MACHINE LEARNING

MSMA-218 PRACTICAL FILE

M.Sc. Mathematics

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

Submitted to:

Prof. Anjana Gupta
Ms. Shweta Kalson

Submitted by:

Ritika Gupta
2K22/MSCMAT/54

INDEX

S.No.	Title
1	To implement the model $f_{w,b}$ for linear regression with one variable.
2	To implement and explore the cost function for linear regression with one variable.
3	To optimize w and b using gradient descent for linear regression.
4	Perform data visualization on the given dataset.
5	Apply Linear Regression on the given dataset.
6	Apply Logistic Regression on the given dataset.
7	Apply K means clustering on the given dataset.
8	Apply Principal Component Analysis on the given dataset.
9	Apply Neural Network on the given dataset.
10	Apply Decision Tree and Random Forest on the given dataset.
11	Annexure: Datasets used

0.1 PRACTICAL 1: To implement the model $f_{w,b}$ for linear regression with one variable.

Problem Statement:

Use the example of housing price prediction. Use a simple data set with only two data points - a house with 1000 square feet(sqft) sold for \$300,000 and a house with 2000 square feet sold for \$500,000. These two points will constitute our data or training set. In this lab, the units of size are 1000 sqft and the units of price are 1000s of dollars. Fit a linear regression model through these two points, so you can then predict price for other houses- say, a house with 1200 sqft.

[1]:
```
# initializing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
# making dataframe for the given training data

df = pd.DataFrame(data = {'Size in feet$^2$':[1000,2000],'Price in \$1000(s)':
 ↪[300,500]})
df
```

[2]:
```
   Size in feet$^2$  Price in \$1000(s)
0              1000                 300
1              2000                 500
```

[3]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

[4]:
```
# fitting linear regression model on the training data

x_train, y_train = np.array(df['Size in feet$^2$']).reshape(-1,1),np.
 ↪array(df['Price in \\$1000(s)'])
model.fit(x_train,y_train)
```

[4]:
```
LinearRegression()
```

[5]:
```
# predicting price for the test data sample size=1200

x_test=np.array([1200]).reshape(1,-1)
y_pred = model.predict(x_test)
y_pred
```

[5]:
```
array([340.])
```

[6]:
```
# visualizing linear regression

fig,ax = plt.subplots(1,1,figsize=(5,3.5))
ax.plot(x_train,model.predict(x_train))
ax.scatter(x_test,y_pred,color='g')
ax.grid(True,alpha=0.4,lw=0.5)
ax.set_xlabel('Size in feet$^2$')
ax.set_ylabel('Price in \$1000(s)')
ax.set_title('House price prediction',fontsize=10)
plt.show()
```

[7]:
```
# linear regression equation for this model
w,b = model.coef_,model.intercept_
print('Price = ',w[0],'* Size +',b)
```

```
Price =  0.2 * Size + 100.0
```

0.2 PRACTICAL 2: To implement and explore the cost function for linear regression with one variable.

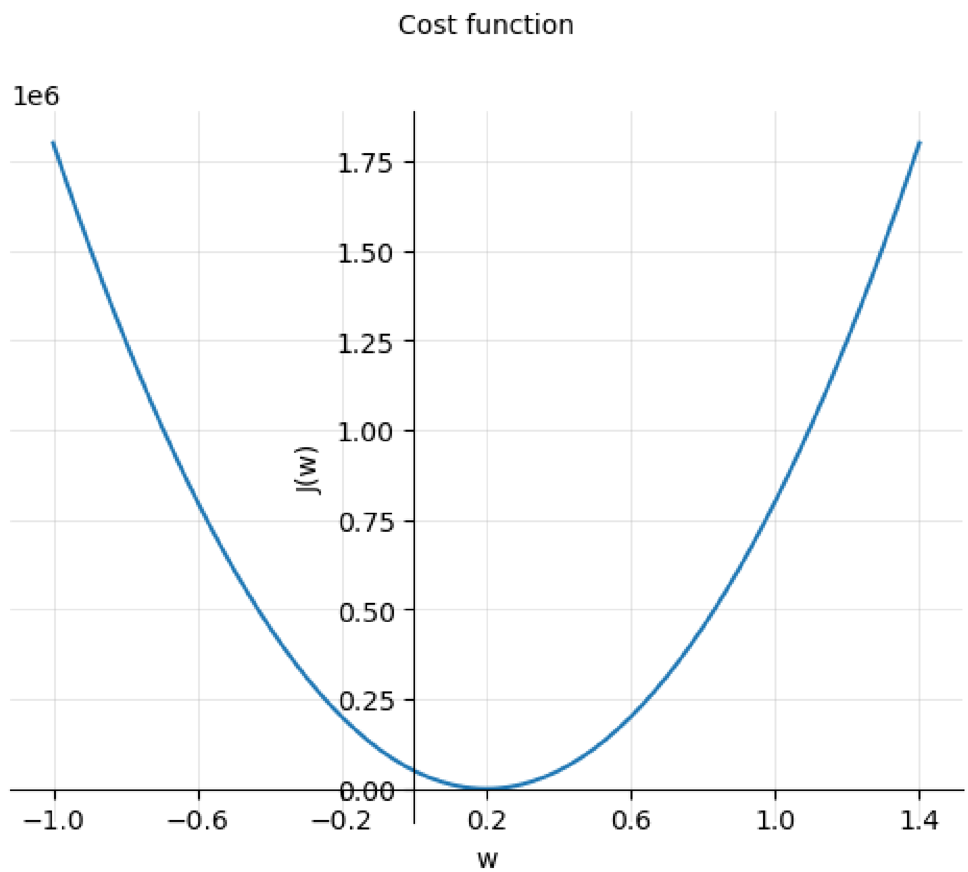
Problem Statement:

Using the same data as in Practical 1, your goal is to find a model $f_{w,b}(x) = wx + b$, with parameters w, b, which will accurately predict house values given an input x. The cost is a measure of how accurate the model is on the training data. The cost function shows that if w and b can be selected such that the predictions $f_{w,b}(x)$ match the target data y, then the $(f_{w,b}(x) - y)^2$ term will be zero and the cost minimized. In the previous lab, we determined that b = 100 provided an optimal solution so let's set b to 100 and focus on w.

[8]:
```
# computing the cost function for varying values of w

def Compute_Cost(x,y,m,w,b):
    j=0
    for i in range(0,m):
        j = j +  ((w*x[i][0] + b) - y[i])**2
    j = j/(2*m)
    return j

k=0
J = np.arange(-1,1.41,0.02)

for w in np.arange(-1,1.41,0.02):
    J[k] = Compute_Cost(x_train,y_train,2,w,model.intercept_)
    k=k+1
```

[9]:
```
# plotting curve for the cost function

fig,ax = plt.subplots(1,1)

ax.plot(np.arange(-1,1.41,0.02),J)
ax.grid(True)
ax.grid(alpha=0.4,lw=0.5)
ax.set_xlabel('w')
ax.set_ylabel('J(w)')
```

```python
ax.set_title('Cost  function\n\n',fontsize=10)

# x tick  labels
ax.set_xticks([-1,-0.6,-0.2,0.2,0.6,1,1.4])
# x axis passes through origin
ax.spines['bottom'].set_position(('data',0))
# y axis passes through origin
ax.spines['left'].set_position(('data',0))
# turning off  right  and  top  axis  spines
for pos in ['right','top']:
    ax.spines[pos].set_visible(False)

plt.show()
```



[10]:
```python
# Thus, w = 0.2 is the ideal value for this model because J(0.2) = 0
```

[ ]:
```python

```

[ ]:
```python

```

0.3 PRACTICAL 3: To optimize *w* and *b* using gradient descent for linear regression.

Problem Statement:

Use the same two data points as before - a house with 1000 square feet sold for \$300,000 and a house with 2000 square feet sold for \$500,000. Implement gradient descent algorithm for one feature using these three functions: - compute_gradient - compute_cost - gradient_descent

[11]:
```
def Compute_Gradient(x,y,m,w,b):
    dj_dw,dj_db = 0,0

    for i in range(0,m):
        dj_dw = dj_dw +  (((w*x[i][0] + b) - y[i])*x[i][0])
        dj_db = dj_db +  ((w*x[i][0] + b) - y[i])

    dj_dw = dj_dw/m
    dj_db = dj_db/m

    return dj_dw,dj_db
```

[12]:
```
def Gradient_Descent(x,y,m,w_init,b_init,a):
    w,b = w_init,b_init
    J = Compute_Cost(x,y,m,w,b)
    dJ_dw,dJ_db = Compute_Gradient(x,y,m,w,b)
    w = w - a*dJ_dw
    b = b - a*dJ_db
    return w,b,J
```

