Name :Yash Dhamecha

Reg_no : 21BCE6096

Branch :Vit Chennai Batch : Morning

Data preprocessing

1.Import the Libraries

In [1]:

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

2.Importing the dataset.

In [2]:

```
dataset = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv") dataset.head(5)
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Rela
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

In [3]:

```
dataset.shape
```

Out[3]:(1470, 35)

3.Checking for Null Values.

In [4]:

```
dataset.isnull().any()
```

Out[4]:

```
Age                       False
Attrition                 False
BusinessTravel            False
DailyRate                 False
Department                False
DistanceFromHome          False
Education                 False
EducationField            False
EmployeeCount             False
EmployeeNumber            False
EnvironmentSatisfaction   False
Gender                    False
HourlyRate                False
JobInvolvement            False
JobLevel                  False
JobRole                   False
JobSatisfaction           False
MaritalStatus             False
MonthlyIncome             False
MonthlyRate               False
NumCompaniesWorked        False
Over18                    False
OverTime                  False
PercentSalaryHike         False
PerformanceRating         False
```

```
RelationshipSatisfaction    False
StandardHours               False
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False dtype:
bool
```

In [5]:

```
dataset.isnull().sum()
```

Out[5]:

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0 dtype:
int64
```

In [6]: dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
0    Age                      1470 non-null   int64
1    Attrition                1470 non-null   int64
2    BusinessTravel           1470 non-null   object
3    DailyRate                1470 non-null   int64
4    Department               1470 non-null   object
5    DistanceFromHome         1470 non-null   int64
6    Education                1470 non-null   int64
7    EducationField           1470 non-null   object
8    EmployeeCount            1470 non-null   int64
9    EmployeeNumber           1470 non-null   int64
10   EnvironmentSatisfaction  1470 non-null   int64
11   Gender                   1470 non-null   object
12   HourlyRate               1470 non-null   int64
13   JobInvolvement           1470 non-null   int64
14   JobLevel                 1470 non-null   int64
15   JobRole                  1470 non-null   object
16   JobSatisfaction          1470 non-null   int64
17   MaritalStatus            1470 non-null   object
18   MonthlyIncome            1470 non-null   int64
```

```
19          MonthlyRate             1470 non-null   int64
20          NumCompaniesWorked      1470 non-null   int64
21          Over18                  1470 non-null   object
22          OverTime                1470 non-null   object
23          PercentSalaryHike       1470 non-null   int64
24          PerformanceRating       1470 non-null   int64
25          RelationshipSatisfaction 1470 non-null  int64
26          StandardHours           1470 non-null   int64
27          StockOptionLevel        1470 non-null   int64
28          TotalWorkingYears       1470 non-null   int64
29          TrainingTimesLastYear   1470 non-null   int64
30          WorkLifeBalance         1470 non-null   int64
31          YearsAtCompany          1470 non-null   int64
32          YearsInCurrentRole      1470 non-null   int64
33          YearsSinceLastPromotion 1470 non-null   int64  34  YearsWithCurrManager     1470 non-null   int64 dtypes: int64(27), object(8) memory usage: 402.1+ KB
```

In [4]:

```
dataset['Attrition'] = dataset['Attrition'].map({'Yes': 1, 'No': 0})
```

In [13]: `dataset.head(5)`

Out[13]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Re
0	41	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	0	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	1	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	0	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

4.Data Visualization.

In [14]:

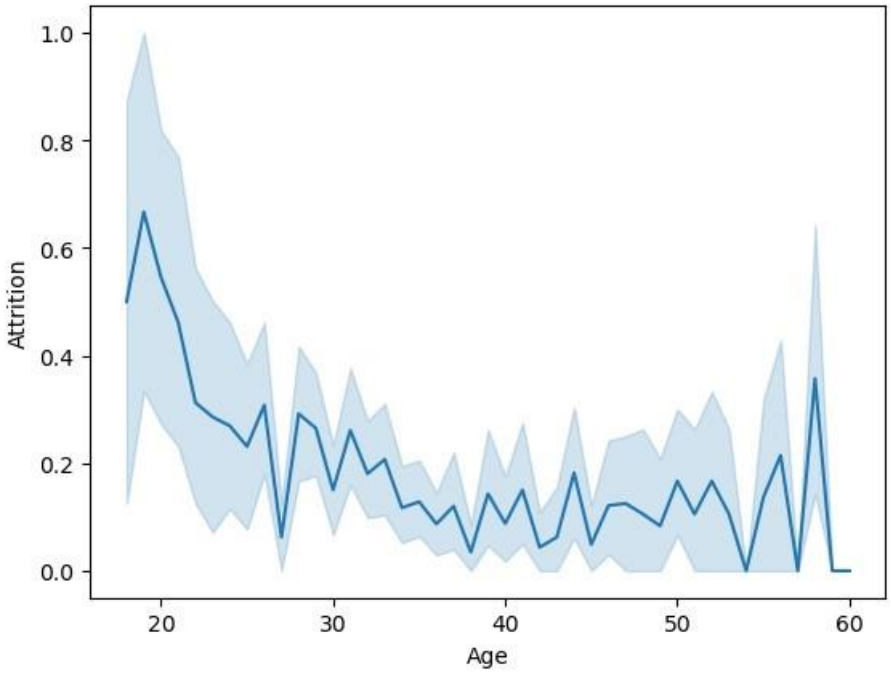
```
correlation_matrix = dataset.corr()
plt.figure(figsize=(20, 15))
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

```
C:\Users\rajes\AppData\Local\Temp\ipykernel_24744\4044223167.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated.
In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.   correlation_matrix =
dataset.corr()
```

In [19]:

```
sns.lineplot(x="Age",y="Attrition",data=dataset)
```

Out[19]:<Axes: xlabel='Age', ylabel='Attrition'>



In [17]: correlation_matrix = dataset.corr()

```
# To get the correlation of "Attrition" with other columns
attrition_correlation = correlation_matrix['Attrition'].drop('Attrition')

print(attrition_correlation)
```

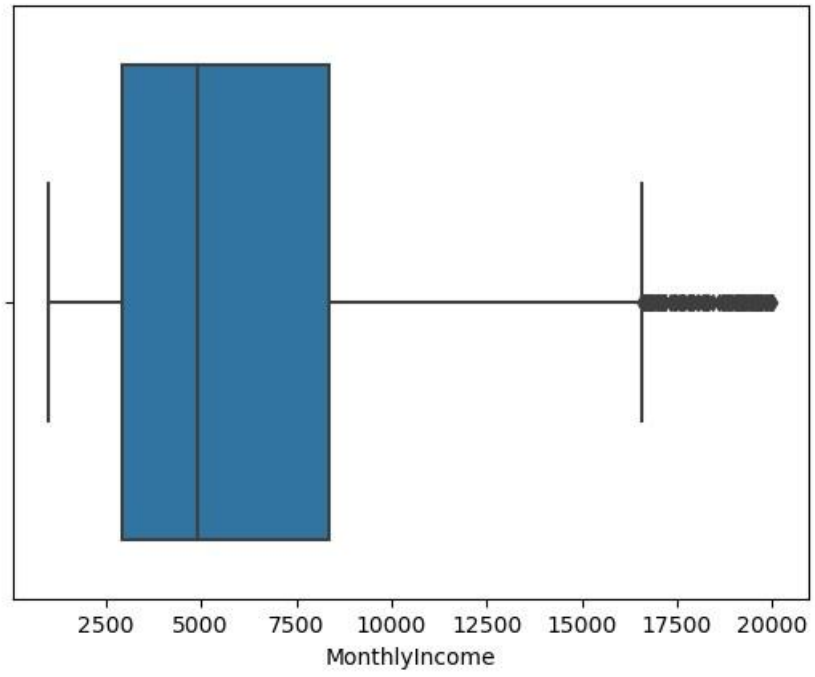
```
Age                      -0.159205
DailyRate                -0.056652
DistanceFromHome          0.077924
Education                -0.031373
EmployeeCount                  NaN
EmployeeNumber           -0.010577
EnvironmentSatisfaction  -0.103369
HourlyRate               -0.006846
JobInvolvement           -0.130016
JobLevel                 -0.169105
JobSatisfaction          -0.103481
MonthlyIncome            -0.159840
MonthlyRate               0.015170
NumCompaniesWorked        0.043494
PercentSalaryHike        -0.013478
PerformanceRating         0.002889
RelationshipSatisfaction -0.045872
StandardHours                  NaN
StockOptionLevel         -0.137145
TotalWorkingYears        -0.171063
TrainingTimesLastYear    -0.059478
WorkLifeBalance          -0.063939
YearsAtCompany           -0.134392
YearsInCurrentRole       -0.160545
YearsSinceLastPromotion  -0.033019
YearsWithCurrManager     -0.156199
Name: Attrition, dtype: float64
C:\Users\rajes\AppData\Local\Temp\ipykernel_24744\4043424376.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated.
In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = dataset.corr()
```

5.Outlier Detection

In [15]:

```
sns.boxplot(x=dataset["MonthlyIncome"])
```

Out[15]:<Axes: xlabel='MonthlyIncome'>



Inference : It shows that MonthlyIncome has outliers

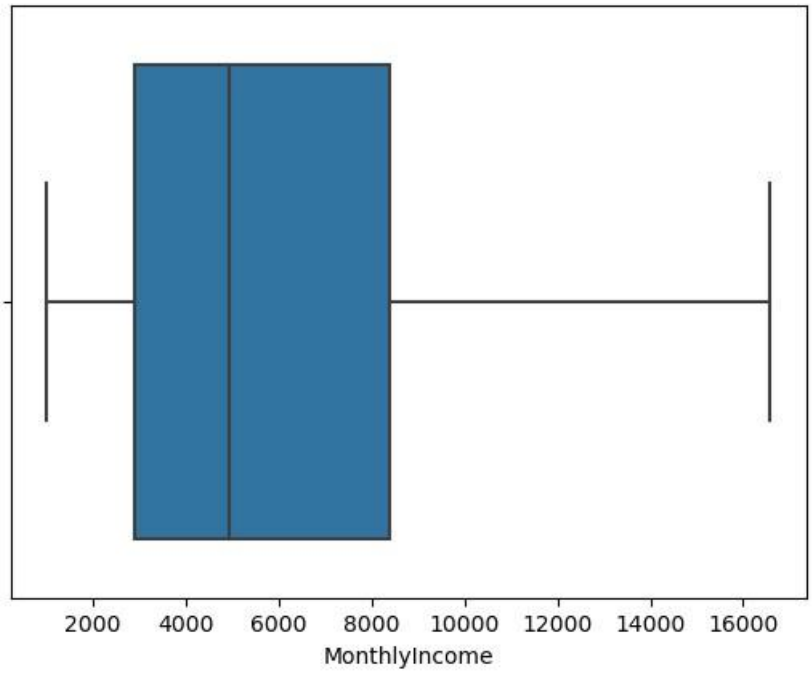
Inference : It shows that fare has outliers that are left skewed.

In [16]:

```
#So we will use flooring and capping for removing outliers
Q1 = dataset['MonthlyIncome'].quantile(0.25) Q3 = dataset['MonthlyIncome'].quantile(0.75) IQR = Q3 - Q1 whisker_width = 1.5 lower_whisker = Q1 -
(whisker_width*IQR) upper_whisker = Q3 + (whisker_width*IQR)
dataset['MonthlyIncome']=np.where(dataset['MonthlyIncome']>upper_whisker,upper_whisker,np.where(dataset['MonthlyIncome']<lower_whisker,lower
```

In [17]: `sns.boxplot(x=dataset["MonthlyIncome"])`

Out[17]:<Axes: xlabel='MonthlyIncome'>



Inference: Hence We have successfully removed outliers.

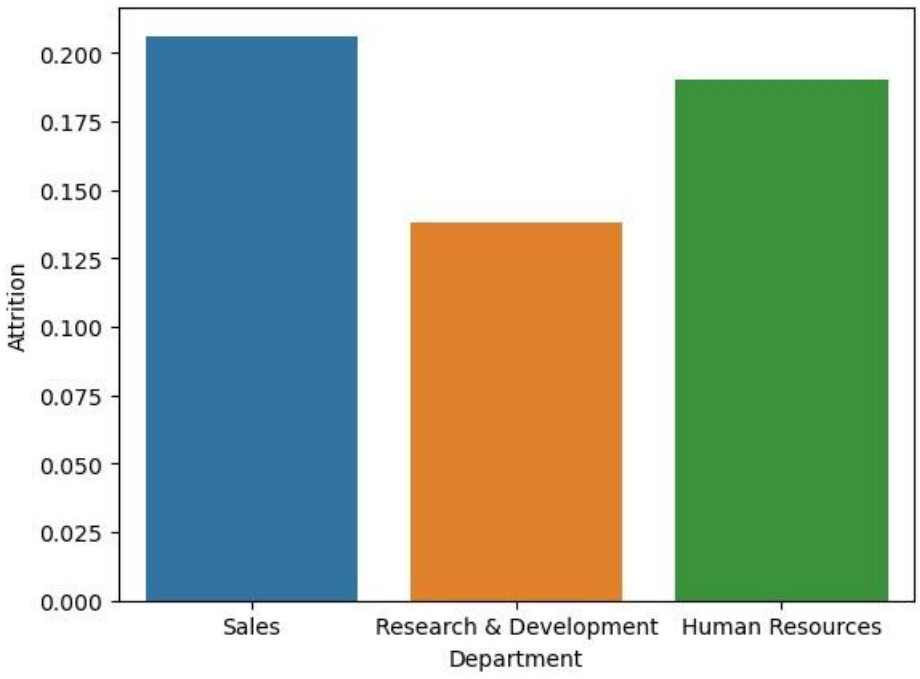
In [21]:

```
sns.barplot(y=dataset['Attrition'],x=dataset['Department'],ci=0)
```

C:\Users\rajes\AppData\Local\Temp\ipykernel_25492\729302506.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

sns.barplot(y=dataset['Attrition'],x=dataset['Department'],ci=0) Out[21]:<Axes: xlabel='Department', ylabel='Attrition'>



In [ ]:

Inference: It shows that Sales columns has the most attrition.

6.Splitting Dependent and Independent variables

In [22]:

```
#dropping unecessary columns
dataset.drop(['Over18','EmployeeCount'],axis=1,inplace=True)
```

In [24]:

```
x=dataset.drop(columns=['Attrition'])
y=dataset.iloc[:,0:1]
```

In [25]:

```
y.shape
```

Out[25]:(1470, 1) 7.Perform Encoding

In [40]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
columns_to_encode = ['BusinessTravel', 'Department', 'EducationField','Gender','JobRole','MaritalStatus','OverTime']

# Create a dictionary to store label encoders and mappings label_encoders = {}
# Perform label encoding and print mappings for
column in columns_to_encode:
   le = LabelEncoder()    x[column] = le.fit_transform(x[column])
label_encoders[column] = dict(zip(le.classes_, le.transform(le.classes_)))

x.head(5)
```

Out[40]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	Gender	..
0	41	2	1102	2	1	2	1	1	2	0	..
1	49	1	279	1	8	1	1	2	3	1	..
2	37	2	1373	1	2	2	4	4	4	1	..
3	33	1	1392	1	3	4	1	5	4	0	..
4	27	2	591	1	2	1	3	7	1	1	..

5 rows × 32 columns

In [42]:

```
# Print the mappings for each column
print("\nMappings:") for column, mapping in
label_encoders.items():
   print(f"{column} Mapping:")    for
key, value in mapping.items():
      print(f"{key}: {value}")
print()
```

```
Mappings:
BusinessTravel Mapping:
Non-Travel: 0
Travel_Frequently: 1
Travel_Rarely: 2

Department Mapping:
Human Resources: 0
Research & Development: 1
Sales: 2

EducationField Mapping:
Human Resources: 0
Life Sciences: 1
Marketing: 2 Medical:
3
Other: 4

Technical Degree: 5

Gender Mapping:
Female: 0
Male: 1

JobRole Mapping:
Healthcare Representative: 0
Human Resources: 1
Laboratory Technician: 2
Manager: 3
Manufacturing Director: 4
Research Director: 5
Research Scientist: 6
Sales Executive: 7 Sales
Representative: 8

MaritalStatus Mapping:
Divorced: 0
Married: 1
Single: 2

OverTime Mapping:
No: 0
Yes: 1
```

8.Feature Scaling

In [43]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [44]:
```
x=sc.fit_transform(x)
```

In [45]:
```
x
```

Out[45]:
```
array([[ 0.4463504 ,  0.59004834,  0.74252653, ..., -0.0632959 ,
        -0.67914568,  0.24583399],
       [ 1.32236521, -0.91319439, -1.2977746 , ...,  0.76499762,
        -0.36871529,  0.80654148],
       [ 0.008343  ,  0.59004834,  1.41436324, ..., -1.16768726,
        -0.67914568, -1.15593471],
       ...,
       [-1.08667552,  0.59004834, -1.60518328, ..., -0.61549158,
        -0.67914568, -0.31487349],
       [ 1.32236521, -0.91319439,  0.54667746, ...,  0.48889978,
        -0.67914568,  1.08689522],
       [-0.32016256,  0.59004834, -0.43256792, ..., -0.33939374,
        -0.36871529, -0.59522723]])
```

9.Splitting Data into Train and Test

In [46]:
```
from sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [49]:
```
print(x_train.shape)
print(y_train.shape) print(x_test.shape)
print(y_test.shape)
```

```
(1029, 32)
(1029, 1)
(441, 32)
(441, 1)
```

In [50]:
```
x.shape
```

Out[50]:
```
(1470, 32)
```

Model Building

1.Logistic Regression

In [57]:
```
from sklearn.linear_model import LogisticRegression from sklearn.metrics import
accuracy_score, classification_report, confusion_matrix

# Create and train the Logistic Regression model
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(x_train, y_train)

# Make predictions y_pred_lr =
logistic_regression_model.predict(x_test)

# Calculate performance metrics accuracy_lr =
accuracy_score(y_test, y_pred_lr) confusion_matrix_lr =
confusion_matrix(y_test, y_pred_lr) classification_report_lr =
classification_report(y_test, y_pred_lr)

# Print the metrics print("Logistic
Regression Metrics:")
print(f"Accuracy: {accuracy_lr}")
print("Confusion Matrix:")
print(confusion_matrix_lr)
print("Classification Report:")
print(classification_report_lr)
```

C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a 1d array was e xpected. Please change the shape of y to (n_samples,), for example using ravel().
  y = column_or_1d(y, warn=True)
Logistic Regression Metrics: Accuracy:
0.11337868480725624
Confusion Matrix:
[[1 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0] [1
0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]
Classification Report:
              precision   recall f1-score   support

          18   0.50   0.50   0.50   2
          19   0.00   0.00   0.00   3
          20   0.00   0.00   0.00   3
          21   0.17   0.50   0.25   2
          22   0.10   0.25   0.14   4
          23   0.00   0.00   0.00   3
          24   0.08   0.11   0.10   9
          25   0.40   0.17   0.24   12
          26   0.08   0.11   0.09   9
          27   0.06   0.06   0.06   16
          28   0.23   0.27   0.25   11
          29   0.14   0.26   0.18   19
          30   0.12   0.07   0.09   14
          31   0.11   0.04   0.06   27
          32   0.14   0.11   0.12   18
          33   0.14   0.13   0.14   15
          34   0.25   0.17   0.20   30
          35   0.09   0.13   0.11   23
          36   0.12   0.14   0.13   21
          37   0.25   0.15   0.19   13
          38   0.08   0.05   0.06   19
          39   0.15   0.22   0.18   9
          40   0.07   0.07   0.07   14
          41   0.00   0.00   0.00   11
          42   0.12   0.08   0.10   12
          43   0.09   0.07   0.08   14
          44   0.07   0.11   0.09   9
          45   0.00   0.00   0.00   10
          46   0.11   0.11   0.11   9
          47   0.00   0.00   0.00   8
          48   0.00   0.00   0.00   5
          49   0.00   0.00   0.00   6
          50   0.09   0.08   0.08   13
          51   0.00   0.00   0.00   5
          52   0.00   0.00   0.00   6
          53   0.14   0.14   0.14   7
          54   0.25   0.50   0.33   4
          55   0.33   0.20   0.25   10
          56   0.00   0.00   0.00   5
          57   0.00   0.00   0.00   1
          58   0.17   0.20   0.18   5
          59   0.00   0.00   0.00   1

          60   0.00   0.00   0.00   4

    accuracy   0.11   441
   macro avg   0.11   0.12   0.11   441
weighted avg   0.12   0.11   0.11   441

C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\linear_model_logistic.py:460: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

```
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2.Decision Tree

In [53]:

```
from sklearn.tree import DecisionTreeClassifier

# Create and train the Decision Tree model decision_tree_model =
DecisionTreeClassifier() decision_tree_model.fit(x_train, y_train)

# Make predictions y_pred_dt =
decision_tree_model.predict(x_test)

# Calculate performance metrics accuracy_dt =
accuracy_score(y_test, y_pred_dt) confusion_matrix_dt =
confusion_matrix(y_test, y_pred_dt) classification_report_dt =
classification_report(y_test, y_pred_dt)

# Print the metrics print("Decision
Tree Metrics:") print(f"Accuracy:
{accuracy_dt}") print("Confusion
Matrix:")
print(confusion_matrix_dt)
print("Classification Report:")
print(classification_report_dt)
```

Decision Tree Metrics:
Accuracy: 0.9909297052154195
Confusion Matrix:
[[2 0 0 ... 0 0 0]
 [0 3 0 ... 0 0 0] [0
0 3 ... 0 0 0]
 ...
 [0 0 0 ... 5 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 4 0]]
Classification Report:

	precision	recall	f1-score	support
18	1.00	1.00	1.00	2
19	1.00	1.00	1.00	3
20	1.00	1.00	1.00	3
21	1.00	1.00	1.00	2
22	1.00	1.00	1.00	4
23	1.00	1.00	1.00	3
24	1.00	1.00	1.00	9
25	1.00	1.00	1.00	12
26	1.00	1.00	1.00	9
27	1.00	1.00	1.00	16
28	1.00	1.00	1.00	11
29	1.00	1.00	1.00	19
30	1.00	1.00	1.00	14
31	1.00	1.00	1.00	27
32	1.00	1.00	1.00	18
33	1.00	1.00	1.00	15
34	1.00	1.00	1.00	30
35	1.00	1.00	1.00	23
36	1.00	1.00	1.00	21
37	1.00	1.00	1.00	13
38	1.00	1.00	1.00	19
39	1.00	1.00	1.00	9
40	1.00	1.00	1.00	14
41	1.00	1.00	1.00	11
42	1.00	1.00	1.00	12
43	1.00	1.00	1.00	14
44	1.00	1.00	1.00	9
45	1.00	1.00	1.00	10
46	1.00	1.00	1.00	9
47	1.00	1.00	1.00	8
48	1.00	1.00	1.00	5
49	1.00	1.00	1.00	6
50	1.00	1.00	1.00	13
51	1.00	1.00	1.00	5
52	1.00	1.00	1.00	6
53	1.00	1.00	1.00	7
54	1.00	1.00	1.00	4
55	1.00	1.00	1.00	10
56	1.00	1.00	1.00	5
57	1.00	1.00	1.00	1
58	1.00	1.00	1.00	5
59	0.20	1.00	0.33	1
				60
accuracy			0.99	441
macro avg	0.96	0.98	0.96	441
weighted avg	0.99	0.99	0.99	441

C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior. _warn_prf(average, modifier, msg_start, len(result))

3.Random Forest

In [58]: **from** sklearn.ensemble **import** RandomForestClassifier

```
# Create and train the Random Forest model random_forest_model
= RandomForestClassifier() random_forest_model.fit(x_train,
y_train)

# Make predictions y_pred_rf =
random_forest_model.predict(x_test) # Calculate
performance metrics accuracy_rf =
accuracy_score(y_test, y_pred_rf)
confusion_matrix_rf = confusion_matrix(y_test,
y_pred_rf) classification_report_rf =
classification_report(y_test, y_pred_rf)

# Print the metrics print("Random
Forest Metrics:") print(f"Accuracy:
{accuracy_rf}") print("Confusion
Matrix:")
print(confusion_matrix_rf)
print("Classification Report:")
print(classification_report_rf)
```

```
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
Random Forest Metrics: Accuracy:
0.6077097505668935
Confusion Matrix:
[[2 0 0 ... 0 0 0]
 [0 1 2 ... 0 0 0]  [0
0 3 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Classification Report:
          precision    recall  f1-score   support

18       1.00      1.00      1.00        2
19       1.00      0.33      0.50        3
20       0.60      1.00      0.75        3
21       0.50      1.00      0.67        2
22       0.33      0.25      0.29        4
23       0.00      0.00      0.00        3
24       0.33      0.33      0.33        9
25       0.62      0.42      0.50       12
26       0.56      0.56      0.56        9
27       0.69      0.56      0.62       16
28       0.80      0.73      0.76       11
29       0.68      0.79      0.73       19
30       0.65      0.93      0.76       14
31       0.90      1.00      0.95       27
32       1.00      0.89      0.94       18
33       1.00      0.93      0.97       15
34       0.94      1.00      0.97       30
35       0.82      1.00      0.90       23
36       0.84      1.00      0.91       21
37       0.65      0.85      0.73       13
38       0.64      0.84      0.73       19
39       0.44      0.44      0.44        9
40       0.43      0.71      0.54       14
41       0.33      0.27      0.30       11
42       0.29      0.50      0.36       12
43       0.20      0.07      0.11       14
44       0.00      0.00      0.00        9
45       0.27      0.40      0.32       10
46       0.27      0.44      0.33        9
47       0.00      0.00      0.00        8
48       0.00      0.00      0.00        5
49       0.20      0.17      0.18        6
50       0.50      0.46      0.48       13
51       0.20      0.20      0.20        5
52       0.00      0.00      0.00        6
53       1.00      0.29      0.44        7
54       0.00      0.00      0.00        4
55       0.25      0.10      0.14       10
56       0.00      0.00      0.00        5
57       0.00      0.00      0.00        1
58       0.00      0.00      0.00        5
59       0.00      0.00      0.00        1        60       0.00      0.00      0.00        4

   accuracy                          0.61       441    macro avg
0.44      0.45      0.43       441 weighted avg       0.57
0.61      0.58       441

C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rajes\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```