```
import pandas as pd
df = pd.read_csv('/content/SET3.csv')
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educatic
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

5 rows × 35 columns

```
df.select_dtypes(include='int').nunique()
```

```
Age                           43
DailyRate                    886
DistanceFromHome              29
Education                      5
EmployeeCount                  1
EmployeeNumber              1470
EnvironmentSatisfaction        4
HourlyRate                    71
JobInvolvement                 4
JobLevel                       5
JobSatisfaction                4
MonthlyIncome               1349
MonthlyRate                 1427
NumCompaniesWorked            10
PercentSalaryHike             15
PerformanceRating              2
RelationshipSatisfaction       4
StandardHours                  1
StockOptionLevel               4
TotalWorkingYears             40
TrainingTimesLastYear          7
WorkLifeBalance                4
YearsAtCompany                37
YearsInCurrentRole            19
YearsSinceLastPromotion       16
YearsWithCurrManager          18
dtype: int64
```

```
df.isnull().sum()
```

```
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
```

```
df.drop(columns = ['DailyRate','EmployeeNumber','EmployeeCount'], inplace= True)
df.head()
```

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	Educatic
0	41	Yes	Travel_Rarely	Sales	1	2	Life S
1	49	No	Travel_Frequently	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	Research & Development	2	2	
3	33	No	Travel_Frequently	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	Research & Development	2	1	

5 rows × 32 columns

```
import pandas as pd

# Assuming you already have a DataFrame named 'df' with integer columns
def check_outliers(df):
  int_cols = df.select_dtypes(include="int")
  outliers_info = pd.DataFrame(columns=["Column", "outlier vals","Outlier Count"])

  q1 = int_cols.quantile(0.25)
  q3 = int_cols.quantile(0.75)
  outlier_columns =[]

  iqr = q3 - q1
  upper_limit = q3 + (1.5 * iqr)
  lower_limit = q1 - (1.5 * iqr)
  print(lower_limit)

  for col in int_cols.columns:
      # Check for outliers in each column
      outlier_vals = ((df[col] < lower_limit[col]) | (df[col] > upper_limit[col]))
      outlier_count = ((df[col] < lower_limit[col]) | (df[col] > upper_limit[col])).sum()

      # If there are outliers, add the column and count to the DataFrame
      if outlier_count > 0:
          outlier_columns.append(col)
          outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_inde

  # Display DataFrame with columns containing outliers and their counts

  #print("columns with outliers = ",outlier_columns)
  return outlier_columns,outliers_info,lower_limit,upper_limit
```

```
outlier_columns,oultiers_df,lower_limit,upper_limit = check_outliers(df)
oultiers_df
outlier_columns
```

```
Age                            10.500
DistanceFromHome              -16.000
Education                      -1.000
EnvironmentSatisfaction        -1.000
HourlyRate                     -5.625
JobInvolvement                  0.500
JobLevel                       -2.000
JobSatisfaction                -1.000
MonthlyIncome               -5291.000
MonthlyRate                -10574.750
NumCompaniesWorked             -3.500
PercentSalaryHike               3.000
PerformanceRating               3.000
RelationshipSatisfaction       -1.000
StandardHours                  80.000
StockOptionLevel               -1.500
TotalWorkingYears              -7.500
TrainingTimesLastYear           0.500
WorkLifeBalance                 0.500
YearsAtCompany                 -6.000
YearsInCurrentRole             -5.500
```

```
YearsSinceLastPromotion        -4.500
YearsWithCurrManager           -5.500
dtype: float64
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
<ipython-input-5-eb1cefb2af8d>:25: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
  outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignore_index=Tr
['MonthlyIncome',
 'NumCompaniesWorked',
 'PerformanceRating',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

```
#Winsorization
import numpy as np

for col in outlier_columns:
    df[col] = np.where(df[col] <= lower_limit[col], lower_limit[col], df[col])
    df[col] = np.where(df[col] >= upper_limit[col], upper_limit[col], df[col])
outlier_columns,oultiers_df,lower_limit,upper_limit = check_outliers(df)
print(oultiers_df)
print(outlier_columns)
```

```
Age                           10.500
DistanceFromHome             -16.000
Education                     -1.000
EnvironmentSatisfaction       -1.000
HourlyRate                    -5.625
JobInvolvement                 0.500
JobLevel                      -2.000
JobSatisfaction               -1.000
MonthlyRate               -10574.750
PercentSalaryHike              3.000
RelationshipSatisfaction      -1.000
StandardHours                 80.000
WorkLifeBalance                0.500
dtype: float64
Empty DataFrame
Columns: [Column, outlier vals, Outlier Count]
Index: []
[]
```

```
dfn = pd.get_dummies(df,columns=['BusinessTravel','Department','EducationField',"Gender",'JobRole','MaritalStatus','Over18','OverTime']
dfn.head()
```

