Employee Promotion Analysis

Model Development

Name : Gope Sai Kumar

Branch : Dilsukhnagar

Email : sai.gope2002@gmail.com

Trainer : Rajan Sir

Employee Promotion Analysis

Model Development

- ❖ **Project Overview**
  - This project analyze employee promotion trends using a dataset containing various employee attributes. The goal is to identify patterns and visualize the promotion distribution using a pie chart.
- ❖ **Dataset Information**
  - **Filename:** HR Analytics Dataset
  - **Total Records:** 54,808
  - **Columns:** 14
  - **Target Variable** (1 = Promoted, 0 = Not Promoted)
- ❖ **Project Overview**:-
  - Problem Statement: Predict employee promotion eligibility based on certain attributes/features.
- ❖ **Dataset Overview:**
  - The dataset consists of multiple employee-related attributes, including demographic, performance, and training data. The target variable is is_promoted, which indicates whether an employee was promoted (1) or not (0).
- ❖ **Key Columns:**
  1. employee_id - Unique identifier for each employee
  2. department - Department the employee belongs to
  3. region - Geographic region of the employee
  4. education - Employee's education level
  5. gender - Gender of the employee
  6. recruitment_channel - Source of recruitment
  7. no_of_trainings - Number of training programs attended

  8.age - Employee age

9.previous_year_rating -Performance rating from the previous year

10.length_of_service - Years of service in the company

11.KPIs_met >80% - Whether performance KPIs exceeded 80%

12.awards_won? - Whether the employee has won any awards

13.avg_training_score - Average score in training programs

14.is_promoted - **Target Variable** (1 = Promoted, 0 = Not Promoted)

❖ Objectives

- Load and explore the dataset
- Analyze the distribution of promotions
- Develop a predictive model for employee promotion. The model utilizes employee-related features to predict whether an employee will be promoted.

❖ Necessary imports:

- import pandas as pd
- import numpy as np
- import joblib
- from sklearn.impute import SimpleImputer
- from sklearn.preprocessing import StandardScaler, MinMaxScaler, OrdinalEncoder, OneHotEncoder
- from sklearn.pipeline import Pipeline
- from sklearn.compose import ColumnTransformer
- from sklearn.model_selection import train_test_split, RandomizedSearchCV
- from sklearn.metrics import accuracy_score, classification_report
- import xgboost as xgb

❖ Implementation Steps

1. Load the Dataset

- Read the CSV file using pandas
- Inspect the dataset structure and missing values

2. Data preprocessing

Features:

- Categorical: department, region, education, gender, recruitment_channel

- Numerical: no_of_trainings, age, previous_year_rating, length_of_service, KPIs_met >80%, awards_won?, avg_training_score

Handle missing values:

- education: Impute missing values using the mode (most frequent value in the columns
- previous_year_rating: Fill missing values with 0, as employees with length_of_service equal to 1 do not have a previous rating

3. Feature engineering :

Feature scaling :

Standardization :

- Standardizing "age" , "avg_training_score" columns as they have outliers
- The skewness of two columns less than 1 we are standardizing the columns

Normalization :

- Normalizing "length_of_service" column cause the skewness of the column is greater than 1

Feature selection:

- Droping the columns like is_promoted ,region ,employee_id

4. Encoding :

One hot enconding by using pd.get_dummies() :

- One hot encoding columns "department", "recruitment_channel", "gender" as they don't have any order and value

Ordinal encoding :

- The column "education" is ordinally encoded as it has order education level as "below secondary", "bachelors" "mastersabove"

❖ MODEL DEVELOPMENT :

1. Evaluation Metrics :

- **Accuracy**: Measures overall correctness.
- **Precision**: Fraction of correctly predicted positive instances.

  Formula:

  $$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives(TP)} + \text{False Positive(FP)}}$$

- **Recall**: Ability to capture actual positives.

  Formula:

  $$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives(TP)} + \text{False Negatives(FN)}}$$

- **F1-Score**: Harmonic mean of precision and recall.

  $$\text{F1-score} = \frac{2 \times \text{Precision} + \text{Recall}}{\text{Precision} \times \text{Recall}}$$

- **ROC-AUC Score**: Distinguishes between classes effectively.

❖ Logistic Regression :

- **Classification Report**:

METRICS	0	1
precision	0.93	0.86
recall	1.00	0.25
F1 score	0.97	0.38
support	10093	869

- **Precision**
  - ➢ Class 0 : 93% predications of 0s are corect
  - ➢ Class 1 : 86% of predicted 1s are correct.

- **Recall :**
  - ➢ Class 0: 100% of actual 0s were predicted correctly.
  - ➢ Class 1: Only 25% of actual 1s were predicted correctly (very low recall).

- **F1-Score:**
  - Class 0 has a strong F1-score (0.97), but Class 1 is poor (0.38).

- **Accuracy:**
  - Accuracy of logistic regression is 0.93

❖ **Random Forest :**

- **Classification Report:**

METRICS	0	1
precision	0.95	0.71
recall	0.99	0.35
F1 score	0.97	0.47
support	10093	869

- **Precision:**
  - Class 0 : 95% predications of 0s are corect
  - Class 1 : 71% of predicted 1s are correct.

- **Recall :**

  - Class 0: 99% of actual 0s were predicted correctly.
  - Class 1: Only 35% of actual 1s were predicted correctly (very low recall).

- **F1-Score:**
  - Class 0 has a strong F1-score (0.97), but Class 1 is poor (0.47) but improved then logistic regression

- **Accuracy:**
  - Accuracy of Random forest is 0.93

❖ **XGBoost Classifier :**

- **Classification Report:**

METRICS	0	1
precision	0.95	0.93
recall	100	0.35

F1 score	0.97	0.51
support	10093	869

- **Precision:**
  - Class 0 : 95% predications of 0s are correct
  - Class 1 : 93% of predicted 1s are correct.

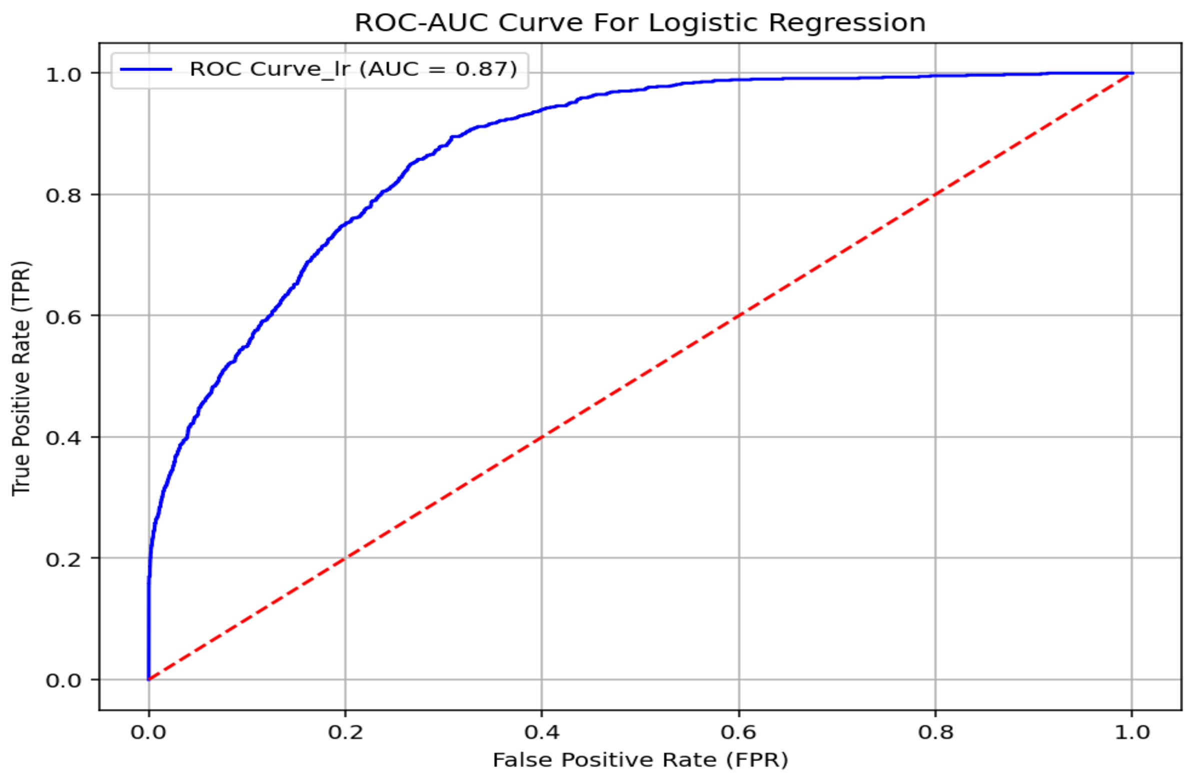
- **Recall:**
  - Class 0: 100% of actual 0s were predicted correctly.
  - Class 1: Only 35% of actual 1s were predicted correctly (very low recall).

- **F1-Score:**
  - Class 0 has a strong F1-score (0.97), but Class 1 is poor (0.51) but improved then random forest

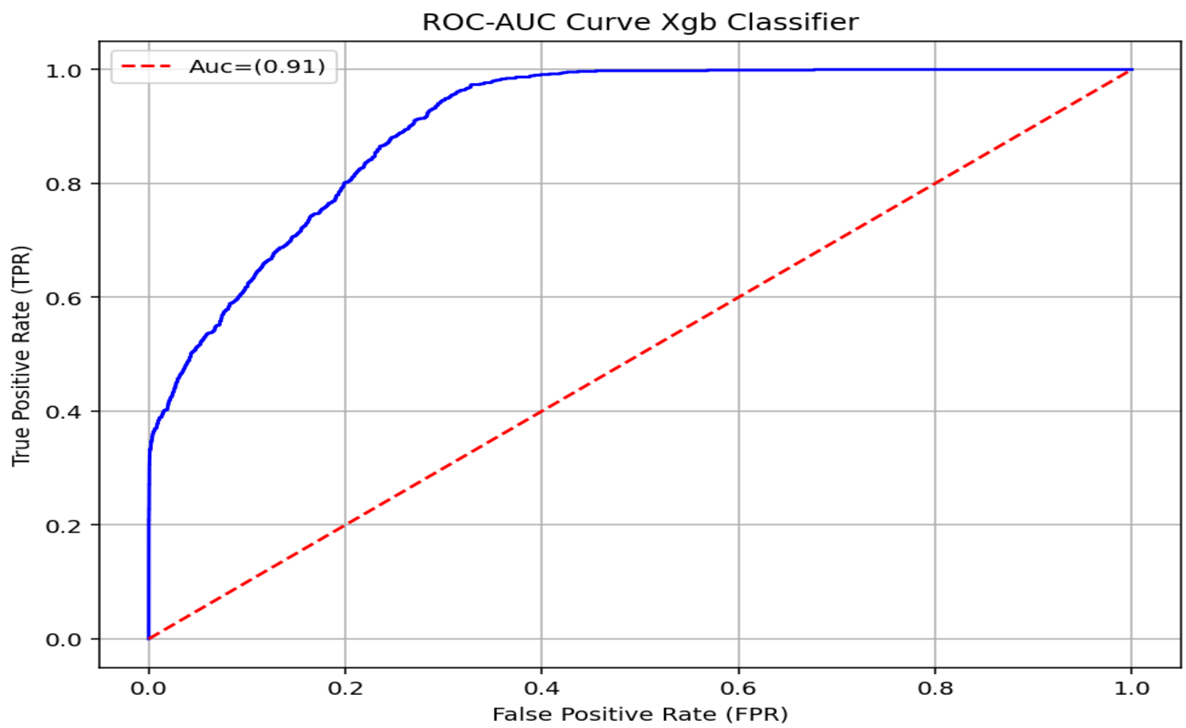
- **Accuracy:**
  - Accuracy of Xgb classifier is 0.94

❖ AUC Score & ROC Curve

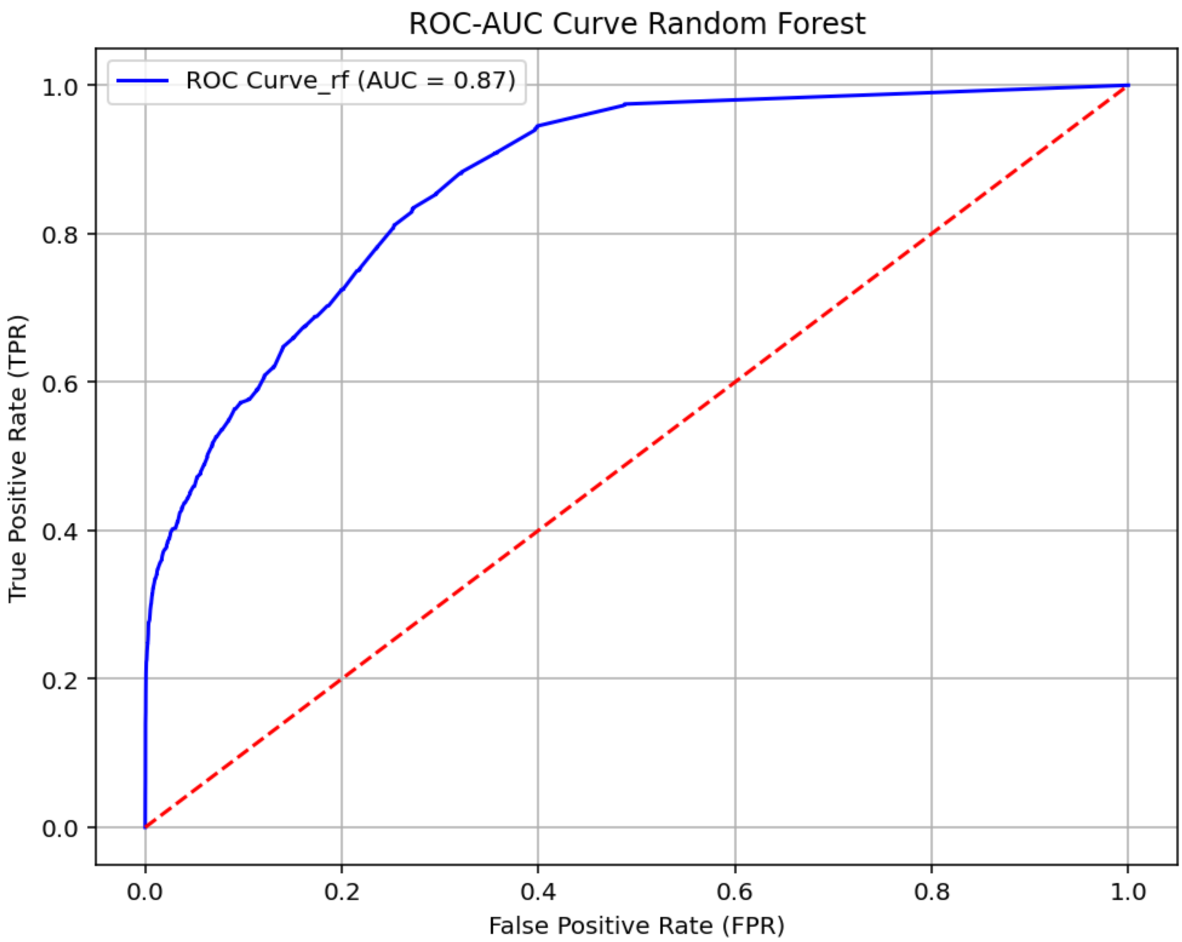
- ROC(**Receiver Operating Charcteristic Curve**) curve plots True Positive Rate against False Positive Rate.
- AUC(**Area Under The Curve**) score measures model's ability to differentiate between classes.
- Comparing ROC-AUC scores for Logistic Regression, Random Forest, and XGBoost



ROC-AUC Curve For Logistic Regression
ROC Curve_lr (AUC = 0.87)
True Positive Rate (TPR)
False Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0



ROC-AUC Curve Xgb Classifier
Auc=(0.91)
True Positive Rate (TPR)
False Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0



❖ **Comparing Auc Scores for Logistic regression , Random Forest,Xgb classifier**

Algorithm Name	**AUC Score**
Logistic regression	0.87
Random forest	0.87
Xgb classifier	0.91

❖ **Model Selection:**

From table the Xgb classifier have more Auc score We are selecting Xgb classifier as our model

❖ **Hyperparameter Tuning For Xgb Classifier:**

Parameters for Xgb Classifier:

- **n_estimators:** Number of boosting rounds.
- **max_depth**: Maximum tree depth.

- **learning_rate:** Step size shrinkage.
- **subsample**: Fraction of samples used for training each tree
- **colsample_bytree**: Fraction of features used for training each tree
- **min_child_weight**: Minimum sum of instance weights needed to make a child node.

❖ **Tuning strategy :**

- **RandomizedSearchCV:** Tests a random subset of hyperparameter combinations.

❖ **Model development & Deployment**

1. Preprocessing Steps & Tools Used :

ColumnTransformer

- **Purpose:** Apply different preprocessing techniques to different columns in a structured way.
- **Process:**
  - Standardized numerical features (age, avg_training_score).
  - Normalized certain numerical features (length_of_service).
  - One-hot encoded categorical features (department, gender, recruitment_channel).
  - Applied ordinal encoding to “education” using a predefined order.
  - Imputed missing values for education (mode) and previous_year_rating (filled with 0) cause for length_of_serivce is equal to there is no rating
  - **Why?**
    ColumnTransformer allows us to process different types of data (numerical, categorical) in a single step.

2. Pipeline

- **Purpose:** Automate sequential data transformations and ensure consistency during training and inference.
- **Process:**
- Created separate pipelines for numerical features, categorical encoding, and missing value imputation.
- Combined these pipelines into the **ColumnTransformer** for seamless preprocessing.

- **Why?**
  Pipelines help us avoid manual transformations and ensure the same preprocessing is applied to both training and test datasets.

❖ Model Deployment Process

1. Model & Preprocessor Saving

- Saved the trained XGBoost model using joblib.dump().
- Saved the preprocessor (ColumnTransformer) so it can be applied to future test data.

2. Predicting on Test Data

- Loaded the test dataset and applied the saved preprocessor.
- Used the trained model to make predictions.
- Saved the results in a submission.csv file.