190031920

A Nikhil Reddy

DS Practical 5

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv("kerala.csv")
df
```

Out[2]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FI
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	3248.6	
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	3326.6	
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	3271.2	
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	3129.7	
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	2741.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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	3046.4	
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	2600.6	
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	2176.6	
116	KERALA	2017	1.9	6.8	8.9	43.6	173.5	498.5	319.6	531.8	209.5	192.4	92.5	38.1	2117.1	
117	KERALA	2018	29.1	52.1	48.6	116.4	183.8	625.4	1048.5	1398.9	423.6	356.1	125.4	65.1	4473.0	

118 rows × 16 columns

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

Out[3]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FLOO
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	3248.6	Y
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	3326.6	Y
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	3271.2	Y
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	3129.7	Y
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	2741.6	

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

Out[4]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG
count	118.000000	118.000000	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	698.220339	430.369492
std	34.207699	15.473766	16.406290	30.063862	44.633452	147.548778	186.181363	228.988966	181.980463
min	1901.000000	0.000000	0.000000	0.100000	13.100000	53.400000	196.800000	167.500000	178.600000
25%	1930.250000	2.175000	4.700000	18.100000	74.350000	125.050000	535.550000	533.200000	316.725000
50%	1959.500000	5.800000	8.350000	28.400000	110.400000	184.600000	625.600000	691.650000	386.250000
75%	1988.750000	18.175000	21.400000	49.825000	136.450000	264.875000	786.975000	832.425000	500.100000
max	2018.000000	83.500000	79.000000	217.200000	238.000000	738.800000	1098.200000	1526.500000	1398.900000

In [5]:
```
df.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 [6]:
```
# Changing the target column to numeric values
df["FLOODS"] = df["FLOODS"].map({"YES": 1, "NO": 0})
```

In [7]:
```
df["JUN_GT_500"] = (df["JUN"] > 500).astype("int")
df["JUL_GT_500"] = (df["JUL"] > 500).astype("int")
df_small = df.loc[:, ["YEAR", "JUN_GT_500", "JUL_GT_500", "FLOODS"]]
df_small["COUNT"] = 1
df_small.head()
```

Out[7]:

	YEAR	JUN_GT_500	JUL_GT_500	FLOODS	COUNT
0	1901	1	1	1	1
1	1902	0	1	1	1
2	1903	1	1	1	1
3	1904	1	1	1	1
4	1905	1	1	0	1

In [8]:
```
df_small.shape
```

Out[8]:
```
(118, 5)
```

In [9]:
```
# Creating the tabular data based on the counts
pd.crosstab(df_small["FLOODS"], df_small["JUN_GT_500"])
```

Out[9]:

JUN_GT_500	0	1
FLOODS		
0	19	39
1	6	54

In [10]:
```
P_F = (6 + 54) / (6 + 54 + 19 + 39)
P_J = (39 + 54) / (6 + 54 + 19 + 39)
P_F_intersect_J = 54 / (6 + 54 + 19 + 39)
print(f"P(F): {P_F}")
print(f"P(J): {P_J}")
print(f"P(F AND J): {P_F_intersect_J}")
```

```
P(F): 0.5084745762711864
P(J): 0.788135593220339
P(F AND J): 0.4576271186440678
```

In [11]:
```
# Now calculate probailitity of flood given it rained more than 500 mm in June (P(A|B))
P_F_J = P_F_intersect_J / P_J
print(f"P(F|J): {P_F_J}")
```

```
P(F|J): 0.5806451612903226
```

In [12]:
```
# Probability of rain more than 500 mm in June given it flooded that year (P(B|A))
P_J_F = (P_F_J * P_J) / P_F
print(f"P(J|F): {P_J_F}")
```

```
P(J|F): 0.9000000000000001
```

In [13]:
```
# We can similarly do it for july
pd.crosstab(df_small["FLOODS"], df_small["JUL_GT_500"])
```

Out[13]:

JUL_GT_500	0	1
FLOODS		
0	19	39
1	3	57

In [14]:
```
P_F = (3 + 57) / (3 + 57 + 19 + 39)
P_J = (39 + 57) / (3 + 57 + 19 + 39)
P_F_intersect_J = 57 / (3 + 57 + 19 + 39)
print(f"P(F): {P_F}")
print(f"P(J): {P_J}")
print(f"P(F AND J): {P_F_intersect_J}")
```

```
P(F): 0.5084745762711864
P(J): 0.8135593220338984
P(F AND J): 0.4830508474576271
```

In [15]:
```
# Now calculate probailitity of flood given it rained more than 500 mm in July
P_F_J = P_F_intersect_J / P_J
print(f"P(F|J): {P_F_J}")
```

```
P(F|J): 0.59375
```

In [16]:
```
# Probability of rain more than 500 mm in July given it flooded that year (P(B|A))
P_J_F = (P_F_J * P_J) / P_F
print(f"P(J|F): {P_J_F}")
```

```
P(J|F): 0.9500000000000002
```