In [712…
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

import itertools
import warnings
warnings.filterwarnings('ignore')
```

In [713…
```
filename = 'pune_1965_to_2002.csv'
```

Data

- Downloaded from http://www.indiawaterportal.org/met_data/
- State: Maharashtra
- District: PUNE
- Data type: Precipitation
- Data Range: Monthly mean precipitation for all the years from 1972 to 2002

Transpose data for easy visualiztion

In [714…
```
rainfall_data_matrix = pd.read_csv(filename, delimiter='\t')
rainfall_data_matrix.set_index('Year', inplace=True)
rainfall_data_matrix = rainfall_data_matrix.transpose()
rainfall_data_matrix
```

Out[714]:

Year	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	...	
Jan	0.029	0.905	0.248	0.318	0.248	0.070	0.000	0.000	0.000	0.000	...	
Feb	0.069	0.000	3.390	3.035	2.524	0.000	0.000	0.029	2.969	0.000	...	
Mar	0.000	0.000	1.320	1.704	0.334	0.001	0.000	0.000	0.234	6.427	...	
Apr	21.667	2.981	13.482	23.307	4.569	16.218	0.812	5.982	3.925	16.864	...	
May	17.859	63.008	11.116	7.441	6.213	68.036	57.691	19.101	14.978	51.209	...	
Jun	102.111	94.088	251.314	179.872	393.682	300.546	297.187	132.413	304.484	148.697	...	23
Jul	606.071	481.942	780.006	379.354	678.354	330.502	122.195	338.484	696.024	405.359	...	45
Aug	402.521	59.386	181.069	171.979	397.335	283.476	372.693	68.741	256.932	319.651	...	15
Sep	69.511	150.624	183.757	219.884	205.413	158.640	286.056	120.415	183.206	288.533	...	10
Oct	5.249	1.308	50.404	73.997	24.014	115.758	39.424	1.078	101.805	188.876	...	22
Nov	16.232	41.214	8.393	23.326	24.385	0.260	0.554	24.089	5.516	0.260	...	
Dec	22.075	4.132	37.685	2.020	1.951	0.000	0.000	0.143	0.000	0.000	...	3

12 rows × 38 columns

Genearete dates from 1965-01(January 1965) to 2002-12(December 2002)

In [715…

```
dates = pd.date_range(start='1965-01', freq='MS', periods=len(rainfall_data_matrix.col
dates
```

Out[715]:

```
DatetimeIndex(['1965-01-01', '1965-02-01', '1965-03-01', '1965-04-01',
               '1965-05-01', '1965-06-01', '1965-07-01', '1965-08-01',
               '1965-09-01', '1965-10-01',
               ...
               '2002-03-01', '2002-04-01', '2002-05-01', '2002-06-01',
               '2002-07-01', '2002-08-01', '2002-09-01', '2002-10-01',
               '2002-11-01', '2002-12-01'],
              dtype='datetime64[ns]', length=456, freq='MS')
```

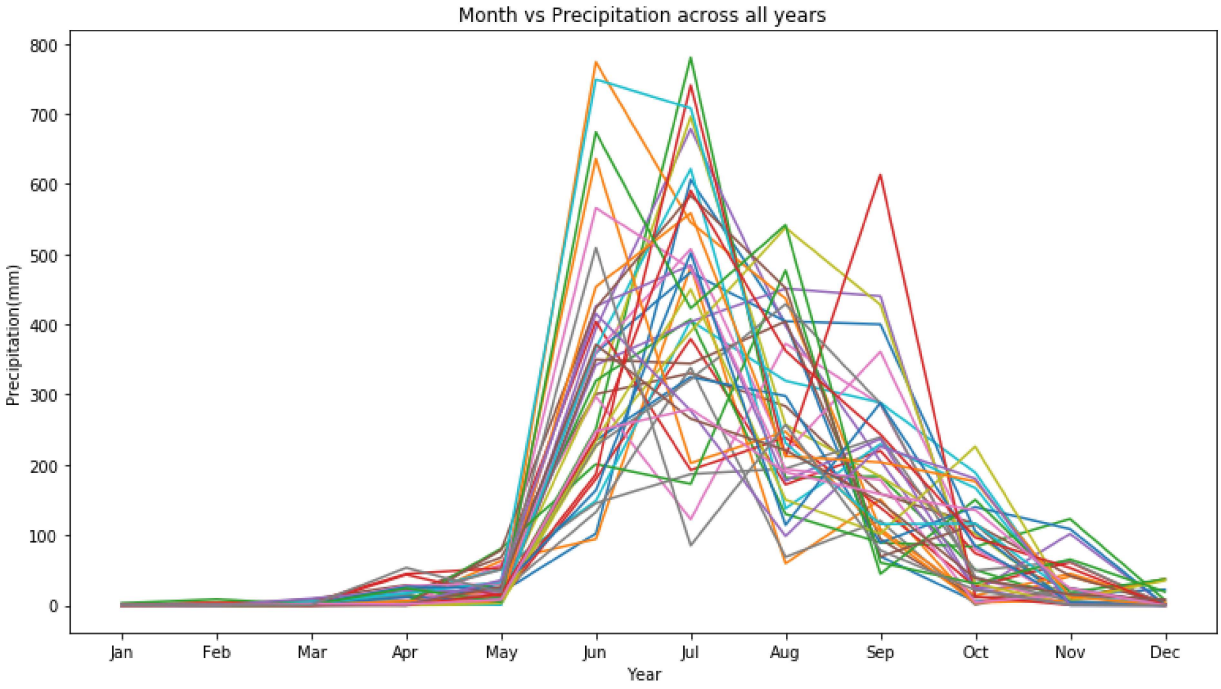
Visualize the whole data

In [716…

