untitled2

November 26, 2024

[1]:
```python
#1
import pandas as pd
```

[2]:
```python
df = pd.read_csv("Experiment7.csv")
```

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

[3]:
```
   Survived  Pclass     Sex  Age  SibSp  Parch  Fare Embarked
0         0       3    male   22      1      0     7        S
1         1       1  female   38      1      0    71        C
2         1       3  female   26      0      0     7        S
3         1       1  female   35      1      0    53        S
4         0       3    male   35      0      0     8        S
```

[4]:
```python
df.tail()
```

[4]:
```
     Survived  Pclass     Sex  Age  SibSp  Parch  Fare Embarked
886         0       2    male   27      0      0    13        S
887         1       1  female   19      0      0    30        S
888         0       3  female   32      1      2    23        S
889         1       1    male   26      0      0    30        C
890         0       3    male   32      0      0     7        Q
```

[5]:
```python
df.isna()
```

[5]:
```
     Survived  Pclass    Sex    Age  SibSp  Parch   Fare  Embarked
0       False   False  False  False  False  False  False     False
1       False   False  False  False  False  False  False     False
2       False   False  False  False  False  False  False     False
3       False   False  False  False  False  False  False     False
4       False   False  False  False  False  False  False     False
..        …       …     …     …     …     …     …          …
886     False   False  False  False  False  False  False     False
887     False   False  False  False  False  False  False     False
888     False   False  False  False  False  False  False     False
889     False   False  False  False  False  False  False     False
890     False   False  False  False  False  False  False     False
```

```
[891 rows x 8 columns]
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       891 non-null    int64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    int64
 7   Embarked  891 non-null    object
dtypes: int64(6), object(2)
memory usage: 55.8+ KB
```

[7]:
```
df.shape
```

[7]: (891, 8)

[8]:
```
df.size
```

[8]: 7128

[9]:
```
df.describe
```

[9]:
```
<bound method NDFrame.describe of      Survived  Pclass     Sex  Age  SibSp
Parch  Fare Embarked
0           0       3    male   22      1      0     7        S
1           1       1  female   38      1      0    71        C
2           1       3  female   26      0      0     7        S
3           1       1  female   35      1      0    53        S
4           0       3    male   35      0      0     8        S
..        …     …     …  …    …    …   …      …
886         0       2    male   27      0      0    13        S
887         1       1  female   19      0      0    30        S
888         0       3  female   32      1      2    23        S
889         1       1    male   26      0      0    30        C
890         0       3    male   32      0      0     7        Q

[891 rows x 8 columns]>
```

[10]:
```python
df.nunique()
```

[10]:
```
Survived      2
Pclass        3
Sex           2
Age          72
SibSp         7
Parch         7
Fare         91
Embarked      3
dtype: int64
```

[11]:
```python
import numpy as np
```

[12]:
```python
mydata = {
    'EmpID' : [111,112,123,114,231],
    'EmpName':['Ashish','Vinod','Abhay','Munna','Deepak']
}
```

[13]:
```python
employee = pd.DataFrame(mydata)
```

[14]:
```python
print(employee)
```

```
   EmpID EmpName
0    111  Ashish
1    112   Vinod
2    123   Abhay
3    114   Munna
4    231  Deepak
```

[15]:
```python
arr = np.array([11,12,13,14,15,16,17])
print(arr)
```

```
[11 12 13 14 15 16 17]
```

[16]:
```python
print(np.mean(arr))
```

```
14.0
```

[17]:
```python
print(np.median(arr))
```

```
14.0
```

[18]:
```python
print(np.max(arr))
```

```
17
```

[19]:
```python
print(np.min(arr))
```

```
11
```

[20]:
```
print(np.percentile(arr,7))
```

```
11.42
```

[21]:
```
print(np.stack(arr))
```

```
[11 12 13 14 15 16 17]
```

[22]:
```
print(np.vstack(arr))
```

```
[[11]
 [12]
 [13]
 [14]
 [15]
 [16]
 [17]]
```

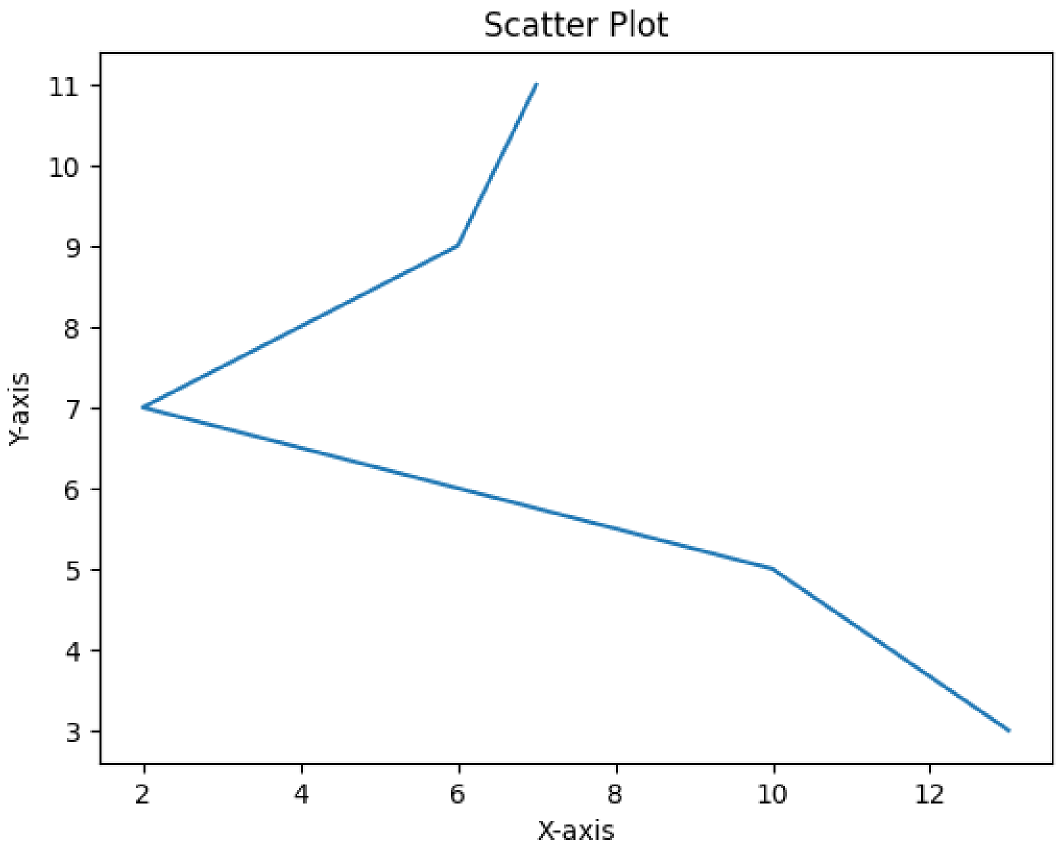
[23]:
```
print(np.hstack(arr))
```

```
[11 12 13 14 15 16 17]
```

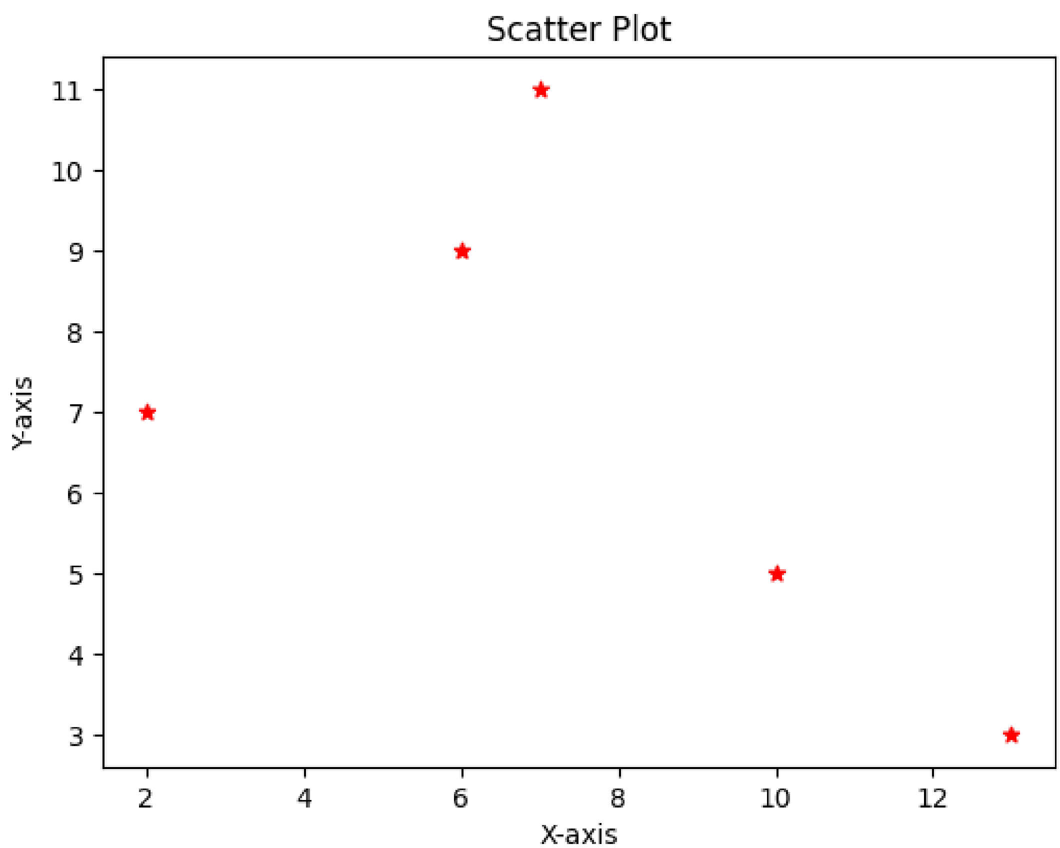
[24]:
```
import matplotlib.pyplot as plt
```

[25]:
```
x = [13,10,2,6,7]
y = [3,5,7,9,11]
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
```

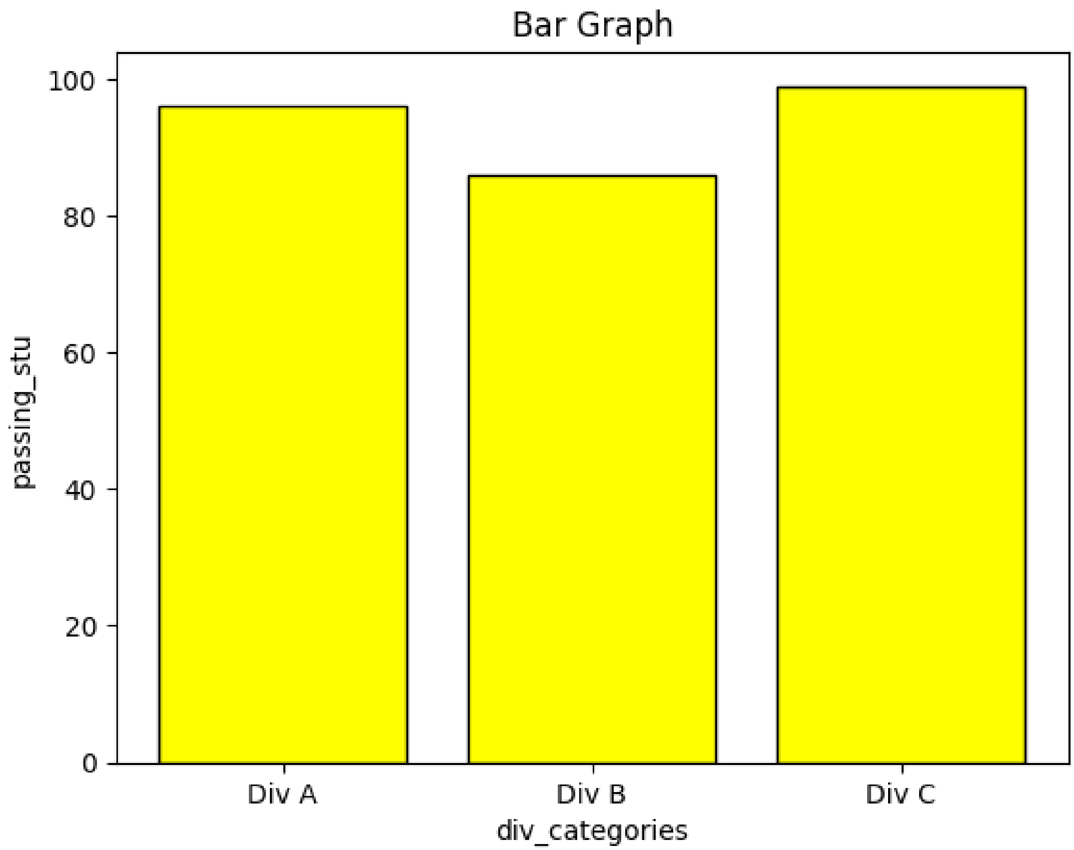
[25]:
```
Text(0.5, 1.0, 'Scatter Plot')
```



[26]:
```
x = [13,10,2,6,7]
y = [3,5,7,9,11]
plt.scatter(x,y,color='red',marker='*')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.show()
```



[27]:
```
div_categories = ['Div A' , 'Div B' , 'Div C']
passing_stu = [96,86,99]

plt.bar(div_categories , passing_stu , color = 'yellow' , edgecolor='black')
plt.xlabel('div_categories')
plt.ylabel('passing_stu')
plt.title('Bar Graph')
plt.show()
```

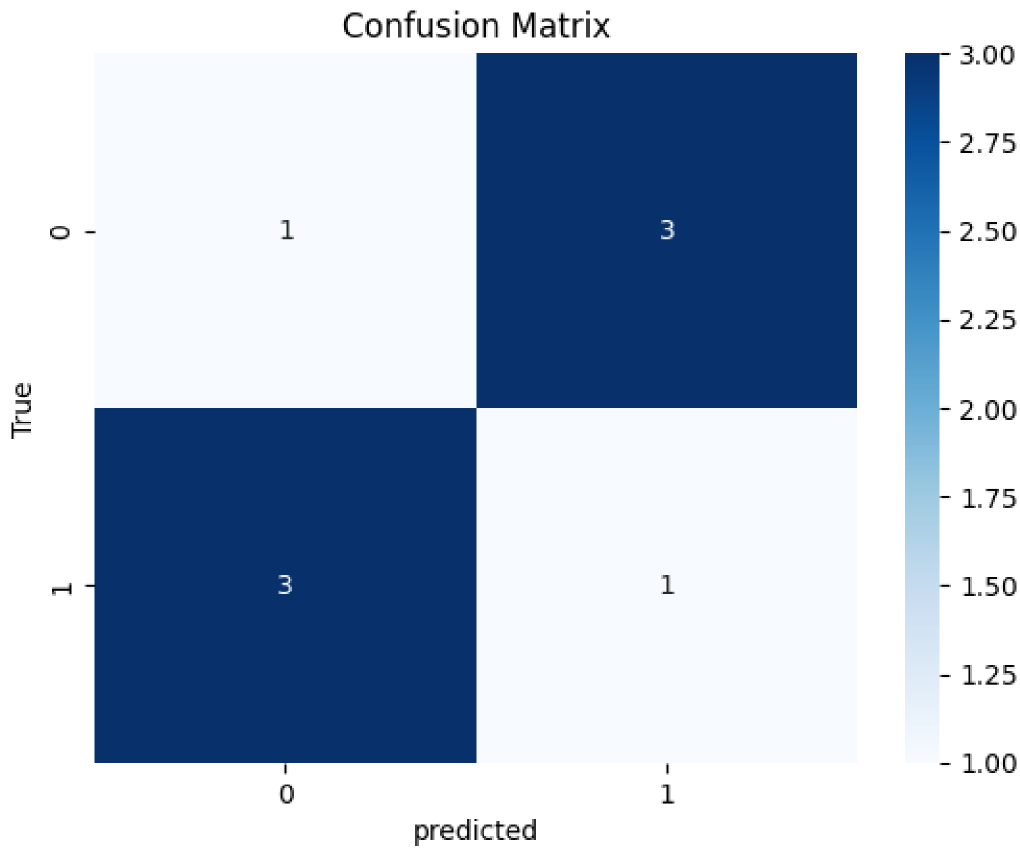


[28]:
```
#2
from sklearn.metrics import confusion_matrix
import numpy as np
true_values = np.array([0,1,1,0,1,0,0,1])
predicted_values = np.array([1,0,1,1,0,1,0,0])
cm = confusion_matrix(true_values,predicted_values)
print("Confusion Matrix")
print(cm)
```

```
Confusion Matrix
[[1 3]
 [3 1]]
```

[29]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cm, annot=True, fmt='d' , cmap='Blues')

plt.xlabel('predicted')
plt.ylabel('True')
```

```
plt.title('Confusion Matrix')
plt.show()
```

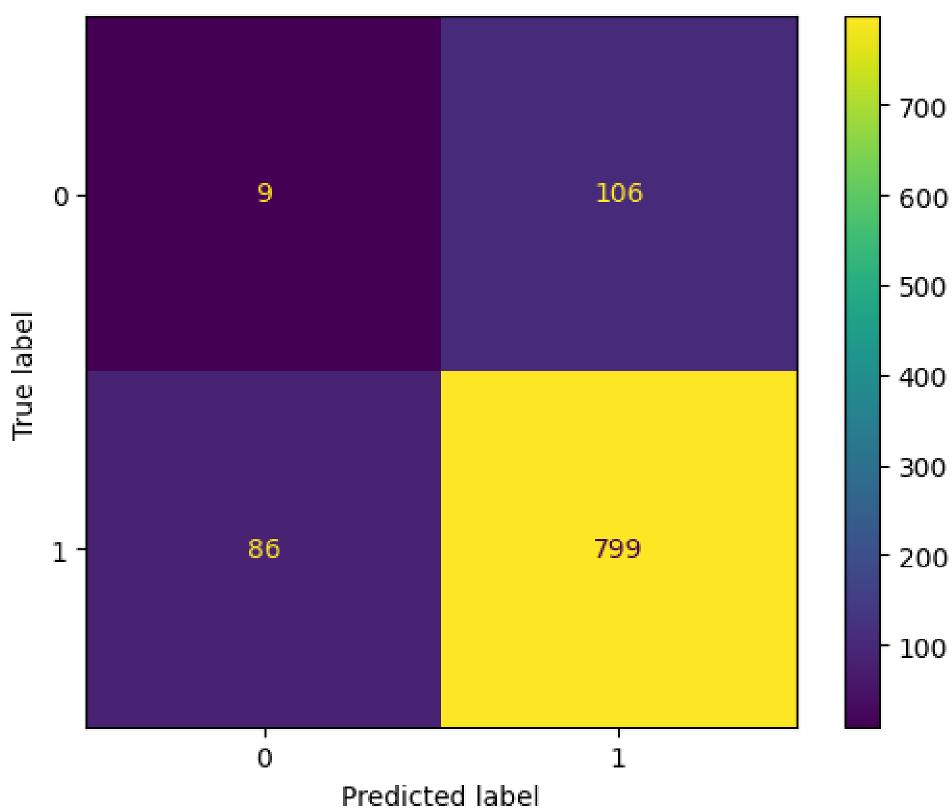


[30]:

```
from sklearn.metrics import confusion_matrix
import numpy as np

true_values = np.array([1,0,0,1,0,1])
predicted_values = np.array([0,1,1,0,0,1])

cm = confusion_matrix(true_values,predicted_values)

TN,FP,FN,TP  = cm.ravel()

print("Confusion_matrix")
print(cm)

print(f"True Positive (TP): {TN}")
print(f"False Positive (FP): {FP}")
print(f"False Negative (FN): {FN}")
print(f"True Negative (TN): {TN}")
```

```
Confusion_matrix
[[1 2]
 [2 1]]
True Positive (TP): 1
False Positive (FP): 2
False Negative (FN): 2
True Negative (TN): 1
```

[31]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import metrics

actual = np.random.binomial(1, 0.9, size=1000)
predicted = np.random.binomial(1, 0.9 , size=1000)

confusion_matrix = metrics.confusion_matrix(actual,predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=confusion_matrix)

cm_display.plot()
plt.show()
```



[32]:

```python
#3
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split



X = X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9],
              [10, 11, 12],
              [13, 14, 15],
              [16, 17, 18],
              [19, 20, 21],
              [22, 23, 24],
              [25, 26, 27],
              [28, 29, 30]])

y = np.array([1,2,3,4,5,6,7,8,9,10])

df = pd.DataFrame(X, columns=['Feature1', 'Feature2', 'Feature3'])

df['Target'] = y

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=0)


print("X_train:")
print(X_train)

print("\nX_test:")
print(X_test)

print("\ny_train:")
print(y_train)

print("\ny_test:")
print(y_test)
```

```
X_train:
   Feature1  Feature2  Feature3
9        28        29        30
1         4         5         6
```

```
6          19        20        21
7          22        23        24
3          10        11        12
0           1         2         3
5          16        17        18

X_test:
   Feature1  Feature2  Feature3
2         7         8         9
8        25        26        27
4        13        14        15

y_train:
9    10
1     2
6     7
7     8
3     4
0     1
5     6
Name: Target, dtype: int64

y_test:
2    3
8    9
4    5
Name: Target, dtype: int64
```

[36]:

```
#4
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Salary_Data.csv')


X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3,␣
↪random_state=0)


from sklearn.linear_model import LinearRegression
```

```python
regressor = LinearRegression()
regressor.fit(X_train, y_train)


viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()


y_pred = regressor.predict(X_test)
for pred in y_pred:
    print("Salary_Data:", pred)
```


Salary VS Experience (Training set)
Salary
120000
100000
80000
60000
40000
2
4
6
8
10
Year of Experience


```
Salary_Data: 40835.105908714744
Salary_Data: 123079.39940819162
Salary_Data: 65134.556260832906
Salary_Data: 63265.36777220843
Salary_Data: 115602.64545369372
Salary_Data: 108125.89149919583
Salary_Data: 116537.23969800596
Salary_Data: 64199.96201652067
Salary_Data: 76349.68719257976
Salary_Data: 100649.13754469794
```

[7]:

```
#5
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, cross_val_score

dp = pd.read_csv("Mobile_Price_Prediction_train.csv")

# Convert "Extracurricular Activities" to numerical (0 for 'No', 1 for 'Yes')
dp['blue'] = dp['blue'].map({'No': 0, 'Yes': 1})

# Select numerical features, including 'Extracurricular Activities'
numerical_features = dp.select_dtypes(include=np.number).columns

# Calculate and print correlation matrix
print(dp[numerical_features].corr())
print(dp.describe())
```

```
               battery_power  blue  clock_speed  dual_sim        fc    four_g  \
battery_power       1.000000   NaN     0.011482 -0.041847  0.033334  0.015665
blue                     NaN   NaN          NaN       NaN       NaN       NaN
clock_speed         0.011482   NaN     1.000000 -0.001315 -0.000434 -0.043073
dual_sim           -0.041847   NaN    -0.001315  1.000000 -0.029123  0.003187
fc                  0.033334   NaN    -0.000434 -0.029123  1.000000 -0.016560
four_g              0.015665   NaN    -0.043073  0.003187 -0.016560  1.000000
int_memory         -0.004004   NaN     0.006545 -0.015679 -0.029133  0.008690
m_dep               0.034085   NaN    -0.014364 -0.022142 -0.001791 -0.001823
mobile_wt           0.001844   NaN     0.012350 -0.008979  0.023618 -0.016537
n_cores            -0.029727   NaN    -0.005724 -0.024658 -0.013356 -0.029706
pc                  0.031441   NaN    -0.005245 -0.017143  0.644595 -0.005598
px_height           0.014901   NaN    -0.014523 -0.020875 -0.009990 -0.019236
px_width           -0.008402   NaN    -0.009476  0.014291 -0.005176  0.007448
ram                -0.000653   NaN     0.003443  0.041072  0.015099  0.007313
sc_h               -0.029959   NaN    -0.029078 -0.011949 -0.011014  0.027166
sc_w               -0.021421   NaN    -0.007378 -0.016666 -0.012373  0.037005
talk_time           0.052510   NaN    -0.011432 -0.039404 -0.006829 -0.046628
three_g             0.011522   NaN    -0.046433 -0.014008  0.001793  0.584246
touch_screen       -0.010516   NaN     0.019756 -0.017117 -0.014828  0.016758
wifi               -0.008343   NaN    -0.024471  0.022740  0.020085 -0.017620
price_range         0.200723   NaN    -0.006606  0.017444  0.021998  0.014772

               int_memory     m_dep  mobile_wt   n_cores  ...  px_height  \
battery_power   -0.004004  0.034085   0.001844 -0.029727  ...   0.014901
blue                  NaN       NaN        NaN       NaN  ...        NaN
clock_speed      0.006545 -0.014364   0.012350 -0.005724  ...  -0.014523
dual_sim        -0.015679 -0.022142  -0.008979 -0.024658  ...  -0.020875
fc              -0.029133 -0.001791   0.023618 -0.013356  ...  -0.009990
four_g           0.008690 -0.001823  -0.016537 -0.029706  ...  -0.019236
```

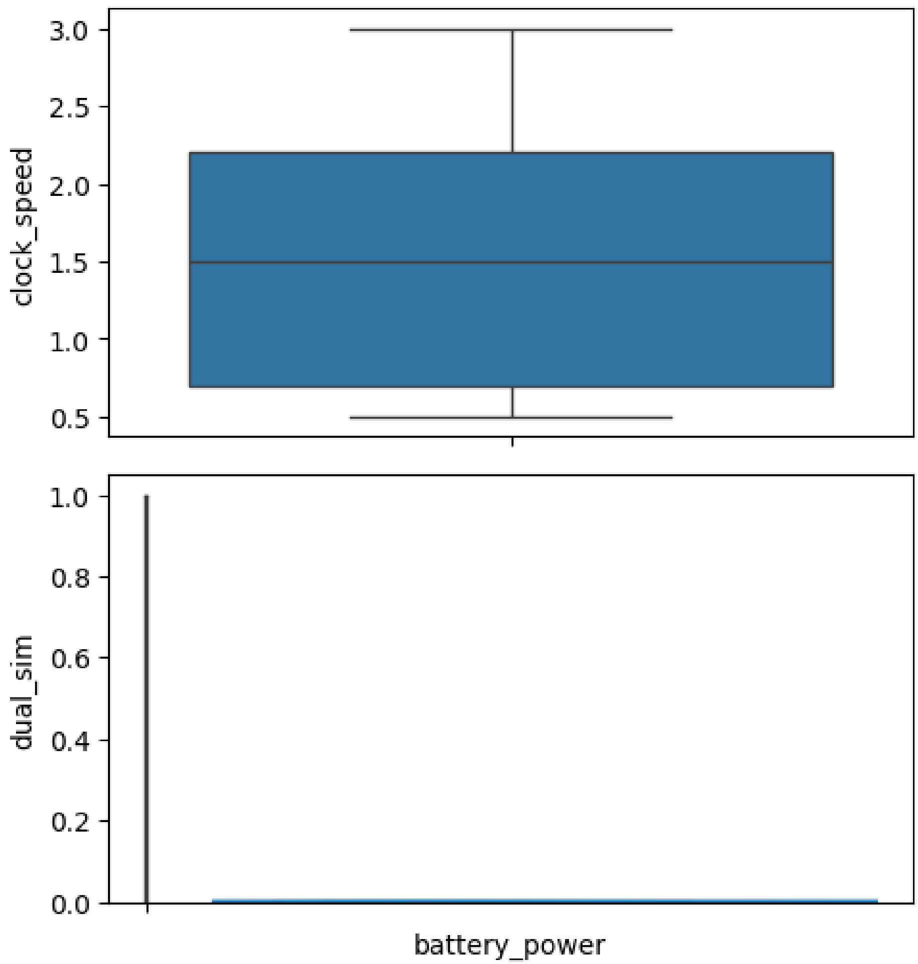
```
int_memory       1.000000  0.006886  -0.034214 -0.028310  …   0.010441
m_dep            0.006886  1.000000   0.021756 -0.003504  …   0.025263
mobile_wt       -0.034214  0.021756   1.000000 -0.018989  …   0.000939
n_cores         -0.028310 -0.003504  -0.018989  1.000000  …  -0.006872
pc              -0.033273  0.026282   0.018844 -0.001193  …  -0.018465
px_height        0.010441  0.025263   0.000939 -0.006872  …   1.000000
px_width        -0.008335  0.023566   0.000090  0.024480  …   0.510664
ram              0.032813 -0.009434  -0.002581  0.004868  …  -0.020352
sc_h             0.037771 -0.025348  -0.033855 -0.000315  …   0.059615
sc_w             0.011731 -0.018388  -0.020761  0.025826  …   0.043038
talk_time       -0.002790  0.017003   0.006209  0.013148  …  -0.010645
three_g         -0.009366 -0.012065   0.001551 -0.014733  …  -0.031174
touch_screen    -0.026999 -0.002638  -0.014368  0.023774  …   0.021891
wifi             0.006993 -0.028353  -0.000409 -0.009964  …   0.051824
price_range      0.044435  0.000853  -0.030302  0.004399  …   0.148858

               px_width       ram      sc_h      sc_w  talk_time   three_g  \
battery_power -0.008402 -0.000653 -0.029959 -0.021421   0.052510  0.011522
blue                NaN       NaN       NaN       NaN        NaN       NaN
clock_speed   -0.009476  0.003443 -0.029078 -0.007378  -0.011432 -0.046433
dual_sim       0.014291  0.041072 -0.011949 -0.016666  -0.039404 -0.014008
fc            -0.005176  0.015099 -0.011014 -0.012373  -0.006829  0.001793
four_g         0.007448  0.007313  0.027166  0.037005  -0.046628  0.584246
int_memory    -0.008335  0.032813  0.037771  0.011731  -0.002790 -0.009366
m_dep          0.023566 -0.009434 -0.025348 -0.018388   0.017003 -0.012065
mobile_wt      0.000090 -0.002581 -0.033855 -0.020761   0.006209  0.001551
n_cores        0.024480  0.004868 -0.000315  0.025826   0.013148 -0.014733
pc             0.004196  0.028984  0.004938 -0.023819   0.014657 -0.001322
px_height      0.510664 -0.020352  0.059615  0.043038  -0.010645 -0.031174
px_width       1.000000  0.004105  0.021599  0.034699   0.006720  0.000350
ram            0.004105  1.000000  0.015996  0.035576   0.010820  0.015795
sc_h           0.021599  0.015996  1.000000  0.506144  -0.017335  0.012033
sc_w           0.034699  0.035576  0.506144  1.000000  -0.022821  0.030941
talk_time      0.006720  0.010820 -0.017335 -0.022821   1.000000 -0.042688
three_g        0.000350  0.015795  0.012033  0.030941  -0.042688  1.000000
touch_screen  -0.001628 -0.030455 -0.020023  0.012720   0.017196  0.013917
wifi           0.030319  0.022669  0.025929  0.035423  -0.029504  0.004316
price_range    0.165818  0.917046  0.022986  0.038711   0.021859  0.023611

               touch_screen      wifi  price_range
battery_power     -0.010516 -0.008343     0.200723
blue                    NaN       NaN          NaN
clock_speed        0.019756 -0.024471    -0.006606
dual_sim          -0.017117  0.022740     0.017444
fc                -0.014828  0.020085     0.021998
four_g             0.016758 -0.017620     0.014772
int_memory        -0.026999  0.006993     0.044435
m_dep             -0.002638 -0.028353     0.000853
```

```
mobile_wt         -0.014368 -0.000409    -0.030302
n_cores            0.023774 -0.009964     0.004399
pc                -0.008742  0.005389     0.033599
px_height          0.021891  0.051824     0.148858
px_width          -0.001628  0.030319     0.165818
ram               -0.030455  0.022669     0.917046
sc_h              -0.020023  0.025929     0.022986
sc_w               0.012720  0.035423     0.038711
talk_time          0.017196 -0.029504     0.021859
three_g            0.013917  0.004316     0.023611
touch_screen       1.000000  0.011917    -0.030411
wifi               0.011917  1.000000     0.018785
price_range       -0.030411  0.018785     1.000000

[21 rows x 21 columns]
       battery_power  blue  clock_speed     dual_sim           fc  \
count    2000.000000   0.0  2000.000000  2000.000000  2000.000000
mean     1238.518500   NaN     1.522250     0.509500     4.309500
std       439.418206   NaN     0.816004     0.500035     4.341444
min       501.000000   NaN     0.500000     0.000000     0.000000
25%       851.750000   NaN     0.700000     0.000000     1.000000
50%      1226.000000   NaN     1.500000     1.000000     3.000000
75%      1615.250000   NaN     2.200000     1.000000     7.000000
max      1998.000000   NaN     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  ...  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  ...
mean      0.521500    32.046500     0.501750   140.249000     4.520500  ...
std       0.499662    18.145715     0.288416    35.399655     2.287837  ...
min       0.000000     2.000000     0.100000    80.000000     1.000000  ...
25%       0.000000    16.000000     0.200000   109.000000     3.000000  ...
50%       1.000000    32.000000     0.500000   141.000000     4.000000  ...
75%       1.000000    48.000000     0.800000   170.000000     7.000000  ...
max       1.000000    64.000000     1.000000   200.000000     8.000000  ...

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean     11.011000     0.761500      0.503000     0.507000     1.500000
```

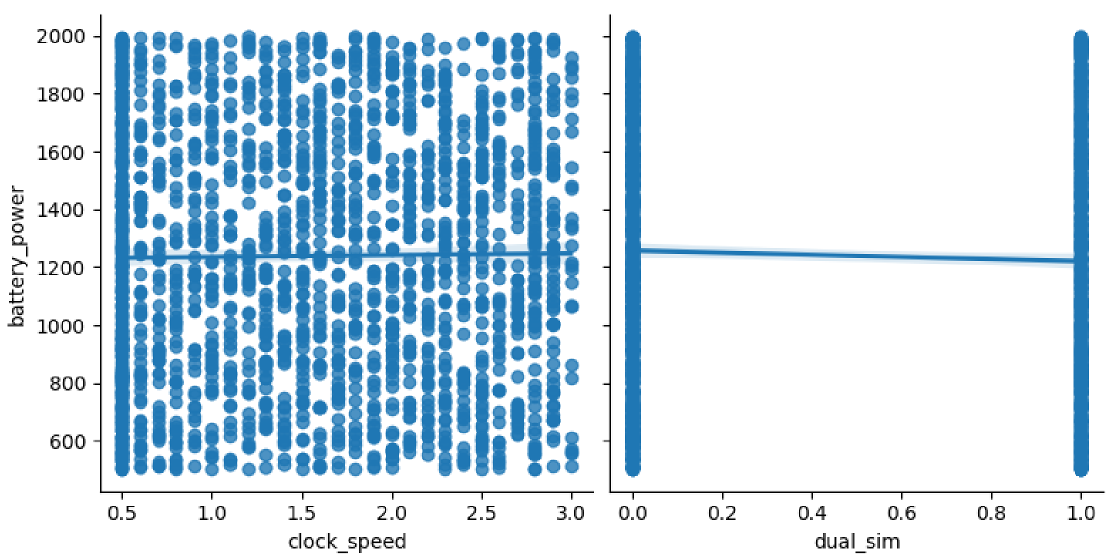
```
std       5.463955      0.426273      0.500116      0.500076      1.118314
min       2.000000      0.000000      0.000000      0.000000      0.000000
25%       6.000000      1.000000      0.000000      0.000000      0.750000
50%      11.000000      1.000000      1.000000      1.000000      1.500000
75%      16.000000      1.000000      1.000000      1.000000      2.250000
max      20.000000      1.000000      1.000000      1.000000      3.000000

[8 rows x 21 columns]
```

[4]:

```python
#5
X = dp[['clock_speed', 'dual_sim']]
y = dp['battery_power']

fig, axs = plt.subplots(2, figsize=(5, 5))

plt1 = sns.boxplot(dp['clock_speed'], ax=axs[0])
plt2 = sns.boxplot(dp['dual_sim'], ax=axs[1])

plt.tight_layout()

sns.distplot(dp['battery_power']);

sns.pairplot(dp, x_vars=['clock_speed', 'dual_sim'],
             y_vars='battery_power', height=4, aspect=1, kind='reg')

plt.show()
```



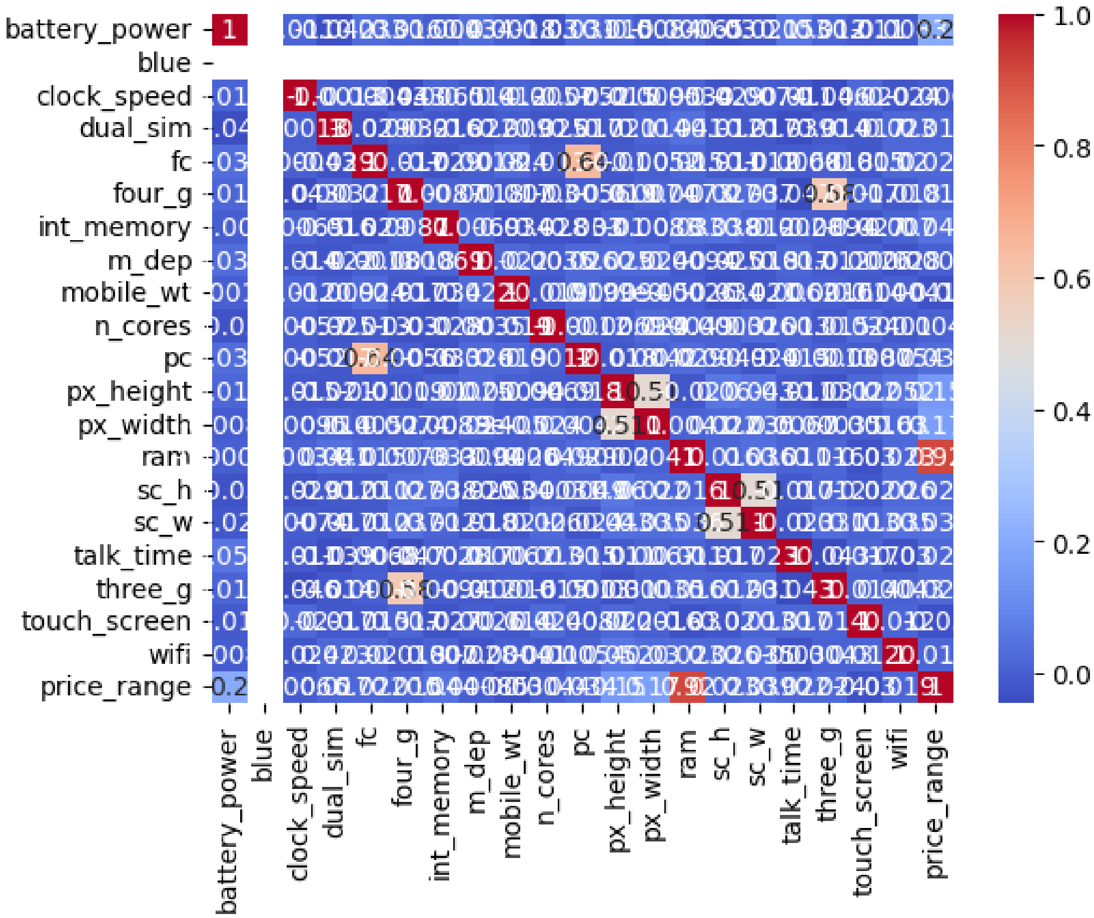
clock_speed
3.0
2.5
2.0
1.5
1.0
0.5
dual_sim
1.0
0.8
0.6
0.4
0.2
0.0
battery_power



battery_power
2000
1800
1600
1400
1200
1000
800
600
0.5
1.0
1.5
2.0
2.5
3.0
clock_speed
0.0
0.2
0.4
0.6
0.8
1.0
dual_sim

[9]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[9], line 4
      2 sns.heatmap(dp[numerical_features].corr(), annot=True, cmap='coolwarm')
      3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 ↪
 ↪random_state=100)
----> 4 reg_model = linear_model.LinearRegression()
      5 reg_model.fit(X_train, y_train)
      8 print('Intercept:', reg_model.intercept_)

NameError: name 'linear_model' is not defined
```



[10]:
```python
#7
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

dn = pd.read_csv("Experiment7.csv")

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dn['Sex'] = le.fit_transform(dn['Sex'])
dn['Embarked'] = le.fit_transform(dn['Embarked'])
print(dn)

# Putting feature variable to X
X = dn.drop('Survived', axis=1)

# Putting response variable to y
y = dn['Survived']

# Splitting the data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,␣
↪random_state=42)

# Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

# Create a Gaussian Classifier
clf = RandomForestClassifier(n_estimators=10)

# Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train, y_train)
Pred = clf.predict(X_test)

print(Pred)
```

```
     Survived  Pclass  Sex  Age  SibSp  Parch  Fare  Embarked
0           0       3    1   22      1      0     7         2
1           1       1    0   38      1      0    71         0
2           1       3    0   26      0      0     7         2
3           1       1    0   35      1      0    53         2
4           0       3    1   35      0      0     8         2
..        …      …   …   …     …     …    …       …
886         0       2    1   27      0      0    13         2
887         1       1    0   19      0      0    30         2
888         0       3    0   32      1      2    23         2
```

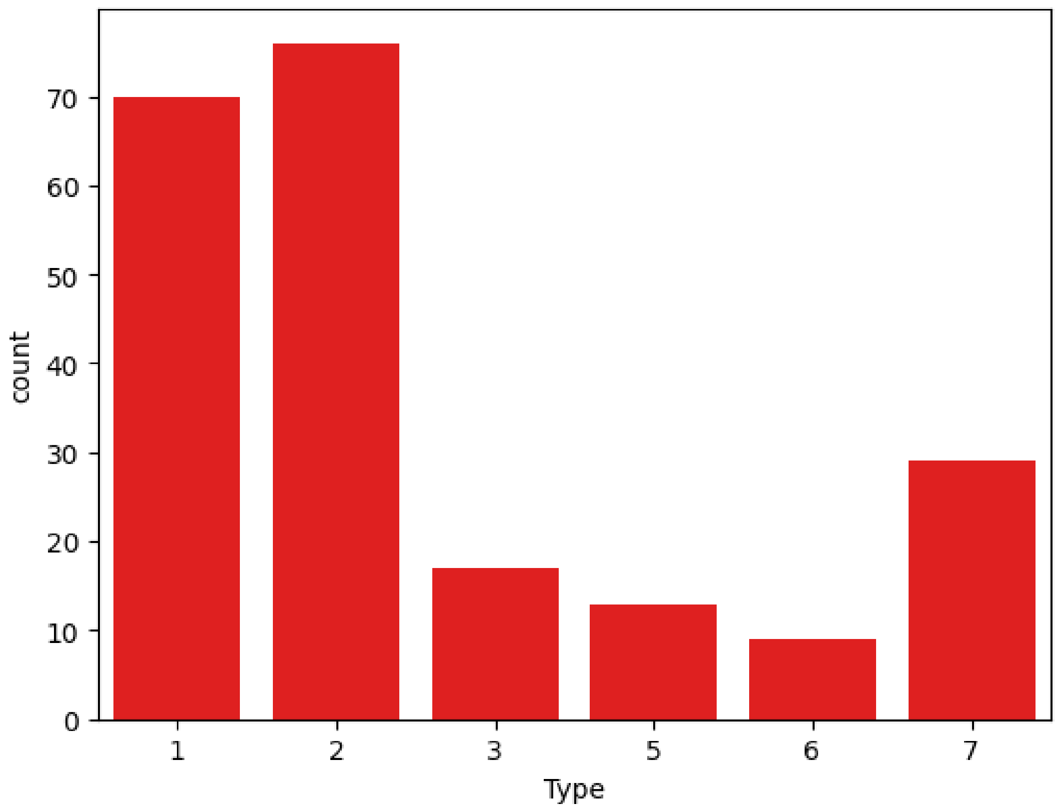
```
889           1        1    1   26       0       0    30          0
890           0        3    1   32       0       0     7          1

[891 rows x 8 columns]
[0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1 1
 0 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1
 1 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0
 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1
 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
 1 1 1 1 0 0 0 0 0]
```

[11]:

```python
#8
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
glass = pd.read_csv("Experiment8.csv")
print(glass.head())
print(glass.tail())
print(glass.shape)
print(glass.isnull().sum())
sns.countplot(x='Type', data=glass, color='red')
plt.show()
nb = GaussianNB()
x = glass.drop(columns=['Type'])
y = glass['Type']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=4)
nb.fit(x_train, y_train)
y_pred = nb.predict(x_test)
print("Predictions:", y_pred)
print("Accuracy Score:", accuracy_score(y_test,y_pred))
```

```
        RI     Na    Mg    Al     Si     K    Ca   Ba   Fe  Type
0  1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.0  0.0     1
1  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.0     1
2  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.0     1
3  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.0     1
4  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.0     1
          RI     Na   Mg    Al     Si     K    Ca    Ba   Fe  Type
209  1.51623  14.14  0.0  2.88  72.61  0.08  9.18  1.06  0.0     7
```

```
210  1.51685  14.92  0.0  1.99  73.06  0.00  8.40  1.59  0.0     7
211  1.52065  14.36  0.0  2.02  73.42  0.00  8.44  1.64  0.0     7
212  1.51651  14.38  0.0  1.94  73.61  0.00  8.48  1.57  0.0     7
213  1.51711  14.23  0.0  2.08  73.36  0.00  8.62  1.67  0.0     7
(214, 10)
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```




```
Predictions: [1 7 5 3 3 1 2 1 1 1 5 1 1 7 1 1 1 7 7 1 1 1 7 1 6 7 3 3 7 2 1 7 1
1 1 1 1
 2 1 1 5 7 2]
Accuracy Score: 0.4883720930232558
```

[12]:

```python
#9
import pandas as pd

# Importing the dataset
dataset = pd.read_csv("Experiment9.csv")

X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,␣
↪random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)

print(ac)
print(cm)
```

```
print(y_pred)
```

```
[[55  3]
 [ 1 21]]
0.95
[[55  3]
 [ 1 21]]
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1]
```

[13]:
```
#10
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

fish = pd.read_csv("Experiment10.csv")


X = fish.drop('Species', axis=1)
y = fish['Species']

# Split data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)


model = SVC(kernel='linear', C=1)


model.fit(X_train, y_train)


svm_pred = model.predict(X_test)


accuracy = model.score(X_test, y_test)
print(f"Accuracy: {accuracy:.4f}")

print(svm_pred)
```

```
Accuracy: 0.9375
['Perch' 'Smelt' 'Pike' 'Roach' 'Perch' 'Bream' 'Smelt' 'Roach' 'Perch'
 'Pike' 'Bream' 'Whitefish' 'Bream' 'Parkki' 'Bream' 'Bream' 'Perch'
 'Perch' 'Perch' 'Bream' 'Smelt' 'Bream' 'Bream' 'Bream' 'Bream' 'Perch'
 'Perch' 'Roach' 'Smelt' 'Smelt' 'Pike' 'Perch']
```

[14]:
```python
#CP1exp
def sum_of_array(arr):
    return sum(arr)

# Example usage
arr = [1, 2, 3, 4, 5]
print("Sum =", sum_of_array(arr))
```

```
Sum = 15
```

[15]:
```python
#1.2
def sort_array(arr):
    return sorted(arr)

# Example usage
arr = [5, 2, 8, 1, 3]
print("Sorted Array =", sort_array(arr))
```

```
Sorted Array = [1, 2, 3, 5, 8]
```

[16]:
```python
# To implement Bubble Sort and Insertion Sort
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Example usage
arr = [64, 34, 25, 12, 22, 11, 90]
print("Sorted Array (Bubble Sort):", bubble_sort(arr))
```

```
Sorted Array (Bubble Sort): [11, 12, 22, 25, 34, 64, 90]
```

[17]:
```python
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

# Example usage
arr = [64, 34, 25, 12, 22, 11, 90]
print("Sorted Array (Insertion Sort):", insertion_sort(arr))
```

```
Sorted Array (Insertion Sort): [11, 12, 22, 25, 34, 64, 90]
```

[18]:
```python
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1

# Example usage
arr = [10, 20, 30, 40, 50]
target = 30
index = linear_search(arr, target)
print(f"Element found at index {index}" if index != -1 else "Element not found")
```

```
Element found at index 2
```

[19]:
```python
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# Example usage
arr = [10, 20, 30, 40, 50]
target = 30
index = binary_search(arr, target)
print(f"Element found at index {index}" if index != -1 else "Element not found")
```

```
Element found at index 2
```

[21]:
```python
#3
text = "Ht India"
print(text.upper())
print(text.lower())
print(text.capitalize())
print(text.swapcase())
print(text.title())
```

```
HT INDIA
ht india
Ht india
hT iNDIA
```

Ht India

[22]:
```
#4
def matrix_multiplication(A, B):
    # Get dimensions of matrices
    rows_A, cols_A = len(A), len(A[0])
    rows_B, cols_B = len(B), len(B[0])

    # Ensure matrices can be multiplied
    if cols_A != rows_B:
        return "Matrix multiplication not possible"

    # Initialize result matrix with zeros
    result = [[0 for _ in range(cols_B)] for _ in range(rows_A)]

    # Perform multiplication
    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):  # or rows_B
                result[i][j] += A[i][k] * B[k][j]
    return result

# Example usage
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
result = matrix_multiplication(A, B)
print("Matrix Multiplication Result:")
for row in result:
    print(row)
```

```
Matrix Multiplication Result:
[19, 22]
[43, 50]
```

[23]:
```
def transpose_matrix(matrix):
    # Number of rows and columns in the matrix
    rows, cols = len(matrix), len(matrix[0])

    # Initialize transposed matrix
    transposed = [[0 for _ in range(rows)] for _ in range(cols)]

    # Perform transpose
    for i in range(rows):
        for j in range(cols):
            transposed[j][i] = matrix[i][j]
    return transposed
```

```python
# Example usage
matrix = [[1, 2, 3], [4, 5, 6]]
transposed = transpose_matrix(matrix)
print("Transpose of Matrix:")
for row in transposed:
    print(row)
```

```
Transpose of Matrix:
[1, 4]
[2, 5]
[3, 6]
```

[26]:

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    # Display the linked list
    def display(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")

    # Create a linked list from a list of values
    def create_linked_list(self, values):
        for value in values:
            new_node = Node(value)
            if self.head is None:
                self.head = new_node
            else:
                temp = self.head
                while temp.next:
                    temp = temp.next
                temp.next = new_node

    # Add a node at the beginning
    def add_at_beginning(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node
```

```python
    # Add a node at a specific index
    def add_at_index(self, index, data):
        if index == 0:  # Add at the beginning
            self.add_at_beginning(data)
            return

        new_node = Node(data)
        temp = self.head
        for _ in range(index - 1):
            if temp is None:
                print("Index out of bounds")
                return
            temp = temp.next

        if temp is None:
            print("Index out of bounds")
            return

        new_node.next = temp.next
        temp.next = new_node

    # Remove a node from the beginning
    def remove_from_beginning(self):
        if self.head is None:
            print("List is empty")
            return
        self.head = self.head.next

    # Remove a node from the end
    def remove_from_end(self):
        if self.head is None:
            print("List is empty")
            return
        if self.head.next is None:
            self.head = None
            return

        temp = self.head
        while temp.next.next:
            temp = temp.next
        temp.next = None

    # Remove a node at a specific index
    def remove_at_index(self, index):
        if index == 0:  # Remove from beginning
            self.remove_from_beginning()
            return
```

```python
        temp = self.head
        for _ in range(index - 1):
            if temp is None:
                print("Index out of bounds")
                return
            temp = temp.next

        if temp is None or temp.next is None:
            print("Index out of bounds")
            return

        temp.next = temp.next.next

# Example usage
if __name__ == "__main__":
    linked_list = LinkedList()

    # Create a linked list
    print("Creating linked list...")
    linked_list.create_linked_list([10, 20, 30])
    linked_list.display()

    # Add at the beginning
    print("\nAdding 5 at the beginning...")
    linked_list.add_at_beginning(5)
    linked_list.display()

    # Add at a specific index
    print("\nAdding 25 at index 2...")
    linked_list.add_at_index(2, 25)
    linked_list.display()

    # Remove from the beginning
    print("\nRemoving from the beginning...")
    linked_list.remove_from_beginning()
    linked_list.display()

    # Remove from the end
    print("\nRemoving from the end...")
    linked_list.remove_from_end()
    linked_list.display()

    # Remove at a specific index
    print("\nRemoving at index 1...")
    linked_list.remove_at_index(1)
    linked_list.display()
```

```
Creating linked list…
10 -> 20 -> 30 -> None

Adding 5 at the beginning…
5 -> 10 -> 20 -> 30 -> None

Adding 25 at index 2…
5 -> 10 -> 25 -> 20 -> 30 -> None

Removing from the beginning…
10 -> 25 -> 20 -> 30 -> None

Removing from the end…
10 -> 25 -> 20 -> None

Removing at index 1…
10 -> 20 -> None
```

[27]:

```python
#6
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

def is_complete_binary_tree(root):
    if not root:
        return True  # An empty tree is complete

    queue = []
    queue.append(root)
    encountered_none = False  # Flag to indicate if a null node has been␣
 ↪encountered

    while queue:
        current = queue.pop(0)

        if current:
            # If we've previously encountered a None node, then the tree is not␣
 ↪complete
            if encountered_none:
                return False

            # Add left and right children to the queue
            queue.append(current.left)
            queue.append(current.right)
        else:
```

```
            # Mark the flag when encountering the first None
            encountered_none = True

    return True

# Example Usage
if __name__ == "__main__":
    # Create a binary tree
    root = Node(1)
    root.left = Node(2)
    root.right = Node(3)
    root.left.left = Node(4)
    root.left.right = Node(5)
    root.right.left = Node(6)
    # Uncomment below to make the tree incomplete
    # root.right.right = Node(7)

    if is_complete_binary_tree(root):
        print("The binary tree is complete.")
    else:
        print("The binary tree is not complete.")
```

```
The binary tree is complete.
```

[28]:
```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Function to check if the tree is a BST
def is_bst(node, min_val=float('-inf'), max_val=float('inf')):
    # Base case: An empty tree is a BST
    if node is None:
        return True

    # Check if the current node violates the min/max constraints
    if node.data <= min_val or node.data >= max_val:
        return False

    # Recursively check the left and right subtrees
    # For left subtree, the max value is the current node's value
    # For right subtree, the min value is the current node's value
    return is_bst(node.left, min_val, node.data) and is_bst(node.right, node.
 ↪data, max_val)

# Example Usage
```

```python
if __name__ == "__main__":
    # Create a binary tree
    root = Node(10)
    root.left = Node(5)
    root.right = Node(15)
    root.left.left = Node(2)
    root.left.right = Node(7)
    root.right.left = Node(12)
    root.right.right = Node(20)

    if is_bst(root):
        print("The binary tree is a Binary Search Tree (BST).")
    else:
        print("The binary tree is NOT a Binary Search Tree (BST).")
```

```
The binary tree is a Binary Search Tree (BST).
```

[ ]:

```python

```