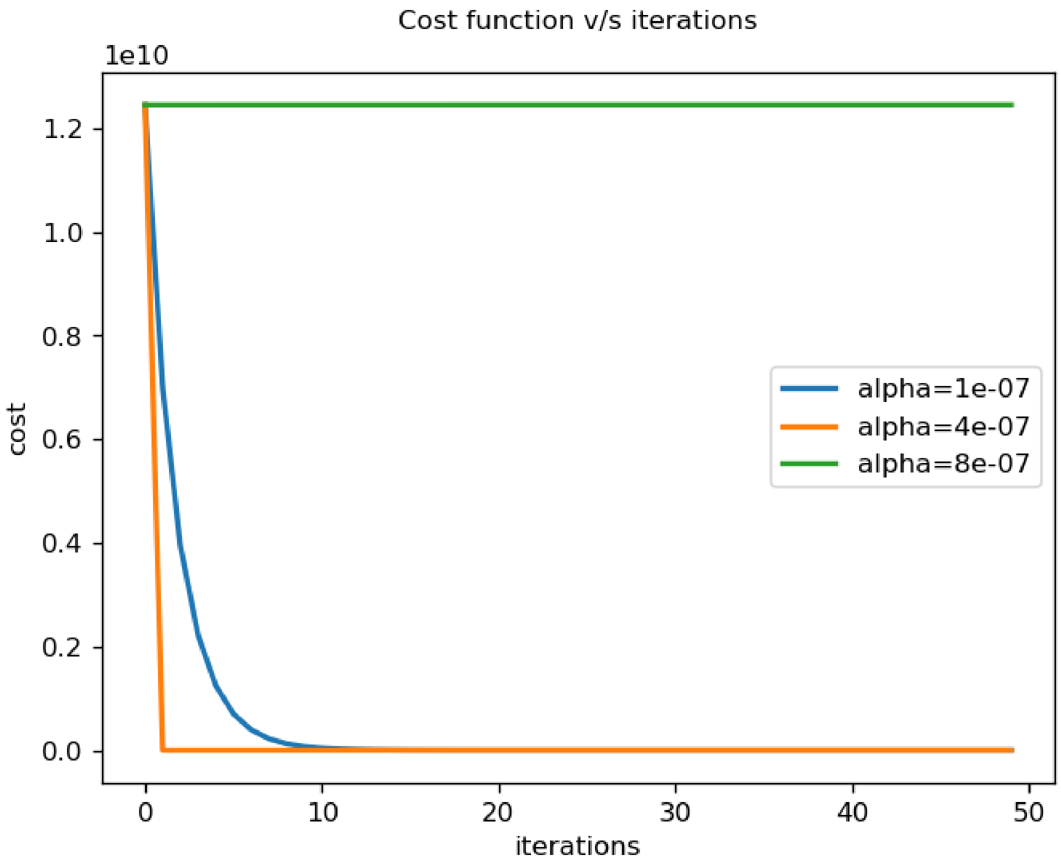
[13]:
```
j = np.zeros((50,3))
k=0
W,B = 0,0

for alpha in [0.0000001,0.0000004,0.0000008]:
    w,b=100,100
    for i in range(0,50):
        w,b,J = Gradient_Descent(x_train,y_train,2,w,b,alpha)
```

```
        j[i][k]=J
    k=k+1
    if J<0.1:
        W,B=w,b
    print('\nAfter 50 iterations for alpha = {},\nw = {}, b = {}, J = {}'.
↪format(alpha,w,b,J))
```

```
After 50 iterations for alpha = 1e-07,
w = 0.20009244651382851, b = 99.94012008301269, J = 0.007277832962203829

After 50 iterations for alpha = 4e-07,
w = 0.20003592791808425, b = 99.94012013892144, J = 0.00017927990248 22805

After 50 iterations for alpha = 8e-07,
w = 100.00359286208413, b = 100.00000239523678, J = 12450928506.240183
```

[14]:

```
# plotting cost v/s iterations curve

fig,ax = plt.subplots(1,1,dpi=120)
x = range(50)

ax.plot(x,j[:,0],label='alpha=1e-07',lw=2)
ax.plot(x,j[:,1],label='alpha=4e-07',lw=2)
ax.plot(x,j[:,2],label='alpha=8e-07',lw=2)
ax.set_xlabel('iterations')
ax.set_ylabel('cost')
ax.set_title('Cost function v/s iterations\n',fontsize=10)
ax.legend()

plt.show()
```



[15]:
```
print('Thus, w converges to {} and b converges to {}\n=> f_wb(x) = {}x + {}'.
  ↪format(round(W,1),round(B),round(W,1),round(B)))
```

```
Thus, w converges to 0.2 and b converges to 100
=> f_wb(x) = 0.2x + 100
```

0.4 PRACTICAL 4: Perform data visualization on crime dataset

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

[7]:
```
df = pd.read_excel("3A.3 Women & Girls Victims of Rape - 2020.xlsx")
df.head(3).transpose()
```

[7]:

	0	1	2
State/UT	Andhra Pradesh	Arunachal Pradesh	Assam
Cases Reported	1095.0	60.0	1657.0
Below 6 Years	27.0	1.0	0.0
6 Years & Above - Below 12 Years	74.0	6.0	0.0
12 Years & Above - Below 16 Years	214.0	14.0	6.0
16 Years & Above - Below 18 Years	272.0	8.0	12.0
Total Girl / Child Victims	587.0	29.0	18.0
18 Years & Above - Below 30 Years	411	20	1006
30 Years & Above - Below 45 Years	87.0	14.0	584.0
45 Years & Above - Below 60 Years	20.0	0.0	50.0
60\nYears & Above	2.0	0.0	0.0
Total Women\n/ Adult Victims	520.0	34.0	1640.0
Total	1107	63	1658

[441]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 13 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   State/UT                           38 non-null     object
 1   Cases Reported                     37 non-null     float64
 2   Below 6 Years                      38 non-null     float64
 3   6 Years & Above - Below 12 Years   38 non-null     float64
 4   12 Years & Above - Below 16 Years  38 non-null     float64
 5   16 Years & Above - Below 18 Years  38 non-null     float64
```

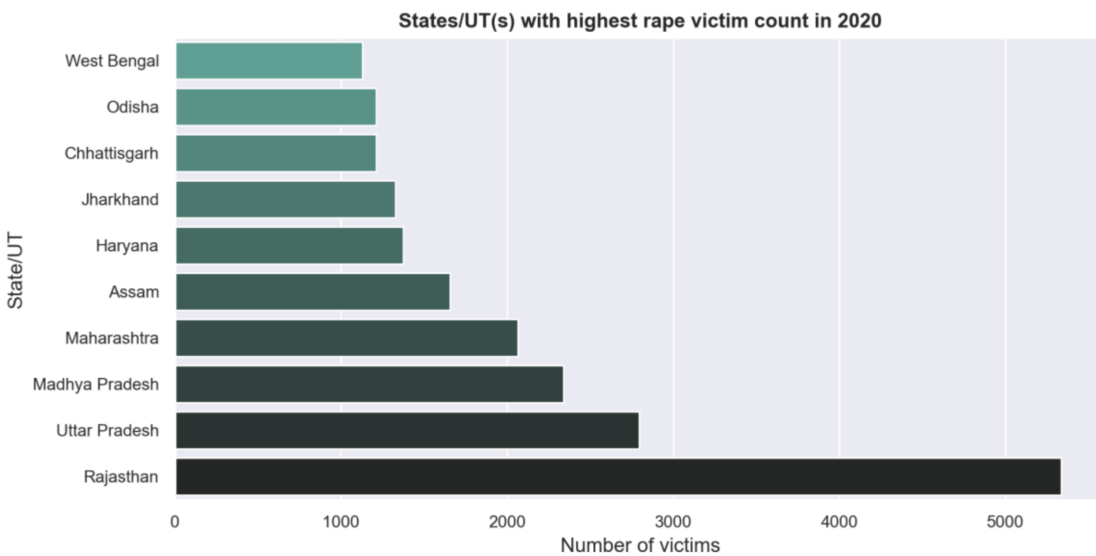
```
 6    Total Girl / Child Victims           38 non-null      float64
 7    18 Years & Above - Below 30 Years  38 non-null      int64
 8    30 Years & Above - Below 45 Years  38 non-null      float64
 9    45 Years & Above - Below 60 Years  38 non-null      float64
 10   60
Years & Above                          38 non-null      float64
 11   Total Women
/ Adult Victims           38 non-null      float64
 12   Total                               38 non-null      int64
dtypes: float64(10), int64(2), object(1)
memory usage: 4.0+ KB
```

[442]:
```
# basic statistical info of the dataset
df.describe().transpose()
```

[442]:
```
                                   count         mean          std  min  \
Cases Reported                      37.0  1516.000000  4603.316805  2.0
Below 6 Years                       38.0     4.218421    13.842626  0.0
6 Years & Above - Below 12 Years    38.0    11.123684    37.058941  0.0
12 Years & Above - Below 16 Years   38.0    47.084211   157.573421  0.0
16 Years & Above - Below 18 Years   38.0    77.557895   270.510249  0.0
Total Girl / Child Victims          38.0   139.984211   476.170602  0.0
18 Years & Above - Below 30 Years   38.0   935.342105  2865.997598  0.0
30 Years & Above - Below 45 Years   38.0   360.218421  1108.574354  0.0
45 Years & Above - Below 60 Years   38.0    45.765789   142.875898  0.0
60\nYears & Above                   38.0     3.057895     9.507767  0.0
Total Women\n/ Adult Victims        38.0  1344.384211  4121.809738  1.0
Total                               38.0  1484.368421  4564.034307  2.0

                                     25%    50%       75%      max
Cases Reported                     60.00  487.0  1210.000  28046.0
Below 6 Years                       0.00    0.0     1.000     80.0
6 Years & Above - Below 12 Years    0.00    0.0     1.675    211.0
12 Years & Above - Below 16 Years   0.00    1.0    12.000    893.0
16 Years & Above - Below 18 Years   0.00    1.0    12.000   1471.0
Total Girl / Child Victims          0.00    2.5    26.250   2655.0
18 Years & Above - Below 30 Years  14.00  310.5   788.000  17740.0
30 Years & Above - Below 45 Years   5.25   79.0   285.250   6832.0
45 Years & Above - Below 60 Years   0.25    6.5    31.000    868.0
60\nYears & Above                   0.00    0.0     2.000     58.0
Total Women\n/ Adult Victims       21.75  462.0  1177.500  25498.0
Total                              60.25  486.5  1190.250  28153.0
```

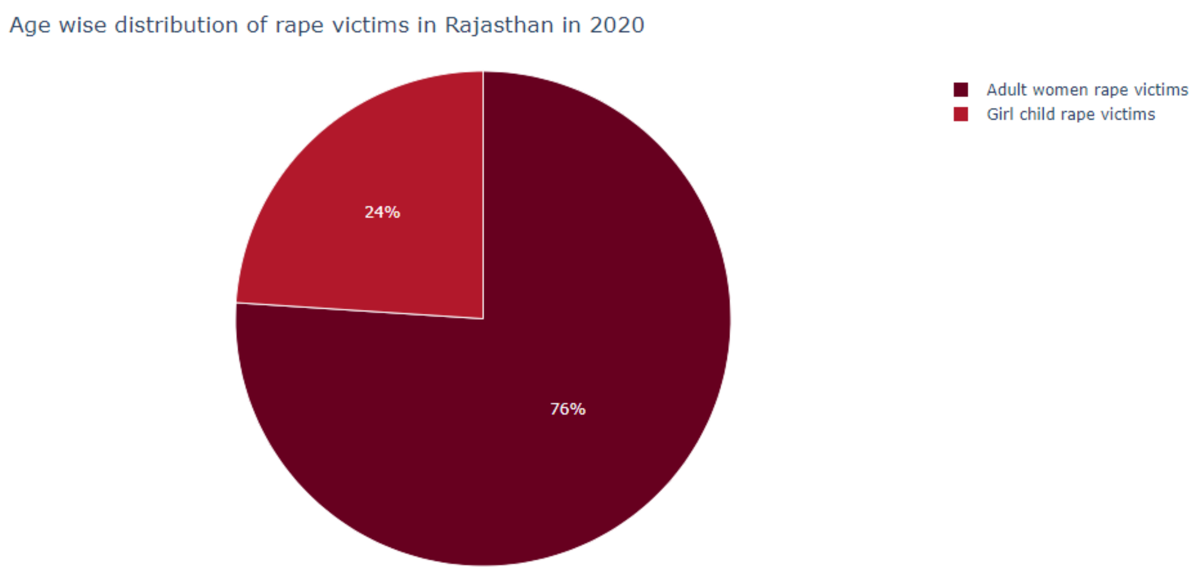
[443]:
```
# bar plot of top 10 states with highest crime

df_top10 = df[0:36].sort_values(by="Total").tail(10)

plt.figure(figsize=(10,5),dpi=200)
```

```python
sns.set_theme()

plot = sns.barplot(data=df_top10,x="Total",y="State/UT",palette="dark:#5A9_r")
plot.set_xlabel("Number of victims")
plot.set_ylabel("State/UT")
plt.tick_params(axis="both", which="major", labelsize=10)
plot.set_title("States/UT(s) with highest rape victim count in 2020",fontsize=␣
↪12,fontweight="bold")

plt.show()
```

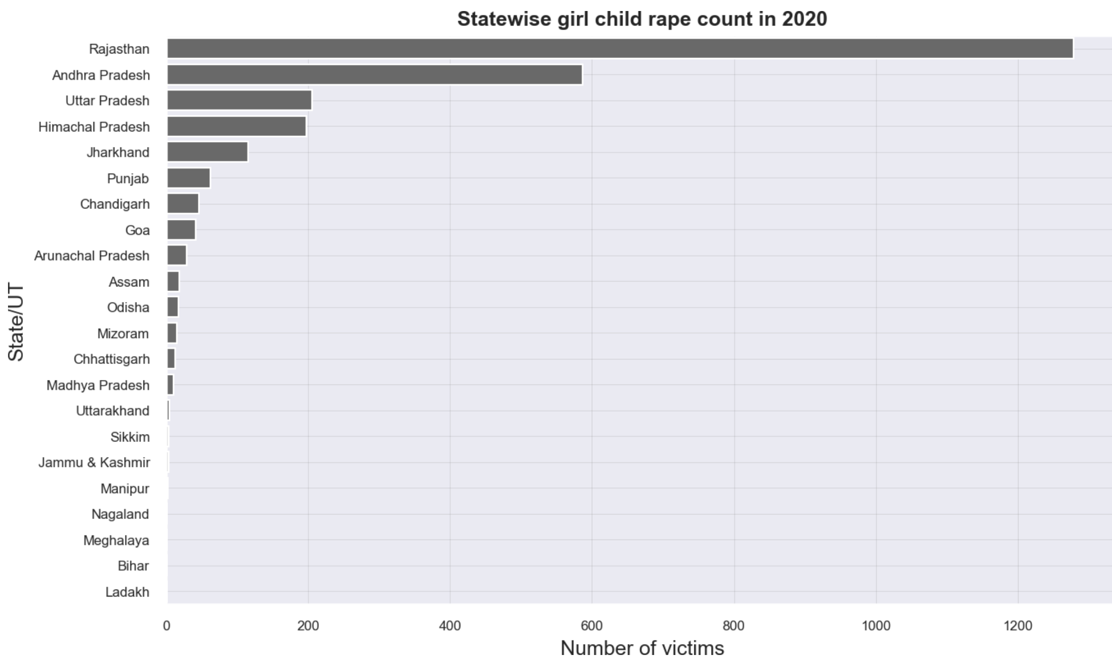


[8]:
```python
# Percentage of girl child victims and adult women victims in Rajasthan

child_raj = np.sum(df[df['State/UT']=='Rajasthan']['Total Girl / Child␣
↪Victims'])
adult_raj = np.sum(df[df['State/UT']=='Rajasthan']['Total Women\n/ Adult␣
↪Victims'])

pie = px.pie(values=[child_raj,adult_raj],names=['Girl child rape␣
↪victims','Adult women rape victims'],
        color_discrete_sequence=px.colors.sequential.RdBu)

pie.update_layout(title="Age wise distribution of rape victims in Rajasthan in␣
↪2020")
pie.update_traces(marker={'line':{'color':'white', 'width':1}})
```

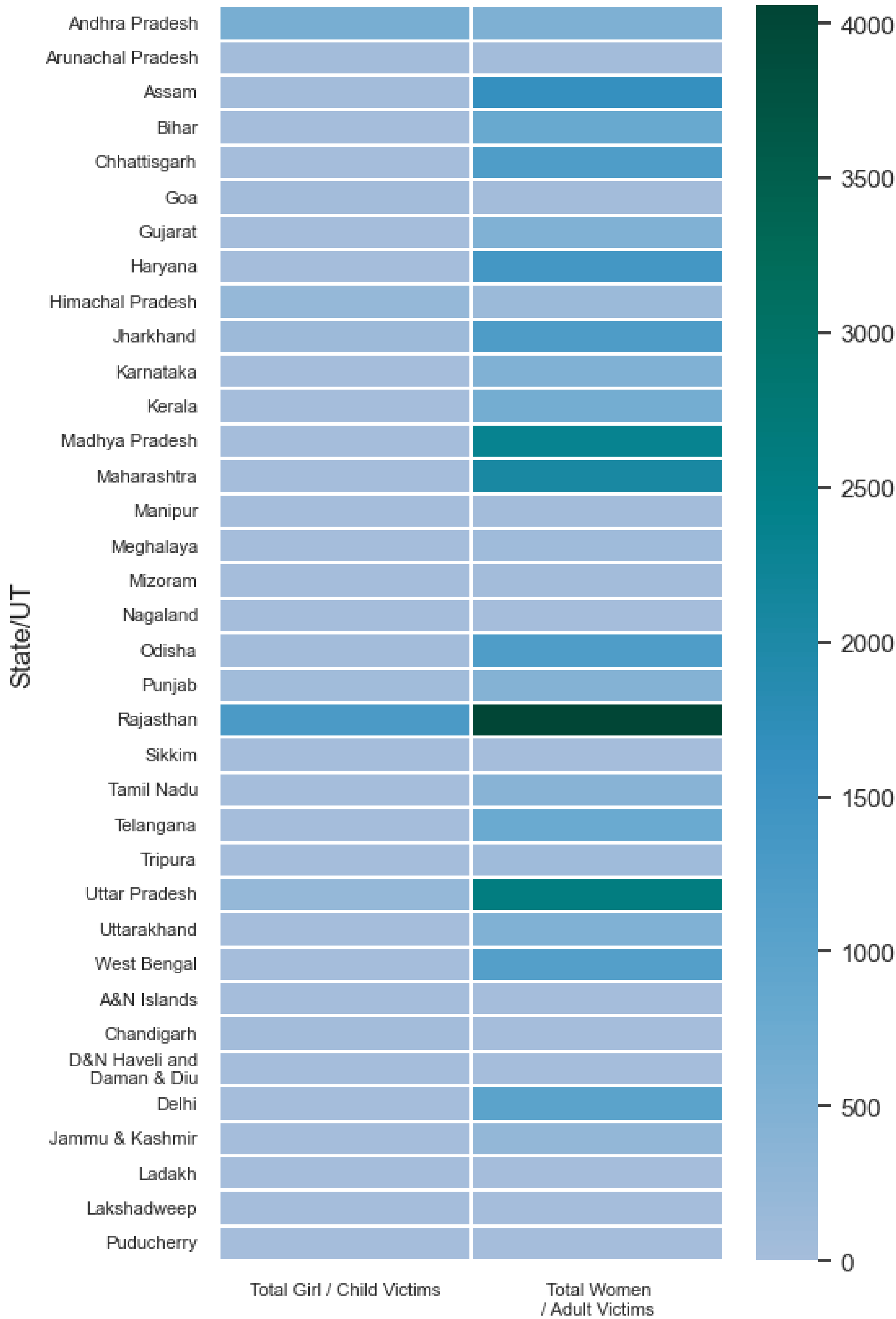
```
pie.show()
```



[445]:
```
# bar plot of statewise child rape count

df_child = df[0:36]
df_child = df_child[df_child['Total Girl / Child Victims']>0].
 ↪sort_values(by='Total Girl / Child Victims',ascending=False)

plt.figure(figsize=(10,6),dpi=150)

plot = sns.barplot(data=df_child,x='Total Girl / Child Victims',y='State/
 ↪UT',color='dimgray')
plot.set_xlabel('Number of victims')
plot.set_ylabel('State/UT')
plt.tick_params(axis='both', which='major', labelsize=8)
plot.set_title('Statewise girl child rape count in 2020',fontsize=␣
 ↪12,fontweight='bold')
plt.grid(color='gray',alpha=0.3,lw=0.3)

plt.show()
```



[446]:
```
# heatmap of agewise victims states/ut(s)

plt.figure(figsize=(5,10))
sns.heatmap(df[0:36][['State/UT','Total Girl / Child Victims','Total Women\n/␣
↪Adult Victims']].set_index('State/UT'),
            lw=0.1,cmap='PuBuGn',center=800)
plt.tick_params(axis='both', which='major', labelsize=8)
plt.show()
```


State/UT
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
A&N Islands
Chandigarh
D&N Haveli and Daman & Diu
Delhi
Jammu & Kashmir
Ladakh
Lakshadweep
Puducherry
Total Girl / Child Victims
Total Women / Adult Victims
4000
3500
3000
2500
2000
1500
1000
500
0

0.5 PRACTICAL 5: Apply Linear Regression on the given dataset

[5]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

[6]:
```
df = pd.read_excel("3A.3 Women & Girls Victims of Rape - 2020.xlsx")[0:
↪36][["State/UT","Total Women\n/ Adult Victims","Total"]].set_index("State/
↪UT")
df = df.rename(columns={"Total Women\n/ Adult Victims": "Adult Victims","Total":
↪"Total Victims"})
df.head(10)
```

[6]:

State/UT	Adult Victims	Total Victims
Andhra Pradesh	520.0	1107
Arunachal Pradesh	34.0	63
Assam	1640.0	1658
Bihar	805.0	806
Chhattisgarh	1199.0	1212
Goa	19.0	61
Gujarat	486.0	486
Haryana	1373.0	1373
Himachal Pradesh	135.0	332
Jharkhand	1210.0	1326

[449]:
```
df.describe().transpose()
```

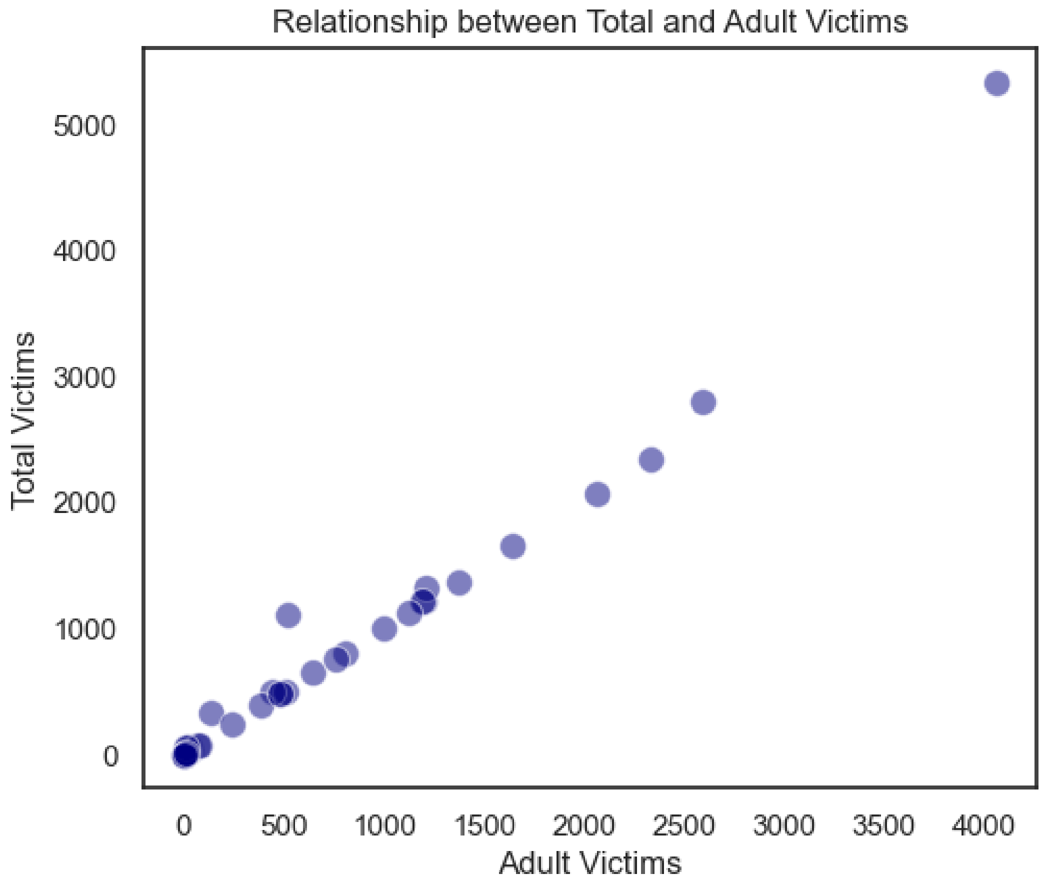
[449]:

	count	mean	std	min	25%	50%	75%	\
Adult Victims	36.0	708.277778	911.338313	1.0	18.75	462.0	1144.50	
Total Victims	36.0	782.027778	1066.832253	2.0	53.25	486.5	1148.75	

	max
Adult Victims	4058.0
Total Victims	5337.0

[450]:
```
# plotting the relationship between total and adult victims

sns.set_theme(style = "white")
plt.figure(figsize=(6,5))
sns.scatterplot(data=df,x="Adult Victims",y="Total Victims",s=100,alpha=0.
↪5,color="navy")
plt.title("Relationship between Total and Adult Victims")
plt.xlabel("Adult Victims")
plt.ylabel("Total Victims")
```

```
plt.show()
```



[626]:
```
# train test split
# 20% data points as test data

X,y = np.array(df['Adult Victims']).reshape(-1,1),np.array(df['Total Victims'])

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,random_state=11)
```

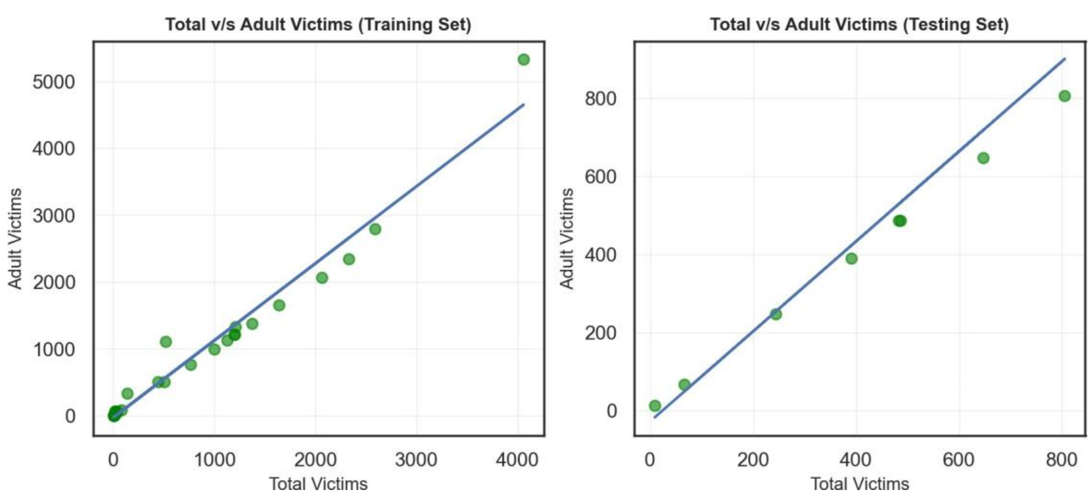
[627]:
```
# creating model

model = LinearRegression()
```

[628]:
```
# training the model

model.fit(X_train,y_train)
```

[628]: LinearRegression()

[629]:
```
# testing

# predicting on testing data
test_pred = model.predict(X_test)

# predicting on training data
train_pred = model.predict(X_train)
```

[630]:
```
# visualizing linear regression

fig,ax = plt.subplots(1,2,figsize=(10,4),dpi=200)

ax[0].plot(X_train,train_pred)
ax[0].scatter(X_train,y_train,color='green',alpha=0.6)
ax[0].grid(True,alpha=0.4,lw=0.5)
ax[0].set_xlabel('Total Victims',fontsize=10)
ax[0].set_ylabel('Adult Victims',fontsize=10)
ax[0].set_title('Total v/s Adult Victims (Training␣
↪Set)',fontsize=10,fontweight='bold')

ax[1].plot(X_test,test_pred)
ax[1].scatter(X_test,y_test,color='green',alpha=0.6)
ax[1].grid(True,alpha=0.4,lw=0.5)
ax[1].set_xlabel('Total Victims',fontsize=10)
ax[1].set_ylabel('Adult Victims',fontsize=10)
ax[1].set_title('Total v/s Adult Victims (Testing␣
↪Set)',fontsize=10,fontweight='bold')

plt.show()
```



[631]:
```python
# linear regression equation for this model

print("Total_Victims = {}*Adult_Victims + {}".format(model.coef_[0],model.
↪intercept_))
```

```
Total_Victims = 1.1528683492417635*Adult_Victims + -27.804003721035997
```

[632]:
```python
# evaluating performance of this model by computing RMSE

from sklearn.metrics import mean_squared_error
import math

performance_rmse = math.sqrt(mean_squared_error(y_test,test_pred))
performance_rmse
```

[632]: 50.127757053535355

0.6 PRACTICAL 6: Apply Logistic Regression on the given dataset

[24]:
```
from sklearn.linear_model import LogisticRegression
```

[25]:
```
df = pd.read_excel('3B.1 Crime against Women in Metropolitan Cities.xlsx')
df.rename(columns = {2018:'Year_2018',2019:'Year_2019',2020:'Year_2020'},␣
↪inplace = True)
df.head(3).transpose()
```

[25]:
```
                                                        0  \
City                                     Ahmedabad\n(Gujarat)
Year_2018                                                1416
Year_2019                                                1633
Year_2020                                                1524
Actual Population (in Lakhs) (2011)                      30.0
Rate of Total Crime against Women (2020)                 50.7
Chargesheeti ng Rate (2020)                              94.8

                                                          1  \
City                                     Bengaluru\n(Karnataka)
Year_2018                                                  3427
Year_2019                                                  3486
Year_2020                                                  2730
Actual Population (in Lakhs) (2011)                        40.6
Rate of Total Crime against Women (2020)                   67.3
Chargesheeti ng Rate (2020)                                71.4

                                                         2
City                                     Chennai\n(Tamil Nadu)
Year_2018                                                  761
Year_2019                                                  729
Year_2020                                                  576
Actual Population (in Lakhs) (2011)                       43.1
Rate of Total Crime against Women (2020)                  13.4
Chargesheeti ng Rate (2020)                               96.8
```

[26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   City                                      20 non-null     object
 1   Year_2018                                 20 non-null     int64
 2   Year_2019                                 20 non-null     int64
 3   Year_2020                                 20 non-null     int64
 4   Actual Population (in Lakhs) (2011)       20 non-null     float64
```

```
 5   Rate of Total Crime against Women (2020)  20 non-null     float64
 6   Chargesheeti ng Rate (2020)               20 non-null     float64
dtypes: float64(3), int64(3), object(1)
memory usage: 1.2+ KB
```

[27]:

```
df = df[0:19].set_index('City')

# creating a new column to perform logistic regression
df['Total Crime Rate > 50'] = (df['Rate of Total Crime against Women (2020)'] >␣
↪50).astype('int64')

df.head(3).transpose()
```

[27]:

```
City                                      Ahmedabad\n(Gujarat)  \
Year_2018                                               1416.0
Year_2019                                               1633.0
Year_2020                                               1524.0
Actual Population (in Lakhs) (2011)                       30.0
Rate of Total Crime against Women (2020)                  50.7
Chargesheeti ng Rate (2020)                               94.8
Total Crime Rate > 50                                      1.0

City                                      Bengaluru\n(Karnataka)  \
Year_2018                                                 3427.0
Year_2019                                                 3486.0
Year_2020                                                 2730.0
Actual Population (in Lakhs) (2011)                         40.6
Rate of Total Crime against Women (2020)                    67.3
Chargesheeti ng Rate (2020)                                 71.4
Total Crime Rate > 50                                        1.0

City                                      Chennai\n(Tamil Nadu)
Year_2018                                                 761.0
Year_2019                                                 729.0
Year_2020                                                 576.0
Actual Population (in Lakhs) (2011)                        43.1
Rate of Total Crime against Women (2020)                   13.4
Chargesheeti ng Rate (2020)                                96.8
Total Crime Rate > 50                                       0.0
```

[28]:

```
# train test split
# 30% data points as test data

X,y = np.array(df['Year_2018']).reshape(-1,1),np.array(df['Total Crime Rate >␣
↪50'])
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
    ↪3,random_state=11)
```

[29]:
```
# creating and training the model

model = LogisticRegression()
model.fit(X_train,y_train)
```

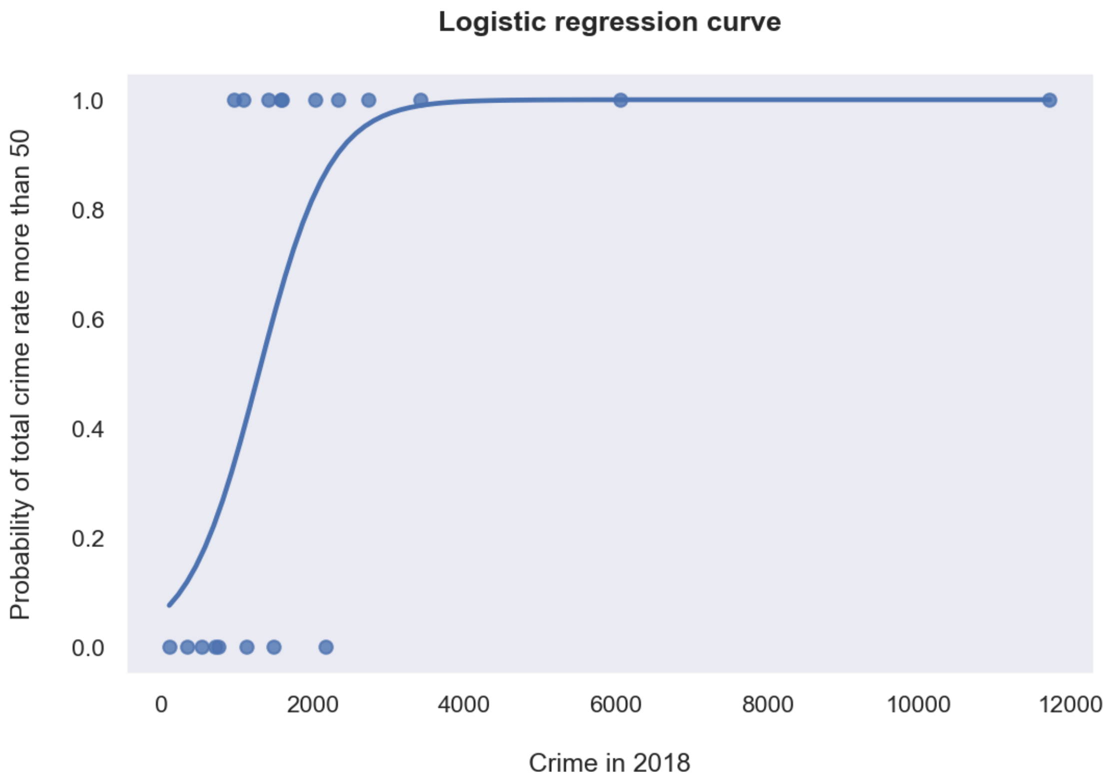
[29]:
```
LogisticRegression()
```

[30]:
```
# testing

# predicting on testing data
test_pred = model.predict(X_test)

# predicting on training data
train_pred = model.predict(X_train)
```

[31]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, test_pred)
cm
```

[31]:
```
array([[3, 0],
       [0, 3]], dtype=int64)
```

[32]:
```
plt.figure(figsize=(8,5),dpi=150)
sns.set_theme(style='dark')

plot = sns.regplot(data=df,x='Year_2018',y='Total Crime Rate > 50',␣
    ↪logistic=True, ci=None)
plot.set_xlabel('\nCrime in 2018')
plot.set_ylabel('Probability of total crime rate more than 50\n')
plot.set_title('Logistic regression curve\n',fontsize= 13,fontweight='bold')

plt.show()
```


Logistic regression curve
Probability of total crime rate more than 50
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
12000
Crime in 2018

```
[ ]:
```

7 PRACTICAL 7: Apply K means clustering on Crime dataset

[33]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[34]:
```
df = pd.read_excel('3B.1 Crime against Women in Metropolitan Cities.xlsx')
df.rename(columns = {2018:'Year_2018',2019:'Year_2019',2020:'Year_2020'},␣
 ↪inplace = True)
df.head(3).transpose()
```

[34]:
```
                                                               0  \
City                                          Ahmedabad\n(Gujarat)
Year_2018                                                     1416
Year_2019                                                     1633
Year_2020                                                     1524
Actual Population (in Lakhs) (2011)                           30.0
Rate of Total Crime against Women (2020)                      50.7
Chargesheeti ng Rate (2020)                                   94.8

                                                                 1  \
City                                          Bengaluru\n(Karnataka)
Year_2018                                                       3427
Year_2019                                                       3486
Year_2020                                                       2730
Actual Population (in Lakhs) (2011)                             40.6
Rate of Total Crime against Women (2020)                        67.3
Chargesheeti ng Rate (2020)                                     71.4

                                                                2
City                                          Chennai\n(Tamil Nadu)
Year_2018                                                      761
Year_2019                                                      729
Year_2020                                                      576
Actual Population (in Lakhs) (2011)                           43.1
Rate of Total Crime against Women (2020)                      13.4
Chargesheeti ng Rate (2020)                                   96.8
```

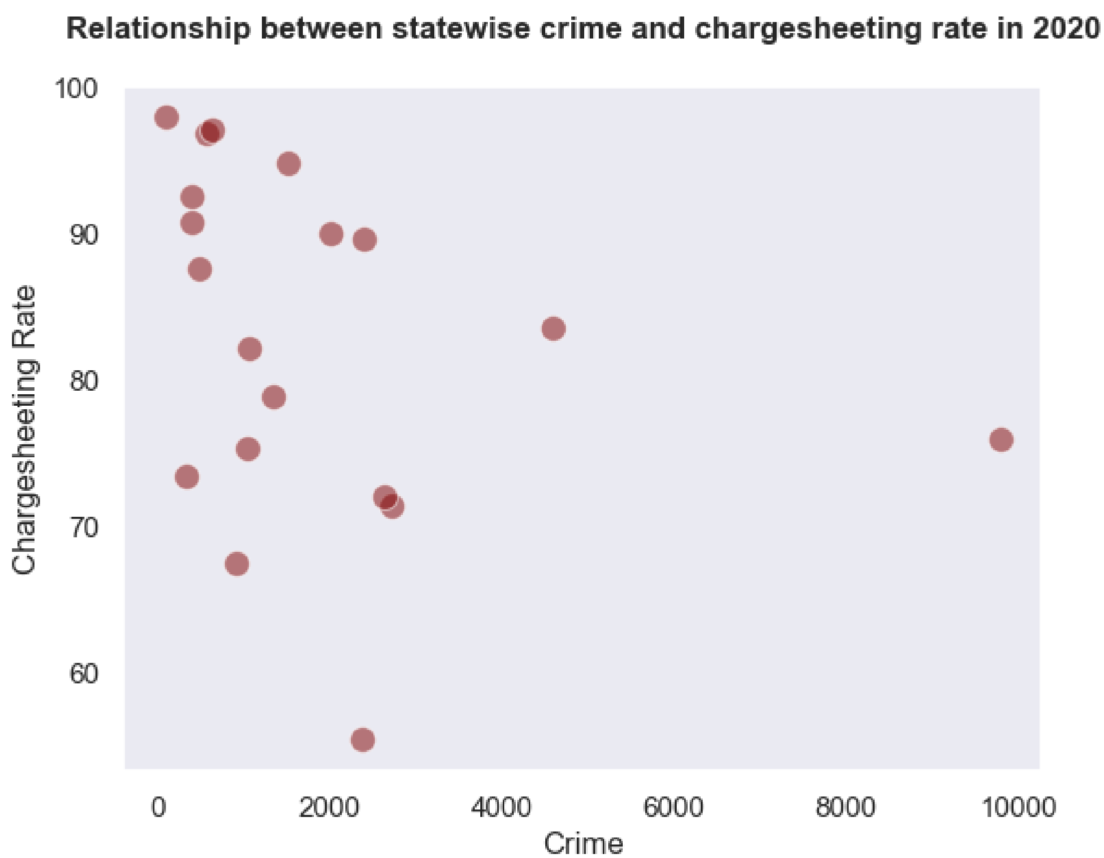
[35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   City                                      20 non-null     object
 1   Year_2018                                 20 non-null     int64
```

```
 2   Year_2019                                 20 non-null     int64
 3   Year_2020                                 20 non-null     int64
 4   Actual Population (in Lakhs) (2011)       20 non-null     float64
 5   Rate of Total Crime against Women (2020)  20 non-null     float64
 6   Chargesheeti ng Rate (2020)               20 non-null     float64
dtypes: float64(3), int64(3), object(1)
memory usage: 1.2+ KB
```

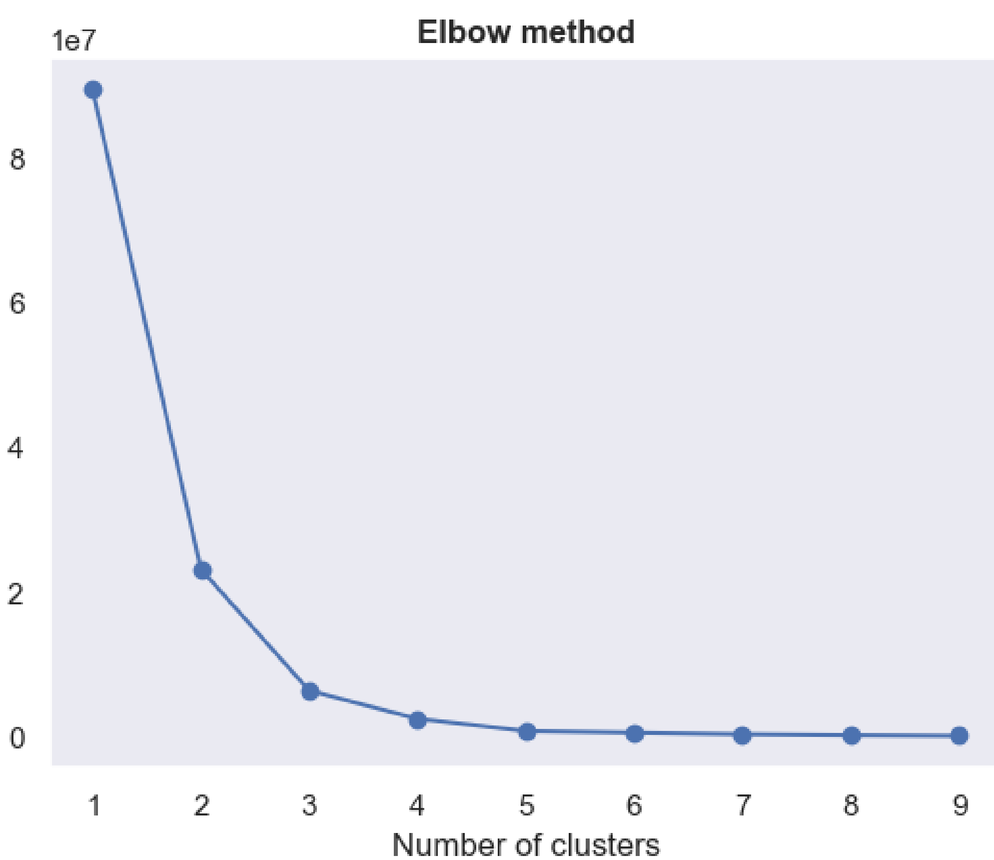
[36]:

```
plt.figure()

sns.scatterplot(data=df[0:19],x='Year_2020',y='Chargesheeti ng Rate␣
 ↪(2020)',s=100,alpha=0.5,color='maroon')
plt.title('Relationship between statewise crime and chargesheeting rate in␣
 ↪2020\n',fontweight='bold')
plt.xlabel('Crime')
plt.ylabel('Chargesheeting Rate')

plt.show()
```



[37]:

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

from sklearn.cluster import KMeans

x = df['Year_2020'][0:19]
y = df['Chargesheeti ng Rate (2020)'][0:19]

data = list(zip(x, y)) # making tuples
Distortion = []

for i in range(1,10):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    Distortion.append(kmeans.inertia_)

plt.plot(range(1,10), Distortion, marker='o')
plt.title('Elbow method',fontweight='bold')
plt.xlabel('Number of clusters')

plt.show()
```

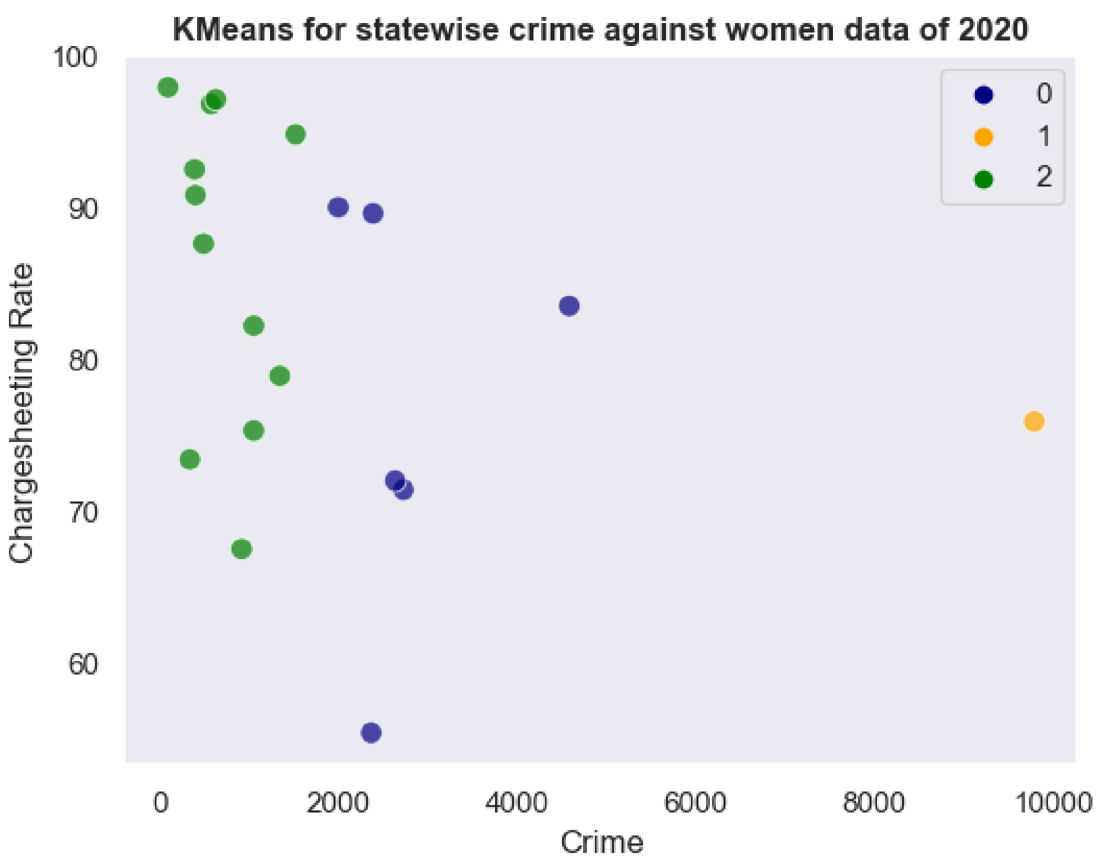


[38]:

```
# According to the elbow method, k=3 is a good choice for the number of clusters
```

[39]:

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)

sns.set_theme(style='dark')
sns.scatterplot(data=df[0:19],x='Year_2020',y='Chargesheeti ng Rate (2020)',
                hue=kmeans.
 ↪labels_,palette=['navy','orange','green'],s=70,alpha=0.7)

plt.title('KMeans for statewise crime against women data of␣
 ↪2020',fontweight='bold')
plt.xlabel('Crime')
plt.ylabel('Chargesheeting Rate')
plt.show()
```



8 PRACTICAL 8: Apply Principal Component Analysis on Crime dataset

[40]:
```
df = pd.read_excel('3B.1 Crime against Women in Metropolitan Cities.xlsx')
df.rename(columns = {2018:'Year_2018',2019:'Year_2019',2020:'Year_2020'},␣
↪inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   City                                      20 non-null     object
 1   Year_2018                                 20 non-null     int64
 2   Year_2019                                 20 non-null     int64
 3   Year_2020                                 20 non-null     int64
 4   Actual Population (in Lakhs) (2011)       20 non-null     float64
 5   Rate of Total Crime against Women (2020)  20 non-null     float64
 6   Chargesheeti ng Rate (2020)               20 non-null     float64
dtypes: float64(3), int64(3), object(1)
memory usage: 1.2+ KB
```

[41]:
```
index = df['City'][0:19]
df = df[0:19].set_index('City')

df['Total Crime Rate > 50'] = (df['Rate of Total Crime against Women (2020)'] >␣
↪50)
df['Total Crime Rate > 50'] = df['Total Crime Rate > 50'].map({True:'yes',␣
↪False:'no'})

df.drop(['Chargesheeti ng Rate (2020)','Rate of Total Crime against Women␣
↪(2020)'],axis=1,inplace=True)
df.head(3).transpose()
```

[41]:
```
City                                 Ahmedabad\n(Gujarat)  \
Year_2018                                            1416
Year_2019                                            1633
Year_2020                                            1524
Actual Population (in Lakhs) (2011)                  30.0
Total Crime Rate > 50                                 yes

City                                 Bengaluru\n(Karnataka)  \
Year_2018                                              3427
Year_2019                                              3486
Year_2020                                              2730
Actual Population (in Lakhs) (2011)                    40.6
```

```
Total Crime Rate > 50                                  yes

City                                   Chennai\n(Tamil Nadu)
Year_2018                                              761
Year_2019                                              729
Year_2020                                              576
Actual Population (in Lakhs) (2011)                   43.1
Total Crime Rate > 50                                   no
```

[42]:
```
# standardizing the data

from sklearn.preprocessing import StandardScaler

features = ['Year_2018', 'Year_2019', 'Year_2020','Actual Population (in Lakhs)␣
 ↪(2011)']

# Separating out the features
x = df[features].values

# Standardizing the features
x = StandardScaler().fit_transform(x)
```

[43]:
```
#first 5 standardized features

x[:5]
```

[43]:
```
array([[-0.30962093, -0.25168779, -0.15472186,  0.06570385],
       [ 0.4648165 ,  0.39247999,  0.40140311,  0.51888037],
       [-0.56186187, -0.56594989, -0.59187482,  0.62576163],
       [-0.8137177 , -0.78982687, -0.81275696, -0.75941945],
       [ 3.65999671,  3.66581176,  3.65330468,  2.02376846]])
```

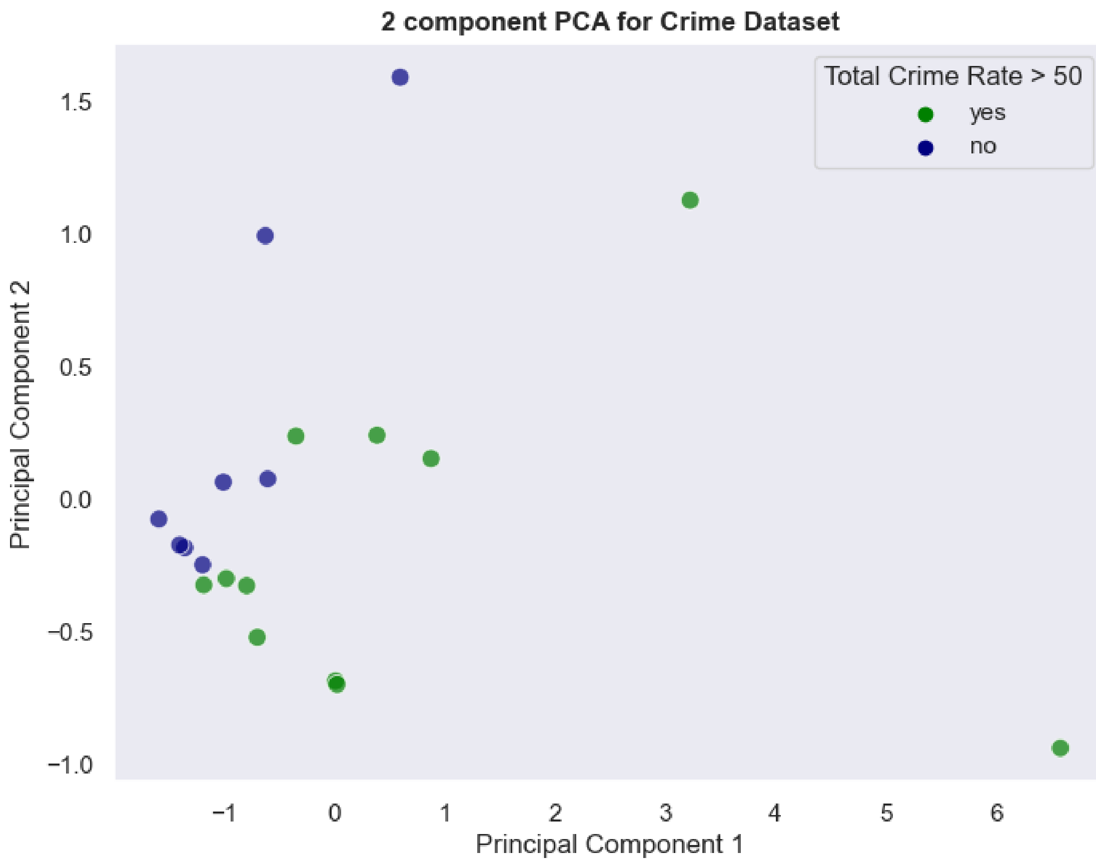
[44]:
```
# performing PCA
# transforming 4 dimensional data to 2 dimensions

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents,index=index,columns =␣
 ↪['Principal Component 1', 'Principal Component 2'])

finalDf = pd.concat([principalDf, df[['Total Crime Rate > 50']]], axis = 1)
finalDf
```

[44]:
```
                            Principal Component 1  Principal Component 2  \
City
```

```
Ahmedabad\n(Gujarat)                    -0.343526      0.237258
Bengaluru\n(Karnataka)                   0.880353      0.152721
Chennai\n(Tamil Nadu)                   -0.620875      0.994273
Coimbatore\n(Tamil Nadu)                -1.586614     -0.076125
Delhi City                               6.587069     -0.941052
Ghaziabad\n(Uttar Pradesh)              -1.190087     -0.248574
Hyderabad\n(Telangana)                   0.391497      0.240756
Indore\n(Madhya Pradesh)                -0.693906     -0.522394
Jaipur\n(Rajasthan)                      0.014409     -0.687503
Kanpur\n(Uttar Pradesh)                 -0.790473     -0.327513
Kochi\n(Kerala)                         -1.351809     -0.183967
Kolkata\n(West Bengal)                   0.601119      1.593103
Kozhikode\n(Kerala)                     -1.398881     -0.173302
Lucknow\n(Uttar Pradesh)                 0.028848     -0.700223
Mumbai\n(Maharashtra)                    3.229610      1.128630
Nagpur\n(Maharashtra)                   -0.974834     -0.300461
Patna\n(Bihar)                          -1.180001     -0.325059
Pune\n(Maharashtra)                     -0.600268      0.075752
Surat\n(Gujarat)                        -1.001632      0.063680

                           Total Crime Rate > 50
City
Ahmedabad\n(Gujarat)                         yes
Bengaluru\n(Karnataka)                       yes
Chennai\n(Tamil Nadu)                         no
Coimbatore\n(Tamil Nadu)                      no
Delhi City                                   yes
Ghaziabad\n(Uttar Pradesh)                    no
Hyderabad\n(Telangana)                       yes
Indore\n(Madhya Pradesh)                     yes
Jaipur\n(Rajasthan)                          yes
Kanpur\n(Uttar Pradesh)                      yes
Kochi\n(Kerala)                               no
Kolkata\n(West Bengal)                        no
Kozhikode\n(Kerala)                           no
Lucknow\n(Uttar Pradesh)                     yes
Mumbai\n(Maharashtra)                        yes
Nagpur\n(Maharashtra)                        yes
Patna\n(Bihar)                               yes
Pune\n(Maharashtra)                           no
Surat\n(Gujarat)                              no
```

[45]:
```
# plotting the reduced data

plt.figure(figsize=(8,6))
sns.set_theme(style='dark')
```

```
sns.scatterplot(data=finalDf,x='Principal Component 1',y='Principal Component␣
↪2',
                hue='Total Crime Rate > 50',palette=['green','navy'],alpha=0.
↪7,s=70)
plt.title('2 component PCA for Crime Dataset',fontweight='bold')

plt.show()
```




```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

9 PRACTICAL 9: Apply Neural Network on Crime dataset

[46]:
```
df.head(3).transpose()
```

[46]:
```
City                                 Ahmedabad\n(Gujarat)  \
Year_2018                                            1416
Year_2019                                            1633
Year_2020                                            1524
Actual Population (in Lakhs) (2011)                  30.0
Total Crime Rate > 50                                 yes

City                                 Bengaluru\n(Karnataka)  \
Year_2018                                                3427
Year_2019                                                3486
Year_2020                                                2730
Actual Population (in Lakhs) (2011)                      40.6
Total Crime Rate > 50                                     yes

City                                 Chennai\n(Tamil Nadu)
Year_2018                                              761
Year_2019                                              729
Year_2020                                              576
Actual Population (in Lakhs) (2011)                   43.1
Total Crime Rate > 50                                   no
```

[47]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
```

[48]:
```
# breaking the dataset into training and testing data

features = ['Year_2018', 'Year_2019', 'Year_2020','Actual Population (in Lakhs)␣
 ↪(2011)']

X = df[features].values
y = df['Total Crime Rate > 50'].values

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪2,random_state=42)
```

[49]:
```
# building the model

model = MLPClassifier(hidden_layer_sizes=(5,5), activation='logistic',␣
 ↪solver='sgd', max_iter=100, random_state=29)
model.fit(X_train,y_train)
```

[49]:
```
MLPClassifier(activation='logistic', hidden_layer_sizes=(5, 5), max_iter=100,
              random_state=29, solver='sgd')
```

[50]:
```python
# making predictions

pred_train = model.predict(X_train)
pred_test = model.predict(X_test)
```

We will observe confusion matrix and classification report for evaluating the model

Confusion matrix:

[51]:
```python
# evaluating model performance on training data

print('Confusion matrix and Classification report for training data:
 ↪\n\n',confusion_matrix(y_train,pred_train),'\n')
print(classification_report(y_train,pred_train))
```

```
Confusion matrix and Classification report for training data:

 [[0 6]
 [0 9]]

              precision    recall  f1-score   support

          no       0.00      0.00      0.00         6
         yes       0.60      1.00      0.75         9

    accuracy                           0.60        15
   macro avg       0.30      0.50      0.37        15
weighted avg       0.36      0.60      0.45        15
```

[52]:
```python
# evaluating model performance on testing data

print('Confusion matrix and Classification report for testing data:
 ↪\n\n',confusion_matrix(y_test,pred_test),'\n')
print(classification_report(y_test,pred_test))
```

```
Confusion matrix and Classification report for testing data:

 [[0 2]
 [0 2]]

              precision    recall  f1-score   support

          no       0.00      0.00      0.00         2
         yes       0.50      1.00      0.67         2
```

```
    accuracy                           0.50         4
   macro avg       0.25      0.50      0.33         4
weighted avg       0.25      0.50      0.33         4
```

It can be observed that the model performs well when predicting target values 0 but very poorly when predicting target values 1

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

10 PRACTICAL 10: Apply Decision Tree and Random Forest on Crime dataset

10.1. Decision Tree

[53]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

[54]:
```
# 30% testing data and 70% training data

features = ['Year_2018', 'Year_2019', 'Year_2020','Actual Population (in Lakhs)␣
↪(2011)']
targets = np.unique(y)

X = df[features].values
y = df['Total Crime Rate > 50'].values

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪3,random_state=101)
```

[55]:
```
# building the model

model_dt = DecisionTreeClassifier(criterion='log_loss',random_state=101)
model_dt.fit(X_train,y_train)
```

[55]:
```
DecisionTreeClassifier(criterion='log_loss', random_state=101)
```

[56]:
```
# making predictions

pred_train = model_dt.predict(X_train)
pred_test = model_dt.predict(X_test)
```

[57]:
```
# evaluating model performance on training data

print('Confusion matrix and Classification report for training data:
↪\n\n',confusion_matrix(y_train,pred_train),'\n')
print(classification_report(y_train,pred_train))
```

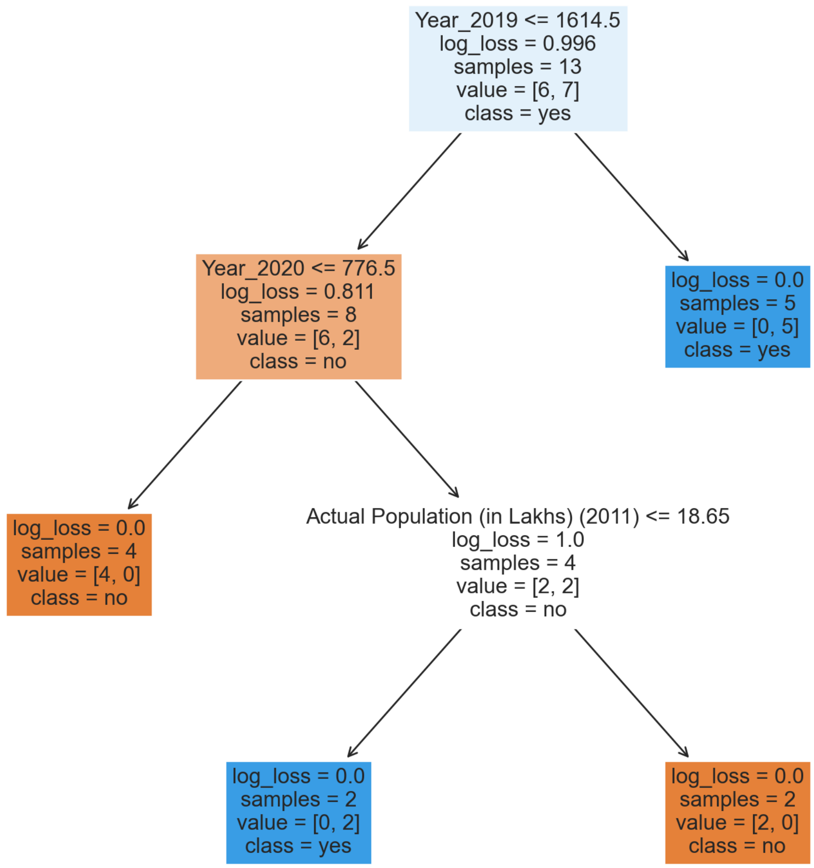
```
Confusion matrix and Classification report for training data:

 [[6 0]
 [0 7]]

              precision    recall  f1-score   support

          no       1.00      1.00      1.00         6
         yes       1.00      1.00      1.00         7
```

```
    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13
```

[58]:
```
# evaluating model performance on testing data

print('Confusion matrix and Classification report for testing data:
↪\n\n',confusion_matrix(y_test,pred_test),'\n')
print(classification_report(y_test,pred_test))
```

```
Confusion matrix and Classification report for testing data:

 [[2 0]
 [1 3]]

              precision    recall  f1-score   support

          no       0.67      1.00      0.80         2
         yes       1.00      0.75      0.86         4

    accuracy                           0.83         6
   macro avg       0.83      0.88      0.83         6
weighted avg       0.89      0.83      0.84         6
```

[65]:
```
plt.figure(figsize=(11,10),dpi=200)
tree.plot_tree(model_dt, feature_names = features, class_names = targets,␣
↪filled = True)
plt.show()
```



10.2. Random Forest

[60]:
```
from sklearn.ensemble import RandomForestClassifier
```

[61]:
```
model_rf = RandomForestClassifier(n_estimators = 1000, random_state = 101)
model_rf.fit(X_train,y_train)
```

[61]:
```
RandomForestClassifier(n_estimators=1000, random_state=101)
```

[62]:
```
pred_train = model_dt.predict(X_train)
pred_test = model_dt.predict(X_test)
```

[63]:
```
print('Confusion matrix and Classification report for training data:
  ↪\n\n',confusion_matrix(y_train,pred_train),'\n')
print(classification_report(y_train,pred_train))
```

```
Confusion matrix and Classification report for training data:
```

```
[[6 0]
 [0 7]]

              precision    recall  f1-score   support

          no       1.00      1.00      1.00         6
         yes       1.00      1.00      1.00         7

    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13
```

[64]:

```
print('Confusion matrix and Classification report for testing data:
 ↪\n\n',confusion_matrix(y_test,pred_test),'\n')
print(classification_report(y_test,pred_test))
```

```
Confusion matrix and Classification report for testing data:

 [[2 0]
 [1 3]]

              precision    recall  f1-score   support

          no       0.67      1.00      0.80         2
         yes       1.00      0.75      0.86         4

    accuracy                           0.83         6
   macro avg       0.83      0.88      0.83         6
weighted avg       0.89      0.83      0.84         6
```

[ ]:

```
```

TABLE 3A.3

Women & Girls Victims of Rape (Age Group-wise) - 2020

SL	State/UT	Cases Reported	Child Victims of Rape (Below 18 Yrs)					Women Victims of Rape (Above 18 Years)					Total Victims (Col.8+Col.13)
			Below 6 Years	6 Years & Above - Below 12 Years	12 Years & Above - Below 16 Years	16 Years & Above - Below 18 Years	Total Girl / Child Victims	18 Years & Above - Below 30 Years	30 Years & Above - Below 45 Years	45 Years & Above - Below 60 Years	60 Years & Above	Total Women / Adult Victims	
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
STATES:													
1	Andhra Pradesh	1095	27	74	214	272	587	411	87	20	2	520	1107
2	Arunachal Pradesh	60	1	6	14	8	29	20	14	0	0	34	63
3	Assam	1657	0	0	6	12	18	1006	584	50	0	1640	1658
4	Bihar	806	0	0	1	0	1	631	156	18	0	805	806
5	Chhattisgarh	1210	0	0	4	9	13	734	398	62	5	1199	1212
6	Goa	60	2	5	23	12	42	12	5	2	0	19	61
7	Gujarat	486	0	0	0	0	0	350	126	8	2	486	486
8	Haryana	1373	0	0	0	0	0	826	494	51	2	1373	1373
9	Himachal Pradesh	331	8	20	91	78	197	78	53	3	1	135	332
10	Jharkhand	1321	0	0	3	113	116	821	364	25	0	1210	1326
11	Karnataka	504	0	0	0	0	0	387	115	5	0	507	507
12	Kerala	637	0	0	0	0	0	359	247	31	10	647	647
13	Madhya Pradesh	2339	2	7	0	1	10	1587	657	81	6	2331	2341
14	Maharashtra	2061	0	0	0	0	0	1420	577	61	7	2065	2065
15	Manipur	32	0	0	0	2	2	25	5	0	0	30	32
16	Meghalaya	67	0	0	1	0	1	45	18	2	1	66	67
17	Mizoram	33	1	0	14	0	15	12	4	1	1	18	33
18	Nagaland	4	0	0	1	0	1	2	1	0	0	3	4
19	Odisha	1211	0	0	4	13	17	1150	44	0	0	1194	1211
20	Punjab	502	7	2	21	32	62	315	110	16	1	442	504
21	Rajasthan	5310	21	64	374	820	1279	2617	1216	225	0	4058	5337
22	Sikkim	12	0	0	3	1	4	6	1	2	0	9	13
23	Tamil Nadu	389	0	0	0	0	0	306	71	5	8	390	390
24	Telangana	764	0	0	0	0	0	546	186	29	4	765	765
25	Tripura	79	0	0	0	0	0	55	18	4	2	79	79
26	Uttar Pradesh	2769	8	30	99	69	206	2033	487	70	0	2590	2796
27	Uttarakhand	487	0	0	3	2	5	301	168	13	0	482	487
28	West Bengal	1128	0	0	0	0	0	806	290	31	1	1128	1128
	TOTAL STATE(S)	**26727**	**77**	**208**	**876**	**1444**	**2605**	**16861**	**6496**	**815**	**53**	**24225**	**26830**
UNION TERRITORIES:													
29	A&N Islands	2	0	0	0	0	0	2	0	0	0	2	2
30	Chandigarh	60	3	3	16	24	46	8	6	0	0	14	60
31	D&N Haveli and Daman & Diu	4	0	0	0	0	0	2	2	0	0	4	4
32	Delhi	997	0	0	0	0	0	677	271	44	5	997	997
33	Jammu & Kashmir	243	0	0	1	2	3	180	55	9	0	244	247
34	Ladakh	2	0	0	0	1	1	0	1	0	0	1	2
35	Lakshadweep	3	0	0	0	0	0	2	1	0	0	3	3
36	Puducherry	8	0	0	0	0	0	8	0	0	0	8	8
	TOTAL UT(S)	**1319**	**3**	**3**	**17**	**27**	**50**	**879**	**336**	**53**	**5**	**1273**	**1323**
	TOTAL ALL INDIA	**28046**	**80**	**211**	**893**	**1471**	**2655**	**17740**	**6832**	**868**	**58**	**25498**	**28153**
	Percentage Share of Age-Group of Victims		**0.3**	**0.7**	**3.2**	**5.2**	**9.4**	**63.0**	**24.3**	**3.1**	**0.2**	**90.6**	**100.0**

● As per data provided by States/UTs

● States/UTs may not be compared purely on the basis of crime figures

TABLE 3B.1

Crime against Women (IPC+SLL) in Metropolitan Cities - 2018-2020

SL	City	2018	2019	2020	Actual Population (in Lakhs) (2011)	Rate of Total Crime against Women (2020)	Chargesheeting Rate (2020)
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
1	Ahmedabad (Gujarat)	1416	1633	1524	30.0	50.7	94.8
2	Bengaluru (Karnataka)	3427	3486	2730	40.6	67.3	71.4
3	Chennai (Tamil Nadu)	761	729	576	43.1	13.4	96.8
4	Coimbatore (Tamil Nadu)	107	85	97	10.7	9.0	97.9
5	Delhi City	11724	12902	9782	75.8	129.1	75.9
6	Ghaziabad (Uttar Pradesh)	1128	793	341	11.0	31.0	73.4
7	Hyderabad (Telangana)	2332	2755	2390	37.6	63.5	89.6
8	Indore (Madhya Pradesh)	1593	1755	1346	10.4	129.7	78.9
9	Jaipur (Rajasthan)	2030	3417	2369	14.5	162.9	55.4
10	Kanpur (Uttar Pradesh)	1574	1315	1056	13.4	79.1	82.2
11	Kochi (Kerala)	537	492	403	10.8	37.5	90.8
12	Kolkata (West Bengal)	2176	1474	2001	67.9	29.5	90.0
13	Kozhikode (Kerala)	349	473	394	10.6	37.0	92.5
14	Lucknow (Uttar Pradesh)	2736	2425	2636	13.8	190.7	72.0
15	Mumbai (Maharashtra)	6058	6519	4583	85.2	53.8	83.5
16	Nagpur (Maharashtra)	1083	1144	920	12.2	75.3	67.5
17	Patna (Bihar)	956	981	495	9.6	51.6	87.6
18	Pune (Maharashtra)	1481	1390	1055	23.9	44.1	75.3
19	Surat (Gujarat)	712	1015	633	19.7	32.1	97.1
	TOTAL CITIES	**42180**	**44783**	**35331**	**540.9**	**65.3**	**78.2**

- Crime Rate is calculated as per one lakh of population

TABLE 3B.1 Page 1 of 1

- Population Source : Registrar General of India Actual Population based on 2011 Census.
- As per data provided by States/UTs