****

**Assesment Report**

on

**“Problem Statement”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**Name of discipline : Shashank Raj**

By

Name (Roll Number) : 202401100300226

**Under the supervision of**

“Name of mentor/guide” : Abhishek Shukla

**KIET Group of Institutions, Ghaziabad**

**Introduction**

Employee attrition, often referred to as employee turnover, is the process where employees leave an organization voluntarily or involuntarily. High attrition rates can be costly for companies in terms of recruitment, training, lost productivity, and organizational knowledge. In a competitive business environment, retaining skilled employees is crucial for maintaining performance and continuity.

The aim of this project is to leverage machine learning techniques to build a predictive model that can identify employees who are likely to leave the company. By analyzing historical employee data, we can uncover patterns and insights related to attrition. These patterns may include factors like job satisfaction, salary, working environment, distance from home, number of years at the company, and more.

Predicting attrition in advance enables the human resources department to take preventive actions such as offering training, improving work-life balance, or modifying job roles to enhance employee retention. Thus, this project not only provides a technical solution but also contributes to better decision-making and workforce management in organizations.

We use a dataset containing several features about employee demographics, job roles, compensation, and work conditions. Our goal is to clean the data, train a machine learning model, and evaluate its performance using relevant metrics.

**Methodology**

1. **Dataset**: We used a structured employee dataset (e.g., employee\_data.csv) containing details such as Age, Gender, Department, Job Role, Monthly Income, and a target variable Attrition (Yes/No).
2. **Preprocessing**:
   * Label encoding for categorical features.
   * Conversion of target column (Attrition) into binary (1 = Yes, 0 = No).
   * Feature scaling using StandardScaler.
3. **Model**:
   * Logistic Regression was used for binary classification.
   * Data was split into training and testing sets (80/20).
4. **Evaluation**:
   * Accuracy score and classification report.
   * Confusion matrix for visual evaluation.
   * Correlation heatmap for understanding feature relationships.

**Code**

1 # Import necessary libraries for data handling, preprocessing, modeling, and visualization

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset from CSV file into a DataFrame

df = pd.read\_csv('employee\_data.csv')

# Create a dictionary to store label encoders for each categorical column

label\_encoders = {}

# Loop through all categorical columns (excluding the target column 'Attrition')

# and apply label encoding to convert them to numerical values

for column in df.select\_dtypes(include=['object']).columns:

    if column != 'Attrition':

        le = LabelEncoder()

        df[column] = le.fit\_transform(df[column])  # Replace the original column with encoded values

        label\_encoders[column] = le  # Save the encoder for possible inverse transformation

# Convert the target variable 'Attrition' to binary values (Yes → 1, No → 0)

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

# Separate the features (X) and the target variable (y)

X = df.drop('Attrition', axis=1)  # Input features

y = df['Attrition']               # Target labels

# Split the dataset into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize a StandardScaler and fit-transform the training data, then transform the test data

# This step ensures all features are on the same scale (important for some models)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize and train a Logistic Regression model using the training data

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Use the trained model to predict attrition on the test dataset

y\_pred = model.predict(X\_test)

# Evaluate model performance using accuracy score and a detailed classification report

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

# Generate and display a confusion matrix to visualize true vs predicted classifications

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

            xticklabels=['No Attrition', 'Yes Attrition'],

            yticklabels=['No Attrition', 'Yes Attrition'])

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

**Output**

Accuracy : 0.891156462585034

Classification Report :

precision recall f1-score support

0 0.91 0.98 0.94 255

1 0.68 0.33 0.45 39

accuracy 0.89 294

macro avg 0.79 0.65 0.69 294

weighted avg 0.88 0.89 0.87 294

A green squares with white text

AI-generated content may be incorrect.

**References**

 **Internet**

 **AI agents : ChatGPT,Gemini**

 **Libraries Used :** pandas, sklearn, matplotlib, seaborn