Employee Turnover Analytics

Aim of the project a comprehensive analysis of employee turnover using various data preprocessing techniques, exploratory data analysis,machine learning modeels, and retention strategy suggestions.

Project statement:

1. perform data quality check by checking for missing values if any.
2. Understand what factors contirubted most to employee turnover by EDA.
3. Peerform clustering of Employees who left based on thier satisfaction and evaluation.
4. Handle the left Class Imbalance using SMOTE technique.
5. Perform k-flod cross-validation model training and evaluated performance.
6. Identify the best model and justify the evalution metrics used.
7. Suggest various retention strategies for targeted employees.

1. Data preprocessing

This step involves the importing required libraries,data loading, data quality check, renaming the columns and exploring the data types.The missing calues and outliers are dealt within this step.This step makes the data ready to apply further analysis and visualization.

In [87]:
```
#importing required libraries
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [88]:
```
data=pd.read_csv("HR_comma_sep.csv")
```

In [89]:
```
#marking copy of dataset
data = data.copy()
```

In [90]:
```
#get to know the dataset
data.head()
```

Out[90]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend
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	

In [91]:
```
data.shape #dimension of dataset i.e.,number of total data entries and features
```

Out[91]:
```
(14999, 10)
```

In [92]:
```
#checking for any missing or null values in dataset
data.isnull().sum() #no missing or null values are there
```

Out[92]:
```
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
```

In [93]:
```
data.info() #checking datatypes of a feature columns
```

```
<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
```

In [94]:
```
data.columns.str.capitalize() #capitalizing the name of features
```

Out[94]:
```
Index(['Satisfaction_level', 'Last_evaluation', 'Number_project',
       'Average_montly_hours', 'Time_spend_company', 'Work_accident', 'Left',
       'Promotion_last_5years', 'Sales', 'Salary'],
      dtype='object')
```

In [95]:
```
data = data.rename(columns={'average_monthly_hours':'avg_monthly_hours','sales':
```

In [96]:
```
data['Department'].unique()
```

Out[96]:
```
array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',
       'IT', 'product_mng', 'marketing', 'RandD'], dtype=object)
```

In [97]:
```
#checking for duplicates
dup = data.duplicated().sum()
dup
```

Out[97]:
```
3008
```

In [98]:
```
dup_row = data[data.duplicated()]
print('Duplicate rows:')
print(dup_row)
```

```
Duplicate rows:
       satisfaction_level  last_evaluation  number_project  \
396                  0.46             0.57               2
866                  0.41             0.46               2
1317                 0.37             0.51               2
1368                 0.41             0.52               2
1461                 0.42             0.53               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  \
396                     139                   3              0     1
866                     128                   3              0     1
1317                    127                   3              0     1
1368                    132                   3              0     1
1461                    142                   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  Department  salary
396                        0       sales     low
866                        0  accounting     low
1317                       0       sales  medium
1368                       0       RandD     low
1461                       0       sales     low
...                      ...         ...     ...
14994                      0     support     low
14995                      0     support     low
14996                      0     support     low
14997                      0     support     low
14998                      0     support     low

[3008 rows x 10 columns]
```

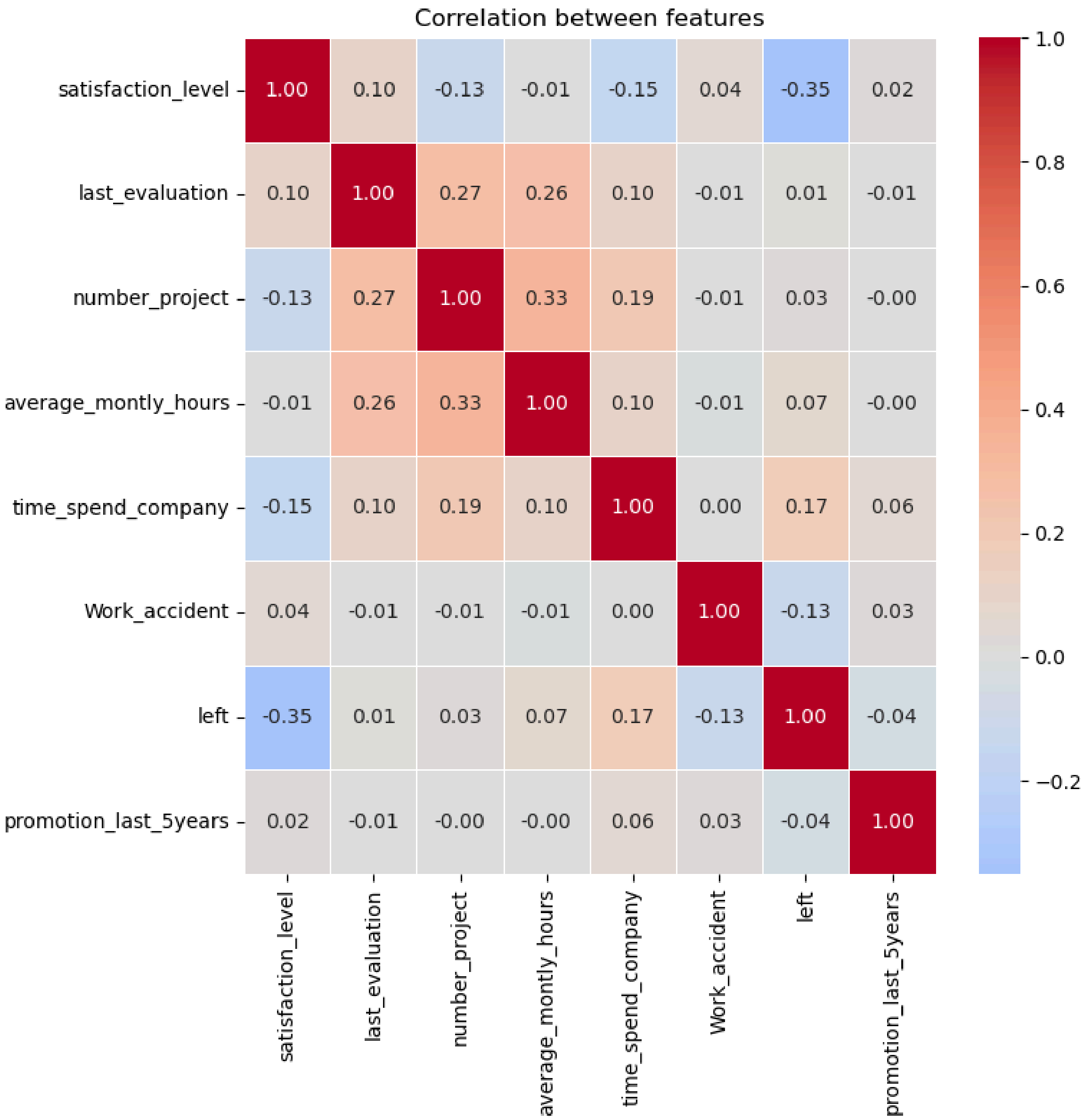
In [99]:
```
#drop the duplicates
data = data.drop_duplicates()
```

In [100…
```
#checking the shape of final datased
data.shape
```

Out[100…
```
(11991, 10)
```

2.EDA

In [101…
```
import matplotlib.pyplot as plt
import seaborn as sns

# Select only numeric columns
numeric_data = data.select_dtypes(include=['int64', 'float64'])

# Create the heatmap
plt.figure(figsize=(8, 8))
sns.heatmap(
    numeric_data.corr(),
    annot=True,
    cmap='coolwarm',
    center=0,
    fmt=".2f",
    linewidths=0.5
)
plt.title("Correlation between features")
plt.tight_layout()
plt.show()
```



	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
satisfaction_level	1.00	0.10	-0.13	-0.01	-0.15	0.04	-0.35	0.02
last_evaluation	0.10	1.00	0.27	0.26	0.10	-0.01	0.01	-0.01
number_project	-0.13	0.27	1.00	0.33	0.19	-0.01	0.03	-0.00
average_montly_hours	-0.01	0.26	0.33	1.00	0.10	-0.01	0.07	-0.00
time_spend_company	-0.15	0.10	0.19	0.10	1.00	0.00	0.17	0.06
Work_accident	0.04	-0.01	-0.01	-0.01	0.00	1.00	-0.13	0.03
left	-0.35	0.01	0.03	0.07	0.17	-0.13	1.00	-0.04
promotion_last_5years	0.02	-0.01	-0.00	-0.00	0.06	0.03	-0.04	1.00

Insights from the correlation: last_evaluation is 35% positively correlated with number_projects and 34% positively with average_monthly_hours. number_project is positively correlated 35% with last_evaluation and 42% with average_monthly_hours. left (employee) have positive correlation with last_evaluation, number_project, avg_monthly_hours, time_spend_company and negative correlation with satisfaction_level, Work_accident and promotion_last_5years. These correlation heatmap provides information about important features or factors contributing to employee turnover.

Higher evaluations, involvement in more projects and longer monthly hours may be associated with lower turnover rates.

Dissatisfaction, lack of promotions, and involvement in work accidents may be potential indicators of higher turnover rates.

In [102…

```
#distribution plot of employee satisfaction
sns.distplot(data['satisfaction_level'])
plt.title('Distribution plot of employee satisfaction')
plt.show()
```

Observation

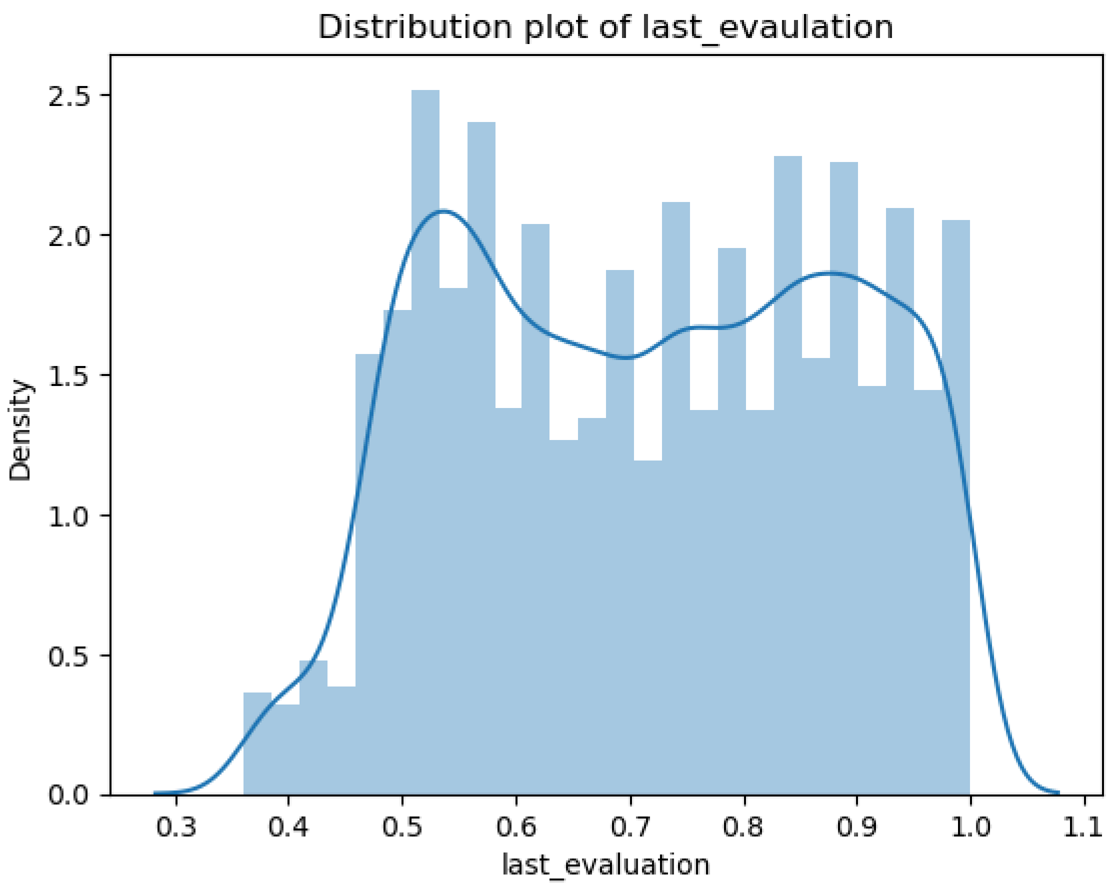
1.The distribution of employee satisfaction levels is skewed towards higher satisfaction.

2.Most of the employees have high satisfaction.

3.Few employees have low satisfaction level.

In [103…

```
#distribution plot of employee evaluation
sns.distplot(data['last_evaluation'])
plt.title('Distribution plot of last_evaulation')
plt.show()
```



Observation

1.It shows binomial distribution, with two distinct peaks in histogram and the density curve.

2.First peak located at 0.5 to 0.8 on x-axis, shows a group of employee received relatively high evaluation scores.

3.Second peak located at 0.2 to 0.4 on x-axis, shows a group of employee recieved relatively low evaluation scores.

In [ ]:
```
#distribution plot of employee average monthly hours
sns.distplot(data['avg_monthly_hours'])
plt.title('Distribution plot of average_montly_hours')
plt.show()
```

Observations

1.It shows mainly three clusters of employees based on the average number of hours work in a month.

2.It shows peaks at around 100,150 and 280 avg.hours spend in month.

3.While some employees work closer to lower end of distribution (around 100 hours), many work longer hours (around 150-280).

4.Density curved provides the smoothed representation of the peaks and valleys of the overall shape and pattern more clearly.

In [125…

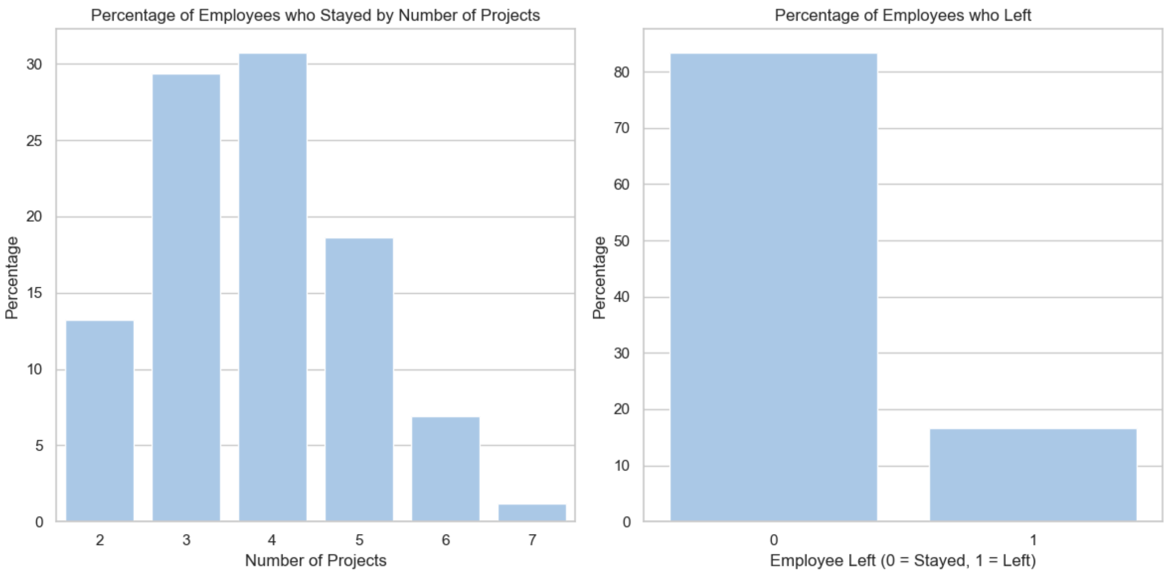
```
total_emp = len(data)

#employee stayed
emp_stayed = data['number_project'].value_counts()
print('\nempployee_stayed', emp_stayed)
per_empstayed = (emp_stayed/total_emp)*100
print("\npercentage of employee stayed:", per_empstayed.round(2))

#employee left
emp_left = data['left'].value_counts()
print('\nemployee_left', emp_left)
per_empleft = (emp_left/total_emp)*100
print("\npercentage of employee who left:", per_empleft.round(2))
```

```
empployee_stayed number_project
4    3685
3    3520
5    2233
2    1582
6     826
7     145
Name: count, dtype: int64

percentage of employee stayed: number_project
4    30.73
3    29.36
5    18.62
2    13.19
6     6.89
7     1.21
Name: count, dtype: float64

employee_left left
0    10000
1     1991
Name: count, dtype: int64

percentage of employee who left: left
0    83.4
1    16.6
Name: count, dtype: float64
```

In [126…

```
# Create a figure and axis objects
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

# Plot for employees who stayed
sns.barplot(x=emp_stayed.index, y=per_empstayed, ax=axes[0])
axes[0].set_title('Percentage of Employees who Stayed by Number of Projects')
axes[0].set_xlabel('Number of Projects')
axes[0].set_ylabel('Percentage')

# Plot for employees who left
sns.barplot(x=emp_left.index, y=per_empleft, ax=axes[1])
axes[1].set_title('Percentage of Employees who Left')
axes[1].set_xlabel('Employee Left (0 = Stayed, 1 = Left)')
axes[1].set_ylabel('Percentage')

# Show plot
```

```
plt.tight_layout()
plt.show()
```



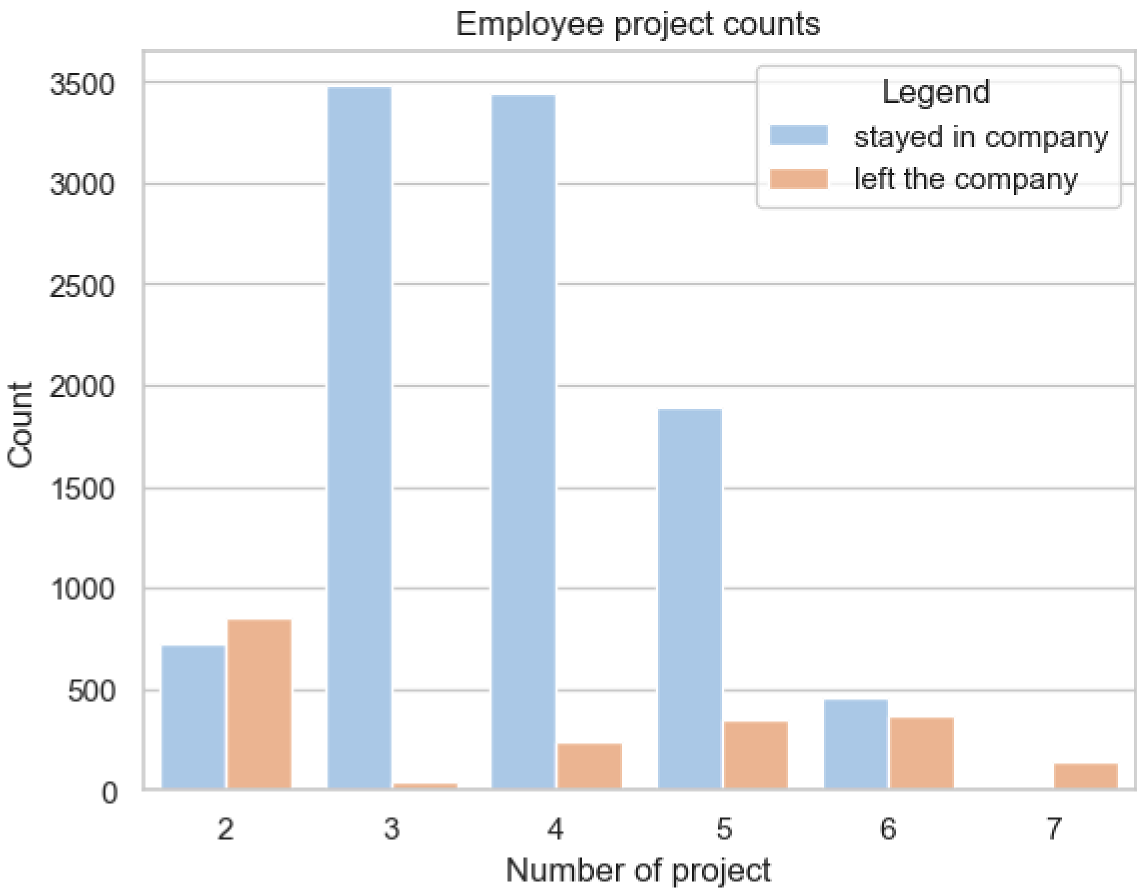
Observations

1.there are 2 to 7 projects assighned to employee

2.tallest bar corrsponds to employees who were involved in 4 projects, representing 30% of total employees who stayed Perentage of employees who left Percentage of employees who stayed by Number of projects .'0' indicates emplooyees who stayed in the company, while '1' represents employee who left .it appears that a larger proportion of employees stayed in the company compared to those who left, as evidenced by the height discrepancy between two bars.

In [128…

```
sns.countplot(x='number_project', hue= 'left', data = data)
plt.title('Employee project counts')
plt.xlabel('Number of project')
plt.ylabel('Count')
plt.legend(title = 'Legend', labels=['stayed in company', 'left the company'])
plt.show()
```



3.Clustering of Employees

Based on the satisfaction and evaluation

In [129…

```
from sklearn.cluster import KMeans

left_emp = data[data['left']==1][['satisfaction_level', 'last_evaluation']]

kmeans = KMeans(n_clusters = 3, random_state = 42)
kmeans.fit(left_emp)

left_emp['cluster']= kmeans.labels_
```

In [130…

```
plt.figure(figsize =(10,6))
sns.scatterplot(x='satisfaction_level', y='last_evaluation', hue = 'cluster', da
plt.title('Clusters of Employees who Left')
plt.xlabel('Satisfaction Level')
plt.ylabel('Last Evaluation')
plt.show()
```

Observations

1.each point of the plot represents an employee who left the company, with different colors indicating the cluster to which they belong.

2.three clusters are identified and differentiated by colors: purple, yellow and teal

3.Teal color : Employee characterized by low satisfaction level and high last evaluation score

4.Yellow color : Employee characterized by high satisfaction level and high last evaluation scores

5.purple color : Employee characterized with moderate satisfaction levels and moderate to high last evaluation scores

4. Left class imbalance using SMOTE

In [131…
```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
```

In [132…
```
cat_col = data.select_dtypes(include=['object']).columns
num_col = data.select_dtypes(include=['int64','float64']).columns
```

In [135…
```
data_cat = pd.get_dummies(data[cat_col], drop_first = True)
dat_num = pd.concat([data[num_col], data_cat], axis=1)
```

In [138…
```
X = dat_num.drop('left', axis = 1)
y = dat_num['left']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.2, stratif
```

```
smote = SMOTE(random_state=123)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

In [139…

```
print('Class distribution after SMOTE:')
print(y_train_resampled.value_counts())
```

```
Class distribution after SMOTE:
left
0    7999
1    7999
Name: count, dtype: int64
```

5. Perfrom 5-fold cross-validation model training and evaluation

Logistic regression model

In [141…

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
```

In [142…

```
log_reg = LogisticRegression()
```

In [143…

```
log_reg_cv_predictions = cross_val_predict(log_reg, X_train_resampled, y_train_r
```

In [144…

```
log_reg_classification_report = classification_report(y_train_resampled, log_reg
print(log_reg_classification_report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.80      0.82      7999
           1       0.81      0.84      0.82      7999

    accuracy                           0.82     15998
   macro avg       0.82      0.82      0.82     15998
weighted avg       0.82      0.82      0.82     15998
```

Random Forest Classifier

In [145…

```
from sklearn.ensemble import RandomForestClassifier
# Initialize Random Forest Classifier model
rf_classifier = RandomForestClassifier()
```

In [146…

```
# Perform 5-fold cross-validation
rf_cv_predictions = cross_val_predict(rf_classifier, X_train_resampled, y_train_

# Generate classification report
rf_classification_report = classification_report(y_train_resampled, rf_cv_predic
print(rf_classification_report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      7999
           1       0.99      0.96      0.98      7999

    accuracy                           0.98     15998
   macro avg       0.98      0.98      0.98     15998
weighted avg       0.98      0.98      0.98     15998
```

Gradient Boosting Classifier

In [147…

```
from sklearn.ensemble import GradientBoostingClassifier

# Initialize Gradient Boosting Classifier model
gb_classifier = GradientBoostingClassifier()
```

In [148…

```
#Perform 5-fold cross-validation
gb_cv_predictions = cross_val_predict(gb_classifier, X_train_resampled, y_train_

# Generate classification report
gb_classification_report = classification_report(y_train_resampled, gb_cv_predic
print(gb_classification_report)
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      7999
           1       0.98      0.95      0.96      7999

    accuracy                           0.96     15998
   macro avg       0.96      0.96      0.96     15998
weighted avg       0.96      0.96      0.96     15998
```

6. Identify the Best Model and Justify the Evaluation Metrics Used

ROC/AUC for each model and plot the ROC curve

In [149…

```
from sklearn.metrics import roc_auc_score, roc_curve
```

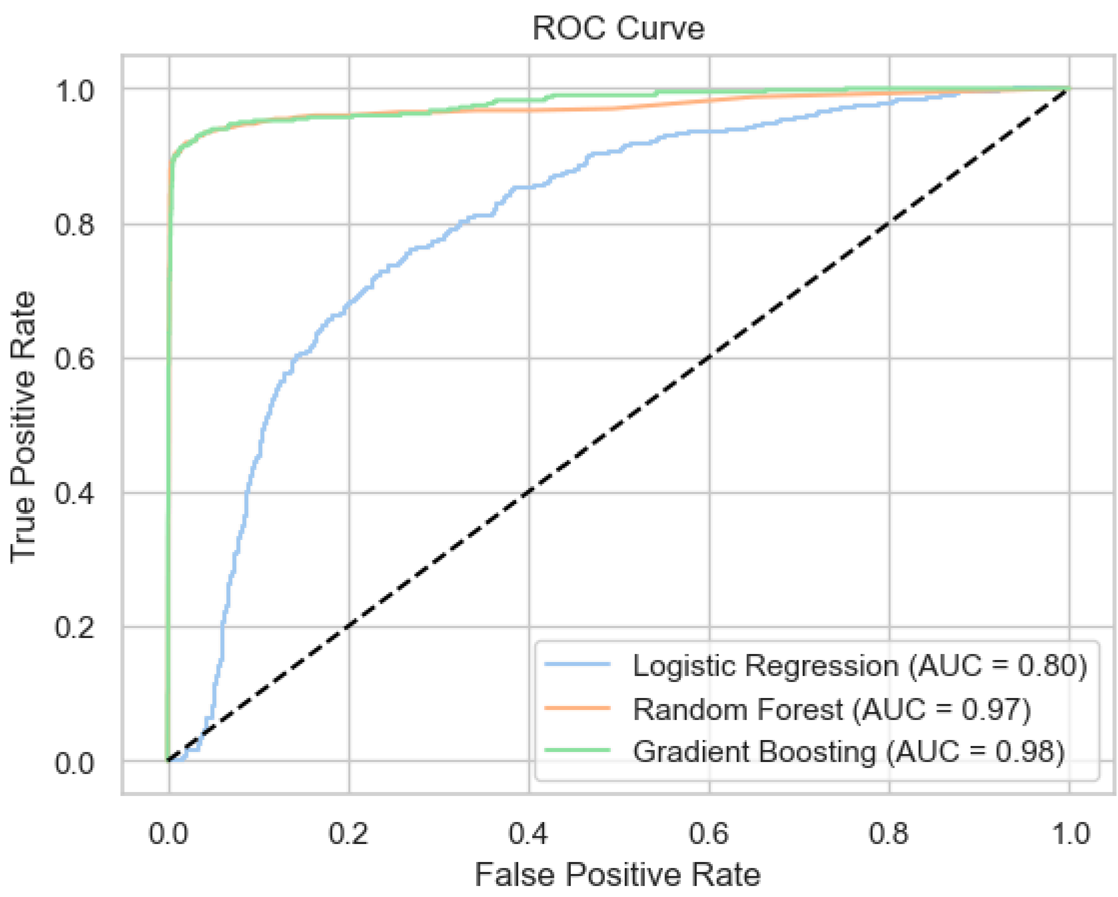
In [150…

```
# Initialize models
models = [log_reg, rf_classifier, gb_classifier]
model_names = ['Logistic Regression', 'Random Forest', 'Gradient Boosting']
```

In [151…

```
# Iterate over models
for model, name in zip(models, model_names):
    model.fit(X_train_resampled, y_train_resampled)
    y_pred_proba = model.predict_proba(X_test)[:, 1]
    auc = roc_auc_score(y_test, y_pred_proba)
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='black')
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



Observations

1.ROC curve plots the trade-off between sensitivity and specificity i.e., True positivie rate and False positive rate, respectively.

2.AUC for Gradient Boosting is high i.e., 0.98.

Confusion matrix

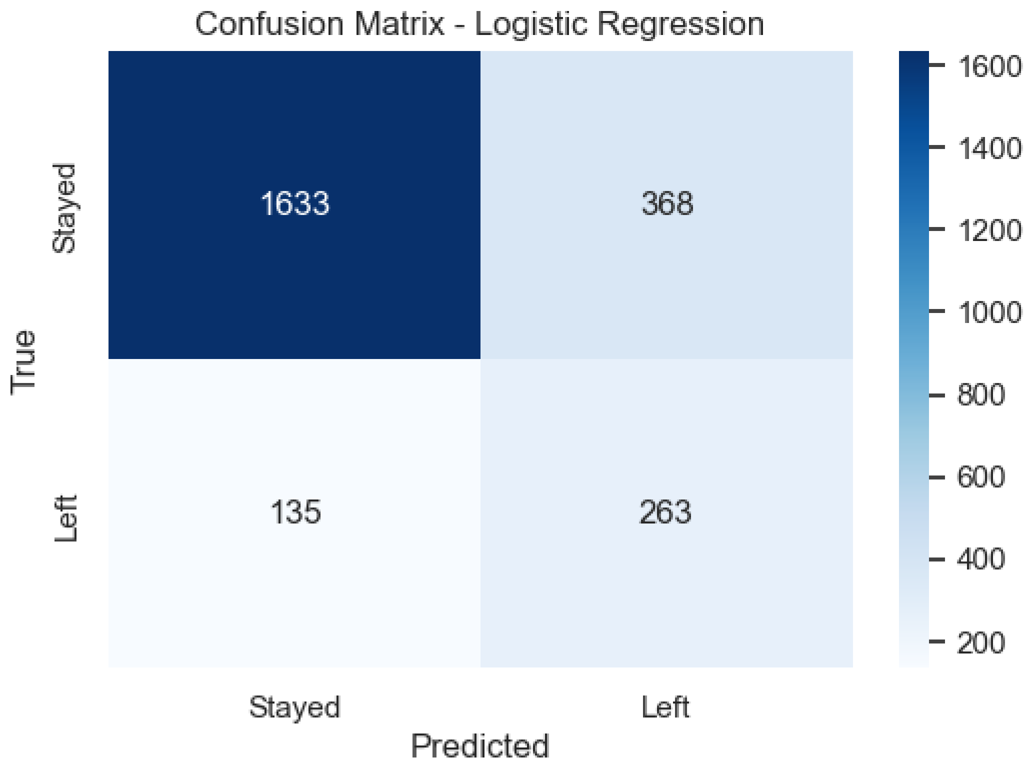
In [152…

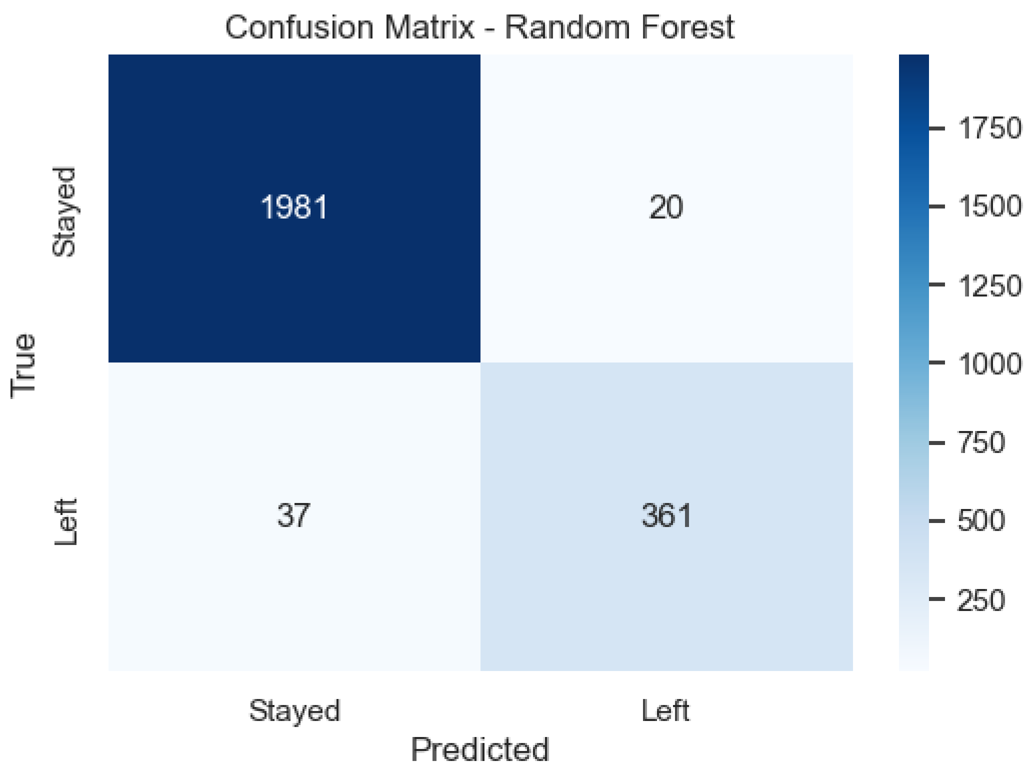
```
from sklearn.metrics import confusion_matrix
```

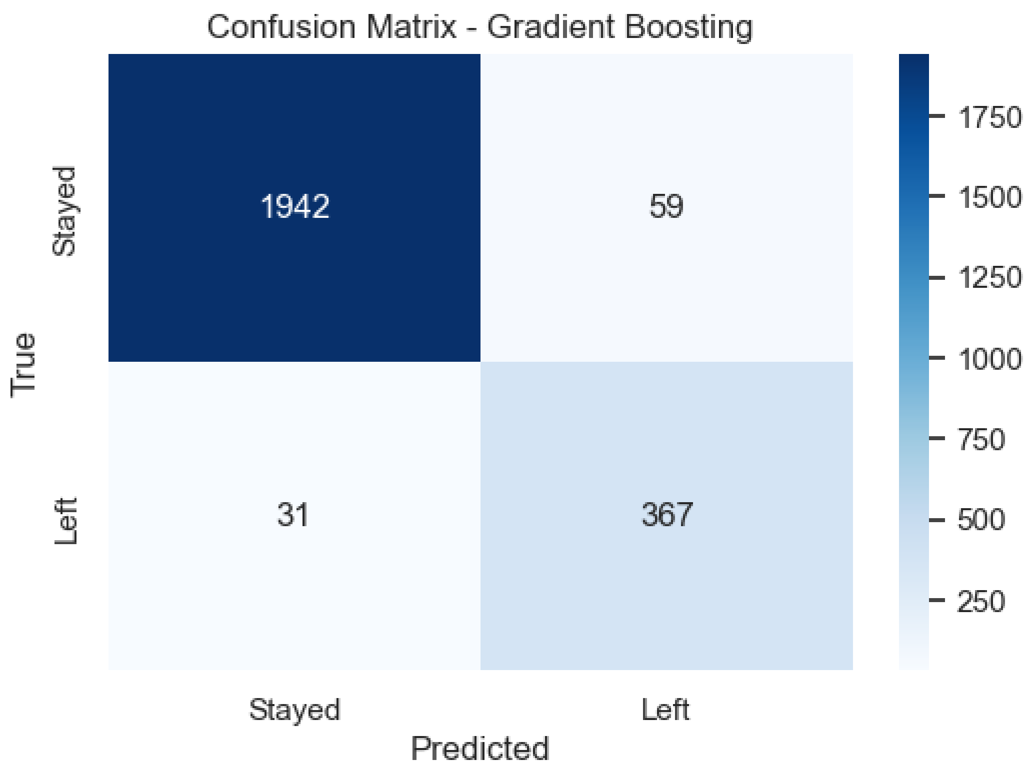
In [153…

```
for model, name in zip(models, model_names):
    # Fit the model
    model.fit(X_train_resampled, y_train_resampled)

    # Predict on the test data
    y_pred = model.predict(X_test)

    # Generate confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Plot confusion matrix
    plt.figure(figsize=(6, 4))
```

```
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['Stayed', 'L
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title(f'Confusion Matrix - {name}')
    plt.show()
```







7.Suggest Various Retention Strategies for Targeted Employees

From the observations, the best-performing model is Gradient Boosting. We have already trained and test data for the Gradient Boosting.

In [154…

```
probabilities = gb_classifier.predict_proba(X_test)[:,1]
```

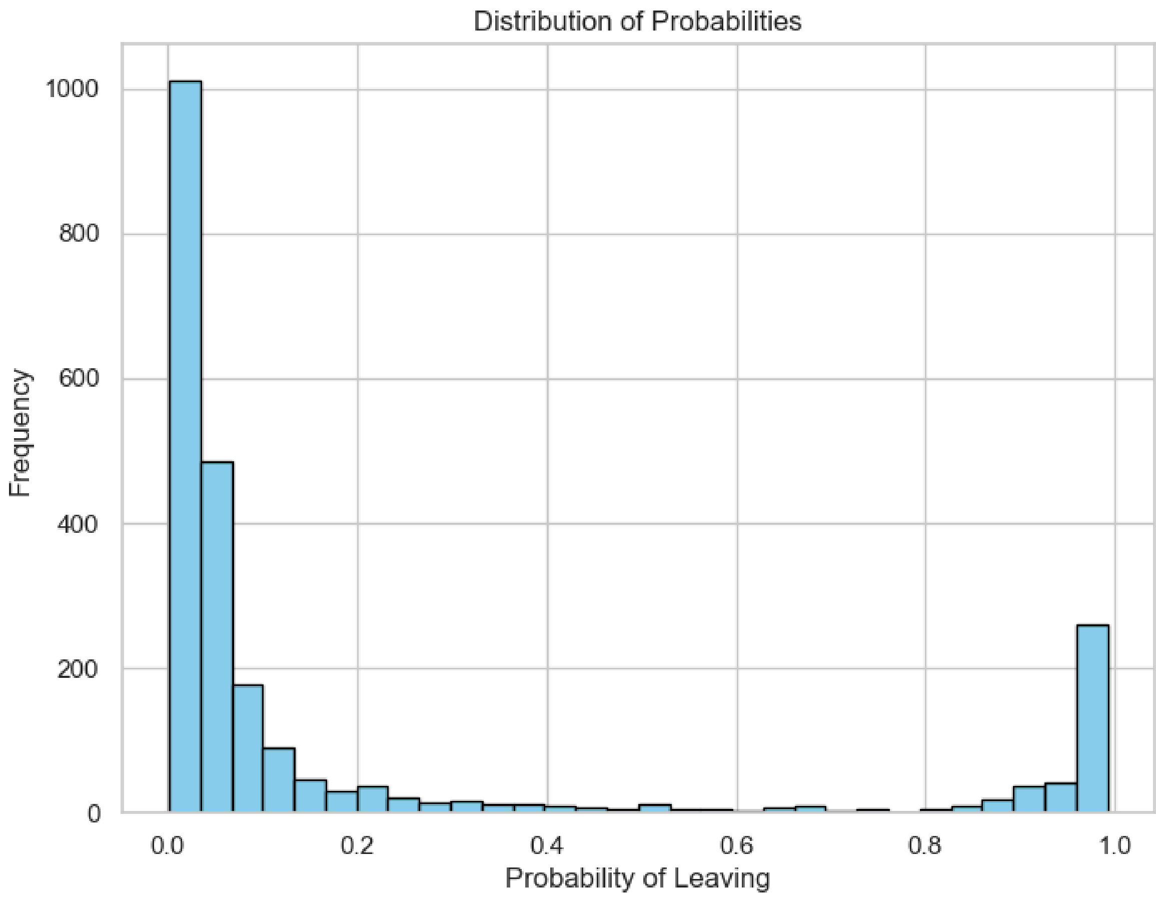
In [155…

```
print(probabilities)
```

```
[0.22155321 0.74152043 0.03250642 ... 0.00924773 0.03250642 0.00908635]
```

In [156…

```
import matplotlib.pyplot as plt

# Plot the distribution of probabilities
plt.figure(figsize=(8, 6))
plt.hist(probabilities, bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Probability of Leaving')
plt.ylabel('Frequency')
plt.title('Distribution of Probabilities')
plt.grid(True)
plt.show()
```



Observations

1.Majority of the employees have low probabilities of leaving, as indicated by taller bars on the left side.

2.At higher probabilities, the frequency of employees decreaases, leading to shorter bars on the right side of the plot.

Categorize Employees into Risk Zones

In [157…

```
thresholds ={
    'Safe Zone':0.2,
    'Low Risk Zone':0.6,
    'Medium Risk Zone':0.9,
    'High Risk Zone':1.0
}

risk_zones = []

for p in probabilities:
    if p < thresholds['Safe Zone']:
        risk_zones.append('Safe Zone')
    elif thresholds['Safe Zone'] <= p < thresholds['Low Risk Zone']:
        risk_zones.append('Low Risk Zone')
    elif thresholds['Low Risk Zone'] <= p < thresholds['Medium Risk Zone']:
        risk_zones.append('Medium Risk Zone')
    else:
        risk_zones.append('High Risk Zone')
```

In [158…
```
# Add the risk zones to the test data as a new column
test_data_with_risk_zones = X_test.copy()
test_data_with_risk_zones['Risk Zone'] = risk_zones
```

Propose appropriate retention strategies tailored to each zone

In [159…
```
print({'Safe Zone\n':'Retention strategies could focus on recognition, rewards, 
print({'Low Risk Zone\n':'Retention strategies could include regular check-ins, 
print({'Medium Risk Zone\n':'Retention strategies may involve addressing specifi
print({'High Risk Zone\n':'Urgent intervention may be needed, such as addressing
```

```
{'Safe Zone\n': 'Retention strategies could focus on recognition, rewards, and ca
reer development to maintain high satisfaction levels.'}
{'Low Risk Zone\n': 'Retention strategies could include regular check-ins, opport
unities for skill development, and providing a positive work environment.'}
{'Medium Risk Zone\n': 'Retention strategies may involve addressing specific conc
erns, conducting stay interviews, and offering promotions or transfers to increas
e engagement.'}
{'High Risk Zone\n': 'Urgent intervention may be needed, such as addressing workl
oad issues, improving communication, and offering retention bonuses or special in
centives.'}
```

In [ ]: