A PROJECT REPORT

ON

RAINFALL IN INDIA

Using

Machine Learning and Neural Networks

Submitted in partial fulfillment for the requirement of the award of
TRAINING and INTERNSHIP
IN
Data Analytics , Machine Learning and AI using Python

DiginiqueTechlabs
Ideal for Geeks

Submitted by

Aditya Kumar Yadav (CSE 3rd year)

REC KANNAUJ

Under the guidance of

Mr. Bipul shahi

ACKNOWLEDGEMENT

I am very thankful and have much gratitude towards my mentor in this Project Bipul Shahi, Corporate Trainer, Developer, Traveller, IOT, Artificial Intelligence, Robotics, Cloud Computing, Android Apps.

My mentor play major role in my project completion. By providing me the best base knowledge through his training and unconditional help by improving my loop holes in this project. I acknowledge my gratitude to his recording lectures of everyday's session. that's why I can easily access them anytime and can pickup every single bit. I acknowledge my batchmates of this Internship they helped me a lot by his amazing skill and their knowledge.I acknowledge India Meteorological Department (IMD) Govt. of India has shared this dataset. Shrutarbaa Banerjee, Komal kumari, Sudipta Mohanty, P. Arun, Rohan Verma, Niveditha, Sushmita and Naveeta Maheshwari. I am thankful to all of you. I had an amazing learning experience in this project session. I acknowledge my gratitude and respect to my parents for supporting me through this project.I also declare to the best of my knowledge and belief that the Project Work has not been submitted anywhere else.

ABSTRACT

Rainfall Prediction and Monsoon Prediction is clearly very important for India and other countries too. Prediction for Countries are categorized in various forms. Prediction can be done for few months or few weeks or few days. And also prediction for a rainfall in day in hours time. we most of the time read in newspapers and articles about rainfall in different regions(Country, state, District).

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.

Dataset:

Dataset has been gathered from https://www.kaggle.com/rajanand/rainfall-in-india

This dataset contains monthly rainfall detail of 36 meteorological sub-divisions of India.

- Time Period: 1901 - 2015.
- Granularity: Monthly
- Location: 36 meteorological sub-divisions of India.
- Rainfall unit: mm.

Techonology and Concepts

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial **intelligence** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine Learning is categorise in two ways:

- **Supervised Learning:** we train the machine using data which is well "labeled." Supervised learning allows us to collect data or produce a data output from the previous experience.

  In supervised Learninf there are features and labels. Features are independent Variables and Labels ar dependent on features.

- **Unsupervised Learning:** It is a machine learning technique, where we do not need to supervise the model.Instead, we need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.

  In unsupervised Learning, there is only features and they are categorised through clustering.

- **Reinforcement Learning**: Reinforcement learning is the training of machine learning models to make a sequence of decisions. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation.

Algorithm used in this model:

Linear Regression: ElasticNet is a linear regression model trained with both $\ell 1$ and $\ell 2$-norm regularization of the coefficients. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge. We control the convex combination of $\ell 1$ and $\ell 2$ using the `l1_ratio` parameter.

Elastic-net is useful when there are multiple features which are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

A practical advantage of trading-off between Lasso and Ridge is that it allows Elastic-Net to inherit some of Ridge's stability under rotation.

Supprt Vector Machine(SVM): The model produced by Support Vector Regression depends only on a subset of the training data, because the cost function ignores samples whose prediction is close to their target.

Neural Networks:

Models: Model groups layers into an object with training and inference features.

Arguments

- inputs: The input(s) of the model: a keras.Input object or list of keras.Input objects.
- outputs: The output(s) of the model. See Functional API example below.
- name: String, the name of the model.

Con1D: This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide an input_shape argument (tuple of integers or None, e.g. (10, 128) for sequences of 10 vectors of 128-dimensional vectors, or (None, 128) for variable-length sequences of 128-dimensional vectors.

Flattens: Flattens the input. Does not affect the batch size.If inputs are shaped (batch,) without a feature axis, then flattening adds an extra channel dimension and output shape is (batch, 1).

Dense: Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

Libraries and Packages:

- import pandas ad pd
- import numpy as np
- import matplotlib as plt
- import tensorflow as tf

Dataset:

rainfall_data=pd.read_csv(r'/Volumes/Aditya Yadav/Study Material/ML and AI/rainfall-in-india/rainfall in india 1901-2015.csv').

In rainfall_data , there are 4116 rows and 19 columns.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.

Column has following attributes:

SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec

Modification in DATA:

Firstly we have to check that Is there any null values in our dataset. So we will use isnull function to check null values in datset.

rainfall_data.isnull().sum()

```
SUBDIVISION 0
YEAR 0
JAN 4
FEB 3
MAR 6
APR 4
MAY 3
JUN 5
JUL 7
AUG 4
SEP 6
OCT 7
NOV 11
DEC 10
ANNUAL 26
Jan-Feb 6
Mar-May 9
Jun-Sep 10
Oct-Dec 13
dtype: int64
```

So we found the null values, Now we will remove the null values by replacing to the mean of their respective columns.

rainfall_data=rainfall_data.fillna(rainfall_data.mean())

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.

