```
import numpy as np #linear algebra
import pandas as pd #data processing,

#data visualization
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# 1. Data Preparation

df= pd.read_excel("Employee_Turnover_Analytics.xlsx")

df.head()

   satisfaction_level  last_evaluation  number_project
average_montly_hours  \
0                0.38             0.53               2
157
1                0.80             0.86               5
262
2                0.11             0.88               7
272
3                0.72             0.87               5
223
4                0.37             0.52               2
159

   time_spend_company  Work_accident  left  promotion_last_5years
sales  \
0                   3              0     1                      0
sales
1                   6              0     1                      0
sales
2                   4              0     1                      0
sales
3                   5              0     1                      0
sales
4                   3              0     1                      0
sales

   salary
0     low
1  medium
2  medium
3     low
4     low

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64  
 3   average_montly_hours   14999 non-null  int64  
 4   time_spend_company     14999 non-null  int64  
 5   Work_accident          14999 non-null  int64  
 6   left                   14999 non-null  int64  
 7   promotion_last_5years  14999 non-null  int64  
 8   sales                  14999 non-null  object 
 9   salary                 14999 non-null  object 
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
# Dataframe Shape
df.shape
```

```
(14999, 10)
```

```
# Checking for Missing Values

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

```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
sales                    0
salary                   0
dtype: int64
```

```
# Display summary statistics
print(df.describe())
```

```
       satisfaction_level  last_evaluation  number_project  \
count        14999.000000     14999.000000    14999.000000   
mean             0.612834         0.716102        3.803054   
std              0.248631         0.171169        1.232592   
min              0.090000         0.360000        2.000000   
25%              0.440000         0.560000        3.000000   
50%              0.640000         0.720000        4.000000   
75%              0.820000         0.870000        5.000000   
max              1.000000         1.000000        7.000000   
```

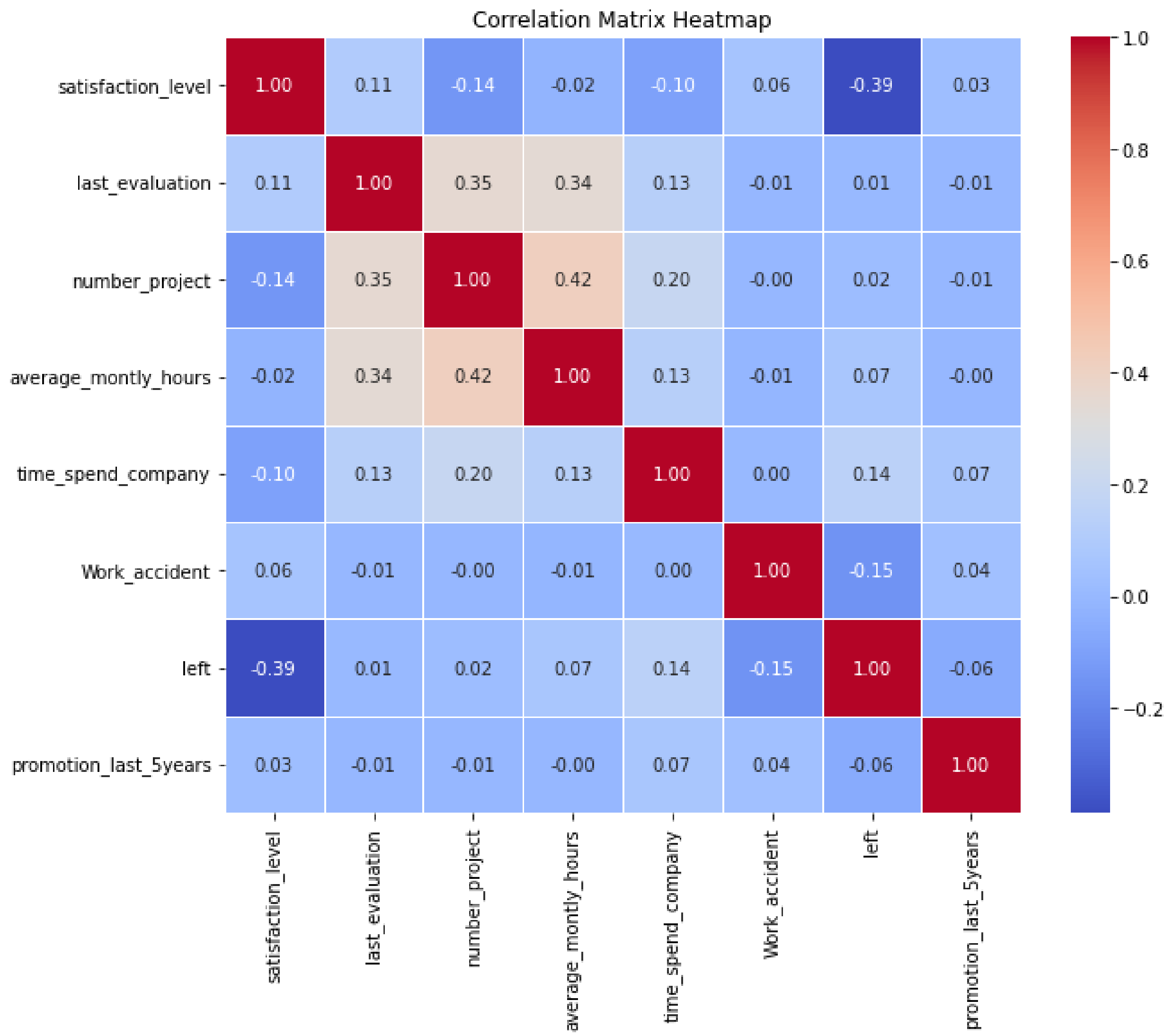
```
       average_montly_hours  time_spend_company  Work_accident
left  \
count          14999.000000        14999.000000   14999.000000
14999.000000
mean             201.050337            3.498233       0.144610
0.238083
std               49.943099            1.460136       0.351719
0.425924
min               96.000000            2.000000       0.000000
0.000000
25%              156.000000            3.000000       0.000000
0.000000
50%              200.000000            3.000000       0.000000
0.000000
75%              245.000000            4.000000       0.000000
0.000000
max              310.000000           10.000000       1.000000
1.000000

       promotion_last_5years
count           14999.000000
mean                0.021268
std                 0.144281
min                 0.000000
25%                 0.000000
50%                 0.000000
75%                 0.000000
max                 1.000000
```

```
df.columns
```

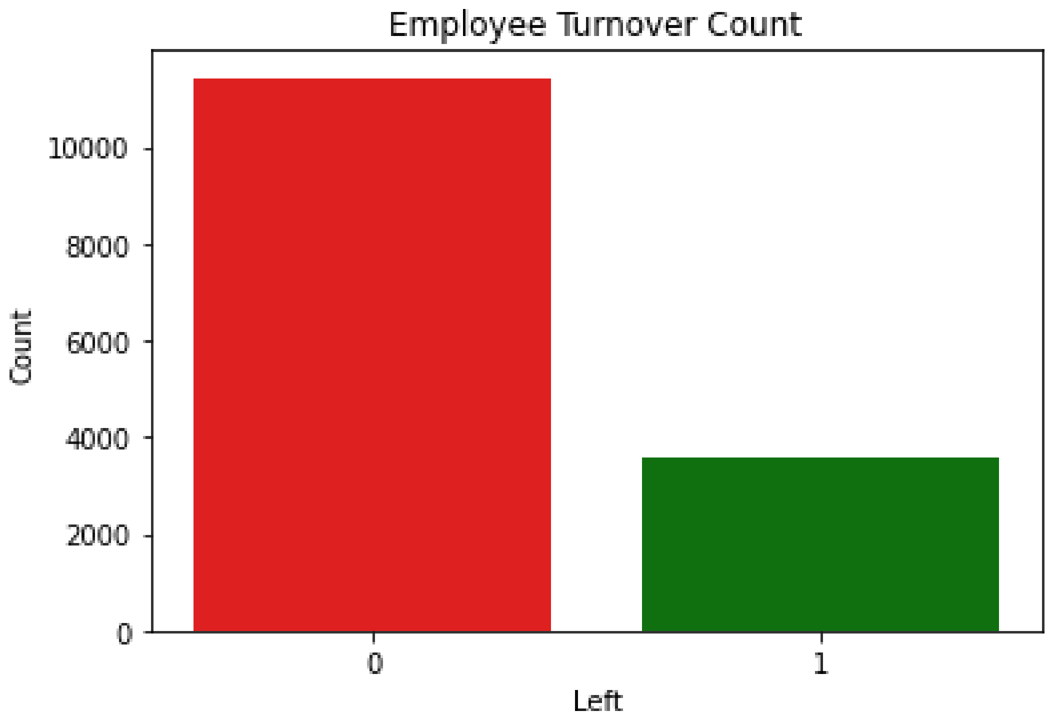
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
'left',
       'promotion_last_5years', 'sales', 'salary'],
      dtype='object')
```

```
# 2.1 Draw a heatmap of the Correlation Matrix between all numerical
features/columns in the data.

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

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Draw the heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f", linewidths=.5)
```

```
# Set the title
plt.title('Correlation Matrix Heatmap')

# Show the plot
plt.show()
```

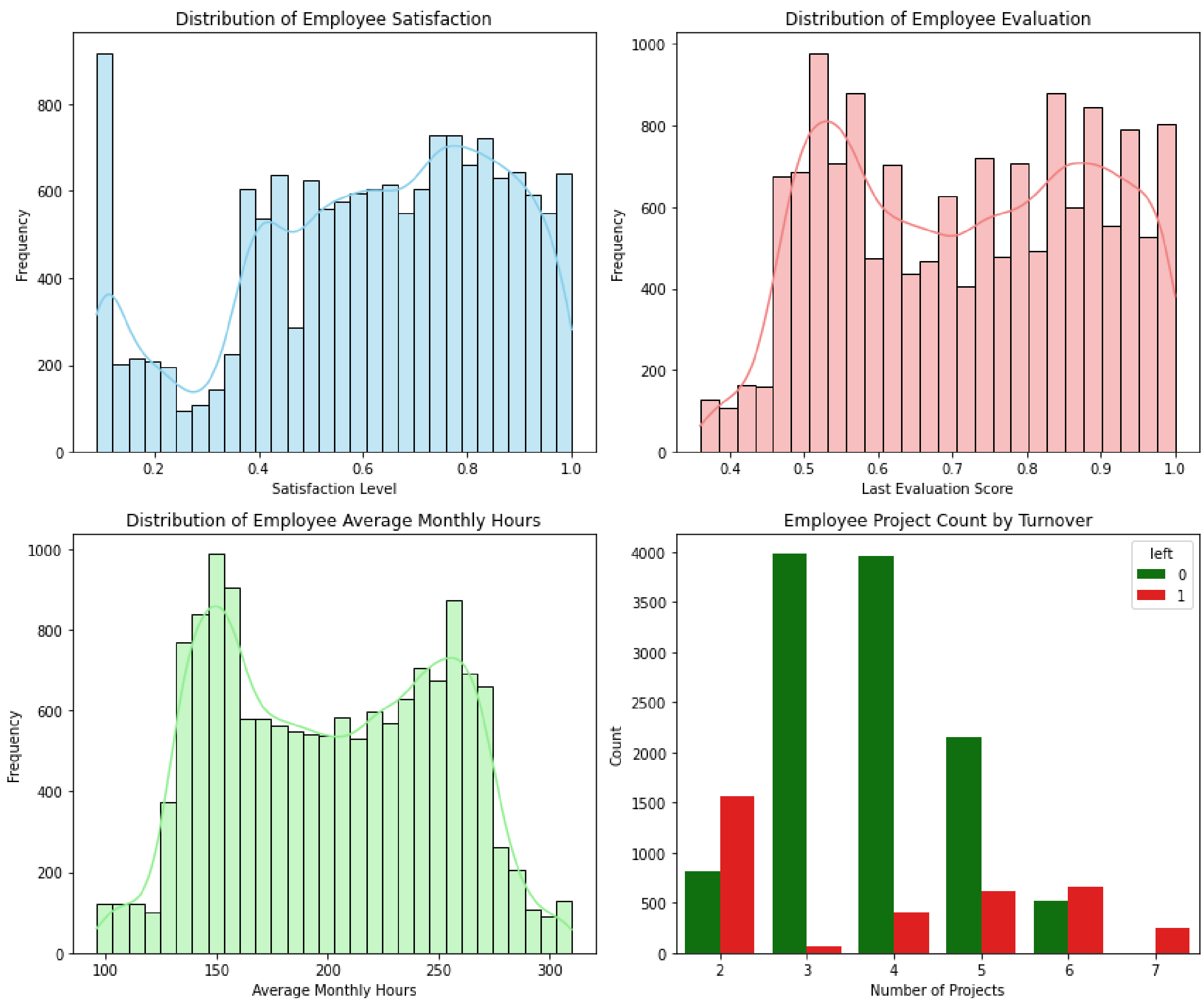



```
# Plotting the count of 'left' values using a bar chart
sns.countplot(x='left', data=df, palette=['red', 'green'])
plt.title('Employee Turnover Count')
plt.xlabel('Left')
plt.ylabel('Count')
plt.show()
```



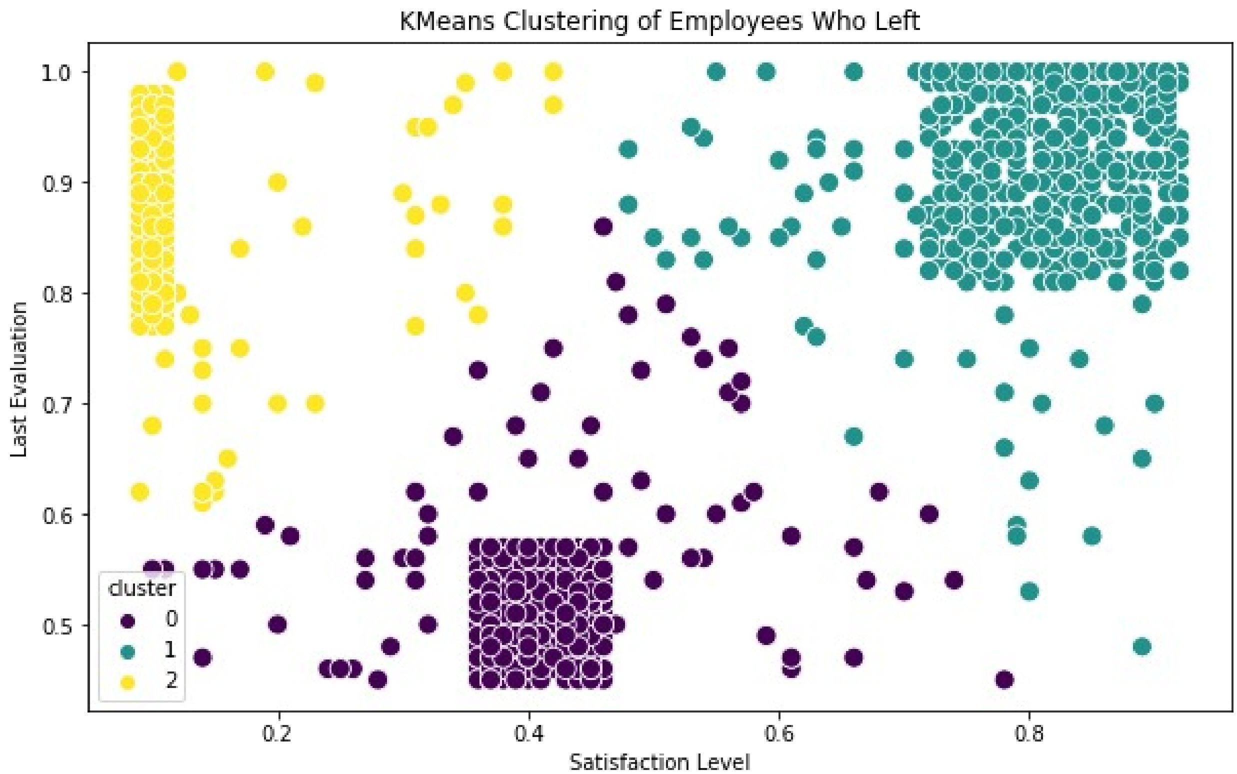

```
2.2 # Draw the distribution plot of
#Employee Satisfaction (use column satisfaction_level)
#Employee Evaluation (use column last_evaluation)
#Employee Average Monthly Hours (use column average_montly_hours)

# Set up the subplots
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Distribution plot for Employee Satisfaction
sns.histplot(df['satisfaction_level'], kde=True, color='skyblue',
ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Employee Satisfaction')
axes[0, 0].set_xlabel('Satisfaction Level')
axes[0, 0].set_ylabel('Frequency')

# Distribution plot for Employee Evaluation
sns.histplot(df['last_evaluation'], kde=True, color='lightcoral',
ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Employee Evaluation')
axes[0, 1].set_xlabel('Last Evaluation Score')
axes[0, 1].set_ylabel('Frequency')

# Distribution plot for Employee Average Monthly Hours
sns.histplot(df['average_montly_hours'], kde=True, color='lightgreen',
ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Employee Average Monthly Hours')
axes[1, 0].set_xlabel('Average Monthly Hours')
```

```
axes[1, 0].set_ylabel('Frequency')

# Bar plot for Employee Project Count
sns.countplot(x='number_project', hue='left', data=df,
palette=['green', 'red'], ax=axes[1, 1])
axes[1, 1].set_title('Employee Project Count by Turnover')
axes[1, 1].set_xlabel('Number of Projects')
axes[1, 1].set_ylabel('Count')

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```




```
#3. Perform clustering of Employees who left based on their
satisfaction and evaluation.
#3.1 Choose columns satisfaction_level, last_evaluation and left.
```

```python
#3.2 Do KMeans clustering of employees who left the company into 3
clusters.

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Select relevant columns for clustering
left_employees = df[df['left'] == 1][['satisfaction_level',
'last_evaluation']]

# Perform KMeans clustering with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
left_employees['cluster'] = kmeans.fit_predict(left_employees)

# Get the count of employees in each cluster
cluster_counts = left_employees['cluster'].value_counts()

# Visualize the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x='satisfaction_level', y='last_evaluation',
hue='cluster', data=left_employees, palette='viridis', s=100)
plt.title('KMeans Clustering of Employees Who Left')
plt.xlabel('Satisfaction Level')
plt.ylabel('Last Evaluation')
plt.show()

# Analyze the clusters
cluster_centers = kmeans.cluster_centers_
print("Cluster Centers:")
print(cluster_centers)

# Display the number of employees in each cluster
print("\nNumber of Employees in Each Cluster:")
print(cluster_counts)

# Give thoughts on the clusters
for i in range(len(cluster_centers)):
    print(f"\nCluster {i + 1}:")
    print(f"Average Satisfaction Level: {cluster_centers[i][0]:.2f}")
    print(f"Average Last Evaluation: {cluster_centers[i][1]:.2f}")
```




```
Cluster Centers:
[[0.41014545 0.51698182]
 [0.80851586 0.91170931]
 [0.11115466 0.86930085]]

Number of Employees in Each Cluster:
0    1650
1     977
2     944
Name: cluster, dtype: int64

Cluster 1:
Average Satisfaction Level: 0.41
Average Last Evaluation: 0.52

Cluster 2:
Average Satisfaction Level: 0.81
Average Last Evaluation: 0.91

Cluster 3:
Average Satisfaction Level: 0.11
Average Last Evaluation: 0.87

# 3.3 Based on the satisfaction and evaluation factors, giving
thoughts on the employee clusters.
 - Based on evaluating and observing 3 clusters, satisfaction level of
```

```
employees has dropped from last evaluation in all clusters.
 - Most Concerning cluster is yellow cluster as employees seem to be
performing well despite very low satisfaction.
 - Retention strategies should focus on identifying and addressing the
root causes of dissatisfaction, understanding their needs,
   and creating a more positive work environment.

#4.Handle the left Class Imbalance using SMOTE technique.
4.1 Pre-Process the data by converting categorical columns to
numerical columns by:
    Separating categorical variables and numeric variables.
    Applying get_dummies() to the categorical variables.
    combining categorical variables and numeric variables.

from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

# Separate categorical and numeric variables
categorical_columns = df.select_dtypes(include='object').columns
numeric_columns = df.select_dtypes(include=['int64',
'float64']).columns

categorical_columns

Index(['sales', 'salary'], dtype='object')

numeric_columns

Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
'left',
       'promotion_last_5years'],
      dtype='object')

# Convert categorical variables to numerical using get_dummies
df_categorical = pd.get_dummies(df[categorical_columns],
drop_first=True)

df_categorical

       sales_RandD  sales_accounting  sales_hr  sales_management  \
0                0                 0         0                 0
1                0                 0         0                 0
2                0                 0         0                 0
3                0                 0         0                 0
4                0                 0         0                 0
...            ...               ...       ...               ...
14994            0                 0         0                 0
14995            0                 0         0                 0
14996            0                 0         0                 0
14997            0                 0         0                 0
```

```
14998            0                  0         0                  0

       sales_marketing  sales_product_mng  sales_sales  sales_support
\
0                    0                  0            1              0

1                    0                  0            1              0

2                    0                  0            1              0

3                    0                  0            1              0

4                    0                  0            1              0

...                ...                ...          ...            ...

14994                0                  0            0              1

14995                0                  0            0              1

14996                0                  0            0              1

14997                0                  0            0              1

14998                0                  0            0              1


       sales_technical  salary_low  salary_medium
0                    0           1              0
1                    0           0              1
2                    0           0              1
3                    0           1              0
4                    0           1              0
...                ...         ...            ...
14994                0           1              0
14995                0           1              0
14996                0           1              0
14997                0           1              0
14998                0           1              0

[14999 rows x 11 columns]
```

```
# Combine categorical and numeric variables
df_processed = pd.concat([df[numeric_columns], df_categorical],
axis=1)

df_processed
```

```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
```

```
2                      0.11             0.88               7
3                      0.72             0.87               5
4                      0.37             0.52               2
...                     ...              ...             ...
14994                  0.40             0.57               2
14995                  0.37             0.48               2
14996                  0.37             0.53               2
14997                  0.11             0.96               6
14998                  0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident  
left  \
0                       157                   3              0     1

1                       262                   6              0     1

2                       272                   4              0     1

3                       223                   5              0     1

4                       159                   3              0     1

...                     ...                 ...            ...   ...

14994                   151                   3              0     1

14995                   160                   3              0     1

14996                   143                   3              0     1

14997                   280                   4              0     1

14998                   158                   3              0     1


       promotion_last_5years  sales_RandD  sales_accounting  sales_hr  
\
0                          0            0                 0         0

1                          0            0                 0         0

2                          0            0                 0         0

3                          0            0                 0         0

4                          0            0                 0         0

...                      ...          ...               ...       ...

14994                      0            0                 0         0
```

```
14995                           0             0                   0          0

14996                           0             0                   0          0

14997                           0             0                   0          0

14998                           0             0                   0          0


       sales_management  sales_marketing  sales_product_mng
sales_sales  \
0                     0                0                  0
1
1                     0                0                  0
1
2                     0                0                  0
1
3                     0                0                  0
1
4                     0                0                  0
1
...                 ...              ...                ...            .
..
14994                 0                0                  0
0
14995                 0                0                  0
0
14996                 0                0                  0
0
14997                 0                0                  0
0
14998                 0                0                  0
0

       sales_support  sales_technical  salary_low  salary_medium
0                  0                0           1              0
1                  0                0           0              1
2                  0                0           0              1
3                  0                0           1              0
4                  0                0           1              0
...              ...              ...         ...            ...
14994              1                0           1              0
14995              1                0           1              0
14996              1                0           1              0
14997              1                0           1              0
14998              1                0           1              0

[14999 rows x 19 columns]
```

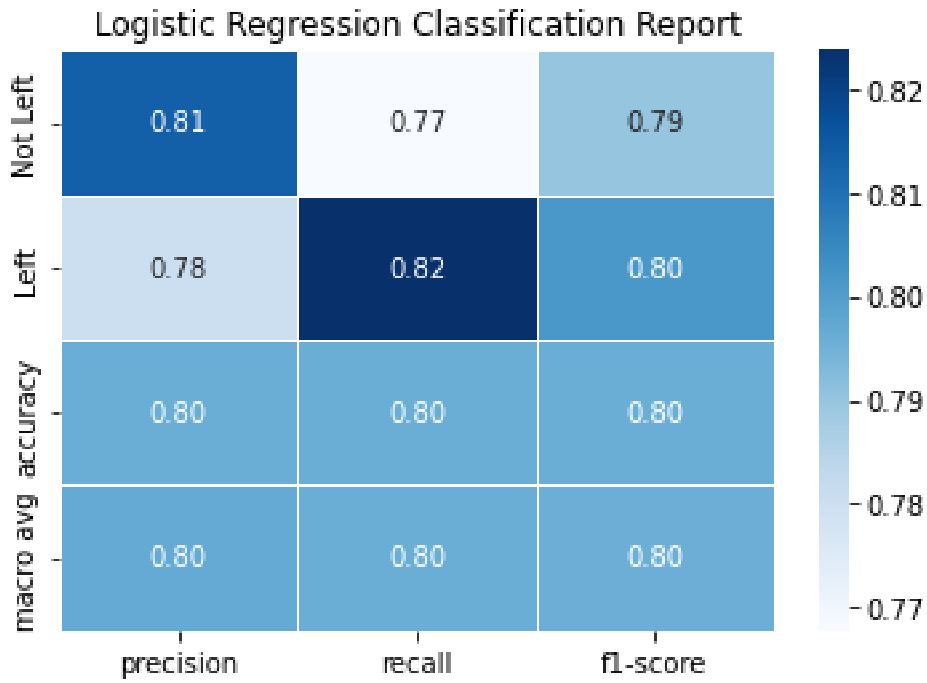
```python
# Define features (X) and target variable (y)
X = df_processed.drop('left', axis=1)
y = df['left']

X
```

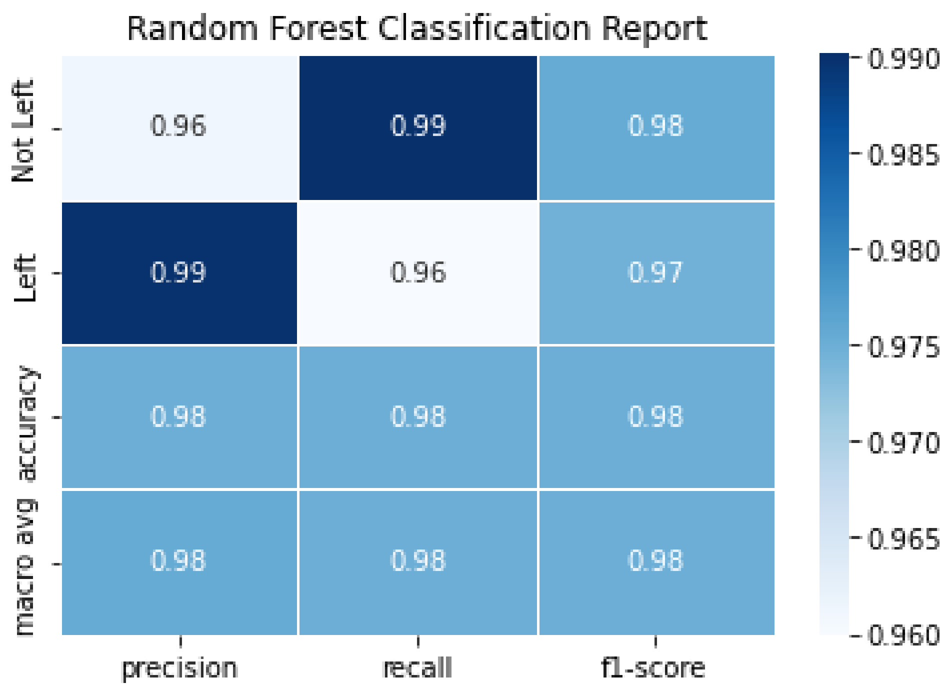
```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
1                    0.80             0.86               5
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
...                   ...              ...             ...
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2

       average_montly_hours  time_spend_company  Work_accident  \
0                       157                   3              0
1                       262                   6              0
2                       272                   4              0
3                       223                   5              0
4                       159                   3              0
...                     ...                 ...            ...
14994                   151                   3              0
14995                   160                   3              0
14996                   143                   3              0
14997                   280                   4              0
14998                   158                   3              0

       promotion_last_5years  sales_RandD  sales_accounting  sales_hr
\
0                          0            0                 0         0

1                          0            0                 0         0

2                          0            0                 0         0

3                          0            0                 0         0

4                          0            0                 0         0

...                      ...          ...               ...       ...

14994                      0            0                 0         0

14995                      0            0                 0         0

14996                      0            0                 0         0
```

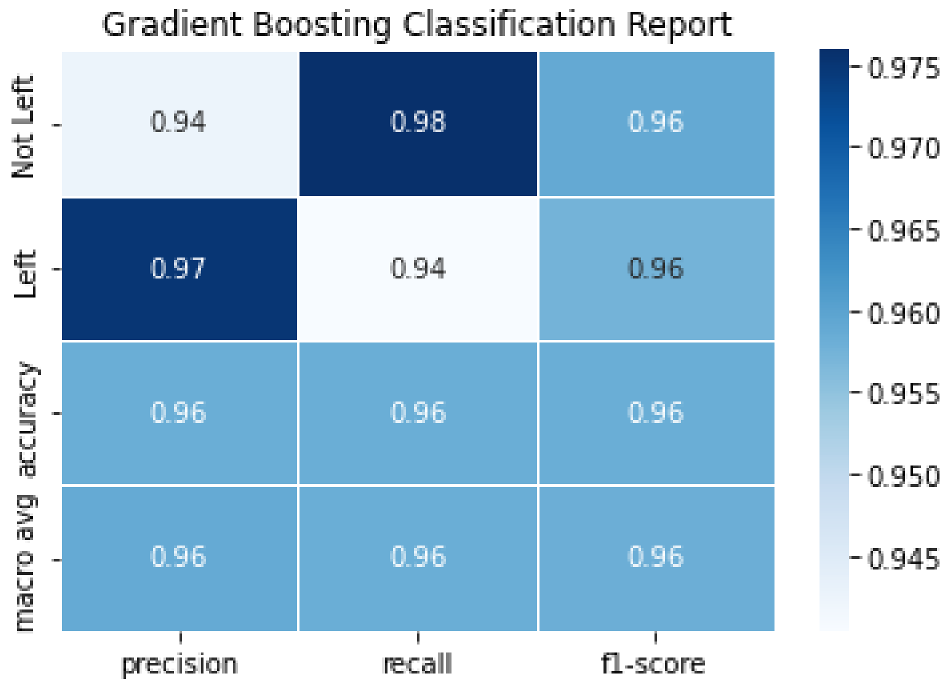
```
14997                          0             0                   0          0

14998                          0             0                   0          0


       sales_management  sales_marketing  sales_product_mng  
sales_sales  \
0                     0                0                  0  
1  
1                     0                0                  0  
1  
2                     0                0                  0  
1  
3                     0                0                  0  
1  
4                     0                0                  0  
1  
...                 ...              ...                ...           .
..  
14994                 0                0                  0  
0  
14995                 0                0                  0  
0  
14996                 0                0                  0  
0  
14997                 0                0                  0  
0  
14998                 0                0                  0  
0  

       sales_support  sales_technical  salary_low  salary_medium  
0                  0                0           1              0  
1                  0                0           0              1  
2                  0                0           0              1  
3                  0                0           1              0  
4                  0                0           1              0  
...              ...              ...         ...            ...  
14994              1                0           1              0  
14995              1                0           1              0  
14996              1                0           1              0  
14997              1                0           1              0  
14998              1                0           1              0  

[14999 rows x 18 columns]

y

0        1
1        1
```

```
2        1
3        1
4        1
        ..
14994    1
14995    1
14996    1
14997    1
14998    1
Name: left, Length: 14999, dtype: int64

# 4.2 Stratified split of the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=123, stratify=y)

# 4.3 Upsample the train dataset using SMOTE
smote = SMOTE(random_state=123)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)

from imblearn.over_sampling import SMOTE
import pandas as pd
from sklearn.model_selection import train_test_split

# Assuming you have your features and target variable ready (X_train,
y_train)
# If not, replace X_train and y_train with your actual variable names

# Split the original training data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.2, random_state=123)

# Initialize SMOTE
smote = SMOTE(random_state=123)

# Fit and transform the training data using SMOTE
X_train_resampled, y_train_resampled = smote.fit_resample(X_train,
y_train)


#5. Perform 5-Fold cross-validation model training and evaluate
performance.

#5.1 Train a Logistic Regression model and apply a 5-Fold CV and plot
the classification report.
#5.2 Train a Random Forest Classifier model and apply the 5-Fold CV
and plot the classification report.
#5.3 Train a  Gradient Boosting Classifier model and apply the 5-Fold
CV and plot the classification report.

from sklearn.model_selection import cross_val_predict
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Function to plot the classification report
def plot_classification_report(y_true, y_pred, title):
    report_dict = classification_report(y_true, y_pred,
target_names=['Not Left', 'Left'], output_dict=True)
    df_report = pd.DataFrame(report_dict).transpose()

    plt.figure(figsize=(6, 4))
    sns.heatmap(df_report.iloc[:-1, :-1], annot=True, cmap='Blues',
fmt='.2f', linewidths=.5)
    plt.title(title)
    plt.show()

# Logistic Regression
logistic_model = LogisticRegression(random_state=123)
logistic_predictions = cross_val_predict(logistic_model,
X_train_resampled, y_train_resampled, cv=5)
plot_classification_report(y_train_resampled, logistic_predictions,
'Logistic Regression Classification Report')

# Random Forest Classifier
rf_model = RandomForestClassifier(random_state=123)
rf_predictions = cross_val_predict(rf_model, X_train_resampled,
y_train_resampled, cv=5)
plot_classification_report(y_train_resampled, rf_predictions, 'Random
Forest Classification Report')

# Gradient Boosting Classifier
gb_model = GradientBoostingClassifier(random_state=123)
gb_predictions = cross_val_predict(gb_model, X_train_resampled,
y_train_resampled, cv=5)
plot_classification_report(y_train_resampled, gb_predictions,
'Gradient Boosting Classification Report')
```


Logistic Regression Classification Report
Not Left
Left
accuracy
macro avg
precision
recall
f1-score
0.81
0.77
0.79
0.78
0.82
0.80
0.80
0.80
0.80
0.80
0.80
0.80
0.82
0.81
0.80
0.79
0.78
0.77

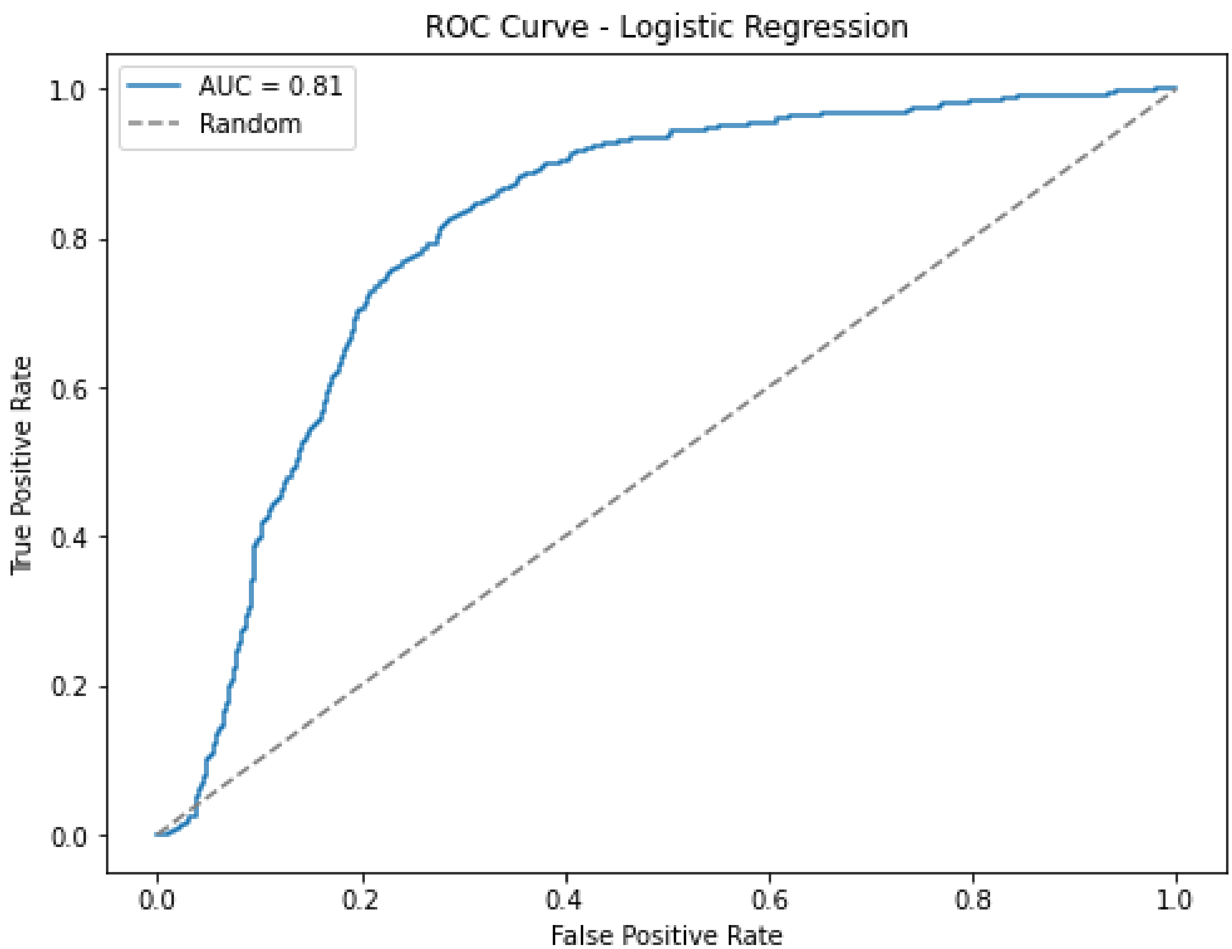

Random Forest Classification Report
Not Left
Left
accuracy
macro avg
precision
recall
f1-score
0.96
0.99
0.98
0.99
0.96
0.97
0.98
0.98
0.98
0.98
0.98
0.98
0.990
0.985
0.980
0.975
0.970
0.965
0.960



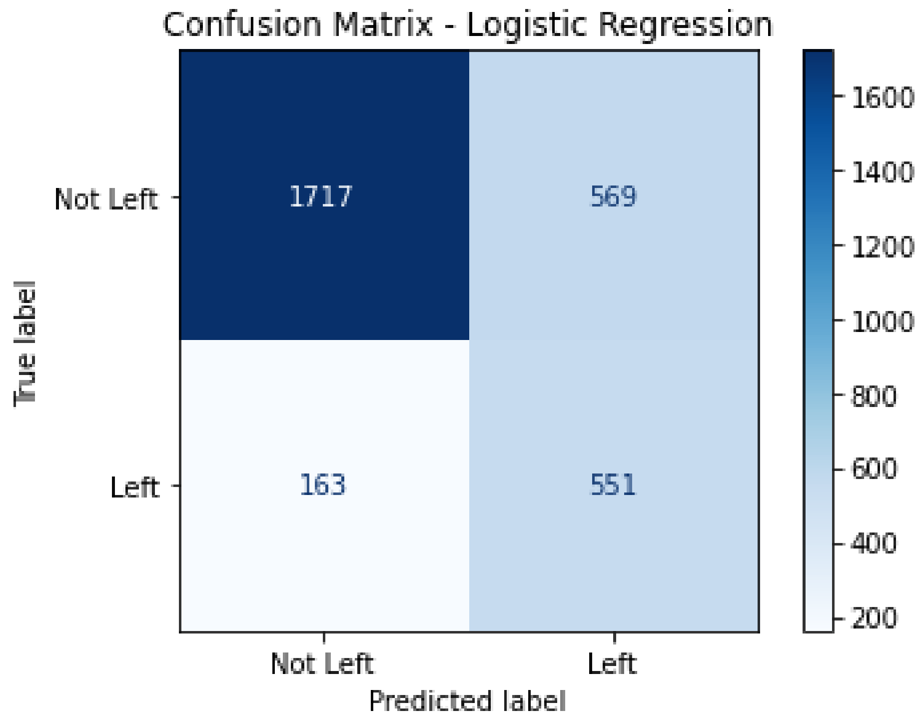

```
# The Logistic Regression Classification
-The logistic regression classification report shows that the model is
performing well enough to be used in practice.
However, there is room for improvement, especially in terms of
precision and recall for the "Not Left" class.


#The Random Forest Classification
-The model's performance is also consistent across all classes, with
all precision, recall, and F1 scores being above 0.97.
This suggests that the model is not biased towards any particular
class.

#The Gradient Boosting Classification
-The gradient boosting classification report in the chart shows a very
good performance.the model is able to correctly
 classify 97.5% of the data points, and it is able to identify both
positive and negative cases with a high degree of accuracy.
-The model's performance is also consistent across all classes, with
all precision, recall, and F1 scores being above 0.95,
This suggests that the model is not biased towards any particular
class.



# 6. Identify the best model and justify the evaluation metrics used.
Find the ROC/AUC for each model and plot the ROC curve.
Find the confusion matrix for each of the models.
```

```
From the confusion matrix, explain which metric needs to be used-
Recall or Precision?

from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score, roc_curve
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Function to calculate ROC/AUC and plot ROC curve
def plot_roc_curve(model, X, y, title):
    y_prob = model.predict_proba(X)[:, 1]
    auc = roc_auc_score(y, y_prob)
    fpr, tpr, _ = roc_curve(y, y_prob)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, label=f'AUC = {auc:.2f}')
    plt.plot([0, 1], [0, 1], '--', color='gray', label='Random')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {title}')
    plt.legend()
    plt.show()

# Function to plot confusion matrix
def plot_confusion_matrix(model, X, y, title):
    y_pred = model.predict(X)
    cm = confusion_matrix(y, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Not Left', 'Left'])

    plt.figure(figsize=(8, 6))
    disp.plot(cmap='Blues')
    plt.title(f'Confusion Matrix - {title}')
    plt.show()

# Function to calculate and print multiple metrics
def print_evaluation_metrics(model, X, y, title):
    y_pred = model.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred)
    recall = recall_score(y, y_pred)
    f1 = f1_score(y, y_pred)

    print(f"{title} Metrics:")
    print(f"  Accuracy: {accuracy:.2f}")
    print(f"  Precision: {precision:.2f}")
    print(f"  Recall: {recall:.2f}")
    print(f"  F1-Score: {f1:.2f}")
    print()

# Assuming logistic_model, rf_model, gb_model are initialized as
```

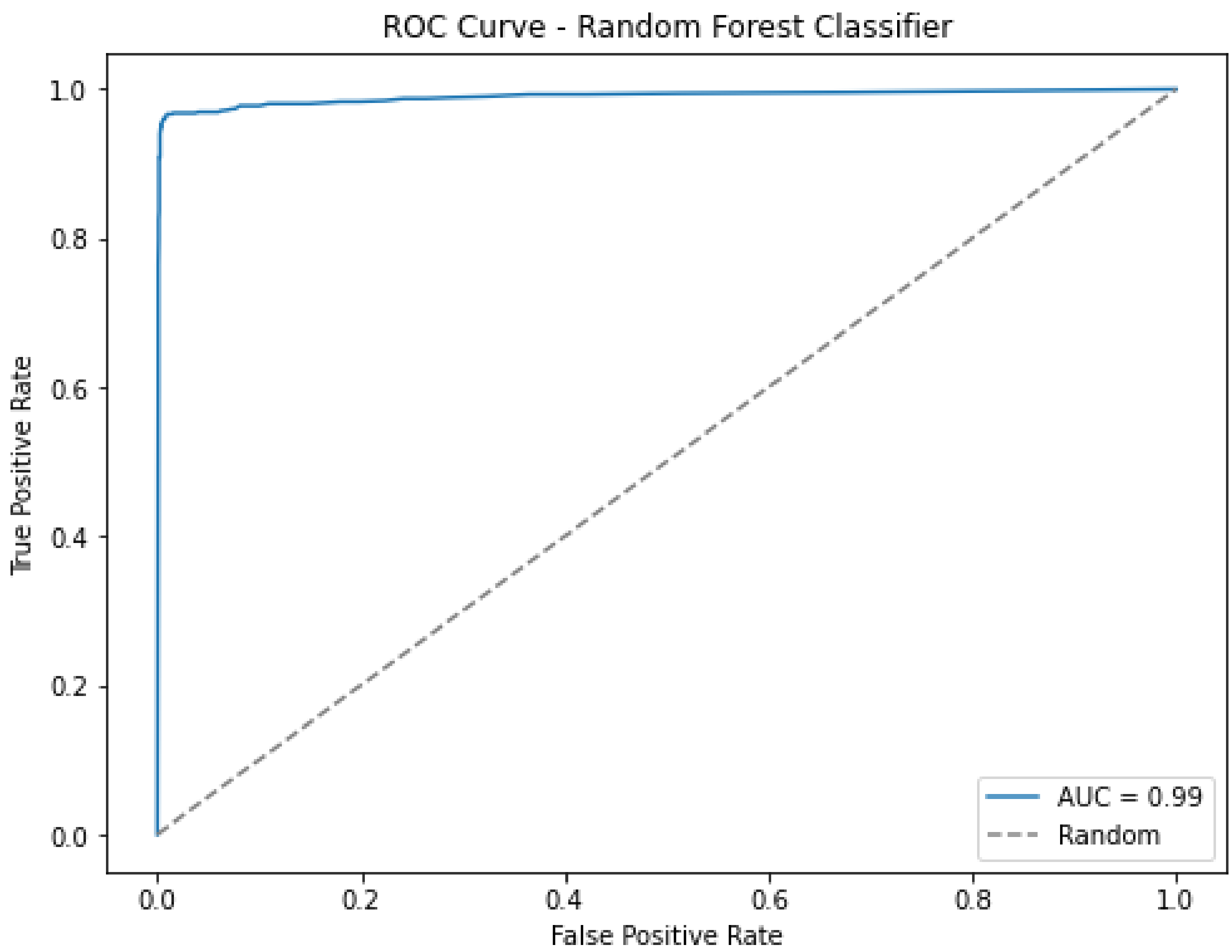
```
mentioned in previous responses

# Logistic Regression
logistic_model = LogisticRegression(random_state=123)
logistic_model.fit(X_train_resampled, y_train_resampled)
plot_roc_curve(logistic_model, X_test, y_test, 'Logistic Regression')
plot_confusion_matrix(logistic_model, X_test, y_test, 'Logistic
Regression')
print_evaluation_metrics(logistic_model, X_test, y_test, 'Logistic
Regression')

# Random Forest Classifier
rf_model = RandomForestClassifier(random_state=123)
rf_model.fit(X_train_resampled, y_train_resampled)
plot_roc_curve(rf_model, X_test, y_test, 'Random Forest Classifier')
plot_confusion_matrix(rf_model, X_test, y_test, 'Random Forest
Classifier')
print_evaluation_metrics(rf_model, X_test, y_test, 'Random Forest
Classifier')

# Gradient Boosting Classifier
gb_model = GradientBoostingClassifier(random_state=123)
gb_model.fit(X_train_resampled, y_train_resampled)
plot_roc_curve(gb_model, X_test, y_test, 'Gradient Boosting
Classifier')
plot_confusion_matrix(gb_model, X_test, y_test, 'Gradient Boosting
Classifier')
print_evaluation_metrics(gb_model, X_test, y_test, 'Gradient Boosting
Classifier')
```



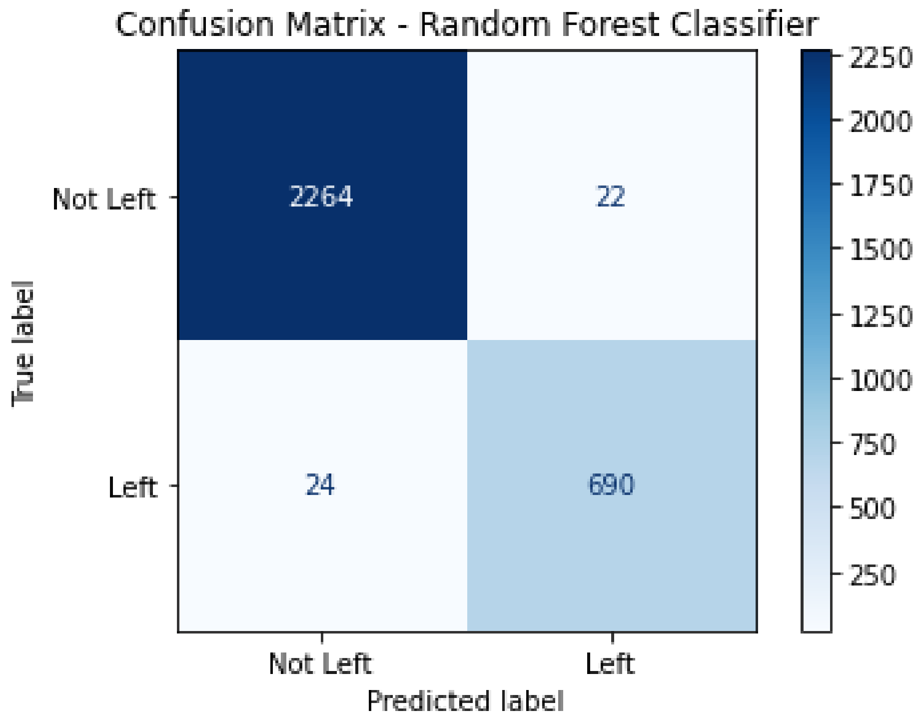

```
<Figure size 576x432 with 0 Axes>
```



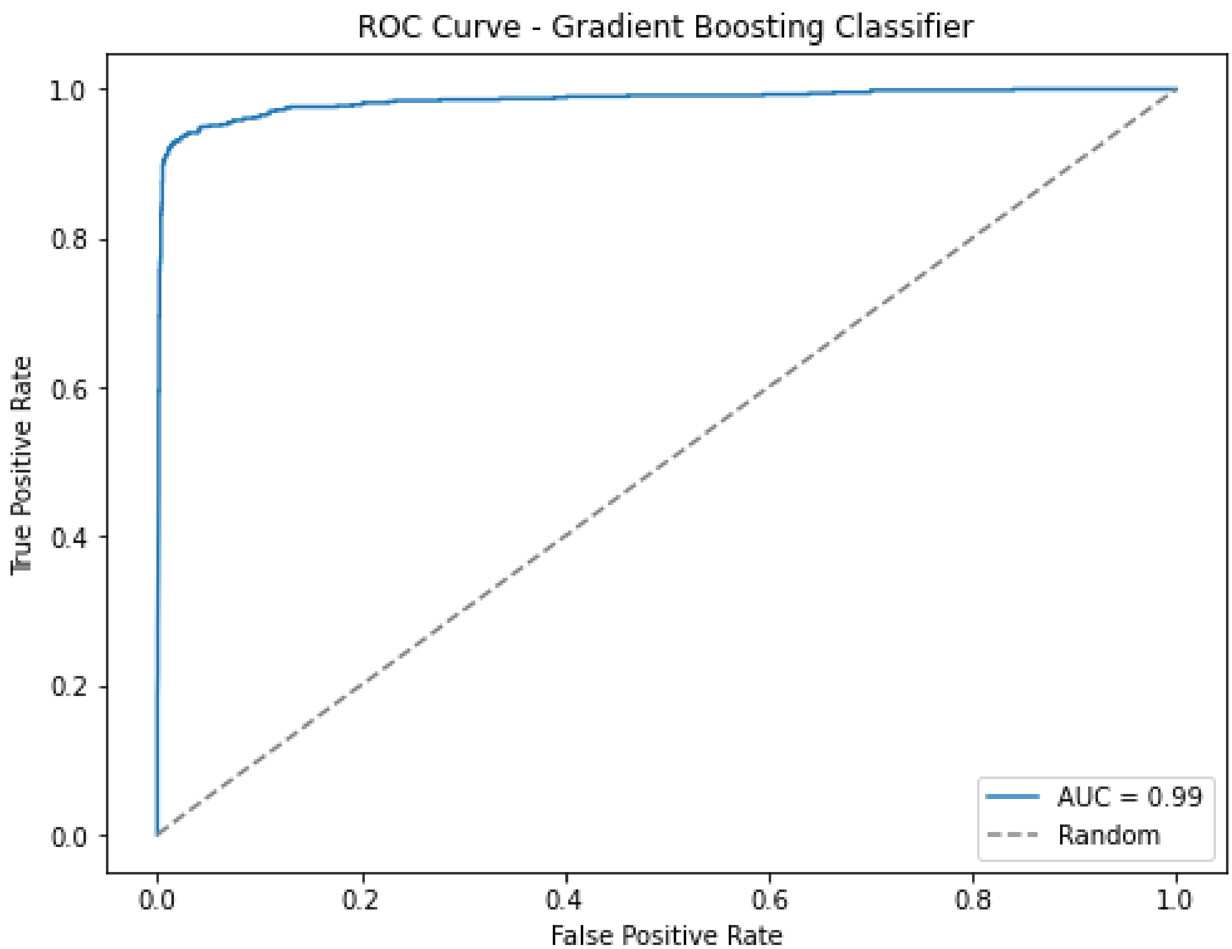

```
Logistic Regression Metrics:
  Accuracy: 0.76
  Precision: 0.49
  Recall: 0.77
  F1-Score: 0.60
```



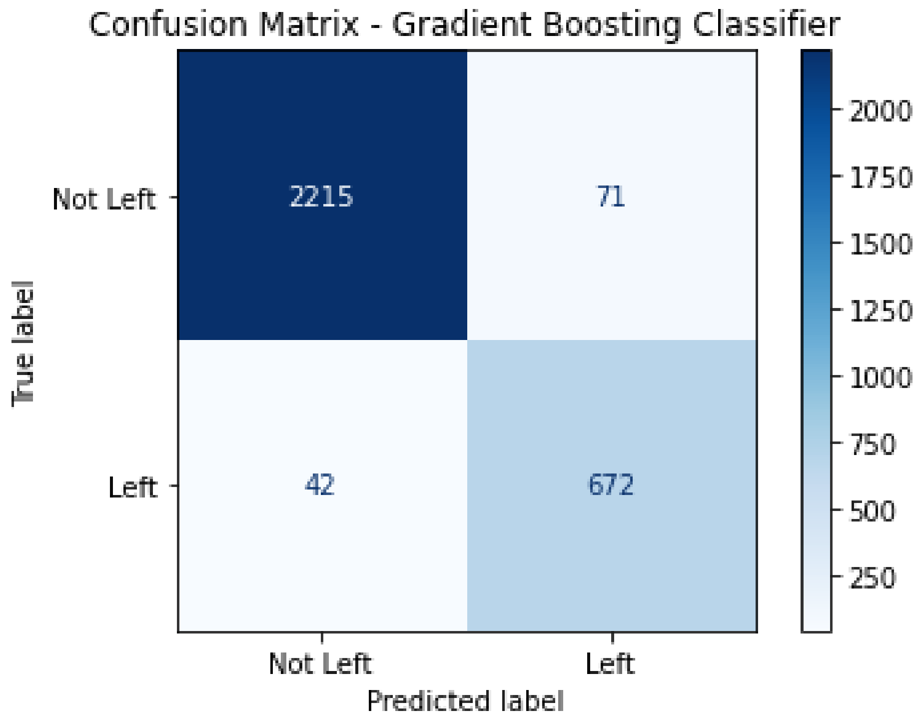

```
<Figure size 576x432 with 0 Axes>
```




```
Random Forest Classifier Metrics:
  Accuracy: 0.98
  Precision: 0.97
  Recall: 0.97
  F1-Score: 0.97
```