```
plt.figure(figsize=(13,7))
plt.plot(rainfall_data_matrix)
plt.xlabel('Year')
plt.ylabel('Precipitation(mm)')
plt.title('Month vs Precipitation across all years')
```

Out[716]:

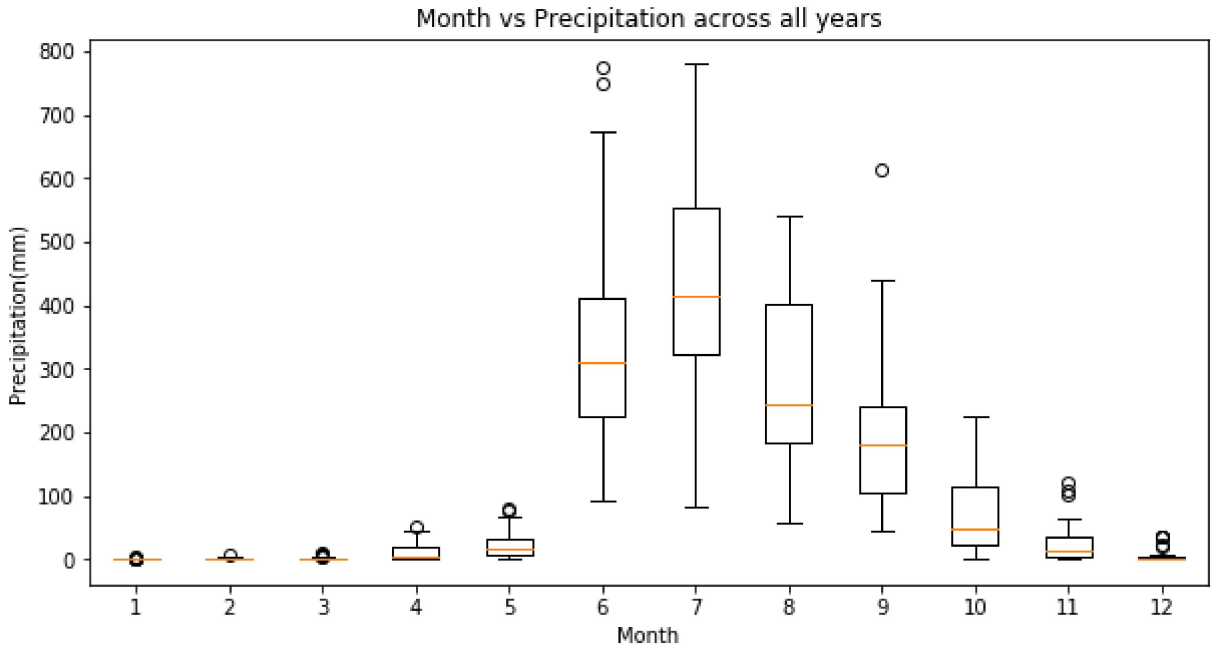
```
Text(0.5,1,'Month vs Precipitation across all years')
```



Box plot

In [717…
```
plt.figure(figsize=(10,5))
plt.boxplot(rainfall_data_matrix)
plt.xlabel('Month')
plt.ylabel('Precipitation(mm)')
plt.title('Month vs Precipitation across all years')
```

Out[717]:
```
Text(0.5,1,'Month vs Precipitation across all years')
```



Insights

- The rainfall in the months November, December, January, February, March and April is very less.
- The rainfall in the months June, July and August are high compared to rainfall in other months of the year.
- We can observe the seasonality effect.

In [718…
```
rainfall_data_matrix_np = rainfall_data_matrix.transpose().as_matrix()