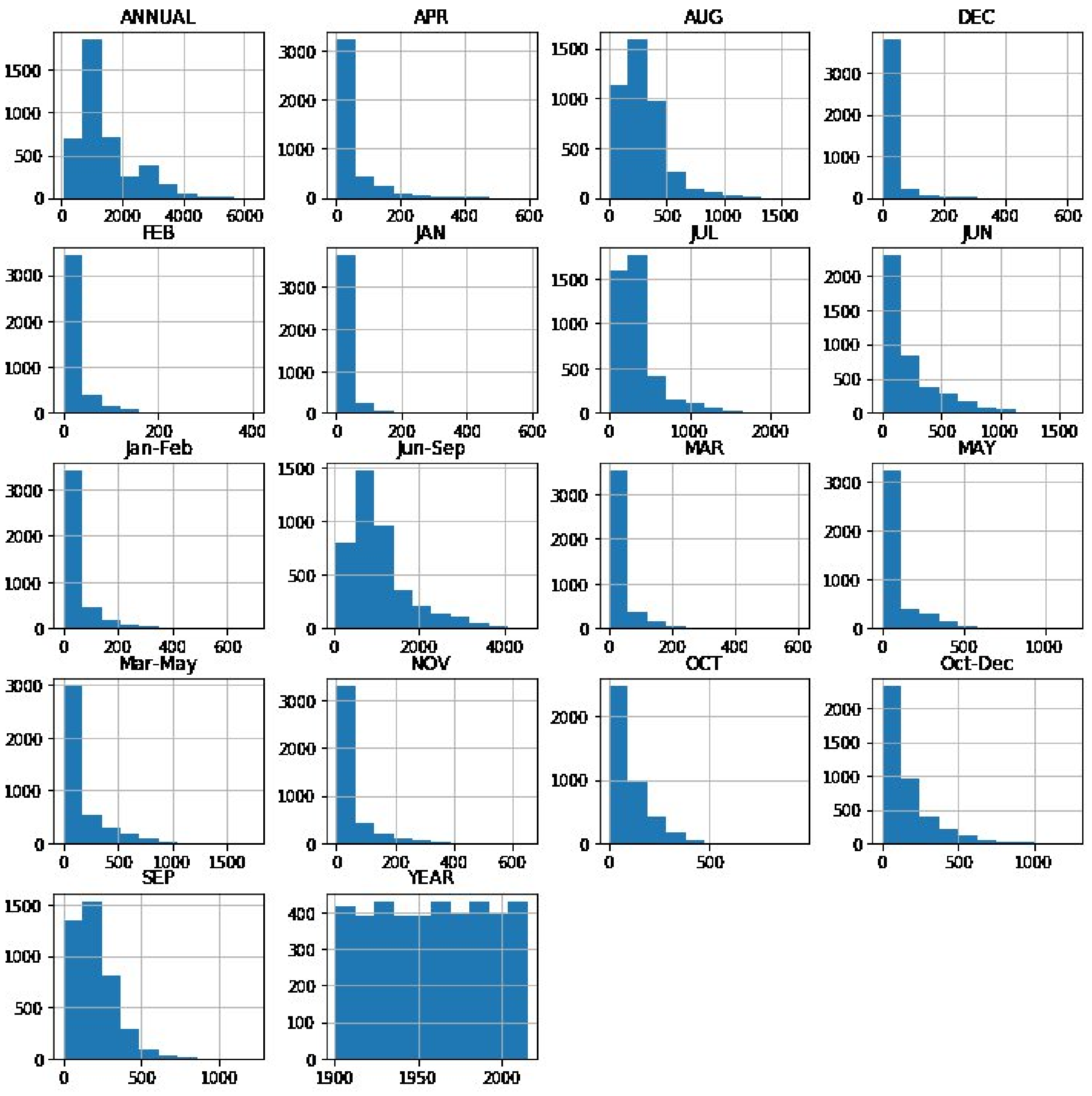
Types of graphs

- 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.

Observations by graph:

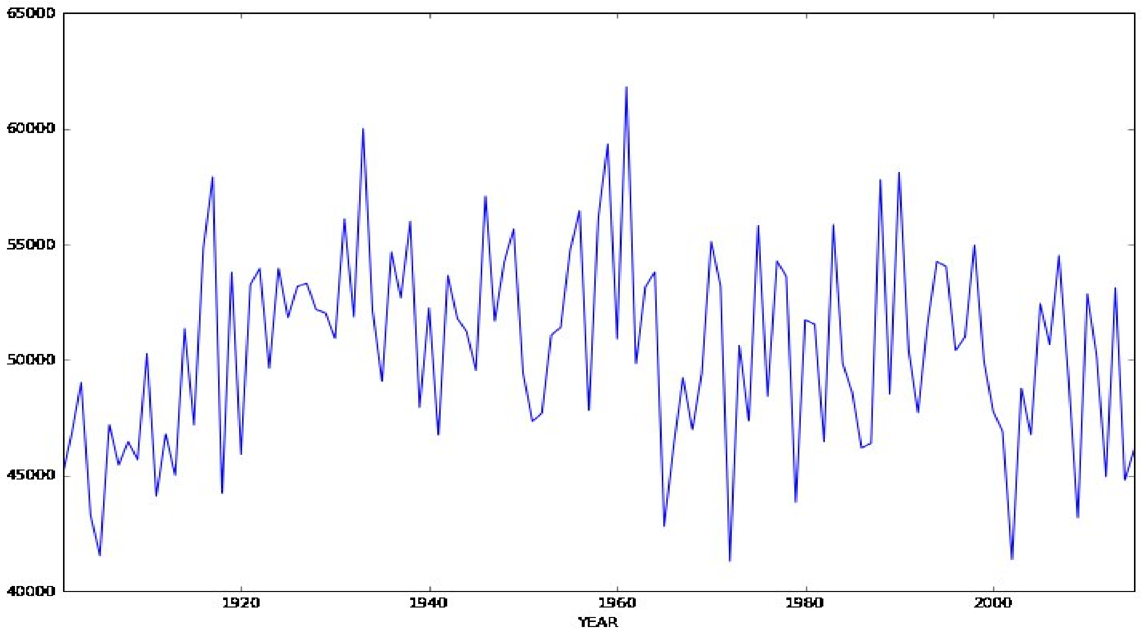
- *rainfall_data.hist(figsize=(12,12));*

This graph shows the distribution of rainfall over months. Increase in amount of rainfall over months July, August, September.

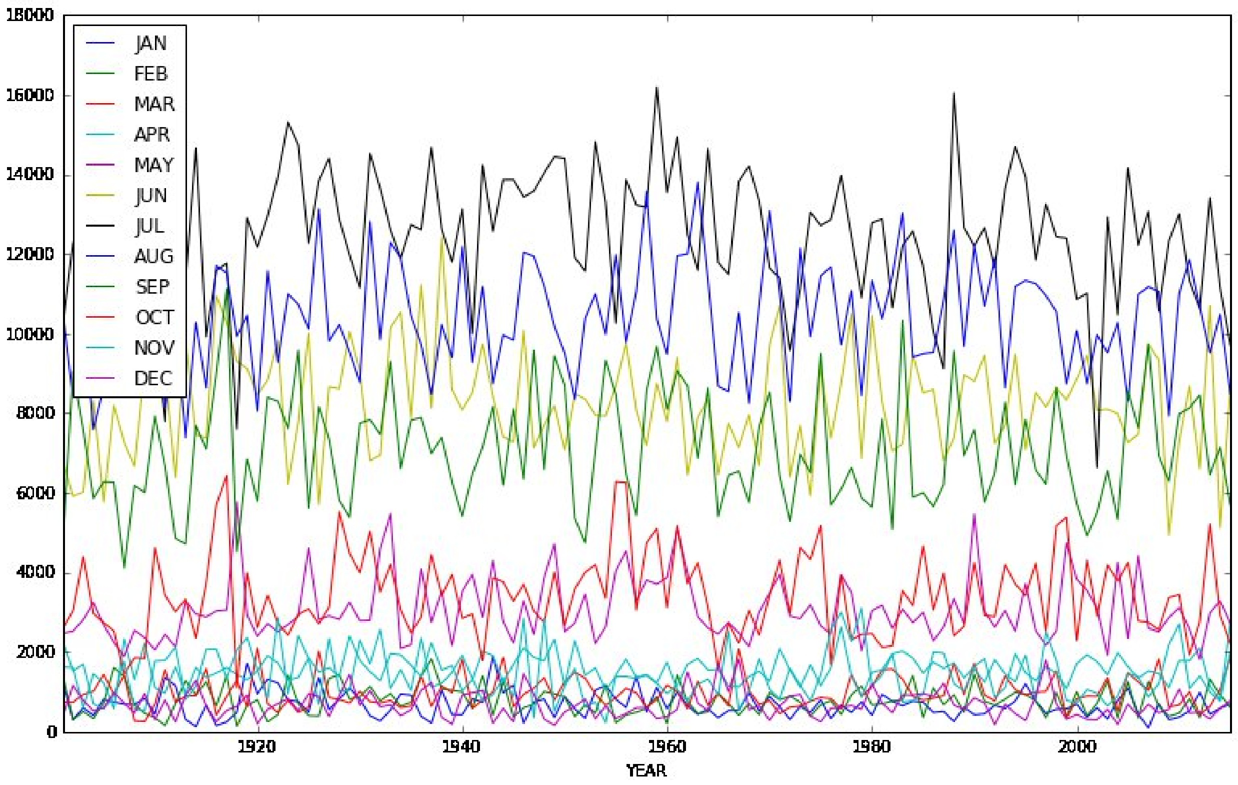


- *rainfall_data.groupby("YEAR").sum()['ANNUAL'].plot(figsize=(12,8));*

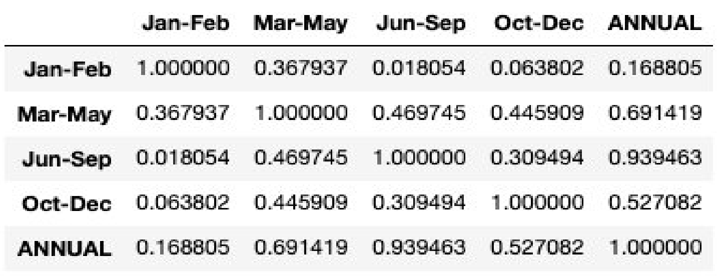
This graph shows the distribution of rainfall over years.Observed high amount of rainfall in 1950s.



- *rainfall_data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(figsize=(13,8));*

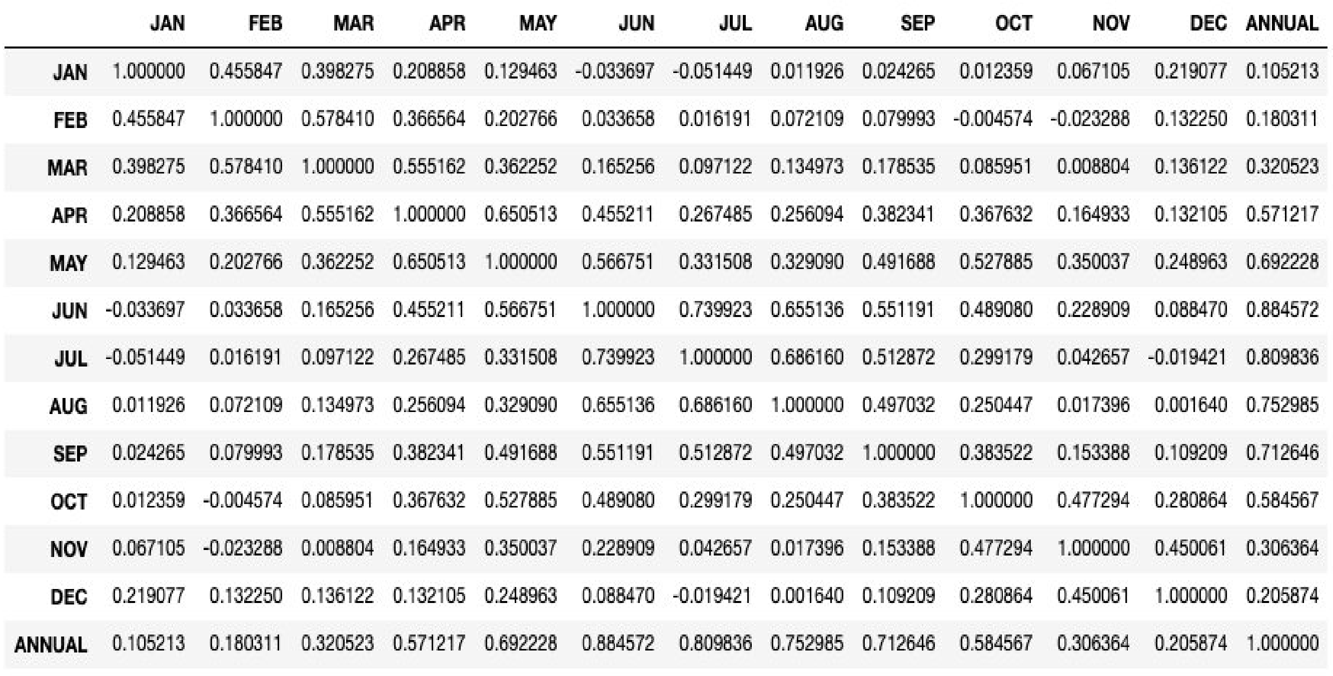


- *rainfall_data[['Jan-Feb','Mar-May','Jun-Sep','Oct-Dec','ANNUAL']].corr()*



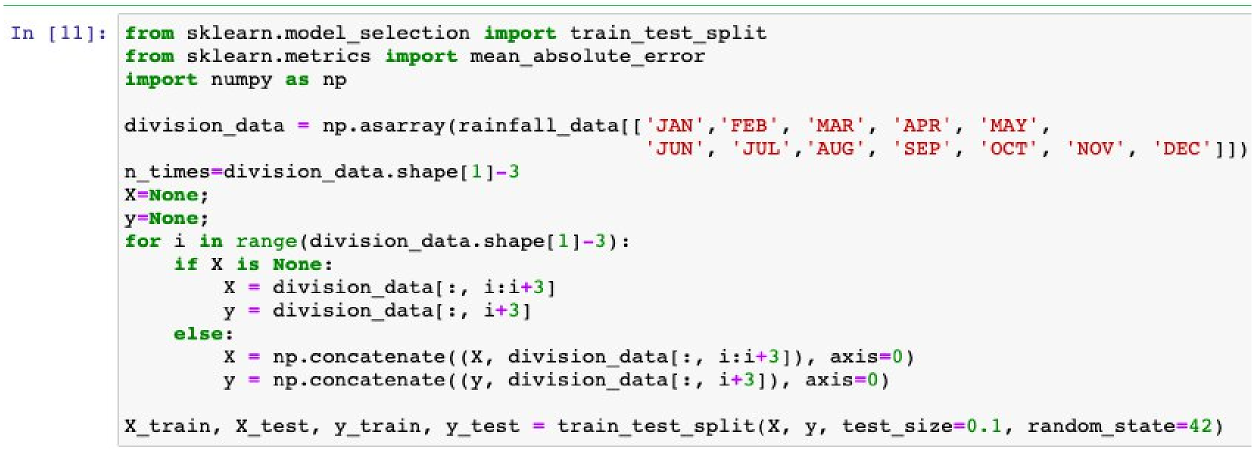
	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec	ANNUAL
Jan-Feb	1.000000	0.367937	0.018054	0.063802	0.168805
Mar-May	0.367937	1.000000	0.469745	0.445909	0.691419
Jun-Sep	0.018054	0.469745	1.000000	0.309494	0.939463
Oct-Dec	0.063802	0.445909	0.309494	1.000000	0.527082
ANNUAL	0.168805	0.691419	0.939463	0.527082	1.000000

- *rainfall_data[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC','ANNUAL']].corr()*



	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL
JAN	1.000000	0.455847	0.398275	0.208858	0.129463	-0.033697	-0.051449	0.011926	0.024265	0.012359	0.067105	0.219077	0.105213
FEB	0.455847	1.000000	0.578410	0.366564	0.202766	0.033658	0.016191	0.072109	0.079993	-0.004574	-0.023288	0.132250	0.180311
MAR	0.398275	0.578410	1.000000	0.555162	0.362252	0.165256	0.097122	0.134973	0.178535	0.085951	0.008804	0.136122	0.320523
APR	0.208858	0.366564	0.555162	1.000000	0.650513	0.455211	0.267485	0.256094	0.382341	0.367632	0.164933	0.132105	0.571217
MAY	0.129463	0.202766	0.362252	0.650513	1.000000	0.566751	0.331508	0.329090	0.491688	0.527885	0.350037	0.248963	0.692228
JUN	-0.033697	0.033658	0.165256	0.455211	0.566751	1.000000	0.739923	0.655136	0.551191	0.489080	0.228909	0.088470	0.884572
JUL	-0.051449	0.016191	0.097122	0.267485	0.331508	0.739923	1.000000	0.686160	0.512872	0.299179	0.042657	-0.019421	0.809836
AUG	0.011926	0.072109	0.134973	0.256094	0.329090	0.655136	0.686160	1.000000	0.497032	0.250447	0.017396	0.001640	0.752985
SEP	0.024265	0.079993	0.178535	0.382341	0.491688	0.551191	0.512872	0.497032	1.000000	0.383522	0.153388	0.109209	0.712646
OCT	0.012359	-0.004574	0.085951	0.367632	0.527885	0.489080	0.299179	0.250447	0.383522	1.000000	0.477294	0.280864	0.584567
NOV	0.067105	-0.023288	0.008804	0.164933	0.350037	0.228909	0.042657	0.017396	0.153388	0.477294	1.000000	0.450061	0.306364
DEC	0.219077	0.132250	0.136122	0.132105	0.248963	0.088470	-0.019421	0.001640	0.109209	0.280864	0.450061	1.000000	0.205874
ANNUAL	0.105213	0.180311	0.320523	0.571217	0.692228	0.884572	0.809836	0.752985	0.712646	0.584567	0.306364	0.205874	1.000000

Seperae training and testing data:

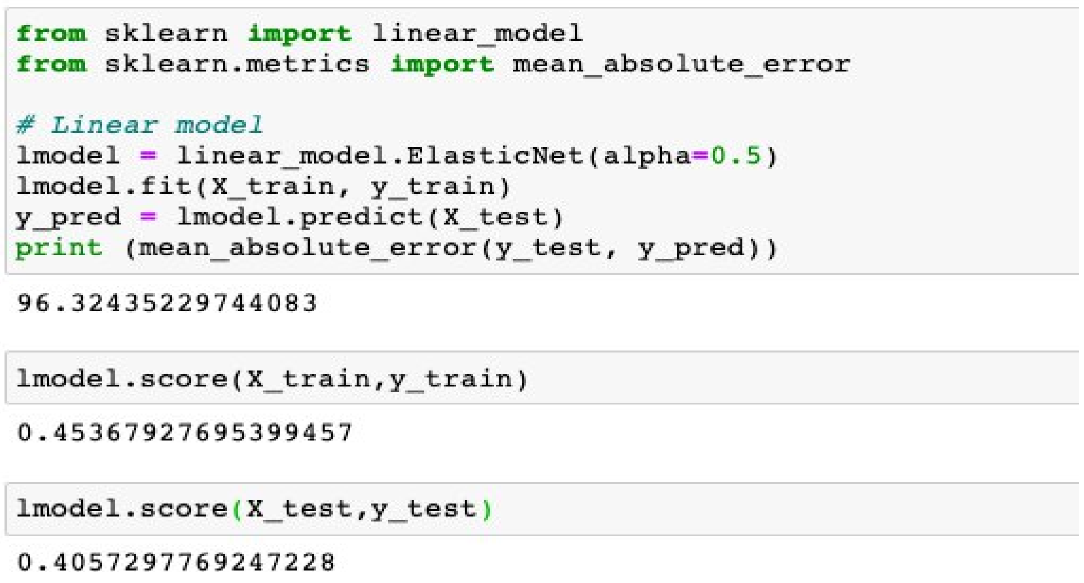


```
In [11]: from sklearn.model_selection import train_test_split
         from sklearn.metrics import mean_absolute_error
         import numpy as np

         division_data = np.asarray(rainfall_data[['JAN','FEB', 'MAR', 'APR', 'MAY',
                                                   'JUN', 'JUL','AUG', 'SEP', 'OCT', 'NOV', 'DEC']])
         n_times=division_data.shape[1]-3
         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)
```

Prediction of Rainfall(in mm) :

Now we will evaluate and analyse the mean absolute value and accuracy between training and testing data with different algoritthm.

Linear Model:



```python
from sklearn import linear_model
from sklearn.metrics import mean_absolute_error

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

```
96.32435229744083
```

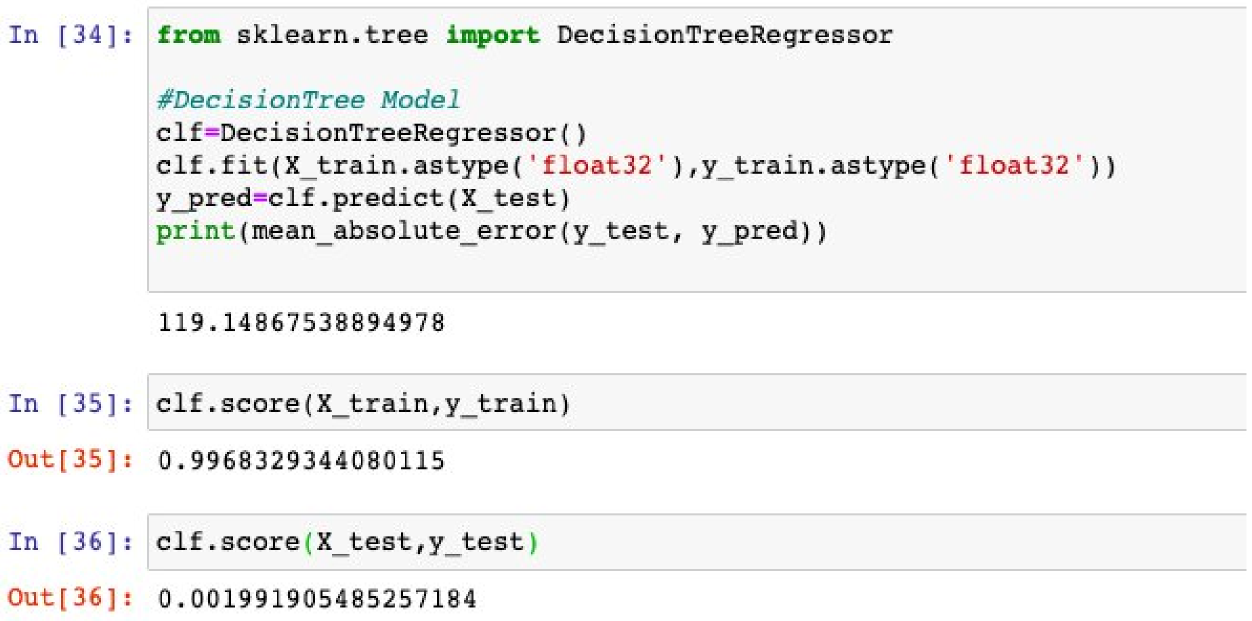
```python
lmodel.score(X_train,y_train)
```

```
0.45367927695399457
```

```python
lmodel.score(X_test,y_test)
```

```
0.4057297769247228
```

Decision Tree Model:



```python
In [34]: from sklearn.tree import DecisionTreeRegressor

         #DecisionTree Model
         clf=DecisionTreeRegressor()
         clf.fit(X_train.astype('float32'),y_train.astype('float32'))
         y_pred=clf.predict(X_test)
         print(mean_absolute_error(y_test, y_pred))
```

```
         119.14867538894978
```

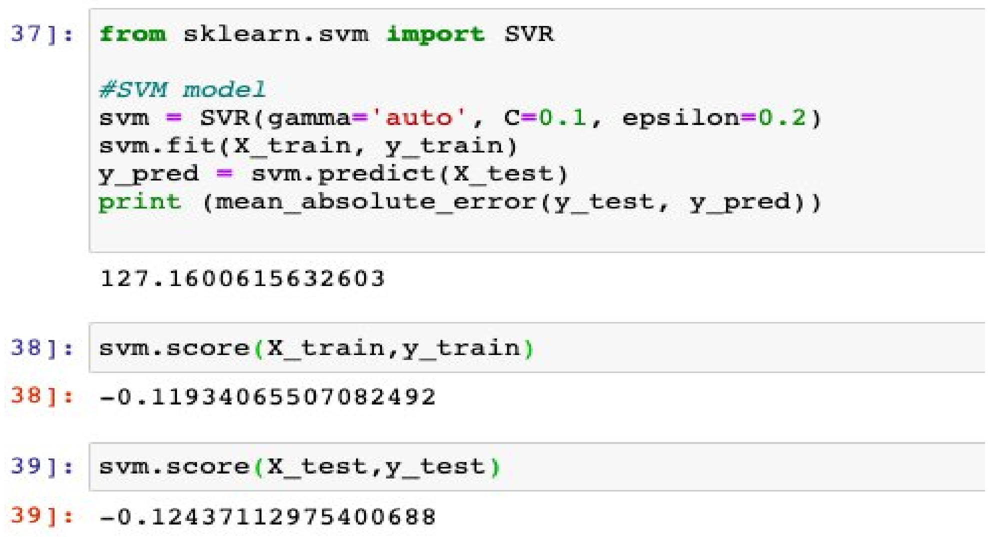
```python
In [35]: clf.score(X_train,y_train)
```

```
Out[35]: 0.9968329344080115
```

```python
In [36]: clf.score(X_test,y_test)
```

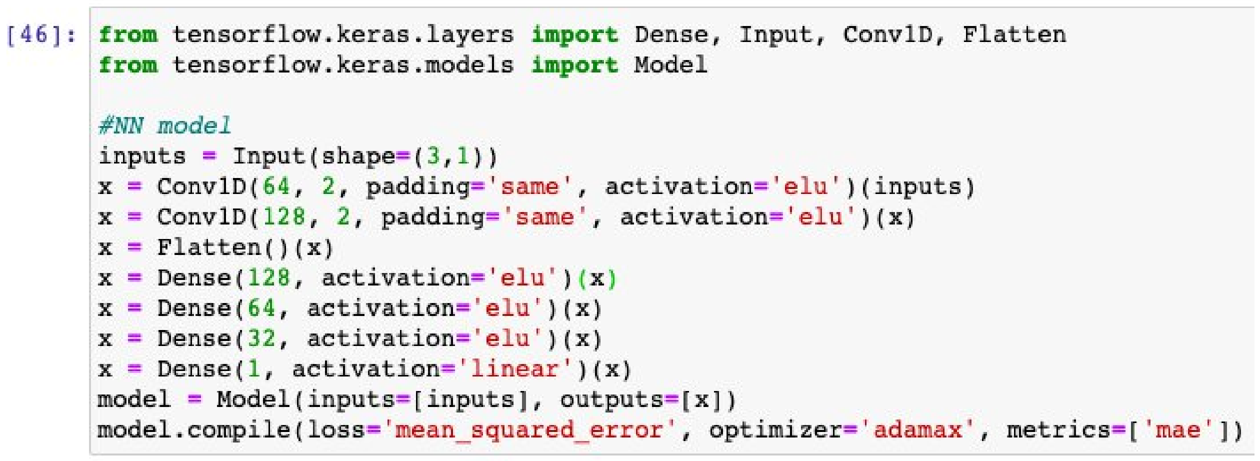
```
Out[36]: 0.001991905485257184
```

SVM Model:

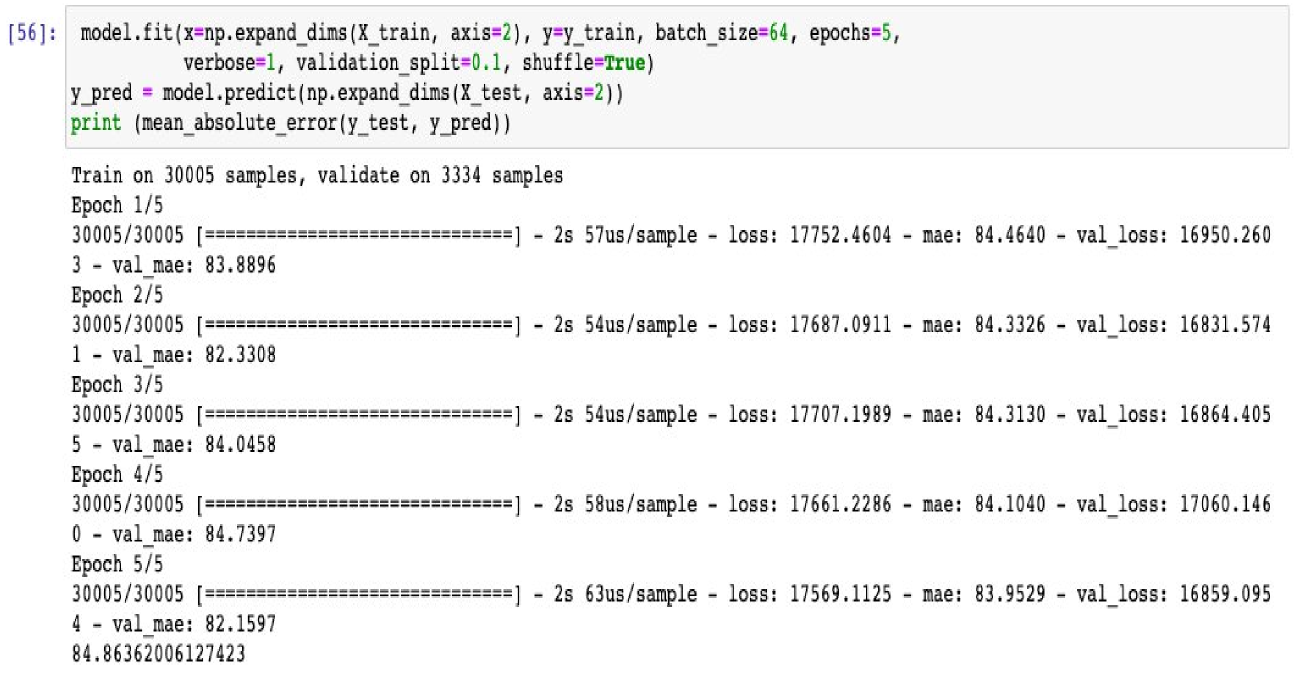


```
37]: from sklearn.svm import SVR

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

     127.1600615632603

38]: svm.score(X_train,y_train)

38]: -0.11934065507082492

39]: svm.score(X_test,y_test)

39]: -0.12437112975400688
```

Neural Network Model:



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

      #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'])
```



```
[56]: model.fit(x=np.expand_dims(X_train, axis=2), y=y_train, batch_size=64, epochs=5,
                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 1/5
      30005/30005 [==============================] - 2s 57us/sample - loss: 17752.4604 - mae: 84.4640 - val_loss: 16950.260
      3 - val_mae: 83.8896
      Epoch 2/5
      30005/30005 [==============================] - 2s 54us/sample - loss: 17687.0911 - mae: 84.3326 - val_loss: 16831.574
      1 - val_mae: 82.3308
      Epoch 3/5
      30005/30005 [==============================] - 2s 54us/sample - loss: 17707.1989 - mae: 84.3130 - val_loss: 16864.405
      5 - val_mae: 84.0458
      Epoch 4/5
      30005/30005 [==============================] - 2s 58us/sample - loss: 17661.2286 - mae: 84.1040 - val_loss: 17060.146
      0 - val_mae: 84.7397
      Epoch 5/5
      30005/30005 [==============================] - 2s 63us/sample - loss: 17569.1125 - mae: 83.9529 - val_loss: 16859.095
      4 - val_mae: 82.1597
      84.86362006127423
```

Analysis and Conclusion:

Algorithm	Mean Error value
Linear Regression	96.3243
Decision Tree Regression	119.1486
SVR	127.16
Neural network	84.8636

From observed Mean Error value, we can analyse that Neural network model is more efficient to predict the rainfall. Decision Tree regression and SVR model, both are worst to predict the rainfalll over months. Linear regression is moderate model to predict the model. As accuracy in Linear model is also not good. 5% difference is enough to reject Linear Model.

So Neural network (NN) model should be preferred over Linear Regreesion, Decision Tree regression, Simple Vector regression (SVR).

Bibiliography:

https://www.kaggle.com/

https://scikit-learn.org/

https://towardsdatascience.com/