	Age	Attrition	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate
0	41	Yes	1	2	2	94
1	49	No	8	1	3	61
2	37	Yes	2	2	4	92
3	33	No	3	4	4	56
4	27	No	2	1	1	40

5 rows × 45 columns

```
#Label Encoding
from sklearn.preprocessing import LabelEncoder

object_cols= ['Attrition']

label_encoder = LabelEncoder()

for col in object_cols:
    dfn[col]= label_encoder.fit_transform(dfn[col])

dfn.head(10)
```

	Age	Attrition	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate
0	41	1	1	2	2	94
1	49	0	8	1	3	61
2	37	1	2	2	4	92
3	33	0	3	4	4	56
4	27	0	2	1	1	40
5	32	0	2	2	4	79
6	59	0	3	3	3	81
7	30	0	24	1	4	67
8	38	0	23	3	4	44
9	36	0	27	3	3	94

10 rows × 45 columns

```
dfn['Attrition'].value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
x = dfn.drop('Attrition', axis=1)
y = dfn['Attrition']

from sklearn.model_selection import train_test_split
x_tr, x_te, y_tr, y_te = train_test_split(x, y, test_size=0.2, stratify=y, random_state=0)
x_tr
```

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvol
237	52	2	4	1	79	
549	34	8	2	2	96	
947	52	5	3	2	64	
1340	36	10	4	2	63	
1273	22	8	1	3	79	
...	...	...	...	...	...	
443	22	4	1	3	99	
449	39	8	1	3	48	
582	40	2	2	3	38	
506	37	3	3	3	36	
813	39	2	3	1	84	

1176 rows × 44 columns

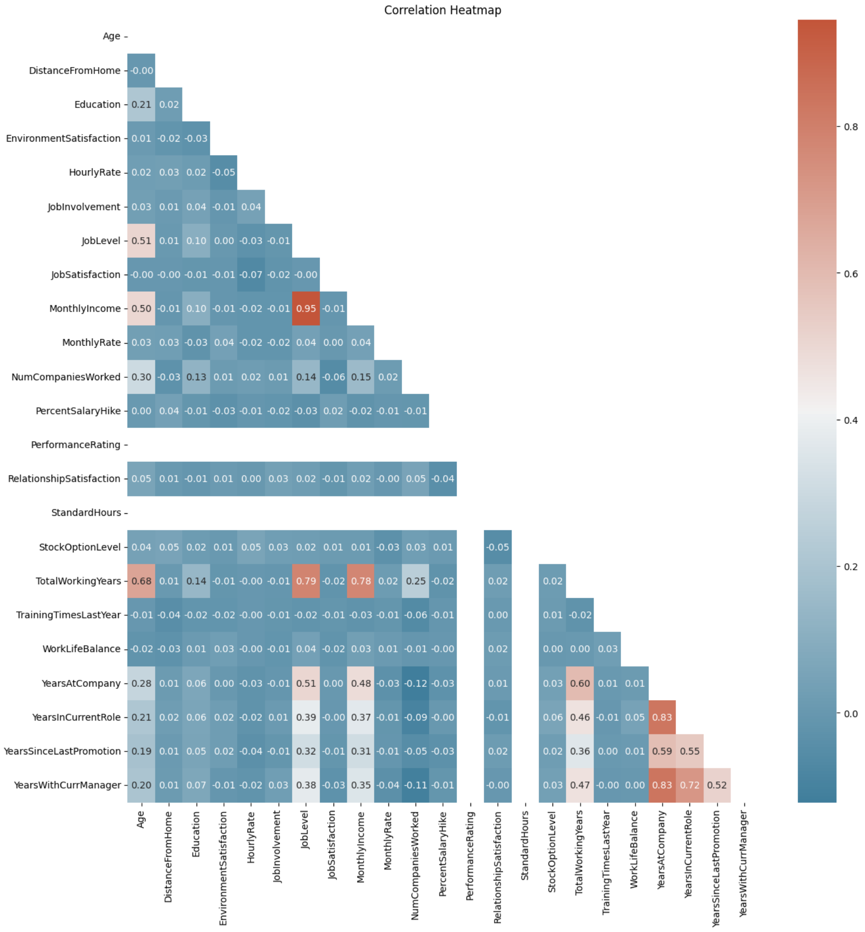
```
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE

# Assuming you have a DataFrame 'dfn' with your dataset
x = dfn.drop('Attrition', axis=1)  # Features
y = dfn['Attrition']  # Target variable
# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the features using the scaler
x_scaled = scaler.fit_transform(x)
df_scaled = pd.DataFrame(x_scaled, columns=x.columns)

# Now, 'df_scaled' is a DataFrame with the scaled features

df_scaled
```

	Age	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	Job
0	0.547619	0.000000	0.25	0.333333	0.914286	
1	0.738095	0.250000	0.00	0.666667	0.442857	
2	0.452381	0.035714	0.25	1.000000	0.885714	
3	0.357143	0.071429	0.75	1.000000	0.371429	
4	0.214286	0.035714	0.00	0.000000	0.142857	
...	...	...	...	...	...	
1465	0.428571	0.785714	0.25	0.666667	0.157143	
1466	0.500000	0.178571	0.00	1.000000	0.171429	
1467	0.214286	0.107143	0.50	0.333333	0.814286	
1468	0.738095	0.035714	0.50	1.000000	0.471429	
1469	0.380952	0.250000	0.50	0.333333	0.742857	

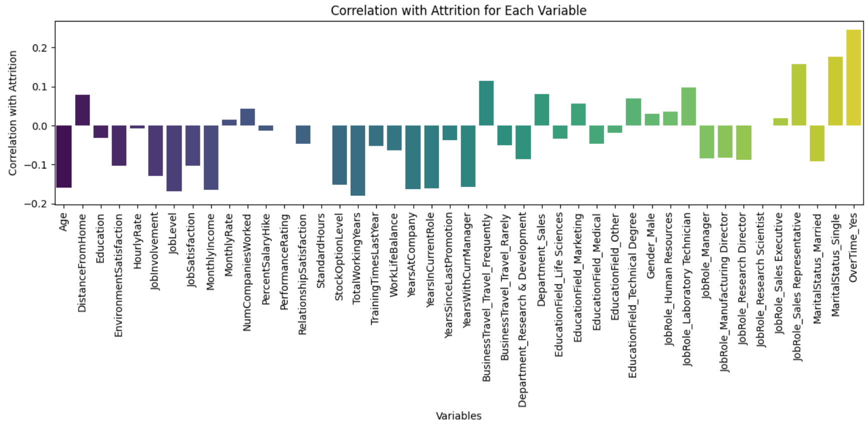
1470 rows × 44 columns

```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
corr = df.corr()

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Configure a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Create a heatmap with annotations formatted to 2 decimal places
plt.figure(figsize=(15, 15))
sns.heatmap(corr, annot=True, fmt=".2f", mask=mask, cmap=cmap)

# Add a title
plt.title("Correlation Heatmap")

# Display the heatmap
plt.show()
```

```
<ipython-input-13-56035030432c>:5: FutureWarning: The default value of numeric_only i
  corr = df.corr()
```



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

# Assuming you have a DataFrame 'df' with your data

# Calculate the correlation between each variable and 'Attrition'
correlation_with_attrition = dfn.corr()['Attrition'].drop('Attrition')

# Create a bar plot to visualize the correlations
plt.figure(figsize=(12, 6))
sns.barplot(x=correlation_with_attrition.index, y=correlation_with_attrition.values, palette='viridis')
plt.xticks(rotation=90)
plt.xlabel('Variables')
plt.ylabel('Correlation with Attrition')
plt.title('Correlation with Attrition for Each Variable')
plt.tight_layout()
plt.show()
```



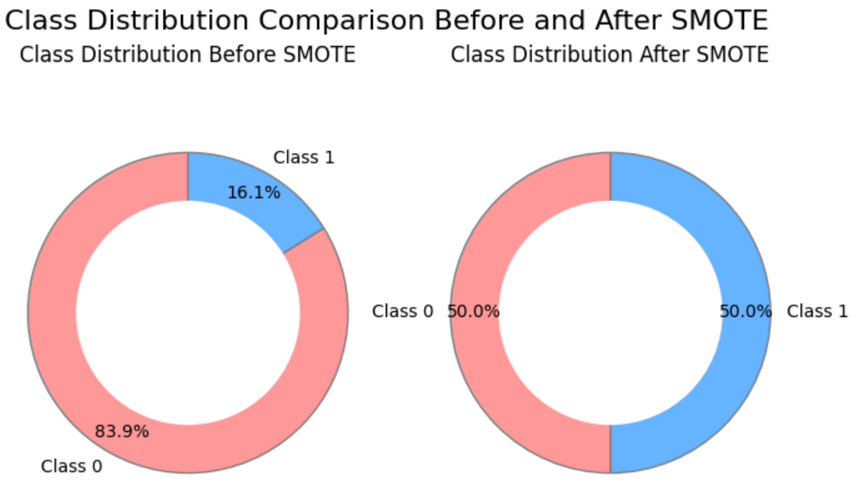
```
dfn['Attrition'].value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
# Initialize SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)

# Apply SMOTE to generate synthetic samples
x_new, y_new = smote.fit_resample(x_scaled, y)

# Now, X_resampled and y_resampled contain the balanced dataset
y_new.value_counts()
```

```
1    1233
0    1233
Name: Attrition, dtype: int64
```

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

# Sample data representing class counts before and after SMOTE
class_counts_before = pd.Series([1233, 237], index=["Class 0", "Class 1"], name="Before SMOTE")
class_counts_after = pd.Series([1233, 1233], index=["Class 0", "Class 1"], name="After SMOTE")

# Create a figure with two subplots (donut charts)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 5))

# Define colors for the donut charts
colors = ['#ff9999', '#66b3ff']

# Plot the donut chart before SMOTE
ax1.pie(class_counts_before, labels=class_counts_before.index, autopct='%1.1f%%', startangle=90, colors=colors,
        wedgeprops={'edgecolor': 'gray'}, pctdistance=0.85)
# Draw a circle in the center to make it a donut chart
centre_circle = plt.Circle((0,0),0.70,fc='white')
ax1.add_artist(centre_circle)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
ax1.set_title("Class Distribution Before SMOTE")

# Plot the donut chart after SMOTE
ax2.pie(class_counts_after, labels=class_counts_after.index, autopct='%1.1f%%', startangle=90, colors=colors,
        wedgeprops={'edgecolor': 'gray'}, pctdistance=0.85)
# Draw a circle in the center to make it a donut chart
centre_circle = plt.Circle((0,0),0.70,fc='white')
ax2.add_artist(centre_circle)
ax2.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
ax2.set_title("Class Distribution After SMOTE")

# Add a common title for both subplots
plt.suptitle("Class Distribution Comparison Before and After SMOTE", fontsize=16)

# Display the donut charts
plt.show()
```



```
import numpy as np

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
logmodel= LogisticRegression()

param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
    'max_iter' : [100, 1000,2500, 5000]
    }
]
from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(logmodel, param_grid = param_grid, cv = 3, verbose=True, n_jobs=-1)
best_clf = clf.fit(x_new, y_new)
best_clf.best_estimator_
```

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validatio
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    raise ValueError(
ValueError: Only 'saga' solver supports elasticnet penalty, got solver=liblinear.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validatio
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got elasticnet pena

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validatio
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    fold_coefs_ = Parallel(n_jobs=self.n_jobs, verbose=self.verbose, prefer=prefer
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/parallel.py", line 6
    return super().__call__(iterable_with_config)
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 1863, in
    return output if self.return_generator else list(output)
  File "/usr/local/lib/python3.10/dist-packages/joblib/parallel.py", line 1792, in
    res = func(*args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/parallel.py", line 1
    return self.function(*args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    alpha = (1.0 / C) * (1 - l1_ratio)
TypeError: unsupported operand type(s) for -: 'int' and 'NoneType'

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validatio
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py'
    raise ValueError("penalty='none' is not supported for the liblinear solver")
ValueError: penalty='none' is not supported for the liblinear solver

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: Us
  warnings.warn(
```

```
▾            LogisticRegression
LogisticRegression(C=1.623776739188721, solver='liblinear')
```

```
best_clf.score(x_new, y_new)
```

```
0.8077858880778589
```

```
#logistic_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
X_train, X_test, y_train, y_test = train_test_split(x_new, y_new, test_size=0.20, random_state=101)
logistic_model =LogisticRegression(C=1.623776739188721, solver='liblinear')
logistic_model.fit(X_train,y_train)
y_pred = logistic_model.predict(X_test)

print(f'Train Accuracy - : {logistic_model.score(X_train,y_train):.3f}')
print (f'Test Accuracy - : {logistic_model.score(X_test,y_test):.3f}')

# Calculate precision, recall, and F1 score for both training and testing sets
acc = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f'Model Accuracy - : {acc}')
print(f'Model precision - : {precision}')
print(f'Model recall - : {recall}')
print(f'Model f1 - : {f1}')
```

```
Train Accuracy - : 0.812
Test Accuracy - : 0.773
Model Accuracy - : 0.7732793522267206
Model precision - : 0.7530364372469636
Model recall - : 0.7848101265822784
Model f1 - : 0.768595041322314
```

```
#Using max_depth, criterion will suffice for DT Models, rest all will remain constant
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
param_grid2 = [
    {'max_depth' : [3,5,7,9,10,15,20,25],
    'criterion' : ['gini', 'entropy'],
    'max_features' : ['auto', 'sqrt', 'log2'],
    'min_samples_split' : [2,4,6]
    }
]
dt= DecisionTreeClassifier()

clf = GridSearchCV(dt, param_grid = param_grid2, cv = 3, verbose=True, n_jobs=-1)
best_clf = clf.fit(x_new, y_new)
best_clf.best_estimator_
```

```
Fitting 3 folds for each of 144 candidates, totalling 432 fits
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning:
  warnings.warn(
▾                       DecisionTreeClassifier
DecisionTreeClassifier(max_depth=15, max_features='auto', min_samples_split=4)
```

```
best_clf.score(x_new, y_new)
```

```
0.9708029197080292
```

```
dt_model =DecisionTreeClassifier(criterion='entropy', max_depth=20, max_features='auto',
                       min_samples_split=6)
dt_model.fit(X_train,y_train)
y_pred = dt_model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f'Model Accuracy - : {acc}')
print(f'Model precision - : {precision}')
print(f'Model recall - : {recall}')
print(f'Model f1 - : {f1}')
print(f'Train Accuracy - : {dt_model.score(X_train,y_train):.3f}')
print (f'Test Accuracy - : {dt_model.score(X_test,y_test):.3f}')
```

```
Model Accuracy - : 0.8319838056680162
Model precision - : 0.8407079646017699
Model recall - : 0.8016877637130801
Model f1 - : 0.8207343412526997
Train Accuracy - : 0.973
Test Accuracy - : 0.832
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
  warnings.warn(
```

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import graphviz
dot_data = export_graphviz(dt_model,
                      out_file=None,
                      feature_names=x_new.columns,       #Provide X Variables Column Names
                      class_names=['Yes','No'],      # Provide Target Variable Column Name
                      filled=True, rounded=True,     # Controls the look of the nodes and colours it
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-33-c42b90b929a3> in <cell line: 3>()
      3 dot_data = export_graphviz(dt_model,
      4                       out_file=None,
----> 5                       feature_names=x_new.columns,       #Provide X
Variables Column Names
      6                       class_names=['Yes','No'],      # Provide Target
Variable Column Name
      7                       filled=True, rounded=True,     # Controls the look of
the nodes and colours it

AttributeError: 'numpy.ndarray' object has no attribute 'columns'
```