# Data Extraction

In [1]:

```
# importing libraries
import numpy as np
import pandas as pd
df = pd.read_csv(r'C:\Users\I DEEPIKA\Documents\kerala.csv')
data=pd.DataFrame(df)
data
```

Out[1]:

| | SUBDIVISION | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KERALA | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 350.8 | 48.4 |
| **1** | KERALA | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 158.3 | 121.5 |
| **2** | KERALA | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 157.0 | 59.0 |
| **3** | KERALA | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 33.9 | 3.3 |
| **4** | KERALA | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 74.4 | 0.2 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **113** | KERALA | 2014 | 4.6 | 10.3 | 17.9 | 95.7 | 251.0 | 454.4 | 677.8 | 733.9 | 298.8 | 355.5 | 99.5 | 47.2 |
| **114** | KERALA | 2015 | 3.1 | 5.8 | 50.1 | 214.1 | 201.8 | 563.6 | 406.0 | 252.2 | 292.9 | 308.1 | 223.6 | 79.4 |
| **115** | KERALA | 2016 | 2.4 | 3.8 | 35.9 | 143.0 | 186.4 | 522.2 | 412.3 | 325.5 | 173.2 | 225.9 | 125.4 | 23.6 |

# Summarization

In [2]:

```
data.head()
```

Out[2]:

| | SUBDIVISION | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KERALA | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 35 |
| **1** | KERALA | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 15 |
| **2** | KERALA | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 15 |
| **3** | KERALA | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 3 |
| **4** | KERALA | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 1 |

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118 entries, 0 to 117
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   SUBDIVISION       118 non-null    object 
 1   YEAR              118 non-null    int64  
 2   JAN               118 non-null    float64
 3   FEB               118 non-null    float64
 4   MAR               118 non-null    float64
 5   APR               118 non-null    float64
 6   MAY               118 non-null    float64
 7   JUN               118 non-null    float64
 8   JUL               118 non-null    float64
 9   AUG               118 non-null    float64
 10  SEP               118 non-null    float64
 11  OCT               118 non-null    float64
 12  NOV               118 non-null    float64
 13  DEC               118 non-null    float64
 14   ANNUAL RAINFALL  118 non-null    float64
 15  FLOODS            118 non-null    object 
dtypes: float64(13), int64(1), object(2)
memory usage: 14.9+ KB
```

In [4]:

```
data.shape
```

Out[4]:

```
(118, 16)
```

In [5]:

```
data.describe()
```

Out[5]:

| | YEAR | JAN | FEB | MAR | APR | MAY | JUN |
|---|---|---|---|---|---|---|---|
| **count** | 118.000000 | 118.000000 | 118.000000 | 118.000000 | 118.000000 | 118.000000 | 118.000000 |
| **mean** | 1959.500000 | 12.218644 | 15.633898 | 36.670339 | 110.330508 | 228.644915 | 651.617797 |
| **std** | 34.207699 | 15.473766 | 16.406290 | 30.063862 | 44.633452 | 147.548778 | 186.181363 |
| **min** | 1901.000000 | 0.000000 | 0.000000 | 0.100000 | 13.100000 | 53.400000 | 196.800000 |
| **25%** | 1930.250000 | 2.175000 | 4.700000 | 18.100000 | 74.350000 | 125.050000 | 535.550000 |
| **50%** | 1959.500000 | 5.800000 | 8.350000 | 28.400000 | 110.400000 | 184.600000 | 625.600000 |
| **75%** | 1988.750000 | 18.175000 | 21.400000 | 49.825000 | 136.450000 | 264.875000 | 786.975000 |
| **max** | 2018.000000 | 83.500000 | 79.000000 | 217.200000 | 238.000000 | 738.800000 | 1098.200000 |

In [6]:

```
data.columns
```

Out[6]:

```
Index(['SUBDIVISION', 'YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JU
L',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC', ' ANNUAL RAINFALL', 'FLOODS'],
      dtype='object')
```

# Data Processing

In [7]:

```
# Data cleaning
data.isnull().sum()
```

Out[7]:

```
SUBDIVISION         0
YEAR                0
JAN                 0
FEB                 0
MAR                 0
APR                 0
MAY                 0
JUN                 0
JUL                 0
AUG                 0
SEP                 0
OCT                 0
NOV                 0
DEC                 0
 ANNUAL RAINFALL    0
FLOODS              0
dtype: int64
```

In [8]:

```
#We want the data in numbers, therefore we will replace the yes/no in floods coloumn by 1/0
data['FLOODS'].replace(['YES','NO'],[1,0],inplace=True)
```

In [9]:

```
data.head()
```

Out[9]:

| | SUBDIVISION | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KERALA | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 35 |
| **1** | KERALA | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 15 |
| **2** | KERALA | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 15 |
| **3** | KERALA | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 3 |
| **4** | KERALA | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 7 |

In [10]:

```
#Now let's seperate the data which we are gonna use for prediction
x = data.iloc[:,1:14]
x.head()
```

Out[10]:

| | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1901 | 28.7 | 44.7 | 51.6 | 160.0 | 174.7 | 824.6 | 743.0 | 357.5 | 197.7 | 266.9 | 350.8 | 48.4 |
| **1** | 1902 | 6.7 | 2.6 | 57.3 | 83.9 | 134.5 | 390.9 | 1205.0 | 315.8 | 491.6 | 358.4 | 158.3 | 121.5 |
| **2** | 1903 | 3.2 | 18.6 | 3.1 | 83.6 | 249.7 | 558.6 | 1022.5 | 420.2 | 341.8 | 354.1 | 157.0 | 59.0 |
| **3** | 1904 | 23.7 | 3.0 | 32.2 | 71.5 | 235.7 | 1098.2 | 725.5 | 351.8 | 222.7 | 328.1 | 33.9 | 3.3 |
| **4** | 1905 | 1.2 | 22.3 | 9.4 | 105.9 | 263.3 | 850.2 | 520.5 | 293.6 | 217.2 | 383.5 | 74.4 | 0.2 |

In [11]:

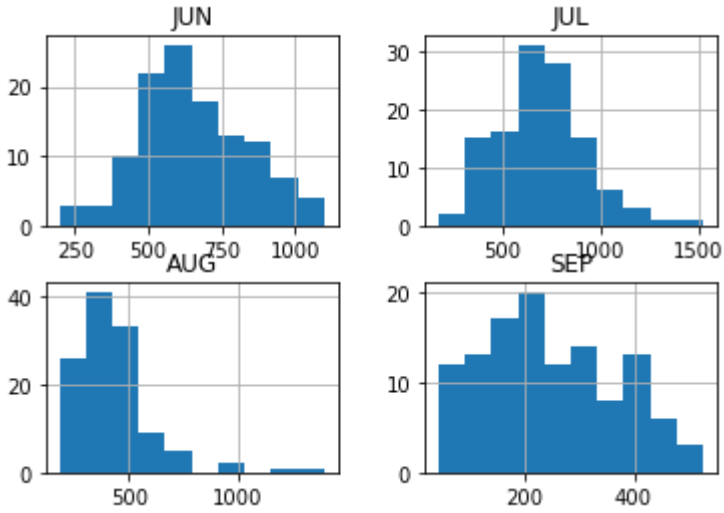
```
#Now seperate the flood label from the dataset
y = data.iloc[:, -1]
y.head()
```

Out[11]:

```
0    1
1    1
2    1
3    1
4    0
Name: FLOODS, dtype: int64
```

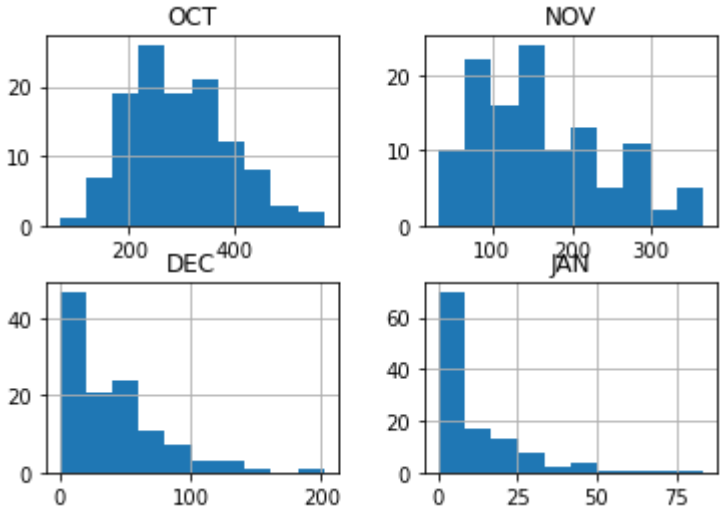
In [12]:

```
#Let's see hoe the rainfall index vary during rainy season
import matplotlib.pyplot as plt
%matplotlib inline
a = data[['JUN','JUL','AUG','SEP']]
a.hist()
plt.show()
```



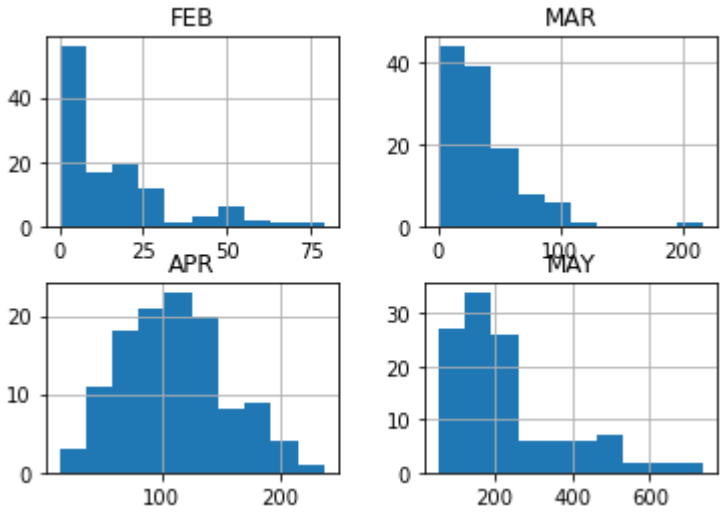
In [13]:

```
import matplotlib.pyplot as plt
%matplotlib inline
b = data[['OCT','NOV','DEC','JAN']]
b.hist()
plt.show()
```




In [14]:

```
import matplotlib.pyplot as plt
%matplotlib inline
c = data[['FEB','MAR','APR','MAY']]
c.hist()
plt.show()
```




In [15]:

```
#Let's divide the dataset into 2 sets:train and test in ratio (4:1)
from sklearn import model_selection,neighbors
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [16]:

```
#Let's see how our train set looks like
x_train.head()
```

Out[16]:

| | YEAR | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **28** | 1929 | 12.8 | 29.8 | 58.9 | 210.7 | 148.0 | 946.6 | 844.0 | 293.9 | 268.9 | 350.4 | 158.2 | 39.4 |
| **104** | 2005 | 19.8 | 7.0 | 25.3 | 205.9 | 134.8 | 619.2 | 832.7 | 291.0 | 414.7 | 240.1 | 184.3 | 56.4 |
| **70** | 1971 | 31.6 | 18.5 | 20.0 | 113.0 | 317.5 | 889.6 | 648.6 | 385.2 | 331.2 | 220.9 | 38.3 | 62.3 |
| **113** | 2014 | 4.6 | 10.3 | 17.9 | 95.7 | 251.0 | 454.4 | 677.8 | 733.9 | 298.8 | 355.5 | 99.5 | 47.2 |
| **47** | 1948 | 43.0 | 8.3 | 48.2 | 125.0 | 212.3 | 910.2 | 619.0 | 487.9 | 166.6 | 183.9 | 215.6 | 19.2 |

In [17]:

```
y_train.head()
```

Out[17]:

```
28     1
104    1
70     1
113    1
47     1
Name: FLOODS, dtype: int64
```

# Logistic regression

In [18]:

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr_clf = lr.fit(x_train,y_train)

lr_accuracy = cross_val_score(lr_clf,x_test,y_test,cv=3,scoring='accuracy',n_jobs=-1)
```

```
C:\Users\I DEEPIKA\anaconda3\lib\site-packages\sklearn\linear_model\_logisti
c.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [19]:

```
lr_accuracy
```

Out[19]:

```
array([0.875, 0.875, 0.75 ])
```

In [20]:

```
y_predict = lr_clf.predict(x_test)
print('Predicted chances of flood')
y_predict
```

```
Predicted chances of flood
```

Out[20]:

```
array([1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 1], dtype=int64)
```

In [21]:

```
print('Actual chances of flood')
print(y_test.values)
```

```
Actual chances of flood
[1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1]
```

In [22]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
print("\naccuracy score: %f"%(accuracy_score(y_test,y_predict)*100))
```

```
accuracy score: 91.666667
```

# Decision tree classification

In [23]:

```
from sklearn.tree import DecisionTreeClassifier
dtc_clf = DecisionTreeClassifier()
dtc_clf.fit(x_train,y_train)
dtc_clf_acc = cross_val_score(dtc_clf,x_train,y_train,cv=3,scoring="accuracy",n_jobs=-1)
dtc_clf_acc
```

Out[23]:

```
array([0.65625   , 0.77419355, 0.77419355])
```

In [24]:

```
#Predicted flood chances
y_pred = dtc_clf.predict(x_test)
print(y_pred)
```

```
[0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 1 1 0 1]
```

In [25]:

```
#Actual flood chances
print("actual values:")
print(y_test.values)
```

```
actual values:
[1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1]
```

In [26]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
print("\naccuracy score:%f"%(accuracy_score(y_test,y_pred)*100))
```

```
accuracy score:66.666667
```

# KNN Classifier

In [27]:

```
clf = neighbors.KNeighborsClassifier()
knn_clf = clf.fit(x_train,y_train)
```

In [28]:

```
#Let's predict chances of flood
y_predict = knn_clf.predict(x_test)
print('predicted chances of flood')
print(y_predict)
```

```
predicted chances of flood
[1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1]
```

In [29]:

```
#Actual chances of flood
print("actual values of floods:")
print(y_test)
```

```
actual values of floods:
48     1
31     1
12     0
87     0
4      0
73     0
80     1
58     1
17     0
13     0
96     1
99     0
6      1
111    0
14     1
79     0
45     1
116    0
76     1
38     1
20     0
72     0
75     0
40     1
Name: FLOODS, dtype: int64
```

In [30]:

```
from sklearn.model_selection import cross_val_score
knn_accuracy = cross_val_score(knn_clf,x_test,y_test,cv=3,scoring='accuracy',n_jobs=-1)
knn_accuracy.mean()
```

Out[30]:

```
0.7916666666666666
```

# Task

In [31]:

```
x1=[[2020,0,0,0,0,0,653.6,687.2,0,0,0,0,0]]
y1=lr.predict(x1)
y1
```

```
C:\Users\I DEEPIKA\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarn
ing: X does not have valid feature names, but LogisticRegression was fitted
with feature names
  warnings.warn(
```

Out[31]:

```
array([0], dtype=int64)
```

In [ ]:

In [ ]: