Assesment Report

on

“Predict Employee Attrition”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

SESSION 2024-25

in

CSE-AI

By:- Shivansh Gupta (202401100300236)

Under the supervision of

“ABHISHEK SHUKLA SIR –(INTRO TO AI) ”

KIET Group of Institutions, Ghaziabad

Affiliated to **Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

(Formerly UPTU)

May, 2025

Introduction

Employee attrition refers to the reduction in workforce due to employees leaving the organization voluntarily or involuntarily. High attrition rates can negatively impact company performance, increase hiring costs, and reduce employee morale. Predicting attrition allows organizations to proactively engage with at-risk employees and improve retention strategies.

In this project, we aim to develop a machine learning model that accurately predicts employee attrition based on features like job satisfaction, salary, work-life balance, and years of experience. The dataset used contains various features that reflect an employee's relationship with the company.

Methodology

1. **Data Acquisition & Inspection**
    - Loaded the dataset using pandas.
    - Checked the number of rows, columns, and column types.
    - Identified missing values and cleaned the data.
2. **Data Preprocessing**
    - Categorical features were encoded using Label Encoding.
    - Numerical features were standardized using StandardScaler to ensure uniform scale.
3. **Train-Test Split**
    - The dataset was split into 80% training and 20% testing data to evaluate model performance.
4. **Model Building**
    - A Random Forest Classifier was selected for its robustness and accuracy in classification problems.
    - The model was trained on the training data and tested on the test set.
5. **Model Evaluation**

- Used Accuracy Score, Confusion Matrix, and Classification Report (Precision, Recall, F1-score) to evaluate performance.
- Plotted Feature Importance to understand which features had the most influence on predictions.

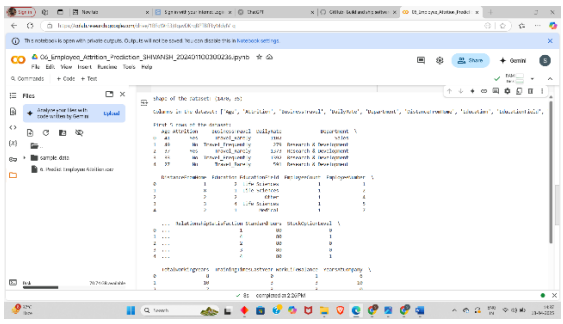
Code

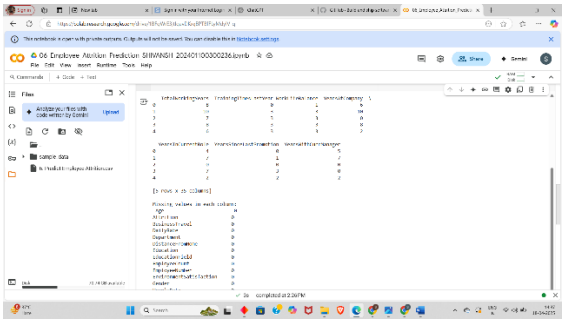
```
# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


# Optional: Uncomment if you want to use XGBoost

# from xgboost import XGBClassifier


# Load the dataset

df = pd.read_csv("6. Predict Employee Attrition.csv")


# Display the shape of the dataset and preview the first few rows

print("Shape of the dataset:", df.shape)

print("\nColumns in the dataset:", df.columns.tolist())

print("\nFirst 5 rows of the dataset:")

print(df.head())
```

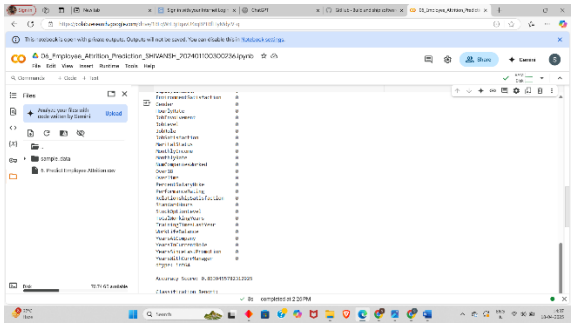
```
# Check for missing values in each column
print("\nMissing values in each column:\n", df.isnull().sum())

# Encode all categorical (object type) columns using Label Encoding
# This converts string labels to numeric codes
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le  # Save encoders in case you need to decode later

# Define feature matrix (X) and target vector (y)
# Make sure 'Attrition' is the correct column name for your target variable
X = df.drop('Attrition', axis=1)
y = df['Attrition']

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features (zero mean, unit variance)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the Random Forest classifier and train it on the training set
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)
```

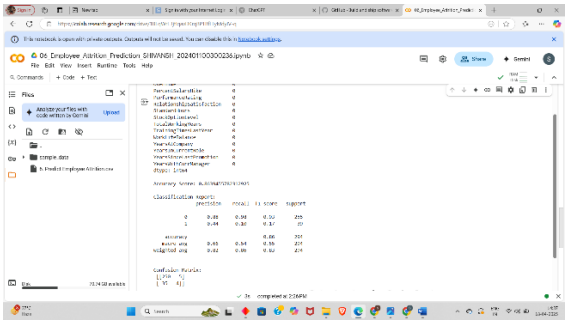
```
# Predict the target for the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model using various metrics
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Plot the feature importance to see which features contribute most to the prediction
importances = model.feature_importances_
features = X.columns
indices = np.argsort(importances)[::-1]  # Sort in descending order

# Create a bar plot of feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x=importances[indices], y=features[indices])
plt.title("Feature Importance from Random Forest")
plt.xlabel("Importance")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```

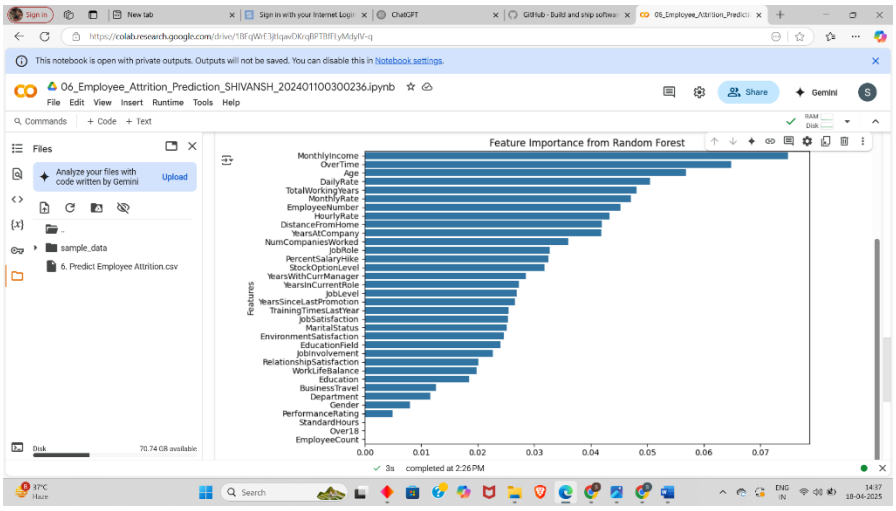
Output / Result











References / Credits

- **Dataset Source:** Provided as part of course materials
- **Tools Used:**
  - Google Colab
  - Python 3
- **Libraries:** pandas, numpy, matplotlib, seaborn, scikit-learn
- **Special Thanks:** Instructor and academic team for guidance