```
<Figure size 576x432 with 0 Axes>
```

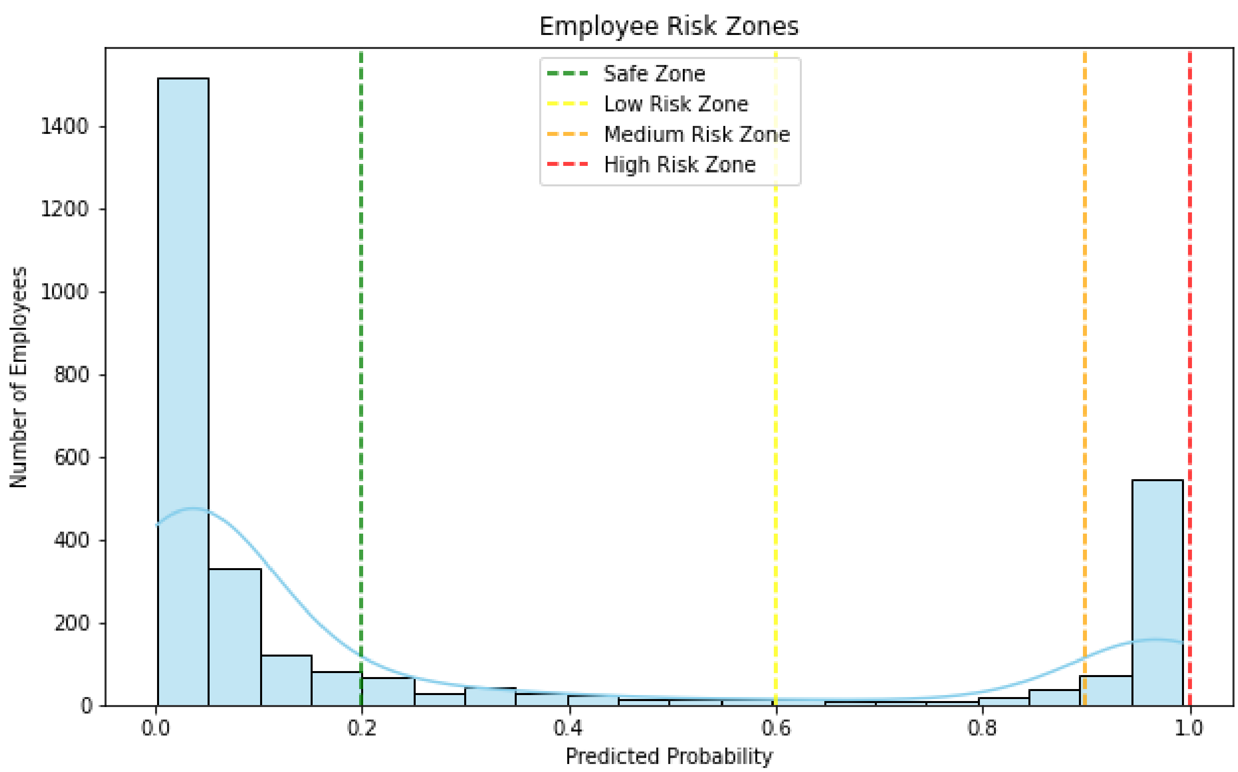


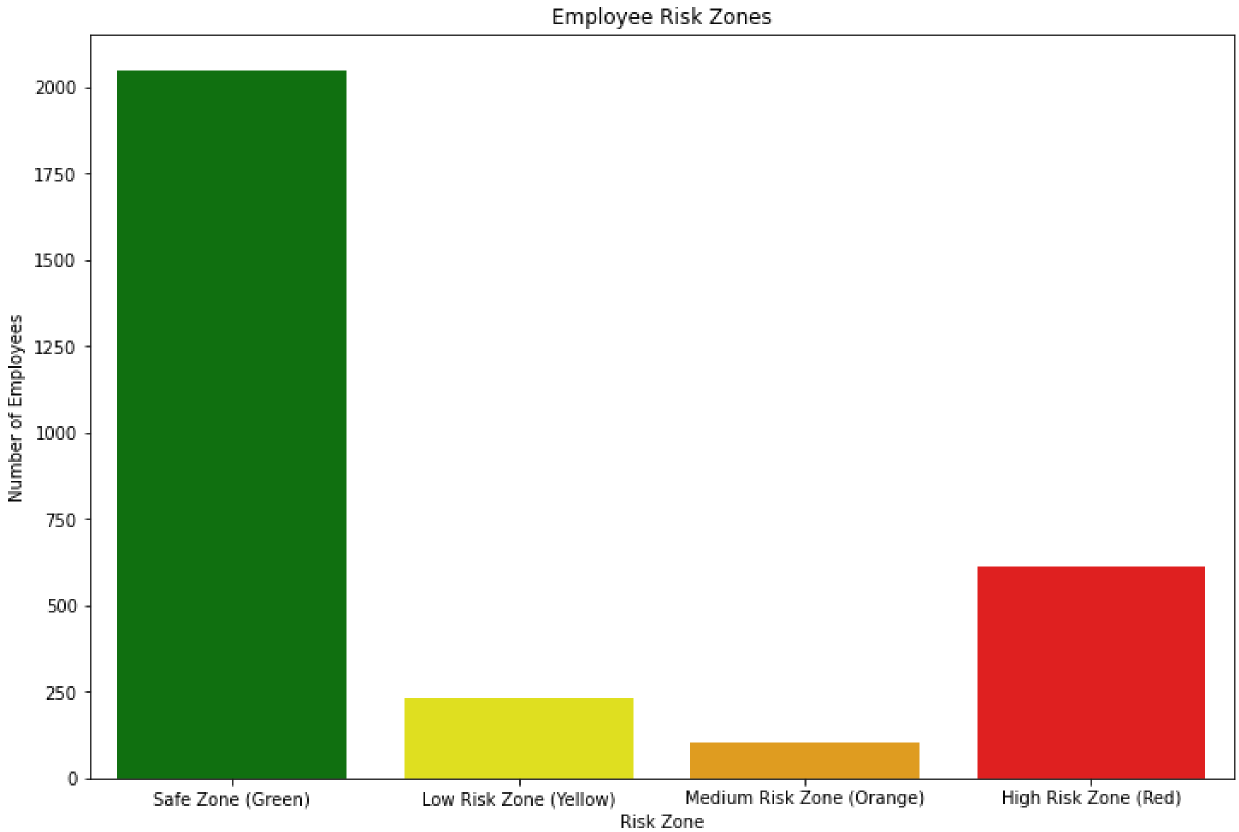

```
Gradient Boosting Classifier Metrics:
  Accuracy: 0.96
  Precision: 0.90
  Recall: 0.94
  F1-Score: 0.92


 #7.Suggest various retention strategies for targeted employees.
-Using the best model, predict the probability of employee turnover in
the test data.
-Based on the below probability score range, categorize the employees
into four zones and suggest your thoughts on the retention
 strategies for each zone.
 Safe Zone (Green) (Score < 20%)
 Low Risk Zone (Yellow) (20% < Score < 60%)
 Medium Risk Zone (Orange) (60% < Score < 90%)
 High Risk Zone (Red) (Score > 90%).

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

# Assuming gb_model is your trained Gradient Boosting Classifier
predicted_probabilities = gb_model.predict_proba(X_test)[:, 1]

# Define probability score ranges for different zones
safe_zone = 0.2  # 20%
low_risk_zone = 0.6  # 60%
```

```
medium_risk_zone = 0.9  # 90%

# Categorize employees into different zones
employee_zones = []
for prob in predicted_probabilities:
    if prob < safe_zone:
        employee_zones.append("Safe Zone (Green)")
    elif safe_zone <= prob < low_risk_zone:
        employee_zones.append("Low Risk Zone (Yellow)")
    elif low_risk_zone <= prob < medium_risk_zone:
        employee_zones.append("Medium Risk Zone (Orange)")
    else:
        employee_zones.append("High Risk Zone (Red)")

# Create a DataFrame to display results
results_df = pd.DataFrame({'Employee': range(1, len(X_test) + 1),
                            'Predicted Probability':
predicted_probabilities,
                            'Risk Zone': employee_zones})

# Display the results
print(results_df)

# Visualize the distribution of employees in different zones
plt.figure(figsize=(10, 6))
sns.histplot(predicted_probabilities, bins=20, kde=True,
color='skyblue')
plt.axvline(x=safe_zone, color='green', linestyle='--', label='Safe
Zone')
plt.axvline(x=low_risk_zone, color='yellow', linestyle='--',
label='Low Risk Zone')
plt.axvline(x=medium_risk_zone, color='orange', linestyle='--',
label='Medium Risk Zone')
plt.axvline(x=1.0, color='red', linestyle='--', label='High Risk
Zone')
plt.xlabel('Predicted Probability')
plt.ylabel('Number of Employees')
plt.title('Employee Risk Zones')
plt.legend()
plt.show()

# Categorize employees into different zones
employee_zones = pd.cut(predicted_probabilities, bins=[-float('inf'),
safe_zone, low_risk_zone, medium_risk_zone, float('inf')],
                        labels=['Safe Zone (Green)', 'Low Risk Zone
(Yellow)', 'Medium Risk Zone (Orange)', 'High Risk Zone (Red)'])

# Create a DataFrame to display results
results_df = pd.DataFrame({'Employee': range(1, len(X_test) + 1),
                            'Predicted Probability':
```

```
predicted_probabilities,
                            'Risk Zone': employee_zones})

# Plot the bar chart
plt.figure(figsize=(12, 8))
sns.countplot(x='Risk Zone', data=results_df, palette=['green',
'yellow', 'orange', 'red'])
plt.xlabel('Risk Zone')
plt.ylabel('Number of Employees')
plt.title('Employee Risk Zones')
plt.show()
```

```
      Employee  Predicted Probability             Risk Zone
0            1               0.004529     Safe Zone (Green)
1            2               0.957684  High Risk Zone (Red)
2            3               0.093000     Safe Zone (Green)
3            4               0.017591     Safe Zone (Green)
4            5               0.035261     Safe Zone (Green)
...        ...                    ...                   ...
2995      2996               0.012131     Safe Zone (Green)
2996      2997               0.038994     Safe Zone (Green)
2997      2998               0.053314     Safe Zone (Green)
2998      2999               0.006474     Safe Zone (Green)
2999      3000               0.929408  High Risk Zone (Red)

[3000 rows x 3 columns]
```






```
# Choosing Gradient Boosting Classfier as the best model, From the
charts we can observe that:
- Majority of the employees are in the safe zone.
- Around 200-225 employees are in the low zone.
- Less than 75 employees are in the Medium risk zone.
- Around 600 employees are in the high risk zone.

# Retention strategies can vary based on the risk zone assigned to
employees:
Safe Zone (Green):
Recognition and Appreciation: Acknowledge and appreciate the efforts
of employees in this zone.
Work-Life Balance: Promote a healthy work-life balance to maintain job
satisfaction.

Low Risk Zone (Yellow):
Training Programs: Provide additional training programs to enhance
skills and knowledge.
Performance Incentives: Introduce performance-based incentives to
boost motivation.

Medium Risk Zone (Orange):
Feedback Mechanism: Establish a feedback system to address concerns
and improve communication.
```

```
Leadership Development: Invest in leadership development programs to
prepare employees for higher responsibilities.

High Risk Zone (Red):
Retention Bonuses: Consider offering retention bonuses to encourage
employees to stay.
Exit Interviews: Conduct exit interviews to understand the reasons for
dissatisfaction and take corrective actions.
```