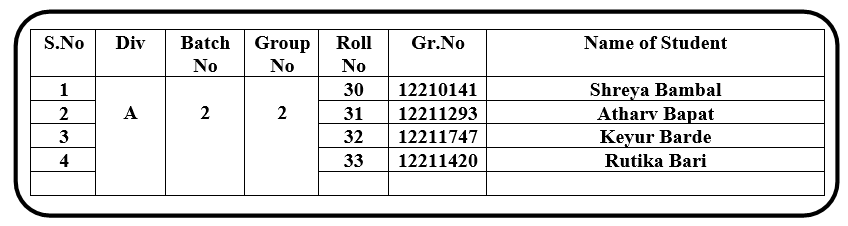
**MD2201 Data Science**

**Course Project**

****

# **Project Title:** Data science in Promotion Prediction

# **Data Set Name:** HR Analytics: Employee Promotion Data

# **Data set Link:** <https://www.kaggle.com/datasets/arashnic/hr-ana>

# **Data Set Description:** The dataset comprises a total of 54,809 rows and 13 columns. Notably, missing values were predominantly observed in the 'previous\_year\_rating' and 'education' fields, encompassing both numerical and character data. Furthermore, an imbalance was evident within the 'is\_promoted' column, wherein the count of zero values, denoting non-promotions, exceeded that of positive values, signifying promotions.

# **Description of Work Done:** Initially, a comprehensive data preprocessing approach was undertaken to address the presence of missing values within the dataset. The missForest algorithm was employed to impute the missing values, ensuring a comprehensive restoration of the dataset's integrity. Additionally, the mode operation was applied to handle any remaining missing values specifically within the 'education' column. Subsequently, efforts were directed towards mitigating the imbalance observed within the dataset. An oversampling technique was systematically utilized to rectify the disparity in class distribution within the 'is\_promoted' column. This method facilitated a rebalancing of the dataset by generating synthetic instances, thereby ensuring a more equitable representation of promotion and non-promotion instances.

# **Literature Survey**:

| **1** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
| --- | --- | --- | --- | --- | --- | --- |
|  | Employee Promotion Prediction by using Machine  Learning Algorithms for Imbalanced Dataset | 2022 | | 1.There are few missing values in data . 2.SMOTE & ROS are used to address class imbalance. 3.Dataset is highly imbalanced so oversampling techniques are used. 4.There are no duplicate values in the dataset. | RF model  has an obvious advantage for employee promotion prediction and achieves the highest classification performance of predicting promotion with 98% F1-score by using ROS imbalanced technique. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| 2nd International Conference on Computing and Machine Intelligence (ICMI) | A data set of Kaggle's publicly accessible employee values was used | | 1.SMOTE 2.Support Vector machine 3.ROS 4.ANN 5.Random Forest | 1. Recall = TP/TP+FN  2.Precision =TP/TP+FP  3.F1 Score is the result which was obtained by combining the calculation of the “recall and precision” values.  4. F1 Score = 2\* (Recall\*Precision)/(Recall+Precision) | |
|  | | | | | | |
| **2** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | PREDICTION OF EMPLOYEE PROMOTION USING HYBRID SAMPLING METHOD WITH MACHINE LEARNING  ARCHITECTURE | 2023 | | A data set of Kaggle's publicly accessible employee values was used | Random Forest with a high | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Malaysian Journal of Computing | A data set of Kaggle's publicly accessible employee values was used | | 1. Logistic Regression 2.Decision Tree 3.Random Forest 4.K-Nearest Neighbors | 1. Recall = TP/TP+FN 2.Precision =TP/TP+FP 3.F1 Score is the result which obtained by combining the calculation of the “recall and precision” values. 4. F1 Score = 2\* (Recall\*Precision)/(Recall+Precision) | |
|  | | | | | | |
| **3** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | KNN and SVM Machine learning to Predict Staff Due for  Promotion and Training | 2022 | | 1.There are no missing values in the data.Also There are no duplicate values.  2.The dataset is not an imbalance dataset so there wasn't any need of normalization. | The prediction of accuracy of SVM(91%)was higher and better than KNN (79%)in terms of prediction accuracy | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| The International Journal of Engineering and Science (IJES) | The dataset used was obtained from a well-structured self study questionnaire distributed | | 1. K-nearest neighbor(KNN)  2. Support vector machines(SVM)  3. Grid search Method | The prediction accuracy, mean score and standard deviation and cross validations curve are employed to evaluate the performance of KNN and SVM classes.  1.Accuracy -> (Total number of correct classifications) / (Total number of classifications) 2.Standard Deviation | |
|  | | | | | | |
| **4** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Machine Learning Application on Employee Promotion | 2023 | | 1.There are few missing values in data .  2.methods of reducing data imbalance and data cleaning methods are not mentioned. | All algorithms show a accuracy around 92% | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Mesopotamian journal of Computer Science | A data set of Kaggle's publicly accessible employee values was used | | 1.Logistic Regression 2.Support vector machine 3.Random Forest | 1. Recall = TP/TP+FN  2. Precision =TP/TP+FP  3. F1 Score is the result obtained by combining the calculation of the “recall and precision” values. 4. F1 Score = 2\* (Recall\*Precision)/(Recall+Precision) | |
|  | | | | | | |
| **5** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Prediction of Employee Promotion Based on Personal  Basic Features and Post Features | 2018 | | 1. Missing values and outliers were deleted before applying algorithms on the data. 2.No need for data normalization. | It is discovered that the prediction effect of random forest model is relatively better because its Accuracy value reaches 96% | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Proceedings of the International Conference on Data Processing and Applications | The experimental data set comes from the staff database of a Chinese state-owned enterprise. | | 1.KNN 2. logistic regression (LR) 3.support vector classifier (SVC),  4.decision tree (DT) 5.random forest (RF) 6. Adaboost. |  | |
|  | | | | | | |
| **6** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Employee Turnover Prediction with Machine  Learning: A Reliable Approach | 2018 | | 1. Missing Value Imputation, imputed based on their data type. For  numeric and categorical, missing entries replaced by median and mode value.  2. Data type conversion and feature selection, converting categorical variables into numeric.  3. Feature scaling, for adjusting range of features, for better performance of classifiers. | The study evaluated classifiers including XGB,  GBT, RF, DT, and neural networks. XGB demonstrated superior performance  with the highest accuracy (ACC), precision  (PRC), and receiver operating characteristic (ROC) improvements: 5.47%, 45%, and 13.26% on 1000\_Bank dataset, respectively. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| IEEE | The first dataset: regional bank in the United  States of America (2013 - 2016)  The second dataset is  created by  IBM Watson Analytics | | 1. Decision Tree  2. K nearest neighbor method  3. Naive Bayes  Method  4. Neural networks  5. Support vector machine  6. Random Forest  7. Gradient boosting trees  8. Extreme gradient boosting  9. Logistic regression  10. Linear discriminant analysis | 1. ACC-> Accuracy, percent correctly classified data.  2. PRC->Precisions,( no. of true positives)/(true positives+false positives)  3. Recall-> (no. of true positives)/(true positives+false negatives)  4. F1-> Harmonic means of precision and recall.  5. ROC-> Receiver Operating Characteristics, Graphical plot of the tradeoff between precision and recall. Area under the ROC curve tells the quality of the classifier used. ROC has primarily been selected in this research for evaluation. | |
|  | | | | | | |
| **7** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Predicting Employee Promotion Using Machine Learning | 2023 | | 1. To cope with imbalance data,  the Synthetic Minority Oversampling Technique  (SMOTE) approach is applied.  2. The SMOTE can  increase classification performance for all classifiers. | It was said that random forest has the highest categorization accuracy. Few parameters were used including the employee's experience, previous year ratings, KPIs performed, average training score, and number of trainings attended. This web program also analyzed basic characteristics such as an employee's gender, age, region, education, and department to compute and reveal whether or not the individual is suitable for promotion. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Journal of Emerging Technologies and Innovative technologies | Open data from Kaggle | | 1. Decision Tree  2. Support vector machine  3. Random Forest |  | |
|  | | | | | | |
| **8** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Employee Performance  Prediction using  different supervised  classifiers | 2021 | | 1. Checking incorrect values, inconsistent data formatting, outliers, missing values, high cardinality, duplicate records, highly correlated variables.  2. Data cleansing and Transformation was made by having a primary key which is the Employee ID and created calculated fields such as tenure group from the hire date,  created bins for age, and made a uniform tagging for the performance evaluation. | In a study using employee performance data, the Two-Class Logistic Regression model outperformed others like Logistic Regression, Decision Tree, and Naïve Bayes in terms of precision.  Permutation Feature Importance  identified top features: Compensation  Grade Profile (0.051), Tenure Group (0.044), and Department (0.017). | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International  Conference on  Industrial  Engineering  and Operations  Management | The data was extracted internally  from the IRIS Team of the  company. | | 1. Two-class Logistic Regression  2. Two-class Bayes point machine  3. Two-class Boosted Decision Tree | 1. Accuracy  2. Precision  3. Recall  4. F1 score  5. AUC | |
|  | | | | | | |
| **9** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Explaining and  Predicting  employees' attrition: a machine learning  approach | 2020 | | 1. Data exploration is the leading procedure in the analytical action of data. In this phase, the techniques of variable identification, univariate analysis, and bivariate analysis have been performed.  2. Uni-variate analysis.  3. Bivariate analysis.  4. Variable identification. | Random Forest achieved 99% accuracy,  making it suitable for predicting  employee retention, alongside DT. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| SN Applied Sciences  (Springer) | The human resource management data set, which is used in this research work is available online and is free  of cost on kaggle.com. | | 1. Support Vector Machine (SVM)  2. Decision Tree (DT)  3. Random Forest (RF) | 1. Confusion matrix is created,  giving a detailed analysis with reports on the  no. of true positives, false positives, true negatives and false negatives.  2. Precision, Recall and F1 score are included in the classification report. | |
|  | | | | | | |
| **10** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Employee Performance  Prediction using  Naïve Bayes | 2019 | | 1. The target variable is regrouped from seven variable into  two variable in order to avoid the tree branch grow  bigger in the result of the model. The regrouping process  is accomplished using Microsoft Excel 2016 by using  nested if formula to regroup the class into two.  2. Data understanding explore  the data definition and its quality. Data cleaning for  preparing data to be more easy, accurate and  representative is done in data preparation | With the Naïve Bayes method and measure an updated performance score as the objective variable, with 96.77%  accuracy. From the evaluation, the correctly classified instance is 95.48% using the proposed model of Naïve Bayes. This is  shows that the naïve bayes technique is very good at predicting. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International Journal  of Advanced Trends in Computer Science and Engineering | C. Patalano, Human Resources Dataset, Kaggle,  2019. (Dataset from Kaggle) | | Naive Bayes | 1. Confusion matrix is used to evaluate how many instances are wrongly classified as a result of this model.  2. AUROC score was used to judge this model whether it was right or not.  3.10-folds cross-validation, the classification technique that we used in every test is evaluated. | |
|  | | | | | | |
| **11** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Evaluation of Machine Learning Models for  Employee Churn Prediction | 2017 | | 1. The categorical values are converted to numeric values in order to make the classification algorithm more efficient. For  example, categorical attribute ‘salary’ contains three values  such as low, medium and high. Hence it is converted to 0, 1  and 2 respectively 2. The misspelled attributes are also corrected | Random Forest outperformed the other machine learning algorithms (k-NN, SVM, Naïve Bayes, Decision Tree) in terms of predictive accuracy and other relevant metrics. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Proceedings of the International Conference on Inventive Computing and Informatics | HR Analytics dataset from the Kaggle website. | | 1. K-Nearest Neighbors (k-NN) Classifier 2. Support Vector Machine (SVM)  3. Naïve Bayes Classifier  4. Decision Tree (C 5.0) 5. Random Forest | 1. Correlation matrix helps to  identify attributes with the strong or weak correlation. 2. Confusion Matrix is used which is a common evaluation criterion for any classification model. Using parameters like Accuracy, Precision, Recall and F-Measures are used and the corresponding values obtained through experiment is displayed in Table 1 with respect to different learning techniques. | |
|  | | | | | | |
| **12** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | A Data driven analysis of Employee Promotion : The role of the position of organization | 2019 | | 1. The categorical values are converted to numeric values in order to make the classification algorithm more efficient. For  example, categorical attribute ‘salary’ contains three values  such as low, medium and high. Hence it is converted to 0, 1  and 2 respectively 2. The misspelled attributes are also corrected |  | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| IEEE International Conference on Systems, Man and Cybernetics | The data come from a state-owned enterprise in China. It contains personnel information of employees from 2006 to 2016, including gender, university, job title, department and so on. Moreover, we select employees who graduate from higher education colleges and major in science and engineering. Thus, employees in the staff dataset are knowledge staff | | 1. K-Nearest Neighbors (k-NN) Classifier 2. Support Vector Machine (SVM)  3. Naïve Bayes Classifier  4. Decision Tree (C 5.0) 5. Random Forest | 1. Accuracy 2. Area Under the Receiver Operating Characteristic Curve (AUC-ROC) 3. Recall (Sensitivity) 4. Precision | |
|  | | | | | | |
| **13** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Application of Data Mining Classification in Employee  Performance Prediction | 2016 | | 1. Data Cleaning  2. Filling in Missing Values  3. Feature Selection  4. Data Splitting:  The collected performance data was divided into multiple datasets  5. Data Format Conversion:  The data was prepared and converted into ARFF 6. Feature Grouping:  The attributes were grouped into three categories for analysis:  D: Age, Gender, Marital status  E: Qualification, Specialization, Professional Training  P: Job Group, Experience, Salary, Designation, Performance Appraisal Score  7. Feature Scaling | C4.5 algorithm (J4.8) was chosen as the best-suited algorithm for developing the classification model due to its highest classification accuracy rate of 92.60%. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International Journal of Computer Applications | Data was collected from the Human Resource Department at the Kenya School of Government and includes attributes such as experience, age, qualification, gender, marital status, training, and performance appraisal scores. | | 1. ID3 (Iterative Dichotomiser 3) 2. Naïve Bayes Algorithm 3. C4.5 Algorithm (J4.8) | Classification Accuracy: This accuracy indicates the proportion of correctly classified instances out of the total instances in the dataset. | |
|  | | | | | | |
| **14** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Early Prediction of Employee Attrition using Data  Mining Techniques | 2018 | | 1. Data Cleaning: null values have been removed from the dataset.  2. Feature Engineering:  Brute-Force Approach  One-Hot Encoding  Feature Selection 3. Data Splitting 4. Target Label Encoding | Paper doesn't explicitly state the best-performing algorithm, it suggests that Random Forest, Decision Tree, and AdaBoost are the stronger contenders | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| IEEE | The dataset is "Human Resource Attrition dataset" from Kaggle website. | | 1. Logistic Regression 2. Support Vector Machine (SVM)  3. Random Forest  4. Decision Tree  5. AdaBoost  6. Neural Network | 1.Accuracy  2. Precision  3. Recall (Sensitivity or True Positive Rate)  4. F1-Score | |
|  | | | | | | |
| **15** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Early Prediction of Employee Attrition using Data  Mining Techniques | 2018 | |  | SVMs were able to capture more than 0.70  using all the features, and more than 0.60 with just two features | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International Conference on Innovations in Information Technology | The dataset used in the above paper is a synthetic dataset created by IBM Watson Analytics. It contains HR-related data for 1470 employees with 32 features. | | 1. Support Vector Machine (SVM) 2. Random Forest 3. K-Nearest Neighbors (KNN) 4. Adaptive Synthetic (ADASYN) Sampling Approach | 1. Accuracy 2. Precision 3. Recall 4. F1 Score | |
|  | | | | | | |
| **16** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Prediction of Employee promotion based on ratings using machine-learning algorithms | 202 | | 1. Unnecessary features were removed from dataset 2. Temporary employees such as foreign interns were not included in the final set. | Research on predicting employee promotion using machine learning found Gradient Boosting and Random Forest models more consistent than the Keras model, based on performance metrics and tests. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| Bulletin "Physical and Mathematical Sciences" | Custom simulated dataset was used which was derived from real HR dataset of employees collected from 2020-2021. | | 1. Gradient Boosting  2. Random Forest  3. Keras | 1. Confusion matrix is created for each model,  2. Classification matrix is created for each model, with parameters: Precision, Recall, F-1 Score | |
|  | | | | | | |
| **17** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Prediction of Employee Promotion Based on Personal  Basic Features and Post Features | 2018 | | 1. Extract personal basic information data and  position information data from the database 2. Delete missing values and  outliers 3. Some categorical attributes that were not encoded when  stored in the database are recorded with numeric values | Through model learning and testing, it is discovered that the prediction effect of a random forest model is relatively better because its AUC value reaches 0.96, which proves the validity of features. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International Conference on Data Processing and Applications | The staff database of a Chinese state-owned enterprise was used. | | 1. Logistic Regression 2. Support vector classifier 3. Random Forest 4. Adaboost |  | |
|  | | | | | | |
| **18** | **Title** | **Year** | | **Data Preprocessing** | **Feature selected** | |
|  | Predict Employee Retention Using Data Science | 2018 | | 1. Feature Scaling 2. Label Encoding 3. One-Hot Encoding | The paper doesn't state which algorithm is to be preferred. When compared to the data, the accuracy came out to be 97%. | |
| **Journal** | **Dataset** | | **Algorithm** | **Evaluation Matrix** | |
| International Journal of Electrical Electronics & Computer Science Engineering |  | | 1. Linear Regression 2. Polynomial Regression 3. Logistic Regression 4. Lasso Regression | Confusion matrix is created and analyzed but not shown in the paper | |

# **Data Preprocessing (if any):**

# Missing values:

There are 54808 rows in the data set and there were — missing values in education and previous\_year\_rating columns, to deal with these missing values we have used Missforest. MissForest is an imputation algorithm used to fill in missing values in datasets. It's designed to handle both continuous and categorical data and is particularly useful when dealing with complex datasets with a mix of variable types. MissForest is a part of the "missMDA" package in R, and it's based on random forests, a machine learning technique.

Class Imbalance:

The column “is\_promoted” is the output column for the dataset which shows class imbalance.

In 54808 rows 50140 are zero values and 4668 are one value. To deal with the class balance we have the split dataset in 7:3 ratio using an oversampling method of the “ROSE '' library and oversampled it to 70600. rows which are double the number of zeros of 70:30 split dataset.

# **Algorithms Implemented:**

# **Random Forest:**

# In the context of our promotion prediction system, the Random Forest algorithm plays a pivotal role as an ensemble learning method. It excels in providing robust and accurate predictions by aggregating multiple decision trees.Random Forest's strength lies in its ability to mitigate the risk of overfitting while capturing complex relationships within the data.

# **Number of Trees :** The default setting for the number of trees in a Random Forest is 100. In our implementation, we have used this default value, providing a balanced trade-off between computational efficiency and predictive **performance.**

# **Maximum Depth of Trees (max\_depth):** We have set the maximum depth of individual trees to default value.

# **Minimum Samples per Leaf (min\_samples\_leaf):** The default setting of 2 for the minimum number of samples required at a leaf node remains unchanged

# **Number of Features to Consider (max\_features):** We've chosen "auto" as the setting for the maximum number of features to consider at each split. This option automatically selects the square root of the total number of features.

# **Artificial Neural Network:**

# Within our promotion prediction system, the Artificial Neural Network (ANN) emerges as a crucial tool, harnessing the power of deep learning to make precise and sophisticated predictions. ANNs have proven highly effective in capturing intricate patterns and relationships in complex data, making them instrumental in enhancing our decision-making processes.

# **Size:** The ‘size’ hyperparameter determines the number of neurons in the hidden layer of the neural network. We are exploring three different values for "size": 5, 10, and 15.

# **Decay:** The "decay" hyperparameter, often referred to as weight decay or regularization It introduces a penalty for large weights, encouraging the model to have smaller weights, which can enhance generalization to unseen data. We are exploring three different values for "decay": 0.1, 0.01, and 0.001.

# Support Vector Machine:

# In our promotion prediction system, the Support Vector Machine (SVM) stands as a pivotal tool, leveraging its capabilities to provide precise and robust predictions. SVM's adeptness at uncovering intricate patterns in diverse promotional data enhances our decision-making processes, making it an indispensable asset in our analytical toolkit.

# C Hyperparameter: The "C" hyperparameter, also known as the regularization parameter, controls the trade-off between maximizing the margin and minimizing the classification error

# Kernel Hyperparameter: The "kernel" hyperparameter determines the type of kernel function used to transform the data into a higher-dimensional space. Different kernel functions can capture various types of decision boundaries, such as linear, radial basis function (RBF), and polynomials.

# Gamma: The "gamma" hyperparameter controls the shape of the decision boundary in the case of non-linear kernels.

# XGboost**:**

# XGBoost (Extreme Gradient Boosting) is a popular and powerful machine learning algorithm known for its efficiency and performance in various data science and machine learning tasks, particularly in classification and regression problems.

# **Objective Function:** XGBoost uses a user-specified objective function that needs to be optimized during the training process. Common objectives include binary logistic regression for binary classification and mean squared error for regression.

# **eval\_metric:** The evaluation metric to be used. "logloss" is used for logistic regression loss.

# **Max Depth (max\_depth):** Controls the maximum depth of each decision tree. Higher values allow capturing complex patterns but may lead to overfitting. Lower values promote simpler models to prevent overfitting.

# **Learning Rate (eta):** Determines the step size during model training.Higher values make the model learn faster but may overshoot the optimal solution.Lower values lead to cautious learning, potentially achieving better convergence.

# Decision tree:

# A decision tree is a widely used machine learning algorithm that's effective in classification and regression tasks. It's known for its simplicity and interpretability, making it a valuable tool for understanding and solving predictive modeling problems.

# Class method: The “Class” method is used when the target variable (the variable you are trying to predict) is a categorical or discrete variable.

* **Minsplit and Minbucket:**These parameters control the minimum number of data points required to split a node and the minimum number of data points allowed in a leaf node.

# Logistic Regression:

# Logistic regression is a widely used and well-established machine learning algorithm known for its simplicity and interpretability, making it a valuable tool for solving classification problems

# Family hyperparameter: binomial(link = "logit") specifies that the model uses a binomial distribution and applies the logit link function, which is common for logistic regression. Depending on the specific problem and data type, we would choose an appropriate combination of probability distribution and link function by using the family argument/

# **CODE:**

setwd("D:/DS\_R")

data <- read.csv("imputed\_dataset.csv")

library(ROSE)

library(randomForest)

library(caret)

library(e1071)

library(neuralnet)

library(rpart)

library(dplyr)

library(xgboost)

library(irr)

########################## Random forest #####################

cat("\n RESULTS FOR RANDOM FOREST")

cat("\n")

data$is\_promoted <- as.factor(data$is\_promoted)

set.seed(123)

ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))

train <- data[ind == 1, ]

test <- data[ind == 2, ]

over\_sampled\_data <- ovun.sample(is\_promoted ~ ., data = train, method = "over", N = 70600)$data

rf\_over <- randomForest(is\_promoted ~ ., data = over\_sampled\_data, ntree = 100, nodesize = 2)

# K-fold Cross-validation

k <- 5

folds <- createFolds(train$is\_promoted, k = k)

accuracy\_values <- vector("numeric", length = k)

precision\_values <- vector("numeric", length = k)

recall\_values <- vector("numeric", length = k)

f1\_score\_values <- vector("numeric", length = k)

specificity\_values <- vector("numeric", length = k)

sensitivity\_values <- vector("numeric", length = k)

for (i in 1:k) {

validation\_indices <- folds[[i]]

train\_fold <- train[-validation\_indices, ]

validation\_fold <- train[validation\_indices, ]

rf\_fold <- randomForest(is\_promoted ~ ., data = train\_fold)

rf\_predictions <- predict(rf\_fold, newdata = validation\_fold)

confusion\_matrix\_fold <- confusionMatrix(rf\_predictions, validation\_fold$is\_promoted, positive = '1')

accuracy\_values[i] <- confusion\_matrix\_fold$overall['Accuracy']

precision\_values[i] <- confusion\_matrix\_fold$byClass['Precision']

recall\_values[i] <- confusion\_matrix\_fold$byClass['Sensitivity']

f1\_score\_values[i] <- confusion\_matrix\_fold$byClass['F1']

specificity\_values[i] <- confusion\_matrix\_fold$byClass['Specificity']

sensitivity\_values[i] <- confusion\_matrix\_fold$byClass['Sensitivity']

}

mean\_accuracy <- mean(accuracy\_values)

mean\_precision <- mean(precision\_values)

mean\_recall <- mean(recall\_values)

mean\_f1\_score <- mean(f1\_score\_values)

mean\_specificity <- mean(specificity\_values)

mean\_sensitivity <- mean(sensitivity\_values)

confusion\_matrix\_oversampled <- confusionMatrix(predict(rf\_over, newdata = test), test$is\_promoted, positive = '1')

cat("\nConfusion Matrix:\n")

print(confusion\_matrix\_fold$table)

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Mean Accuracy:", mean\_accuracy, "\n")

cat("Mean Precision:", mean\_precision, "\n")

cat("Mean Recall:", mean\_recall, "\n")

cat("Mean F1 Score:", mean\_f1\_score, "\n")

cat("Mean Specificity:", mean\_specificity, "\n")

cat("Mean Sensitivity:", mean\_sensitivity, "\n")

######################## SVM ##########################

cat("\n RESULTS FOR SVM")

cat("\n")

set.seed(42)

data\_size <- floor(0.8 \* nrow(data))

train\_indices <- sample(1:nrow(data), size = data\_size) # Corrected line

train\_data <- data[train\_indices, ]

test\_data <- data[-train\_indices, ]

oversampled\_data <- ovun.sample(is\_promoted ~ ., data = train\_data, method = "over",

N = 70600, seed = 123)$data

k <- 5

folds <- createFolds(oversampled\_data$is\_promoted, k = k)

accuracy\_values <- vector("numeric", length = k)

precision\_values <- vector("numeric", length = k)

recall\_values <- vector("numeric", length = k)

f1\_score\_values <- vector("numeric", length = k)

specificity\_values <- vector("numeric", length = k)

sensitivity\_values <- vector("numeric", length = k)

for (i in 1:k) {

validation\_indices <- folds[[i]]

train\_fold <- oversampled\_data[-validation\_indices, ]

validation\_fold <- oversampled\_data[validation\_indices, ]

svm\_model <- svm(is\_promoted ~ ., data = train\_fold, type = "C-classification", kernel = "linear", cost = 1, gamma = 0.1, scale = TRUE)

svm\_predictions <- predict(svm\_model, validation\_fold)

confusion\_matrix <- table(Actual = validation\_fold$is\_promoted, Predicted = svm\_predictions)

accuracy\_values[i] <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

precision\_values[i] <- confusion\_matrix[2, 2] / sum(confusion\_matrix[, 2])

recall\_values[i] <- confusion\_matrix[2, 2] / sum(confusion\_matrix[2, ])

f1\_score\_values[i] <- 2 \* (precision\_values[i] \* recall\_values[i]) / (precision\_values[i] + recall\_values[i])

specificity\_values[i] <- confusion\_matrix[1, 1] / sum(confusion\_matrix[1, ])

sensitivity\_values[i] <- recall\_values[i]

}

mean\_accuracy <- mean(accuracy\_values)

mean\_precision <- mean(precision\_values)

mean\_recall <- mean(recall\_values)

mean\_f1\_score <- mean(f1\_score\_values)

mean\_specificity <- mean(specificity\_values)

mean\_sensitivity <- mean(sensitivity\_values)

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Mean Accuracy:", mean\_accuracy, "\n")

cat("Mean Precision:", mean\_precision, "\n")

cat("Mean Recall:", mean\_recall, "\n")

cat("Mean F1 Score:", mean\_f1\_score, "\n")

cat("Mean Specificity:", mean\_specificity, "\n")

cat("Mean Sensitivity:", mean\_sensitivity, "\n")

############################## ANN ##################################

cat("\n RESULTS FOR ANN")

cat("\n ")

# Ensure that is\_promoted is a binary factor

data$is\_promoted <- factor(data$is\_promoted, levels = c("0", "1"))

# Normalize input features

preProcessDesc <- preProcess(data[, c("previous\_year\_rating", "length\_of\_service")], method = c("center", "scale"))

data[, c("previous\_year\_rating", "length\_of\_service")] <- predict(preProcessDesc, data[, c("previous\_year\_rating", "length\_of\_service")])

# Define your formula

formula <- as.formula("is\_promoted ~ previous\_year\_rating + length\_of\_service")

# Perform a 70:30 split

set.seed(123)

ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))

train\_data <- data[ind == 1, ]

test\_data <- data[ind == 2, ]

# Create a trainControl object for cross-validation on the training data

ctrl <- trainControl(method = "cv", # k-fold cross-validation

number = 5, # Number of folds (adjust as needed)

verboseIter = TRUE)

# Define the tuning grid with size and decay

tuning\_grid <- expand.grid(size = c(15, 20, 25), decay = c(0.01, 0.001, 0.0001))

# Perform oversampling with ovun.sample on the training data

oversampled\_train\_data <- ovun.sample(is\_promoted ~ ., data = train\_data, method = "over", N = 70600, seed = 123)$data

# Train the neural network model using cross-validation with oversampled training data

set.seed(123)

sink("training\_output.txt")

model <- train(formula,

data = oversampled\_train\_data, # Use oversampled training data

method = "nnet", # Specify "nnet" for classification

trControl = ctrl,

preProcess = c("center", "scale"),

tuneGrid = tuning\_grid, # Specify tuning grid

linout = FALSE, # linear.output = FALSE

lifesign = "full",

stepmax = 100000)

sink()

# Access other performance metrics from the model object

conf\_matrix <- confusionMatrix(predict(model, newdata = test\_data), test\_data$is\_promoted)

cat("\nConfusion Matrix:\n")

print(conf\_matrix$table)

# Extract metrics

accuracy <- conf\_matrix$overall["Accuracy"]

precision <- conf\_matrix$byClass["Precision"]

recall <- conf\_matrix$byClass["Sensitivity"]

f1 <- conf\_matrix$byClass["F1"]

sensitivity <- conf\_matrix$byClass["Sensitivity"]

specificity <- conf\_matrix$byClass["Specificity"]

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Accuracy:", accuracy, "\n")

cat("Precision:", precision, "\n")

cat("Recall:", recall, "\n")

cat("F1 Score:", f1, "\n")

cat("Sensitivity:", sensitivity, "\n")

cat("Specificity:", specificity, "\n")

########################### Decision Tree ###############################

cat("\n")

cat("\n RESULTS FOR DECISION TREE")

cat("\n")

set.seed(123)

data$is\_promoted <- as.factor(data$is\_promoted)

k <- 5

results <- list()

for (i in 1:k) {

folds <- createFolds(data$is\_promoted, k = k, list = TRUE)

train\_indices <- unlist(folds[-i])

test\_indices <- unlist(folds[i])

train\_data <- data[train\_indices, ]

test\_data <- data[test\_indices, ]

oversampled\_data <- ovun.sample(is\_promoted ~ ., data = train\_data, method = "over", N = 70600)$data

dt\_model\_after\_oversampling <- rpart(is\_promoted ~ ., data = oversampled\_data, method = "class", control = rpart.control(minsplit = 10, minbucket = 5))

dt\_predictions\_after\_oversampling <- as.factor(predict(dt\_model\_after\_oversampling, test\_data, type = "class"))

dt\_confusion\_matrix\_after\_oversampling <- confusionMatrix(data = dt\_predictions\_after\_oversampling, reference = test\_data$is\_promoted)

results[[i]] <- dt\_confusion\_matrix\_after\_oversampling

}

cat("\nConfusion Matrix:\n")

print(dt\_confusion\_matrix\_after\_oversampling$table)

mean\_accuracy <- mean(sapply(results, function(result) result$overall["Accuracy"]))

mean\_precision <- mean(sapply(results, function(result) result$byClass["Precision"]))

mean\_f1\_score <- mean(sapply(results, function(result) {

precision <- result$byClass["Precision"]

recall <- result$byClass["Sensitivity"]

2 \* (precision \* recall) / (precision + recall)

}))

mean\_recall <- mean(sapply(results, function(result) result$byClass["Sensitivity"]))

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Mean Accuracy: ", mean\_accuracy, "\n")

cat("Mean Precision: ", mean\_precision, "\n")

cat("Mean F1 Score: ", mean\_f1\_score, "\n")

cat("Mean Recall: ", mean\_recall, "\n")

####################### Logistic Regression ##################################

cat("\n")

cat("\n RESULTS FOR LOGISTIC REGRESSION")

cat("\n")

char\_columns <- c("department", "education", "gender", "recruitment\_channel", "region")

for (col in char\_columns) {

data[[col]] <- as.numeric(factor(data[[col]], levels = unique(data[[col]])))

}

num\_folds <- 5

fold\_metrics\_before <- list()

oversample\_target <- 70600

oversampled\_data <- ROSE(is\_promoted ~ ., data = data, N = oversample\_target)$data

fold\_metrics\_after <- list()

for (fold in 1:num\_folds) {

train\_indices <- createDataPartition(oversampled\_data$is\_promoted, p = 0.75, list = FALSE)

train\_data <- oversampled\_data[train\_indices, ]

test\_data <- oversampled\_data[-train\_indices, ]

logistic\_model <- glm(is\_promoted ~ ., data = train\_data, family = "binomial")

predictions <- predict(logistic\_model, newdata = test\_data, type = "response")

threshold <- 0.5

binary\_predictions <- ifelse(predictions > threshold, 1, 0)

confusion\_matrix <- table(Actual = test\_data$is\_promoted, Predicted = binary\_predictions)

TP <- confusion\_matrix[2, 2]

TN <- confusion\_matrix[1, 1]

FP <- confusion\_matrix[1, 2]

FN <- confusion\_matrix[2, 1]

accuracy <- (TP + TN) / sum(confusion\_matrix)

precision <- TP / (TP + FP)

recall <- TP / (TP + FN)

f1\_score <- 2 \* (precision \* recall) / (precision + recall)

fold\_metrics\_after[[fold]] <- list(

accuracy = accuracy,

precision = precision,

recall = recall,

f1\_score = f1\_score

)

}

cat("\nConfusion Matrix:\n")

print(confusion\_matrix)

avg\_metrics\_after <- sapply(fold\_metrics\_after, function(fold) c(fold$accuracy,

fold$precision, fold$recall, fold$f1\_score))

avg\_metrics\_after <- colMeans(avg\_metrics\_after)

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Average Accuracy:", avg\_metrics\_after[1], "\n")

cat("Average Precision:", avg\_metrics\_after[2], "\n")

cat("Average Recall:", avg\_metrics\_after[3], "\n")

cat("Average F1 Score:", avg\_metrics\_after[4], "\n")

######################### XG Boost #####################################

cat("\n RESULTS FOR XGBoost")

cat("\n")

character\_columns\_to\_convert <- c("employee\_id", "department", "region", "education", "gender", "recruitment\_channel")

data[character\_columns\_to\_convert] <- lapply(data[character\_columns\_to\_convert], as.factor)

data[character\_columns\_to\_convert] <- lapply(data[character\_columns\_to\_convert], as.numeric)

set.seed(123)

xgb\_params <- list(

objective = "binary:logistic",

eval\_metric = "logloss",

eta = 0.01,

max\_depth = 6,

nrounds = 0,

early\_stopping\_rounds = 0

)

response\_variable <- "is\_promoted"

calculate\_metrics <- function(predictions, actual) {

accuracy <- sum(predictions == actual) / length(actual)

conf\_matrix <- table(Actual = actual, Predicted = predictions)

precision <- posPredValue(conf\_matrix)

recall <- sensitivity(conf\_matrix)

f1\_score <- 2 \* (precision \* recall) / (precision + recall)

kappa\_value <- kappa2(as.matrix(conf\_matrix))

return(list(accuracy = accuracy, precision = precision, recall = recall,

f1\_score = f1\_score, kappa = kappa\_value, conf\_matrix = conf\_matrix))

}

train\_model\_and\_evaluate <- function(train\_data, test\_data, xgb\_params, response\_variable) {

x\_train <- as.matrix(train\_data[, !names(train\_data) %in% response\_variable])

y\_train <- as.numeric(train\_data[, response\_variable])

x\_test <- as.matrix(test\_data[, !names(test\_data) %in% response\_variable])

y\_test <- as.numeric(test\_data[, response\_variable])

dtrain <- xgb.DMatrix(data = x\_train, label = y\_train)

xgb\_model <- xgboost(data = dtrain, params = xgb\_params, nthread = -1, nrounds = xgb\_params$nrounds, verbose = 0)

dtest <- xgb.DMatrix(data = x\_test)

xgb\_predictions <- predict(xgb\_model, newdata = dtest)

xgb\_predictions\_binary <- ifelse(xgb\_predictions > 0.5, 1, 0)

metrics <- calculate\_metrics(xgb\_predictions\_binary, y\_test)

return(metrics)

}

num\_folds <- 5

fold\_metrics <- list()

mean\_confusion\_matrix <- matrix(0, nrow = 2, ncol = 2)

mean\_accuracy <- 0

mean\_precision <- 0

mean\_recall <- 0

mean\_f1\_score <- 0

for (fold in 1:num\_folds) {

fold\_indices <- createDataPartition(data$'is\_promoted', p = 0.7, list = FALSE)

train\_data <- data[fold\_indices, ]

test\_data <- data[-fold\_indices, ]

oversampled\_data <- ovun.sample(is\_promoted ~ ., data = train\_data, method = "over", N = 70600)$data

fold\_metrics[[fold]] <- train\_model\_and\_evaluate(oversampled\_data, test\_data, xgb\_params, response\_variable)

mean\_confusion\_matrix <- mean\_confusion\_matrix + fold\_metrics[[fold]]$conf\_matrix

mean\_accuracy <- mean\_accuracy + fold\_metrics[[fold]]$accuracy

mean\_precision <- mean\_precision + fold\_metrics[[fold]]$precision

mean\_recall <- mean\_recall + fold\_metrics[[fold]]$recall

mean\_f1\_score <- mean\_f1\_score + fold\_metrics[[fold]]$f1\_score

}

mean\_confusion\_matrix <- mean\_confusion\_matrix / num\_folds

cat("\nConfusion Matrix:\n")

print(mean\_confusion\_matrix)

mean\_accuracy <- mean\_accuracy / num\_folds

mean\_precision <- mean\_precision / num\_folds

mean\_recall <- mean\_recall / num\_folds

mean\_f1\_score <- mean\_f1\_score / num\_folds

cat("\nMEAN METRICS ACROSS ALL FOLDS\n")

cat("Mean Accuracy:", mean\_accuracy, "\n")

cat("Mean Precision:", mean\_precision, "\n")

cat("Mean Recall:", mean\_recall, "\n")

cat("Mean F1 Score:", mean\_f1\_score, "\n")

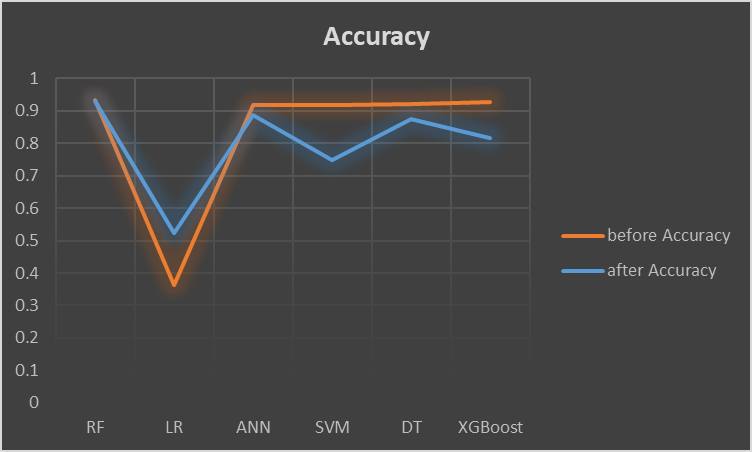
# **Evaluation Parameters used:**

# The evaluation parameters we used are accuracy, precision, recall, f1 score and kappa.

# **Results and Discussions:**

# We implemented 6 algorithms which were, Random Forest, Support Vector Machine, XGboost, ANN (Artificial Neural Networks), Decision Tree, and Logistic regression.

# The Accuracies of all the algorithms compared is as follows:



# **12. Conclusion:**

In conclusion, the analysis and preprocessing of the dataset, consisting of 54,809 rows and 13 columns, revealed several critical aspects. The dataset contained missing values, primarily in the “previous\_year\_rating” and “education” field, which had numerical as well as character data type. The MissForest algorithm was used to deal with these missing values, and the missing values in the “education” column were dealt with using the mode operation.

One of the most prominent challenges occurred in the analysis of class imbalance within the “is\_promoted” column. The dataset displayed a significant disparity between the number of 0 values, indicating non-promotions, and the number of 1 values as promotions. Specifically, there were 50,140 instances of non-promotion and 4,668 instances of promotion, leading to a class imbalance. We used a systematic oversampling technique, from the ROSE library, to effectively rebalance the dataset by generating synthetic instances, doubling the number of non-promotion instances to 70,600 rows. By doing so this problem was mitigated.

We used 6 algorithms RF, SVM, DT, XGBoost, ANN, and LR. We used them once without over sampling and once with to check the difference so as to which gives more accurate results. The respective values of accuracies for both the cases is as follows:

| **Algorithm** | **Before Oversampling Accuracy** | **After Oversampling Accuracy** | **Kappa** |
| --- | --- | --- | --- |
| **RF** | 0.9339 | 0.9302 | 0.3747 |
| **SVM** | 0.9183 | 0.7471 | 0.2521 |
| **ANN** | 0.9183 | 0.8869 | 0.1425 |
| **DT** | 0.9210 | 0.8751 | 0.2923 |
| **LR** | 0.3639 | 0.5226 | 0.2561 |
| **XGBoost** | 0.9286 | 0.8172 | 0.1321 |

From this we can conclude that the algorithm which works the best with this dataset to predict the promotion is Random Forest.

# **References:**

[1] K. Şahinbaş, "Employee Promotion Prediction by using Machine Learning Algorithms for Imbalanced Dataset," presented at the 2nd International Conference on Computing and Machine Intelligence (ICMI), 2022.

[2] S. bin Shafie, "PREDICTION OF EMPLOYEE PROMOTION USING HYBRID SAMPLING METHOD WITH MACHINE LEARNING ARCHITECTURE," Malaysian Journal of Computing, 2023.

[3] I.L. Emmanuel-Okereke, "KNN and SVM Machine Learning to Predict Staff Due for Promotion and Training," The International Journal of Engineering and Science (IJES), 2022.

[4] Mr. Shaik Waris Basha, Mrs. K. JAYASRI, "Predicting Employee Promotion Using Machine Learning," Journal of Emerging Technologies and Innovative Technologies, 2023.

[5] MGT Li, "Employee Performance Prediction using different supervised classifiers," International Conference on Industrial Engineering and Operations Management, 2021.

[6] PK Jain, "Explaining and Predicting employees' attrition: a machine learning approach," SN Applied Sciences (Springer), 2020.

[7] Riyanto Jayadi, Hafizh M. Firmantyo, Muhammad T. J. Dzaka, Muhammad F. Suaidy, Alfitra M. Putra, "Employee Performance Prediction using Naïve Bayes," International Journal of Advanced Trends in Computer Science and Engineering, 2019.

[8] Dilip Singh Sisodia, Somdutta Vishwakarma, Abinash Pujahari, "Evaluation of Machine Learning Models for Employee Churn Prediction," Proceedings of the International Conference on Inventive Computing and Informatics, 2017.

[9] J. Liu, T. Wang, J. Li, J. Huang, F. Yao and R. He, "A Data-driven Analysis of Employee Promotion: The Role of the Position of Organization," 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 2019, pp. 4056-4062, doi: 10.1109/SMC.2019.8914449.

[10] John M., Christopher A, "Application of Data Mining Classification in Employee Performance Prediction," International Journal of Computer Applications, 2016.

[11] S. Yadav, A. Jain and D. Singh, "Early Prediction of Employee Attrition using Data Mining Techniques," 2018 IEEE 8th International Advance Computing Conference (IACC), Greater Noida, India, 2018, pp. 349-354, doi: 10.1109/IADCC.2018.8692137.

[12] S. S. Alduayj and K. Rajpoot, "Predicting Employee Attrition using Machine Learning," 2018 International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, 2018, pp. 93-98, doi: 10.1109/INNOVATIONS.2018.8605976.

[13]Жанузаков, М. Б., and Г. Т. Балакаева. "PREDICTION OF EMPLOYEE PROMOTION BASED ON RATINGS USING MACHINE-LEARNING ALGORITHMS." Вестник «Физико-математические науки» 77.1 (2022): 106-111.

[14] Y Long, "Prediction of Employee Promotion Based on Personal Basic Features and Post

Features," International Conference on Data Processing and Applications, 2018.

[15] Dr Anil Kumar Dubey, ABES Engineering College, Ila Maheshwari, Ashutosh Mishra, "Predict Employee Retention Using Data Science," International Journal of Electrical Electronics & Computer Science Engineering, 2018.