shape = rainfall_data_matrix_np.shape
rainfall_data_matrix_np = rainfall_data_matrix_np.reshape((shape[0] * shape[1], 1))
```

Split the whole data into train(1965 - 1995) and test data(1995 - 2002)

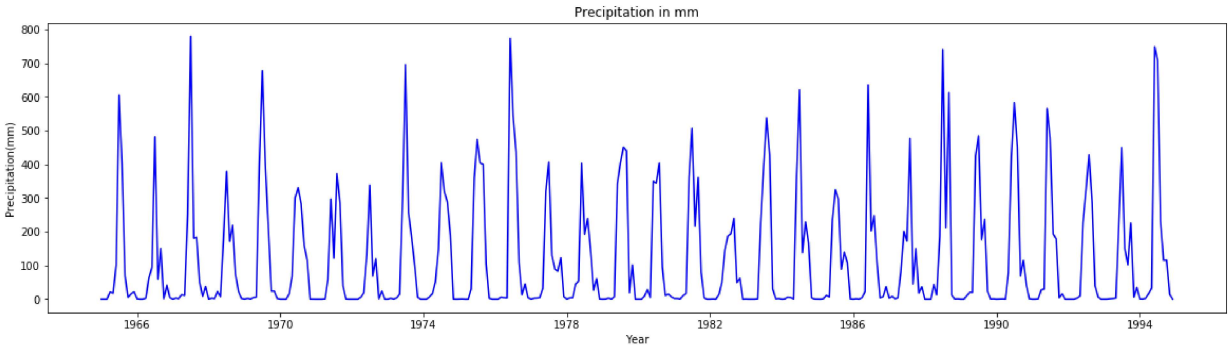
In [719…
```
rainfall_data = pd.DataFrame({'Precipitation': rainfall_data_matrix_np[:,0]})
rainfall_data.set_index(dates, inplace=True)

test_rainfall_data = rainfall_data.ix['1995': '2002']
rainfall_data = rainfall_data.ix[: '1994']
rainfall_data.head()
```

Out[719]:

	Precipitation
1965-01-01	0.029
1965-02-01	0.069
1965-03-01	0.000
1965-04-01	21.667
1965-05-01	17.859

In [720…
```
plt.figure(figsize=(20,5))
plt.plot(rainfall_data, color='blue')
plt.xlabel('Year')
plt.ylabel('Precipitation(mm)')
plt.title('Precipitation in mm')
```

Out[720]:
```
Text(0.5,1,'Precipitation in mm')
```

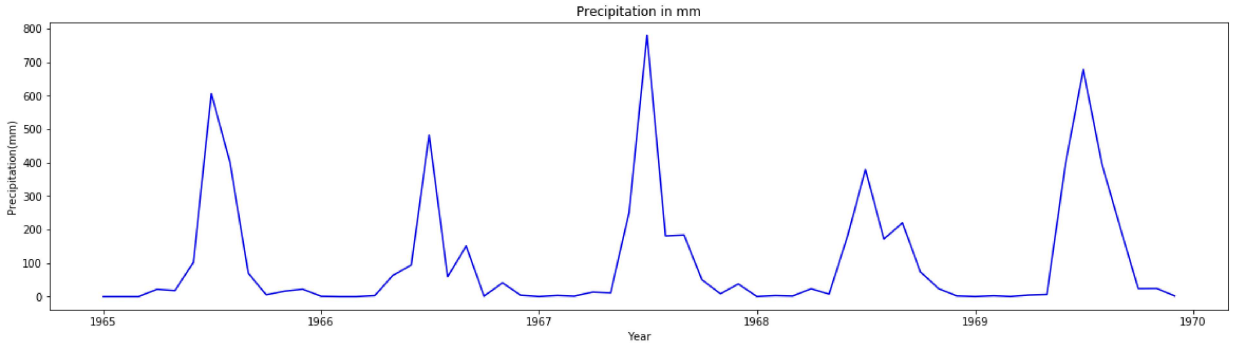


Visualize data(zoomed version) - 5 years(1972 - 1977)

In [721…
```
plt.figure(figsize=(20,5))
plt.plot(rainfall_data.ix[:60], color='blue')
```

```
plt.xlabel('Year')
plt.ylabel('Precipitation(mm)')
plt.title('Precipitation in mm')
```

Out[721]:
```
Text(0.5,1,'Precipitation in mm')
```



LSTM

Scaling

In [722…
```
scaler = MinMaxScaler(feature_range=(0, 1))

train_data = scaler.fit_transform(rainfall_data)
test_data = scaler.fit_transform(test_rainfall_data)
```

In [723…
```
rainfall_data.head()
```

Out[723]:

	Precipitation
1965-01-01	0.029
1965-02-01	0.069
1965-03-01	0.000
1965-04-01	21.667
1965-05-01	17.859

In [724…
```
train_data[:5]
```

Out[724]:
```
array([[3.71792012e-05],
       [8.84608580e-05],
       [0.00000000e+00],
       [2.77779915e-02],
       [2.28959777e-02]])
```

In [725…
```
scaler.inverse_transform(train_data[:5])
```

Out[725]:
```
array([[ 0.0250525 ],
       [ 0.05960767],
       [ 0.        ],
       [18.71767176],
       [15.42801957]])
```

In [726…
```
print("Shape of train data: " + str(train_data.shape))
print("Shape of test data: " + str(test_data.shape))
```

```
Shape of train data: (360, 1)
Shape of test data: (96, 1)
```

Create dataset

- Convert time series data into input and labels.

In [727…
```
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []

    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])

    return np.array(dataX), np.array(dataY)
```

In [728…
```
look_back = 12
trainX, trainY = create_dataset(train_data, look_back)
testX, testY = create_dataset(test_data, look_back)
```

In [729…
```
testY
```

Out[729]:
```
array([2.37448262e-05, 4.12417950e-03, 1.29112493e-04, 9.37623826e-03,
       7.61912111e-03, 6.73176212e-01, 8.28970469e-01, 3.15344649e-01,
       3.02215244e-01, 2.61892077e-01, 1.81083981e-02, 0.00000000e+00,
       1.90552230e-03, 0.00000000e+00, 1.10858657e-03, 3.54866428e-02,
       6.92161684e-03, 1.00000000e+00, 6.27624731e-01, 8.03731203e-01,
       8.97509910e-02, 4.60798034e-02, 9.70451048e-02, 2.81836247e-02,
       1.18724131e-05, 3.96390193e-03, 1.08335770e-04, 5.59487468e-04,
       2.33248396e-02, 3.54108078e-01, 8.76574393e-01, 5.37756500e-01,
       3.61283467e-01, 1.42949790e-01, 7.86027951e-02, 5.19418074e-05,
       0.00000000e+00, 2.12367790e-03, 0.00000000e+00, 9.52761152e-04,
       5.31706021e-02, 6.16580419e-01, 4.11285619e-01, 1.46351236e-01,
       3.35339276e-01, 2.67574511e-01, 3.90305581e-04, 4.89737041e-05,
       0.00000000e+00, 2.52288779e-04, 0.00000000e+00, 1.77047361e-03,
       3.89370629e-02, 5.51069927e-01, 3.93892534e-01, 3.27699379e-01,
       2.18446465e-01, 5.67590390e-02, 2.26674047e-02, 1.21499308e-02,
       2.18155591e-04, 0.00000000e+00, 3.23226447e-03, 2.26763090e-03,
       1.16646459e-02, 3.68018094e-01, 4.14862184e-01, 2.81085317e-01,
       2.34517260e-01, 2.01115710e-01, 3.50725924e-02, 4.45215492e-06,
       3.42815929e-04, 1.35197104e-03, 5.75812036e-04, 7.90494946e-02,
       2.73510717e-02, 7.55597472e-01, 1.26049410e-01, 3.81705502e-01,
       1.16155238e-01, 3.18863335e-02, 9.11207706e-04])
```

In [730…
```
print("Shape of train input: " + str(trainX.shape))
print("Shape of train labels: " + str(trainY.shape))
print("Shape of test input: " + str(testX.shape))
print("Shape of test labels: " + str(testY.shape))
```

```
Shape of train input: (347, 12)
Shape of train labels: (347,)
Shape of test input: (83, 12)
Shape of test labels: (83,)
```

Reshape data

In [731…
```
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

In [732…
```
print("Shape of train input: " + str(trainX.shape))
print("Shape of train labels: " + str(trainY.shape))
print("Shape of test input: " + str(testX.shape))
print("Shape of test labels: " + str(testY.shape))
```

```
Shape of train input: (347, 1, 12)
Shape of train labels: (347,)
Shape of test input: (83, 1, 12)
Shape of test labels: (83,)
```

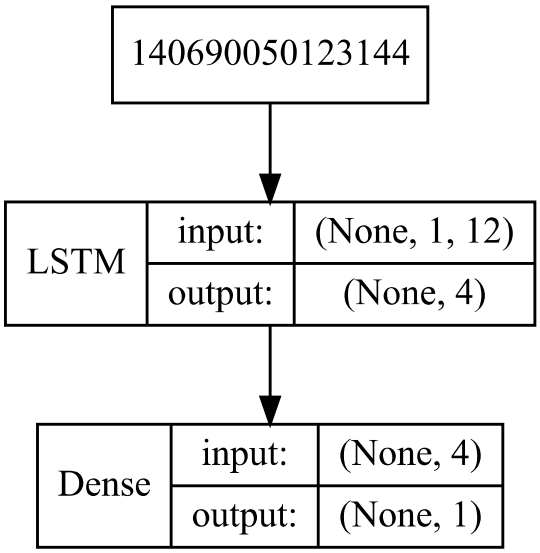
Create model using Keras

In [733…
```
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

Display the model

In [734…
```
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

def plot_keras_model(model, show_shapes=True, show_layer_names=True):
    return SVG(model_to_dot(model, show_shapes=show_shapes, show_layer_names=show_laye

plot_keras_model(model, show_shapes=True, show_layer_names=False)
```

Out[734]:



Train model using train data

In [1]:
```
model.fit(trainX, trainY, epochs=200, batch_size=1, verbose=2)
```

Predict values using the trained model

In [736…
```
trainPredict = model.predict(trainX)
# testPredictInTrain = model.predict(trainX[-look_back:, :, :])
testPredict = model.predict(testX)
```

In [737…
```
testY.shape
```

Out[737]:
```
(83,)
```

Inverse transform the data

In [738…
```
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
# testPredictInTrain = scaler.inverse_transform(testPredictInTrain)
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
```

Calculate RMSE for train and test predictions

In [739…
```
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Train Score: 75.13 RMSE
Test Score: 96.19 RMSE
```

Generate Dates for index

In [740…
```
dates = pd.date_range(start='1965-04', freq='MS', periods=trainY.shape[1])
```

Create Dataframes for actual values and predicted values of train data

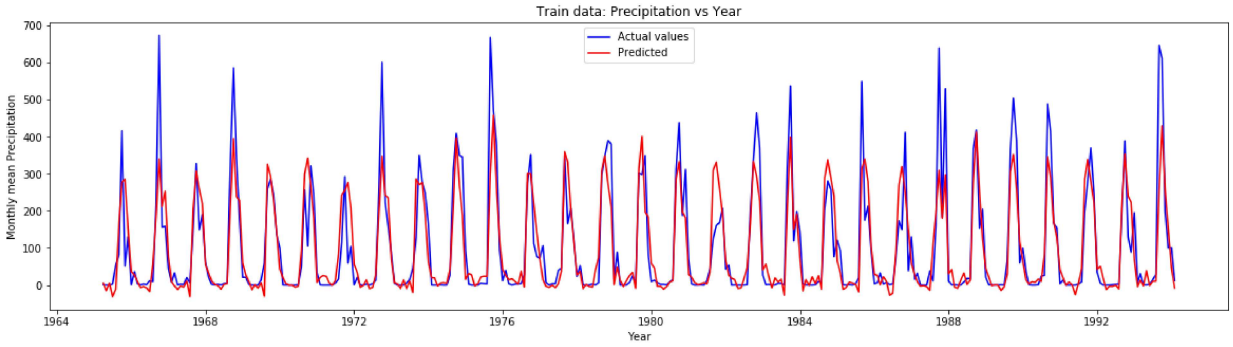
In [741…
```
trainActual = pd.DataFrame({'Precipitation': trainY[0]})
trainActual.index = dates

trainPredictdf = pd.DataFrame({'Precipitation': trainPredict[:,0]})
trainPredictdf.index = dates
```

Plot the predicted values

- Actual values - blue
- Predicted values - red

In [742…
```
plt.figure(figsize=(20,5))
plt.plot(trainActual, color='blue', label='Actual values')
plt.plot(trainPredictdf, color='red', label='Predicted')
plt.title('Train data: Precipitation vs Year')
plt.xlabel('Year')
plt.ylabel('Monthly mean Precipitation')
plt.legend(loc='best')
```

Out[742]:
```
<matplotlib.legend.Legend at 0x7ff4f3e60400>
```



Generate Dates for index

In [747…
```
testDates = pd.date_range(start=test_rainfall_data.index[0]+12, freq='MS', periods=tes
```

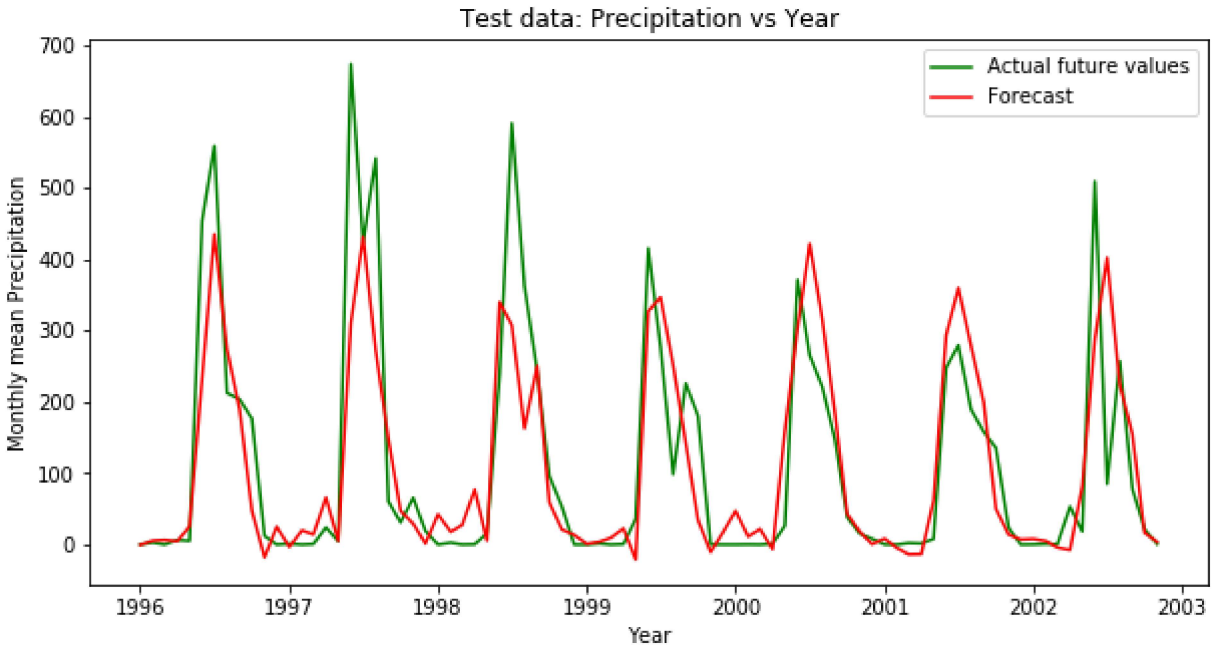
Create Dataframes for actual values and predicted values of test data

In [748…
```
# testActual = pd.DataFrame({'Precipitation': testPredictInTrain[:,0] + testY[0]})
testActual = pd.DataFrame({'Precipitation': testY[0]})
testActual.index = testDates

testPredictdf = pd.DataFrame({'Precipitation': testPredict[:,0]})
testPredictdf.index = testDates
```

Plot the forecasted values

In [749…
```
plt.figure(figsize=(10,5))
plt.plot(testActual, color='green', label='Actual future values')
plt.plot(testPredictdf, color='red', label='Forecast')
plt.title('Test data: Precipitation vs Year')
plt.xlabel('Year')
plt.ylabel('Monthly mean Precipitation')
plt.legend(loc='best')
```

Out[749]:
```
<matplotlib.legend.Legend at 0x7ff4f3a1ea20>
```



In [ ]:

In [673…
```
index = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Train_RMSE = [126.23, 117.23, 104.45, 96.37, 93.37, 90.76, 89.29, 85.00, 83.40, 81.89,
Test_RMSE = [145.11, 140.89, 127.52, 121.96, 116.52, 113.09, 106.36, 107.95, 106.25, 1
```

In [677…
```
plt.plot(Train_RMSE, color='blue', label='Train RMSE')
plt.plot(Test_RMSE, color='red', label='Test RMSE')
plt.legend(loc='best')
plt.title('RMSE of Train and Test data at different look backs')
```

Out[677]:
```
Text(0.5,1,'RMSE of Train and Test data at different look backs')
```

In [ ]: