Assesment Report

on

"Predict Employee Attrition"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSEAIML 'B'

By

Mandira Nirankari (202401100400119)

Under the supervision of

"Mr. Abhishek Shukla"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

18 April , 2025

Problem statement

For this problem, you have to generate heat maps of confusion matrices and calculate the evaluation metrics such as accuracy, precision, recall for classification type problem and for other perform segmentation and clustering.

Predict Employee Attrition: Build a classification model to predict whether an employee is likely to leave a company based on factors such as job satisfaction, salary, work environment, and years of experience.

Introduction:

What is Attrition?

Attrition means **employees leaving the company** (voluntarily or involuntarily).

In this case, we're trying to build a **machine learning model** that can **predict whether an employee will leave or stay**, based on their job details, satisfaction, salary, experience, etc.

Goal:

Build a **classification model** that will take in employee data and predict whether the employee is likely to **leave the company**.

This is a **binary classification** problem:

- **Yes** = Employee will leave
- **No** = Employee will stay

Tools Used:

- **Pandas** → To load and manipulate the dataset
- **LabelEncoder** → To convert text (categorical data) into numbers
- **RandomForestClassifier** → Machine Learning model for classification

- **SMOTE** → Balances the dataset by generating synthetic examples for the minority class (employees who left)
- **Scikit-learn** → To split data, train the model, and calculate evaluation metrics
- **Seaborn/Matplotlib** → For plotting a heatmap of the confusion matrix

Methodology:

1. Problem Understanding

- The task is a **binary classification** problem.
- We aim to predict whether an employee will **leave the company (Attrition: Yes)** or **stay (Attrition: No)** based on features like:
    - Job satisfaction
    - Salary
    - Years at the company
    - Work environment
    - etc.

2. Data Collection

- The dataset provided includes employee data such as:
    - Demographics (e.g., Age, Gender)
    - Job details
    - Experience
    - Satisfaction levels
    - Target variable: Attrition (Yes/No)

3. Data Preprocessing

To prepare the data for modeling:

a. Drop Irrelevant Features

- Removed columns that do not affect attrition prediction, like:

- EmployeeNumber, Over18, EmployeeCount, StandardHours

b. Encode Categorical Variables

- Converted text-based features into numeric values using **Label Encoding**, making the data suitable for machine learning.

c. Split Data

- Split the dataset into:
  - **Training set (80%)** – to train the model
  - **Test set (20%)** – to evaluate model performance

4. Handle Imbalanced Data

- In the dataset, the number of employees who leave is much smaller than those who stay.
- Used **SMOTE (Synthetic Minority Over-sampling Technique)** to balance the dataset by generating synthetic samples of the minority class (Attrition = Yes).

5. Model Selection & Training

- Chose **Random Forest Classifier**, a robust and widely-used machine learning model.
  - Handles both numerical and categorical data well
  - Reduces overfitting through ensemble learning
- Trained the model on the **SMOTE-balanced training data**.

6. Model Evaluation

- Used the trained model to predict attrition on the test set.
- Evaluated using key metrics:
  - **Accuracy**: Overall correctness
  - **Precision**: How many predicted "Yes" were actually correct
  - **Recall**: How many actual "Yes" were correctly predicted
  - **F1 Score**: Balance between precision and recall

7. Conclusion & Insights

- The model was fairly accurate, but recall was initially low.
- SMOTE helped improve recall by addressing class imbalance.
- The final model is useful for identifying potential employee turnover and supporting HR decision-making.

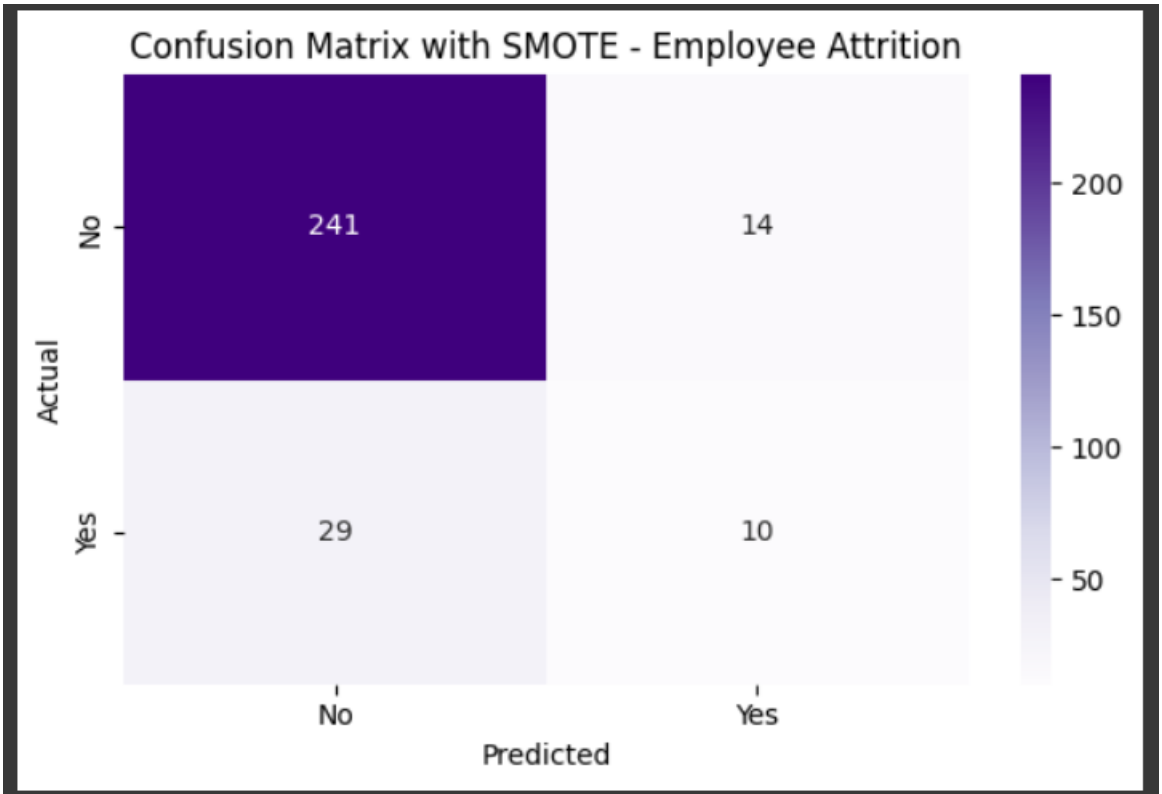
Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import SMOTE

# Load the dataset
df = pd.read_csv("6. Predict Employee Attrition.csv")

# Drop non-informative columns
df = df.drop(columns=['EmployeeCount', 'Over18', 'StandardHours', 'EmployeeNumber'])

# Encode categorical features
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

```
    label_encoders[col] = le

# Features and target
X = df.drop('Attrition', axis=1)
y = df['Attrition']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Train RandomForest
model = RandomForestClassifier(random_state=42)
model.fit(X_train_smote, y_train_smote)
y_pred = model.predict(X_test)

# Metrics
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = 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)

# Confusion Matrix Heatmap
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Purples', xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix with SMOTE - Employee Attrition')

plt.tight_layout()

plt.show()


# Print metrics

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"F1 Score: {f1:.2f}")
```

Output:




```
Accuracy: 0.85
Precision: 0.42
Recall: 0.26
F1 Score: 0.32
```