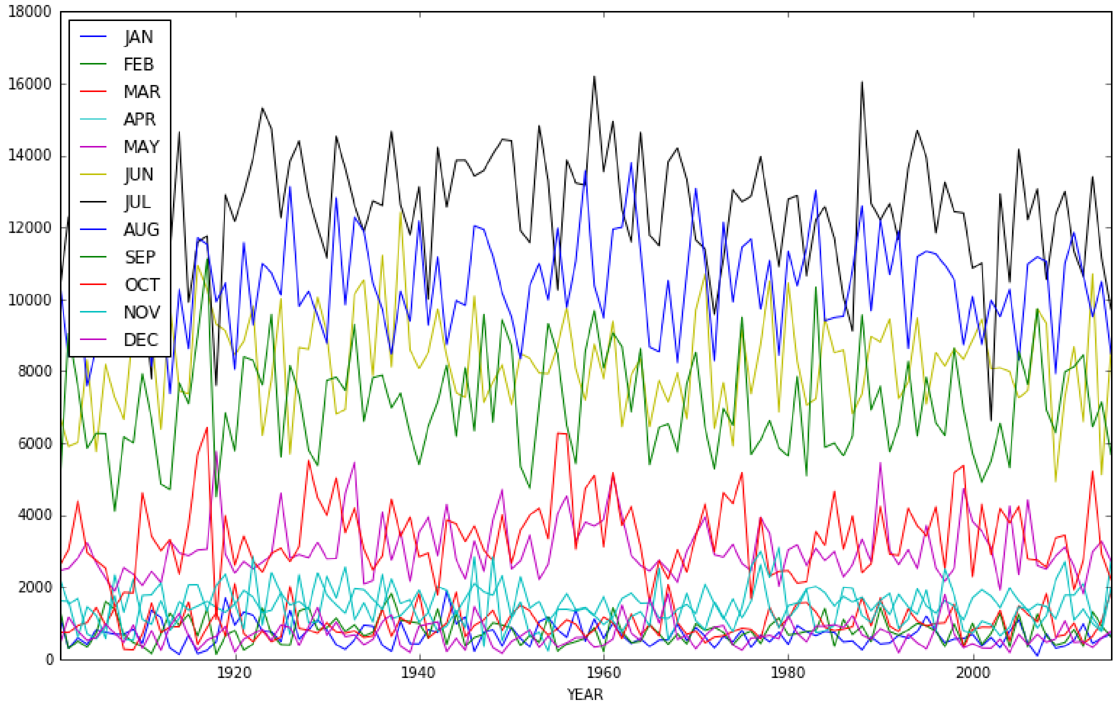
```
In [6]: data.groupby("YEAR").sum()['ANNUAL'].plot(figsize=(12,8));
```



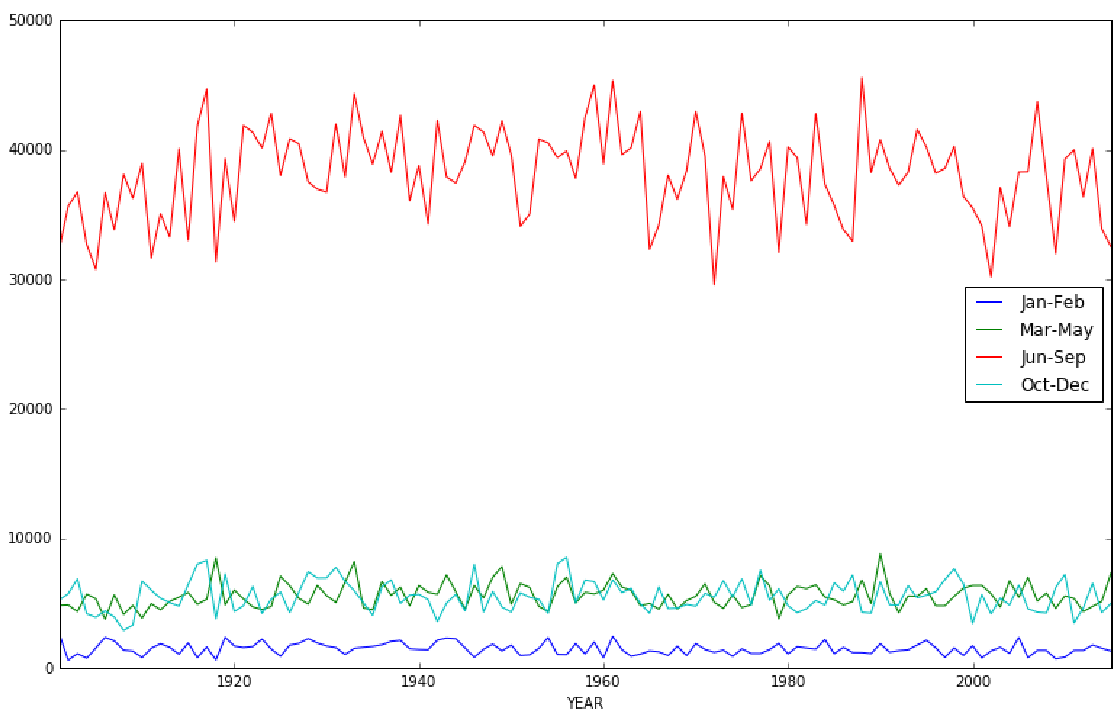
1.7 Observations

- 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));
```



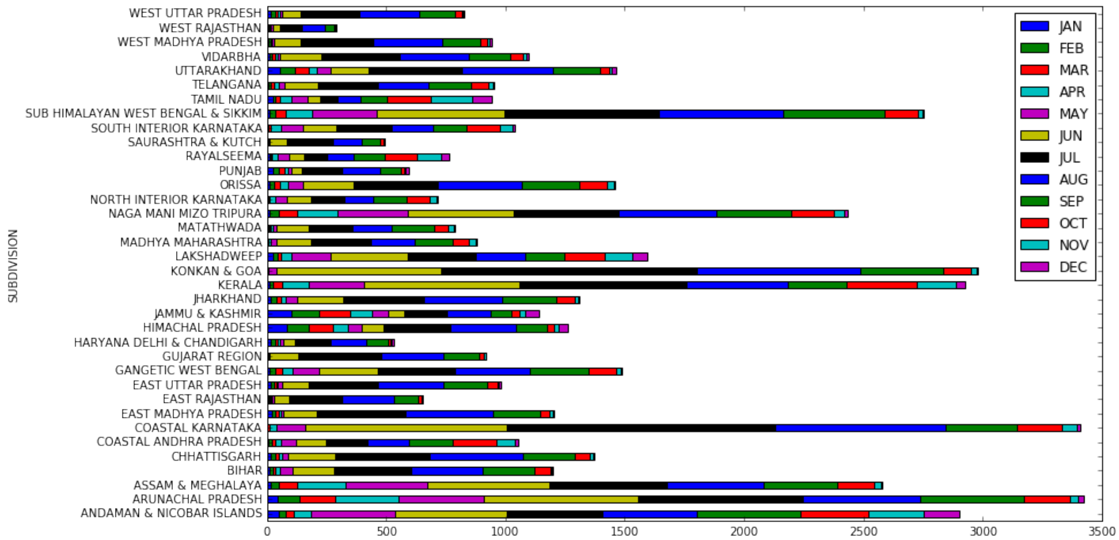

```
In [8]: data[['YEAR','Jan-Feb', 'Mar-May',
         'Jun-Sep', 'Oct-Dec']].groupby("YEAR").sum().plot(figsize=(13,8));
```



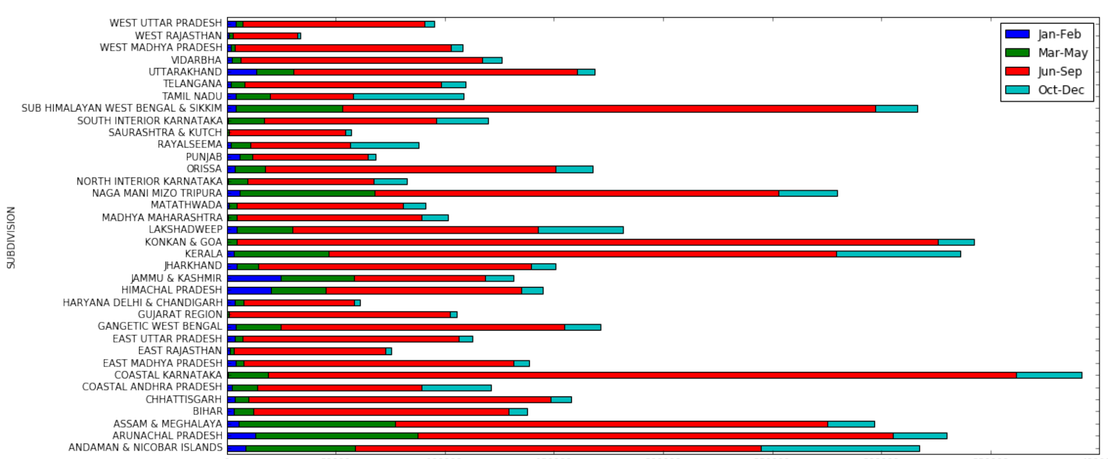
1.8 Observations

- 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
```

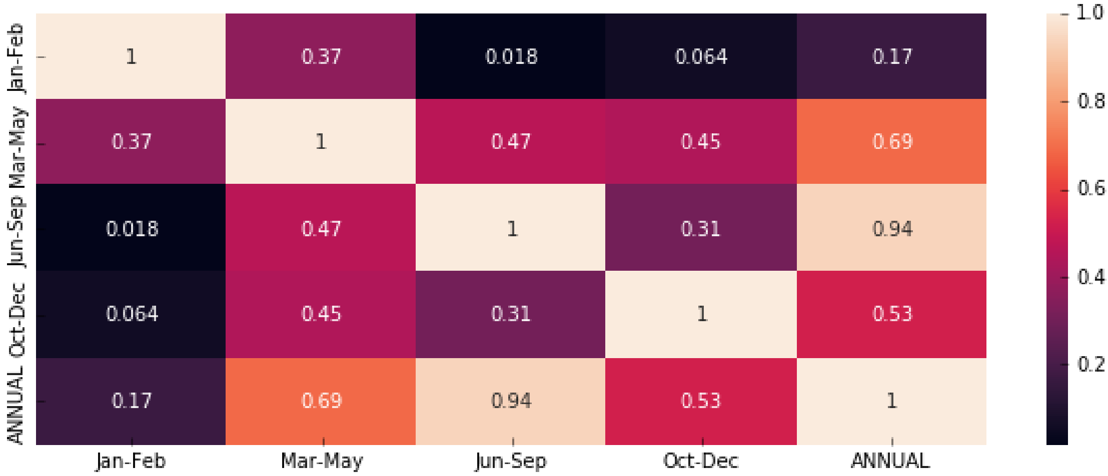


1.9 Observations

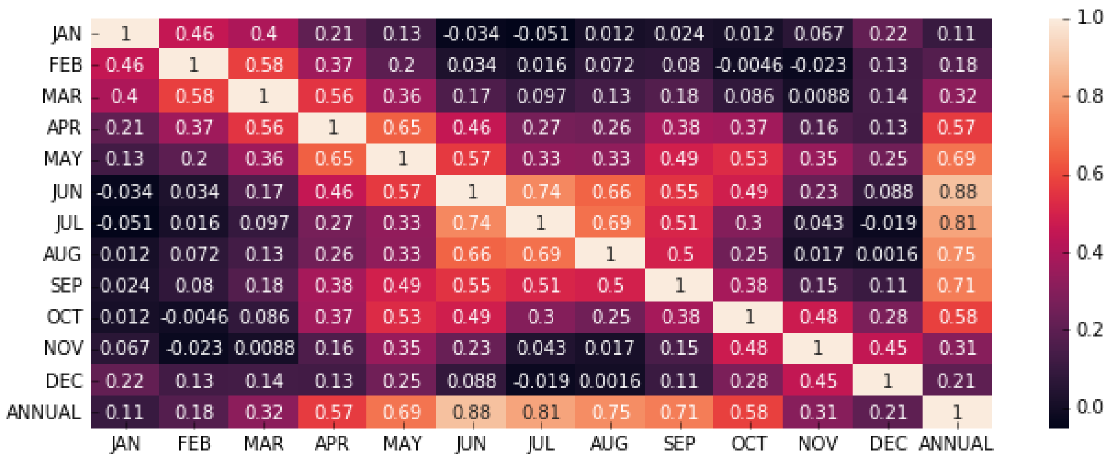
- Above two graphs shows that the amount of rainfall is reasonably good in the months of march, april, may in eastern India.

```
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/sudheer.achary/.local/lib/python2.7/site-packages/pandas/core/computation/check.py:17: UserWarning
The 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','ANNU
         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)
```

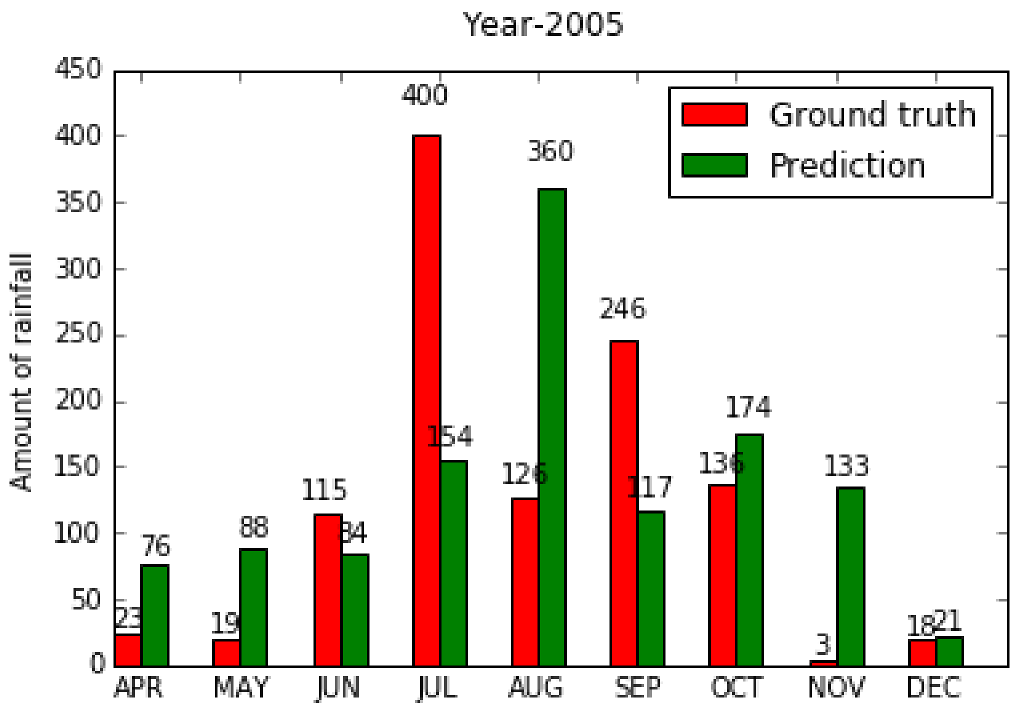
In [18]:
```
from sklearn import linear_model

# 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)
```

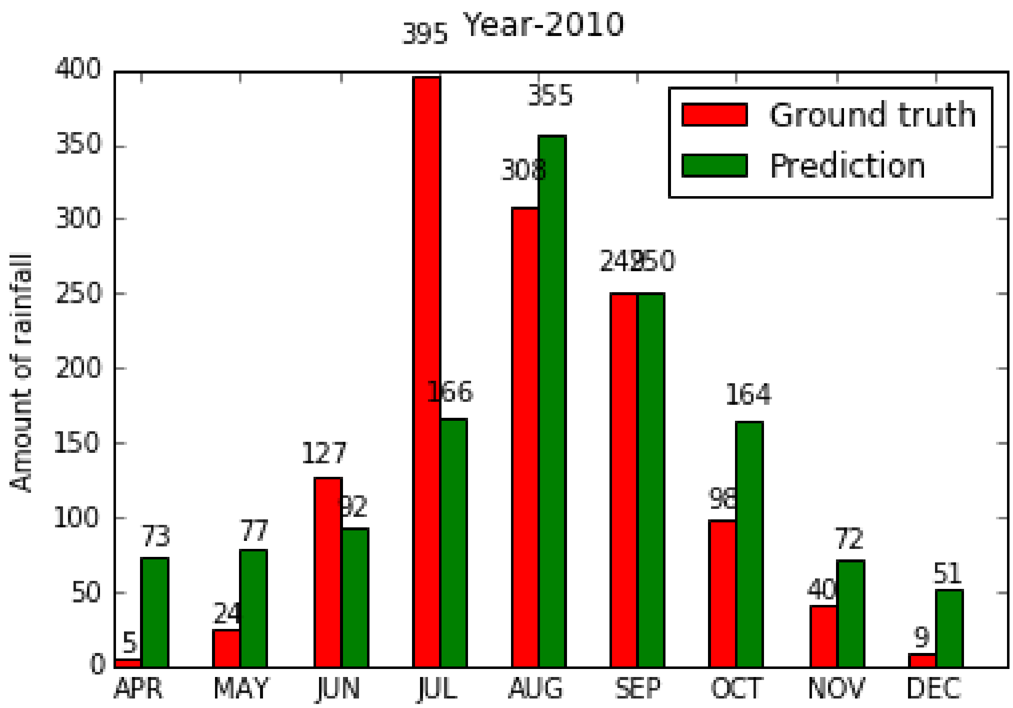
```
96.32435229744095
```

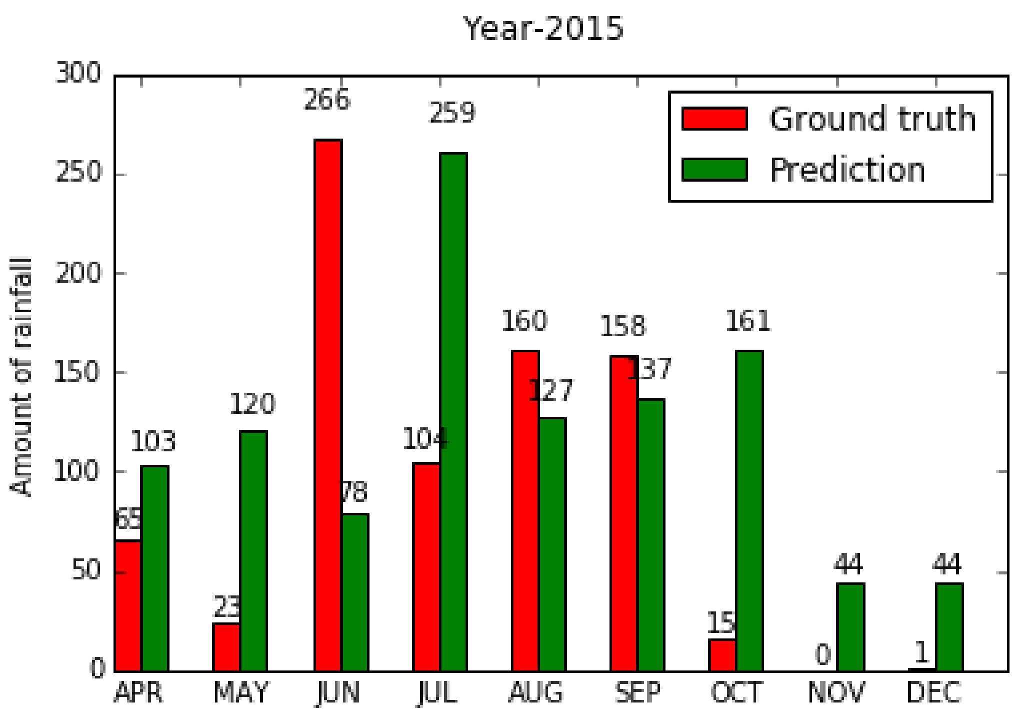
In [19]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)


#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
```







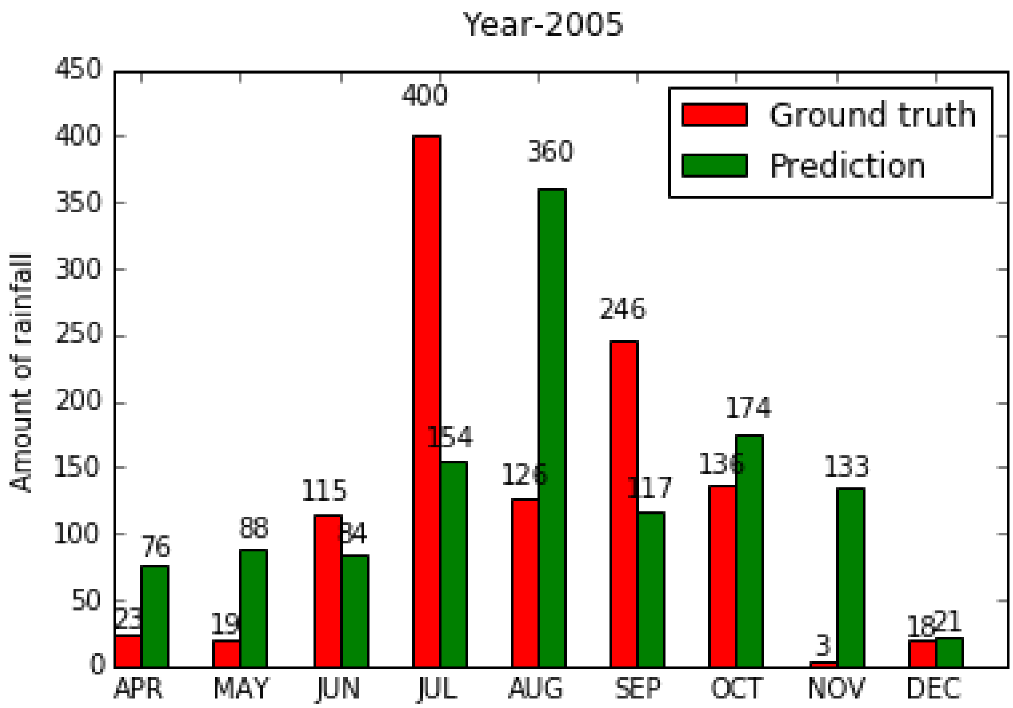

```
In [20]: from sklearn.svm import SVR
```

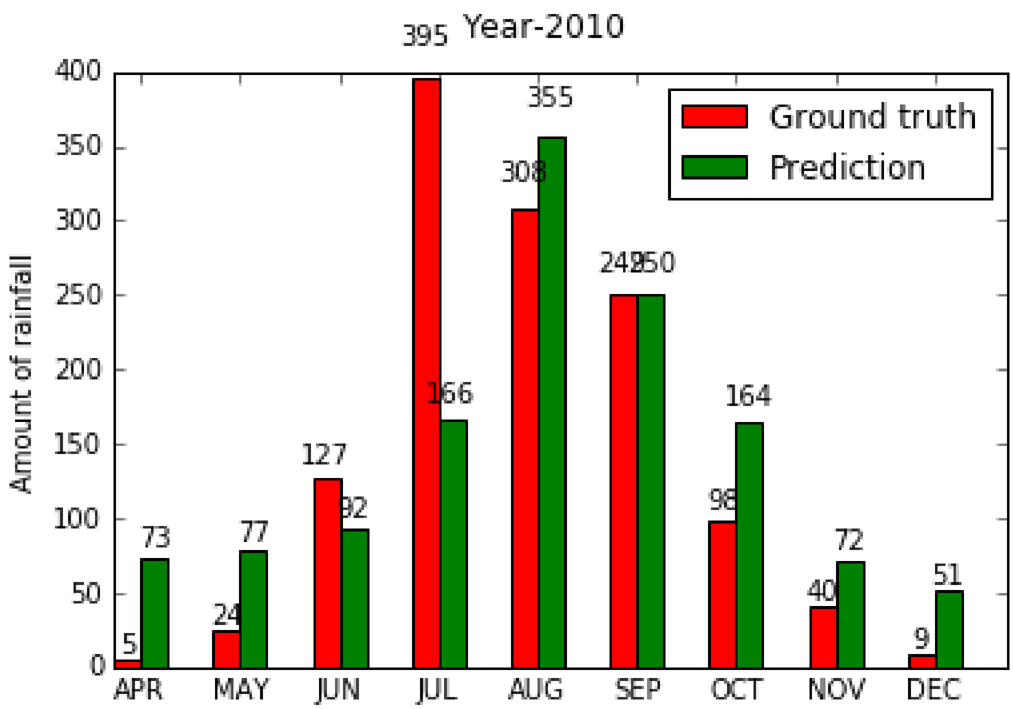
```
# SVM model
clf = SVR(gamma='auto', C=0.1, epsilon=0.2)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print mean_absolute_error(y_test, y_pred)
```

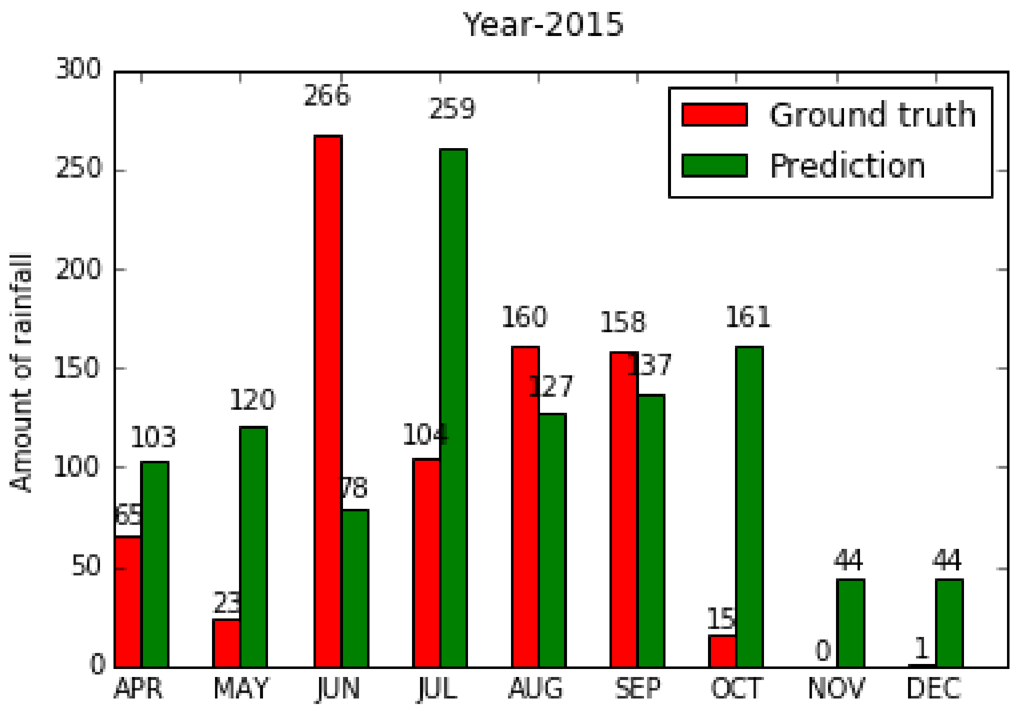
```
127.1600615632603
```

```
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
Ground truth
Prediction
Amount of rainfall
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
Ground truth
Prediction
Amount of rainfall
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5
73
24
77
127
92
395
166
308
355
249
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/sudheer.achary/.local/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion o
  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
_________________________________________________________________
```

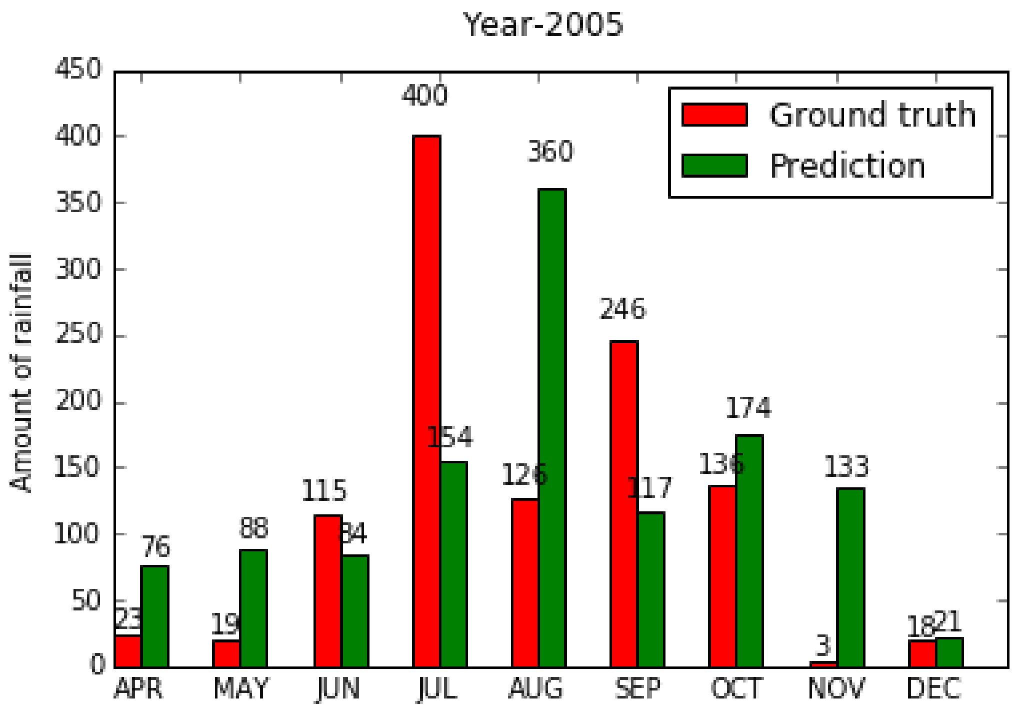
In [23]:
```
model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=10, verbose=1, va
y_pred = model.predict(np.expand_dims(X_test, axis=2))
print mean_absolute_error(y_test, y_pred)
```

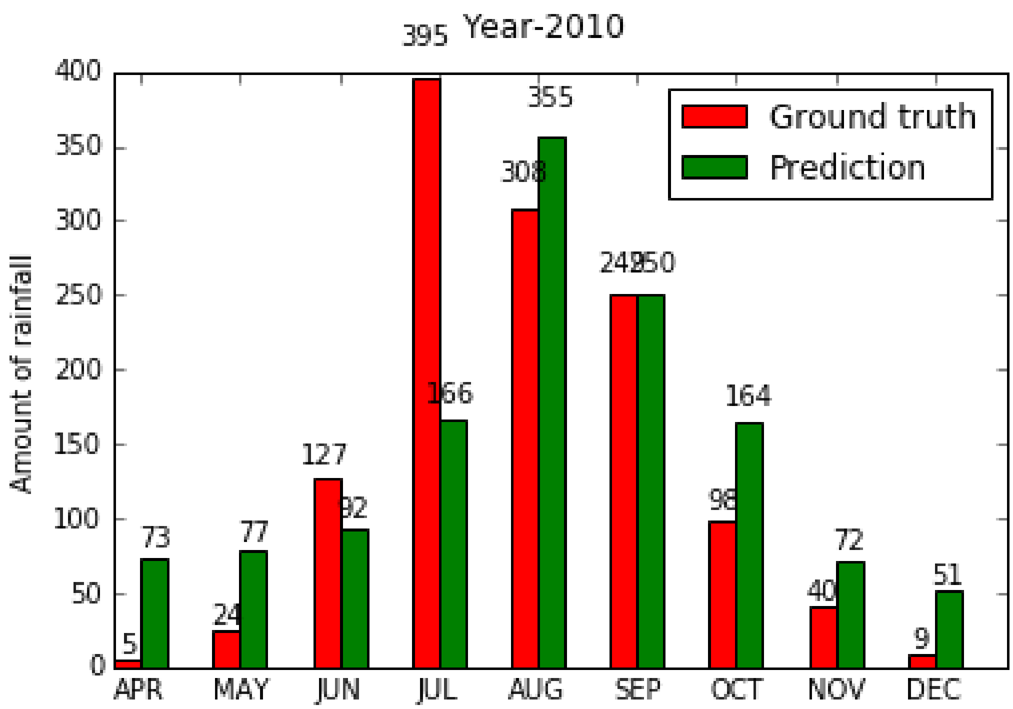
```
Train on 30005 samples, validate on 3334 samples
Epoch 1/10
30005/30005 [==============================] - 3s 111us/step - loss: 19589.7591 - mean_absolute_error: 8
Epoch 2/10
30005/30005 [==============================] - 2s 53us/step - loss: 18582.2211 - mean_absolute_error: 86
Epoch 3/10
30005/30005 [==============================] - 2s 54us/step - loss: 18466.6604 - mean_absolute_error: 86
Epoch 4/10
30005/30005 [==============================] - 2s 54us/step - loss: 18482.5326 - mean_absolute_error: 86
Epoch 5/10
30005/30005 [==============================] - 2s 54us/step - loss: 18358.7726 - mean_absolute_error: 86
Epoch 6/10
30005/30005 [==============================] - 2s 53us/step - loss: 18312.5666 - mean_absolute_error: 85
Epoch 7/10
30005/30005 [==============================] - 2s 54us/step - loss: 18236.9615 - mean_absolute_error: 85
Epoch 8/10
30005/30005 [==============================] - 2s 54us/step - loss: 18118.3601 - mean_absolute_error: 85
Epoch 9/10
30005/30005 [==============================] - 2s 54us/step - loss: 18193.9362 - mean_absolute_error: 85
Epoch 10/10
30005/30005 [==============================] - 2s 54us/step - loss: 18007.9055 - mean_absolute_error: 84
92.28250624049363
```

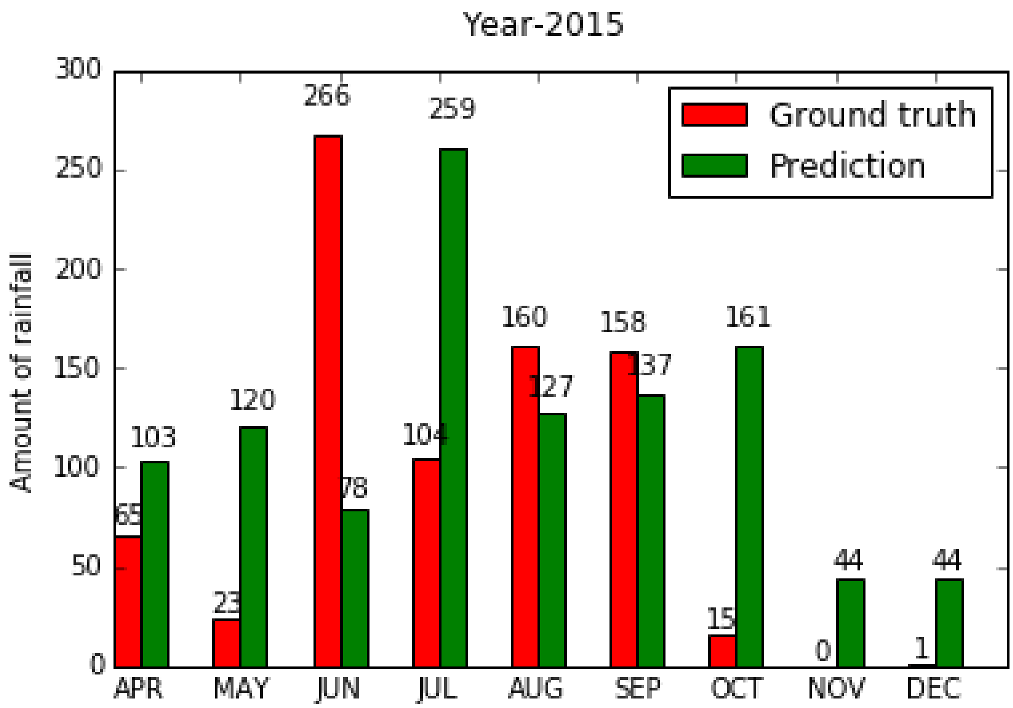
In [24]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)

#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
395
166
308
355
249
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)
```

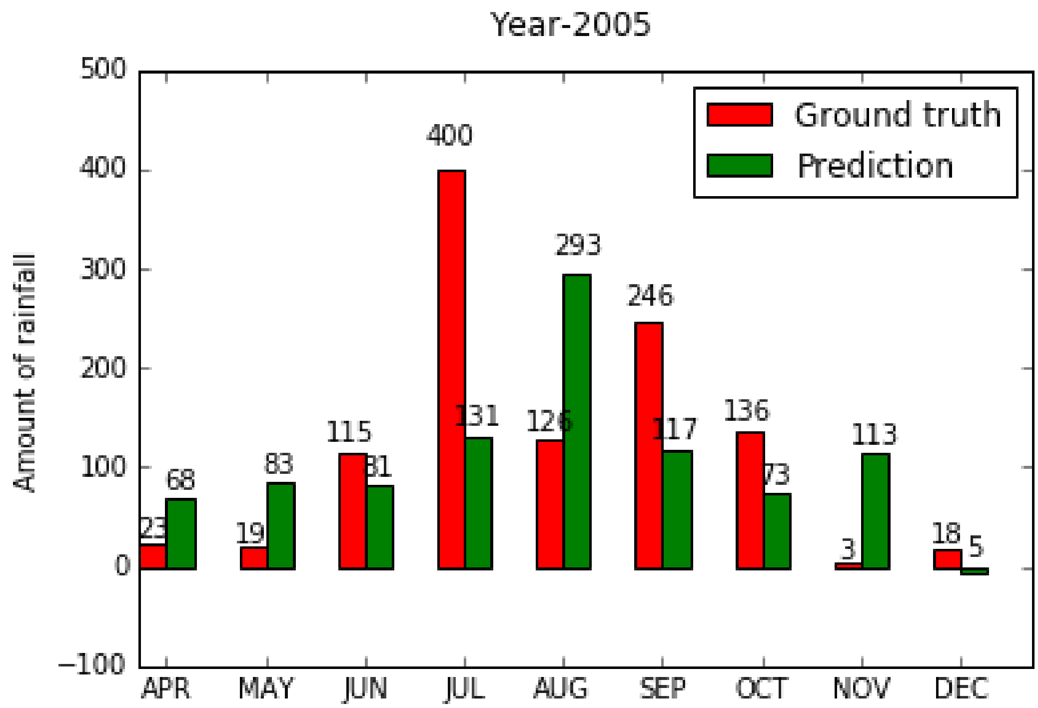
```
In [26]: from sklearn import linear_model

         # 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)
```

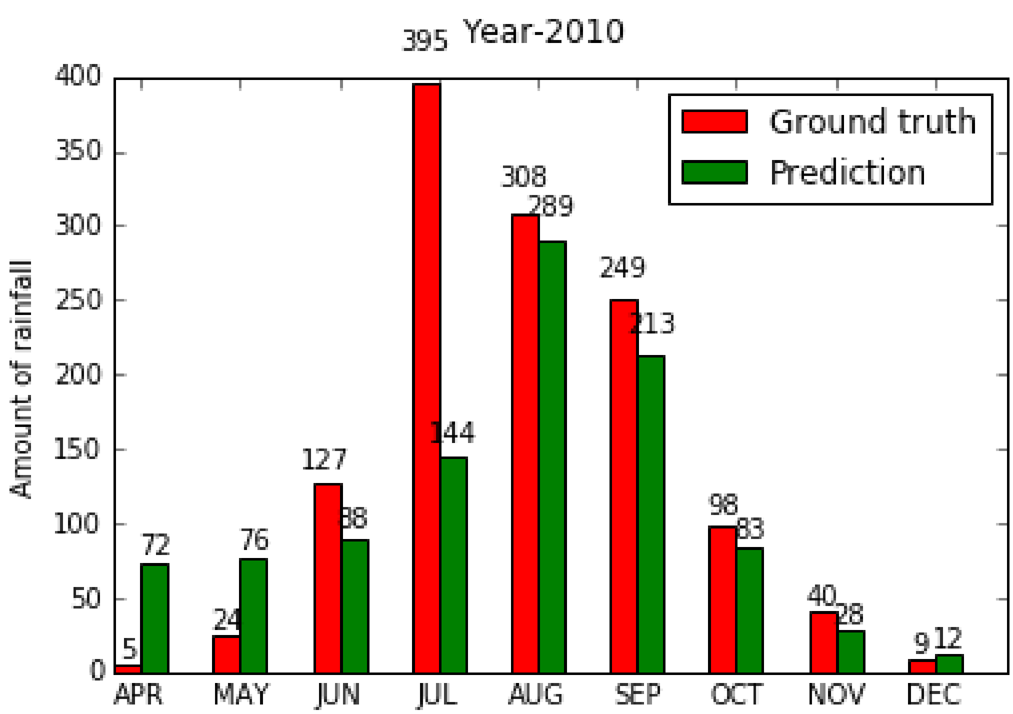
```
64.72601914484643
```

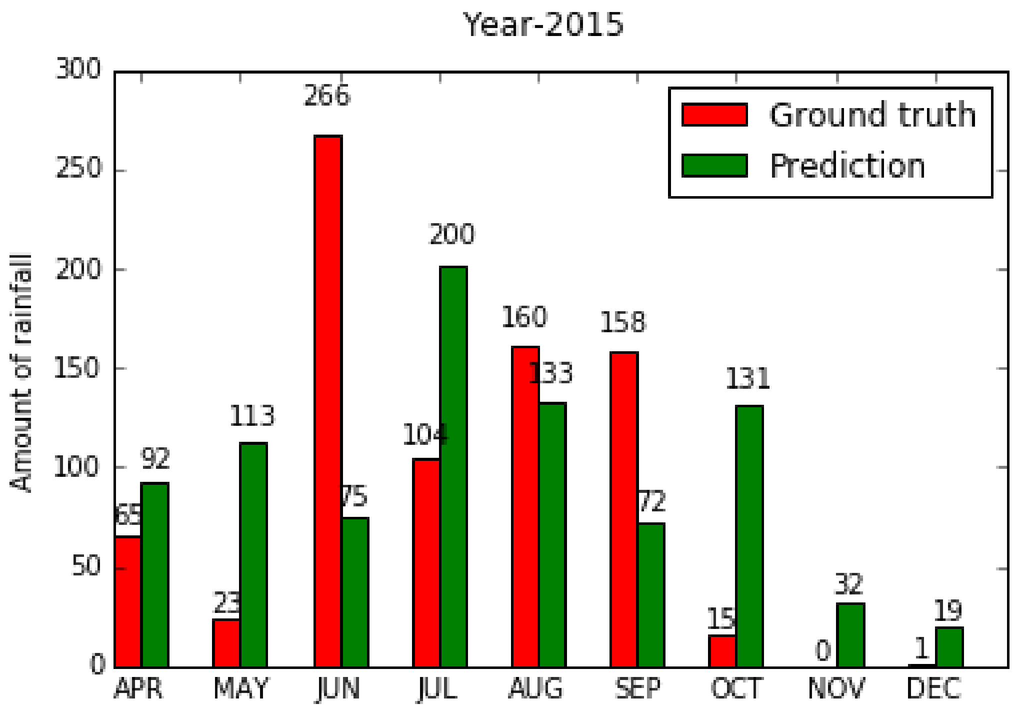
```
In [27]: #2005
         y_year_pred_2005 = reg.predict(X_year_2005)
```

```python
#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


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
395
144
308
289
249
213
98
83
40
28
9
12



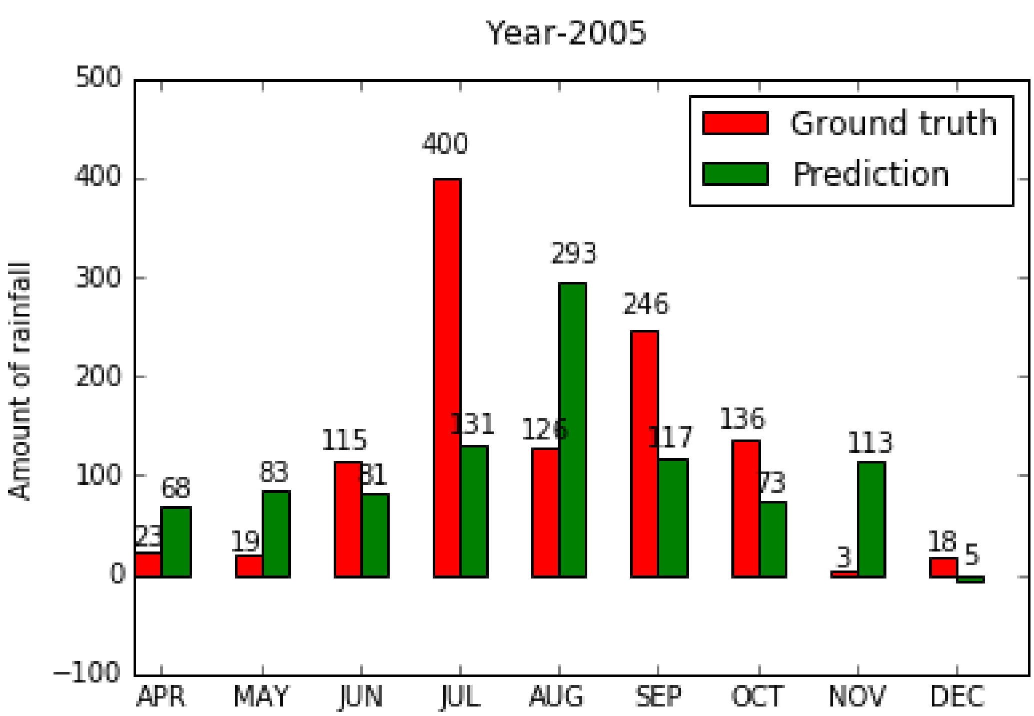
In [28]:
```
from sklearn.svm import SVR

# 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)
```

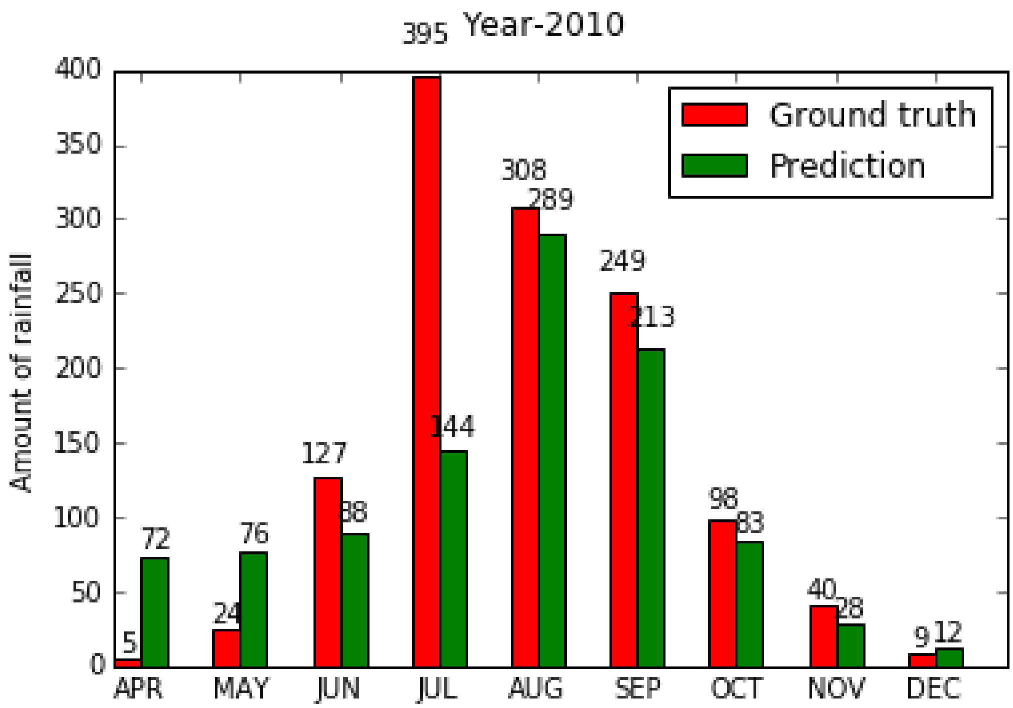
```
115.32415990638656
```

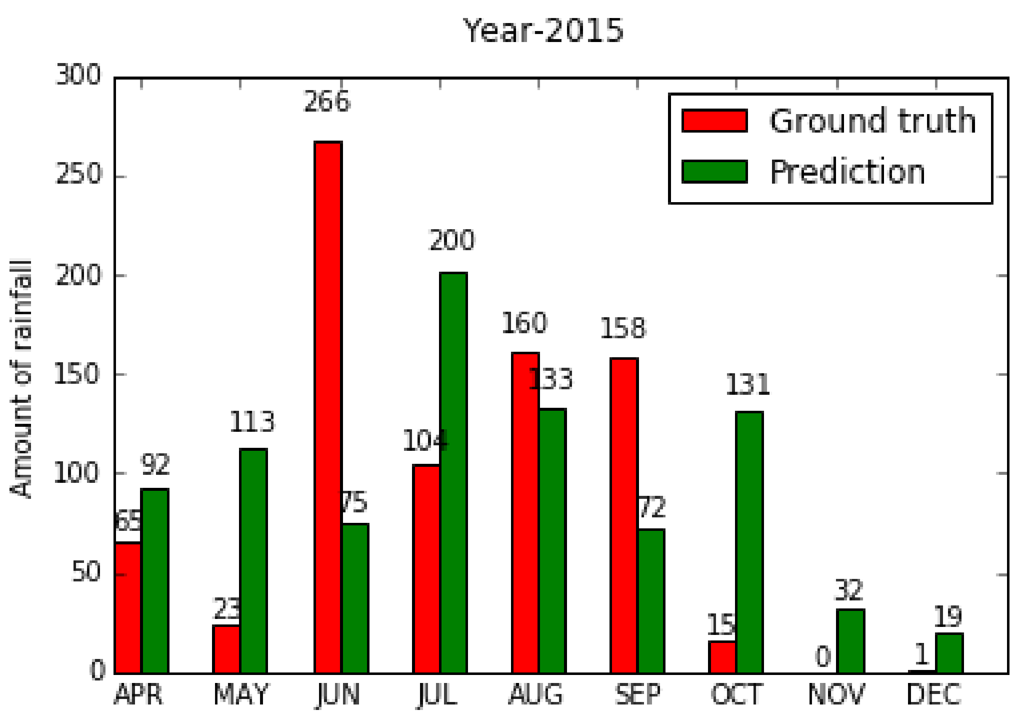
In [29]:
```
#2005
y_year_pred_2005 = reg.predict(X_year_2005)

#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, va
```

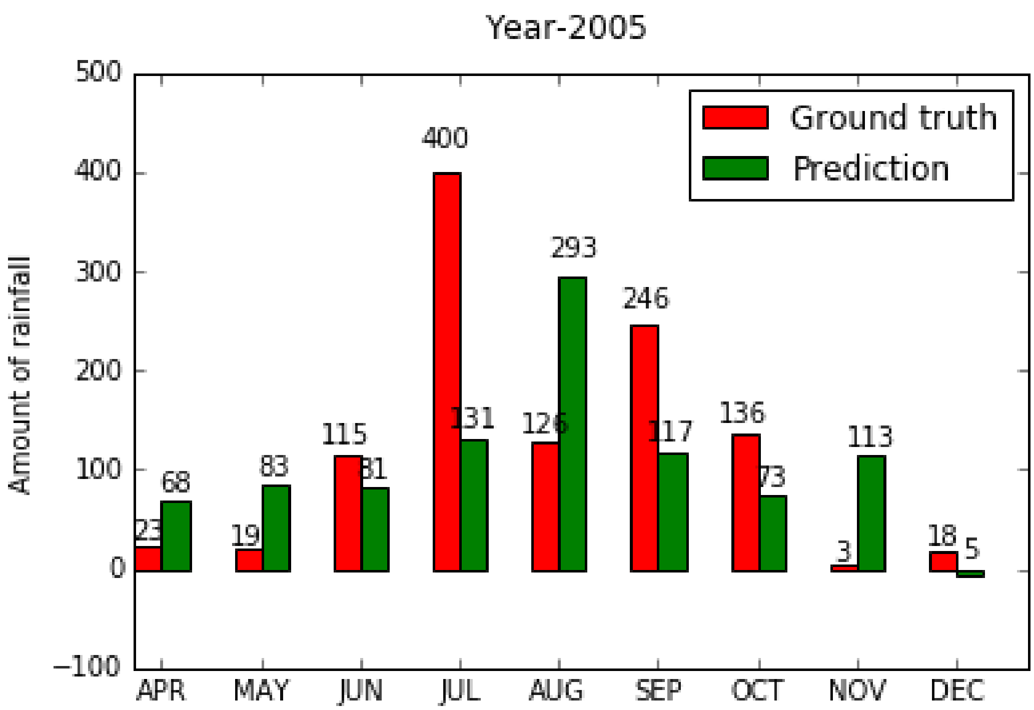
```
        y_pred = model.predict(np.expand_dims(X_test, axis=2))
        print mean_absolute_error(y_test, y_pred)
```

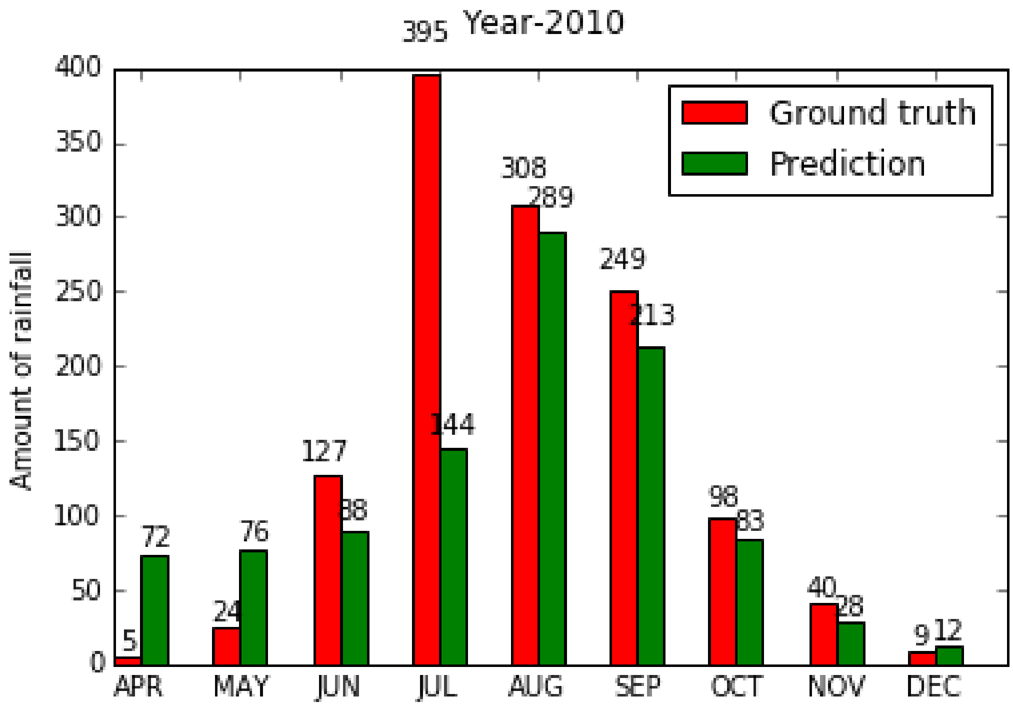
```
Train on 921 samples, validate on 103 samples
Epoch 1/10
921/921 [==============================] - 0s 66us/step - loss: 7274.9487 - mean_absolute_error: 63.5027
Epoch 2/10
921/921 [==============================] - 0s 56us/step - loss: 6431.8426 - mean_absolute_error: 56.7670
Epoch 3/10
921/921 [==============================] - 0s 56us/step - loss: 6046.0127 - mean_absolute_error: 58.4860
Epoch 4/10
921/921 [==============================] - 0s 56us/step - loss: 5883.5181 - mean_absolute_error: 56.4383
Epoch 5/10
921/921 [==============================] - 0s 57us/step - loss: 5764.2698 - mean_absolute_error: 55.1780
Epoch 6/10
921/921 [==============================] - 0s 55us/step - loss: 5706.7510 - mean_absolute_error: 55.3464
Epoch 7/10
921/921 [==============================] - 0s 56us/step - loss: 5636.2414 - mean_absolute_error: 54.4525
Epoch 8/10
921/921 [==============================] - 0s 57us/step - loss: 5564.0726 - mean_absolute_error: 54.5664
Epoch 9/10
921/921 [==============================] - 0s 57us/step - loss: 5529.6288 - mean_absolute_error: 54.0025
Epoch 10/10
921/921 [==============================] - 0s 57us/step - loss: 5478.9296 - mean_absolute_error: 53.5252
65.82400645938786
```

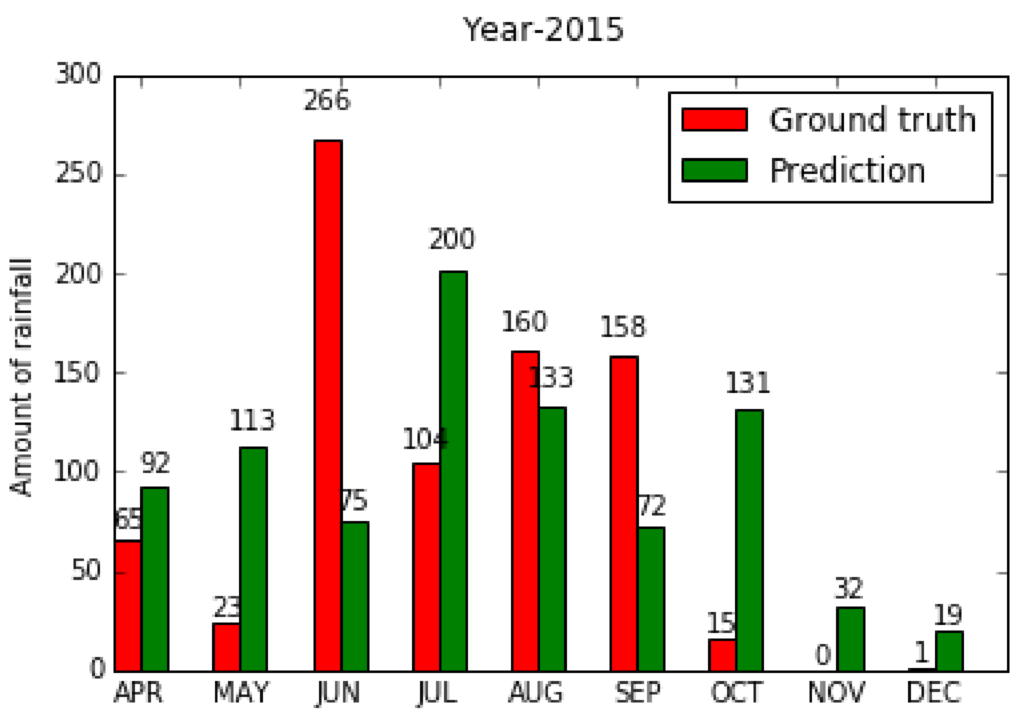
```
In [31]: #2005
         y_year_pred_2005 = reg.predict(X_year_2005)

         #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
0
50
100
150
200
250
300
350
400
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
0
50
100
150
200
250
300
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()
```

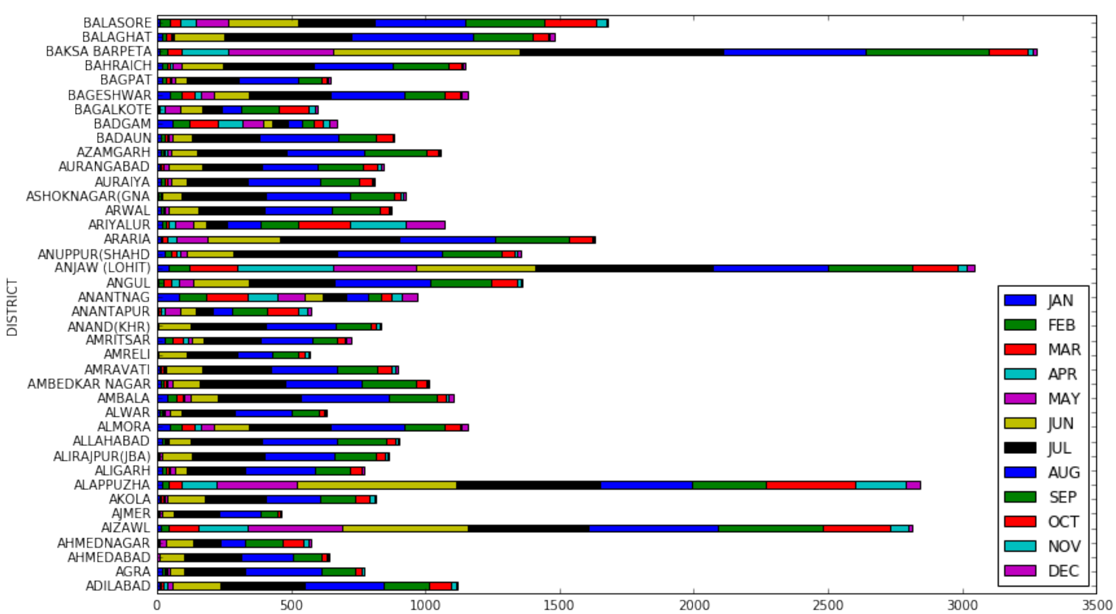
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 641 entries, 0 to 640
Data columns (total 19 columns):
STATE_UT_NAME    641 non-null object
DISTRICT         641 non-null object
JAN              641 non-null float64
FEB              641 non-null float64
MAR              641 non-null float64
APR              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
```

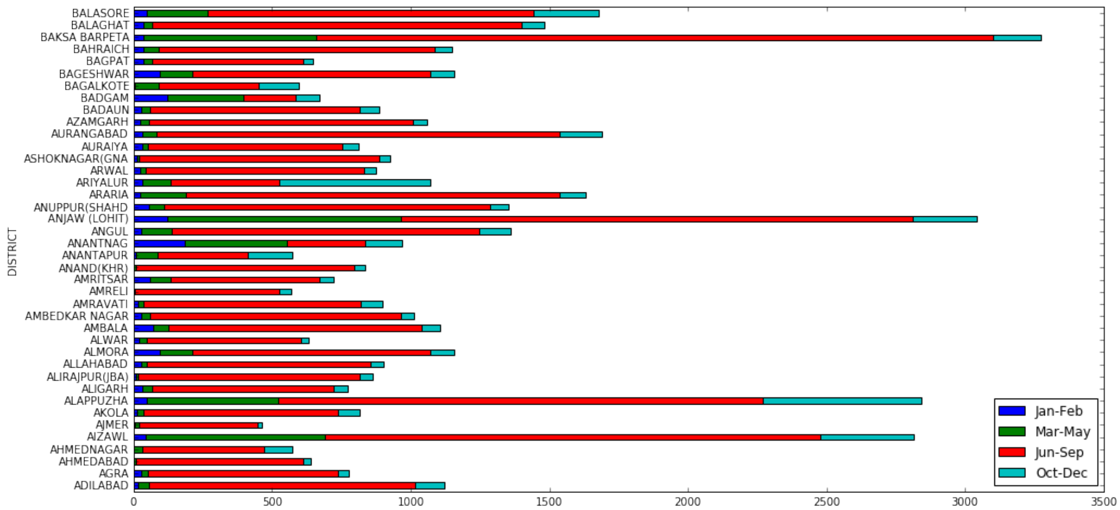
In [33]:
```
district.head()
```

Out[33]:
```
                 STATE_UT_NAME       DISTRICT    JAN   FEB    MAR    APR  \
0  ANDAMAN And NICOBAR ISLANDS        NICOBAR  107.3  57.9   65.2  117.0
1  ANDAMAN And NICOBAR ISLANDS  SOUTH ANDAMAN   43.7  26.0   18.6   90.5
2  ANDAMAN And NICOBAR ISLANDS  N & M ANDAMAN   32.7  15.9    8.6   53.4
3            ARUNACHAL PRADESH          LOHIT   42.2  80.8  176.4  358.5
4            ARUNACHAL PRADESH     EAST SIANG   33.3  79.5  105.9  216.5

     MAY    JUN    JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb  \
0  358.5  295.5  285.0  271.9  354.8  326.0  315.2  250.9  2805.2    165.2
1  374.4  457.2  421.3  423.1  455.6  301.2  275.8  128.3  3015.7     69.7
2  343.6  503.3  465.4  460.9  454.8  276.1  198.6  100.0  2913.3     48.6
3  306.4  447.0  660.1  427.8  313.6  167.1   34.1   29.8  3043.8    123.0
4  323.0  738.3  990.9  711.2  568.0  206.9   29.5   31.7  4034.7    112.8

   Mar-May  Jun-Sep  Oct-Dec
0    540.7   1207.2    892.1
1    483.5   1757.2    705.3
2    405.6   1884.4    574.7
3    841.3   1848.5    231.0
4    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
```



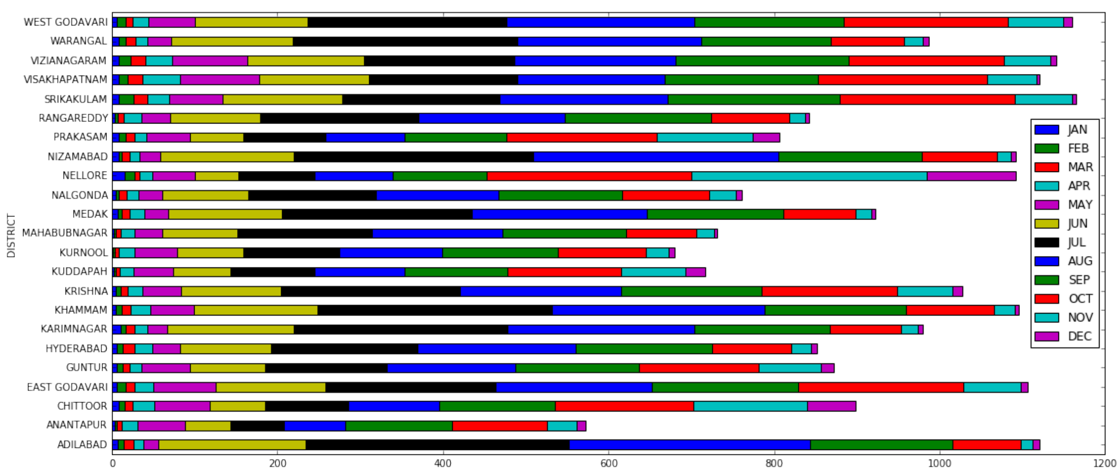
1.14 Observations

- 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=Tr
```

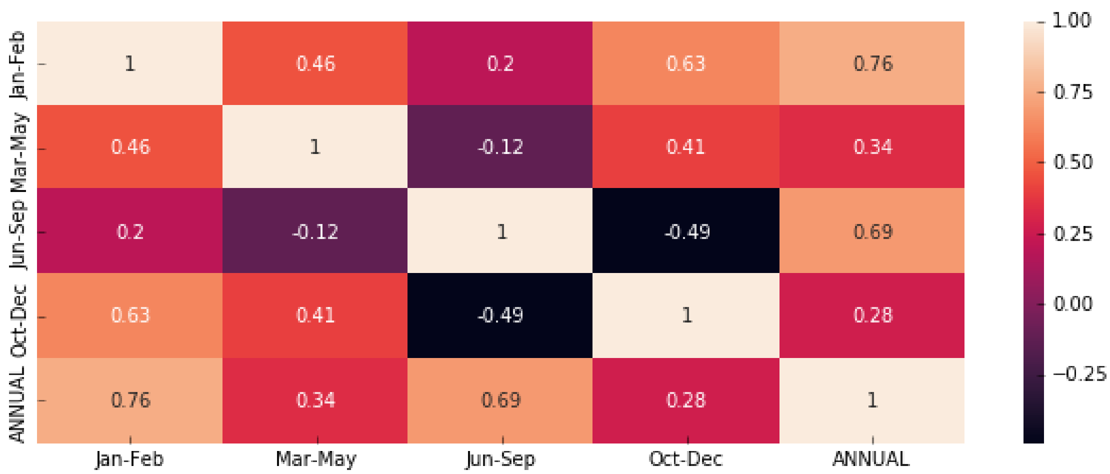



```
In [38]: ap_data[['DISTRICT', 'Jan-Feb', 'Mar-May',
                   'Jun-Sep', 'Oct-Dec']].groupby("DISTRICT").sum()[:40].plot.barh(stacked=True,figsize=(16
```

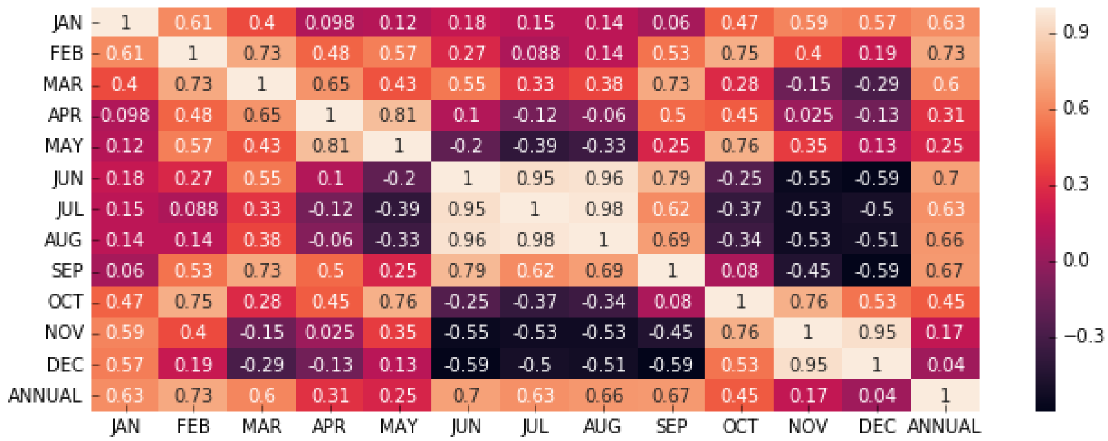
1.15 Observations

- 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','
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
         hyd = np.asarray(temp[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL','AUG', 'SEP', 'OCT', 'N
         # print temp
         X_year = None; y_year = None
         for i in range(hyd.shape[1]-3):
             if X_year is None:
```

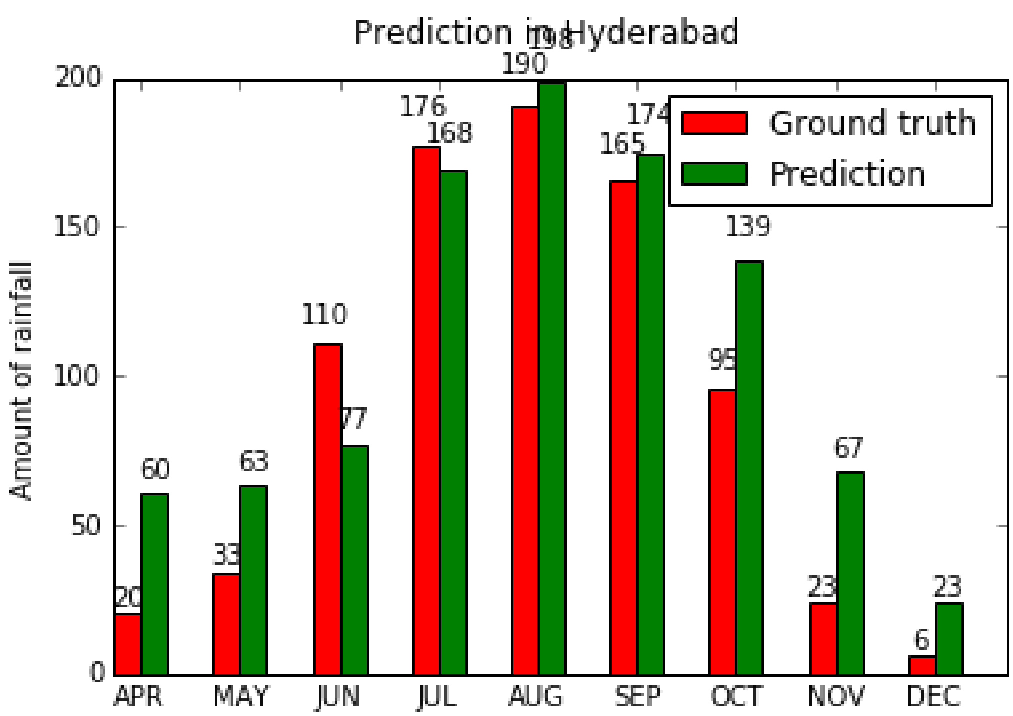
```
            X_year = hyd[:, i:i+3]
            y_year = hyd[:, i+3]
        else:
            X_year = np.concatenate((X_year, hyd[:, i:i+3]), axis=0)
            y_year = np.concatenate((y_year, hyd[:, i+3]), axis=0)
```

```
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
```

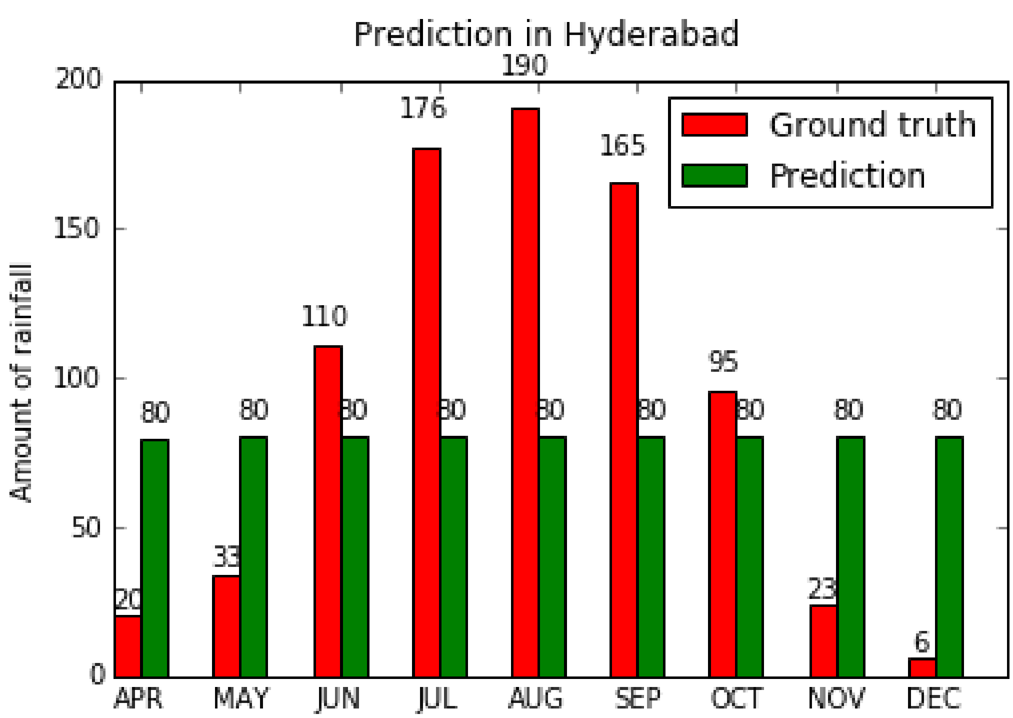


```
In [45]: from sklearn.svm import SVR

         # 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
```

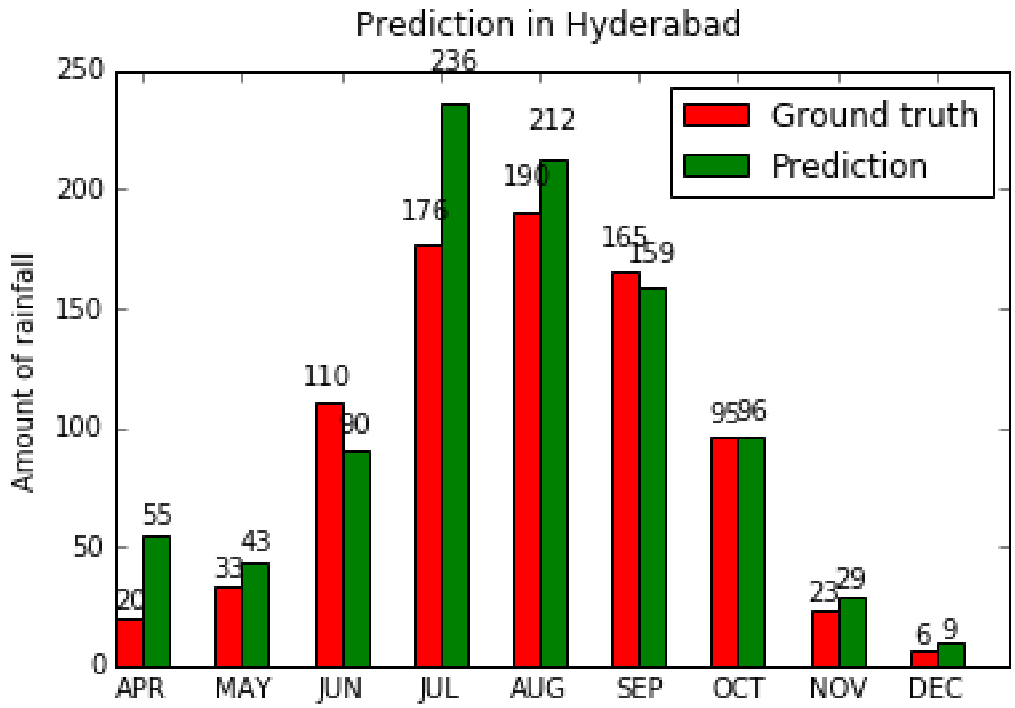


In [47]:
```
model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=10, verbose=1, va
y_pred = model.predict(np.expand_dims(X_test, axis=2))
print mean_absolute_error(y_test, y_pred)
```

```
Train on 4153 samples, validate on 462 samples
Epoch 1/10
4153/4153 [==============================] - 0s 57us/step - loss: 6957.2028 - mean_absolute_error: 50.94
Epoch 2/10
4153/4153 [==============================] - 0s 78us/step - loss: 5258.5688 - mean_absolute_error: 43.73
Epoch 3/10
4153/4153 [==============================] - 0s 56us/step - loss: 5122.4481 - mean_absolute_error: 42.58
Epoch 4/10
4153/4153 [==============================] - 0s 54us/step - loss: 5036.6459 - mean_absolute_error: 42.03
Epoch 5/10
4153/4153 [==============================] - 0s 53us/step - loss: 4964.9985 - mean_absolute_error: 41.43
Epoch 6/10
4153/4153 [==============================] - 0s 54us/step - loss: 5006.1478 - mean_absolute_error: 41.76
Epoch 7/10
4153/4153 [==============================] - 0s 54us/step - loss: 4917.7239 - mean_absolute_error: 41.21
Epoch 8/10
4153/4153 [==============================] - 0s 55us/step - loss: 4861.6742 - mean_absolute_error: 41.15
Epoch 9/10
4153/4153 [==============================] - 0s 55us/step - loss: 4792.0339 - mean_absolute_error: 40.51
Epoch 10/10
4153/4153 [==============================] - 0s 54us/step - loss: 4796.9758 - mean_absolute_error: 40.56
42.14205056426844
```

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
```

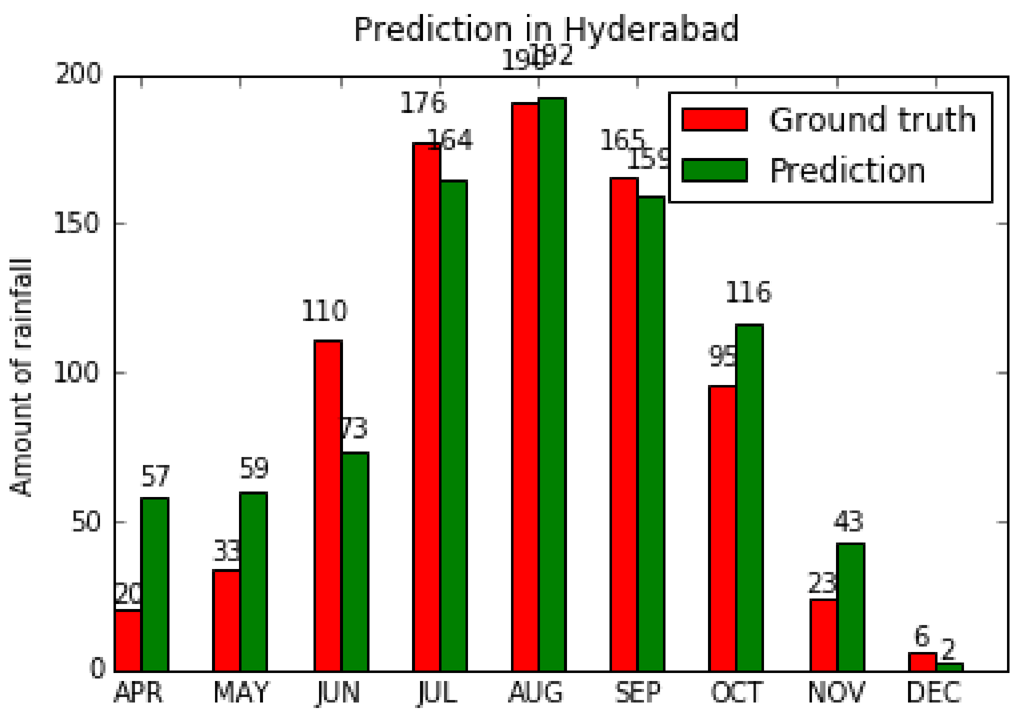



```
In [49]: # training and testing sets for only andhra pradesh data
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import mean_absolute_error

         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
```

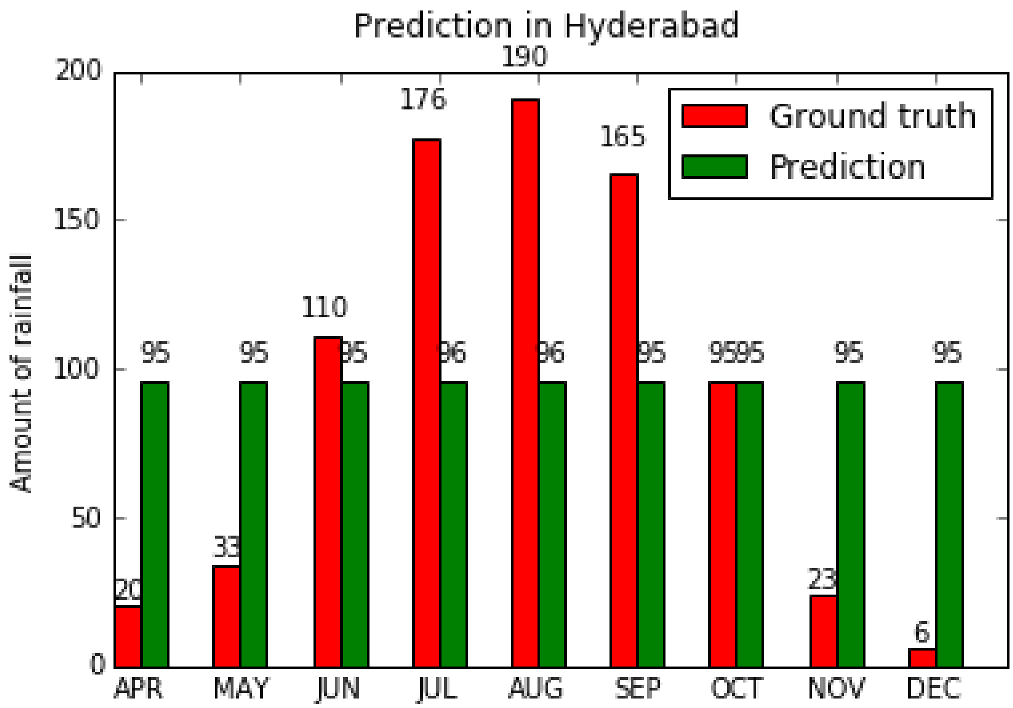



```
In [52]: from sklearn.svm import SVR

         # 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
```



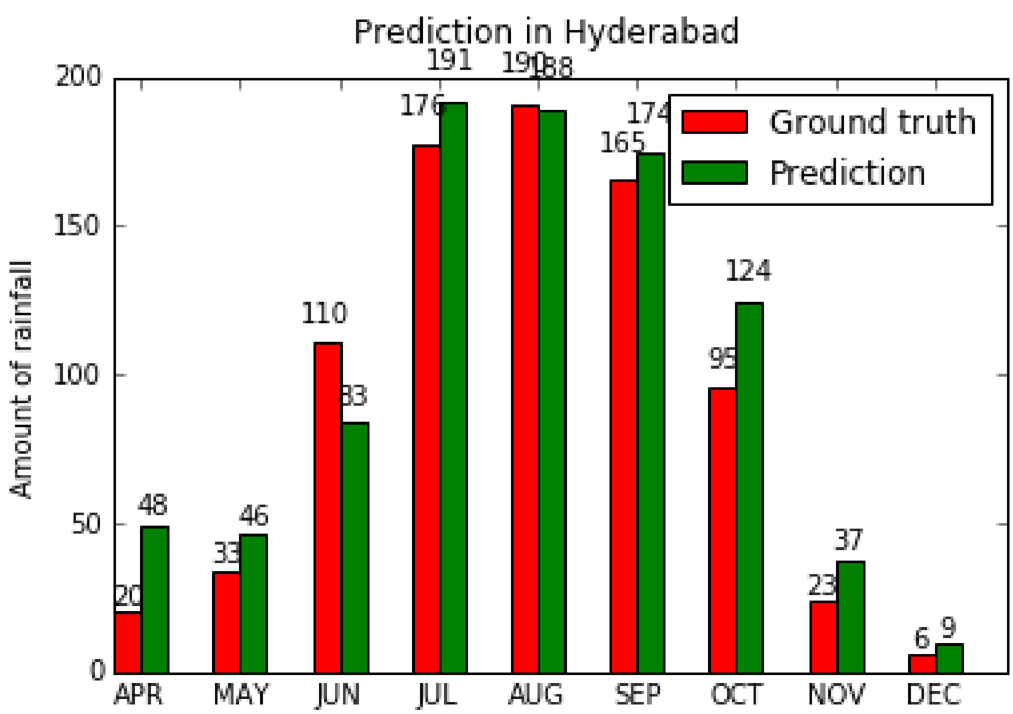

```
In [54]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=10, verbose=1, va
         y_pred = model.predict(np.expand_dims(X_test, axis=2))
         print mean_absolute_error(y_test, y_pred)
```

```
Train on 148 samples, validate on 17 samples
Epoch 1/10
148/148 [==============================] - 0s 120us/step - loss: 1778.6490 - mean_absolute_error: 30.945
Epoch 2/10
148/148 [==============================] - 0s 91us/step - loss: 1664.5006 - mean_absolute_error: 29.9809
Epoch 3/10
148/148 [==============================] - 0s 92us/step - loss: 1504.4134 - mean_absolute_error: 28.0781
Epoch 4/10
148/148 [==============================] - 0s 89us/step - loss: 1405.6685 - mean_absolute_error: 26.9898
Epoch 5/10
148/148 [==============================] - 0s 84us/step - loss: 1399.6531 - mean_absolute_error: 26.8409
Epoch 6/10
148/148 [==============================] - 0s 83us/step - loss: 1364.1109 - mean_absolute_error: 26.3740
Epoch 7/10
148/148 [==============================] - 0s 79us/step - loss: 1321.5538 - mean_absolute_error: 26.1438
Epoch 8/10
148/148 [==============================] - 0s 83us/step - loss: 1302.3495 - mean_absolute_error: 26.0448
Epoch 9/10
```

```
148/148 [==============================] - 0s 81us/step - loss: 1290.8667 - mean_absolute_error: 25.858
Epoch 10/10
148/148 [==============================] - 0s 82us/step - loss: 1273.7824 - mean_absolute_error: 25.566
32.25840494065058
```

```
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*
- Github repo: link