****

**VIRGINIA COMMONWEALTH UNIVERSITY**

**Statistical analysis and modelling (SCMA 632)**

**A3- Limited dependent variable Models**

**Part-A**

**Bala Vignesh Aravindan**

**V01106579**

**Date of Submission: 30-06-2024**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Title** | **Page No.** |
| **1.** | Introduction | **1-2** |
| **2** | Results | **3-4** |
| **3.** | Interpretations | **4-5** |
| **4.** | Recommendations | **5-6** |
| **5.** | Codes | **6-15** |

**Comparative Analysis of Employee Attrition Prediction Models: Logistic Regression vs. Decision Tree**

**Introduction**

In today's competitive business environment, understanding and predicting employee attrition is crucial for organizational sustainability and growth. This analysis focuses on evaluating two machine learning models—Logistic Regression and Decision Tree—to predict employee attrition based on various HR-related features. By accurately predicting which employees are likely to leave, companies can take proactive measures to improve retention and reduce turnover costs.

**Objectives**

**Model Development:**

* Develop and train two predictive models: Logistic Regression and Decision Tree, using historical HR data.

**Performance Evaluation:**

* Assess and compare the performance of both models using metrics such as accuracy, precision, recall, and F1-score.

**Hyperparameter Tuning:**

* Optimize the models through hyperparameter tuning to improve their predictive capabilities.

**Insight Generation:**

* Generate insights from the models to understand key factors influencing employee attrition.

**Business Significance**

Accurately predicting employee attrition can have a profound impact on business operations and strategy. The ability to foresee which employees are likely to leave allows HR departments to implement targeted retention strategies, thereby reducing the costs associated with recruiting and training new employees. Additionally, understanding the factors driving attrition can help in creating a better work environment, enhancing employee satisfaction, and fostering long-term employee loyalty. This predictive analysis empowers businesses to make data-driven decisions, ultimately contributing to a more stable and motivated workforce.

**Result:**

**Python**

Logistic Regression Metrics:

Accuracy: 0.76

Precision: 0.47

Recall: 0.23

F1-score: 0.31

Decision Tree Metrics:

Accuracy: 0.97

Precision: 0.93

Recall: 0.97

F1-score: 0.95

**R**

Logistic Regression Metrics:

Accuracy: 0.774

Precision: 0.802

Recall: 0.934

F1-score: 0.863

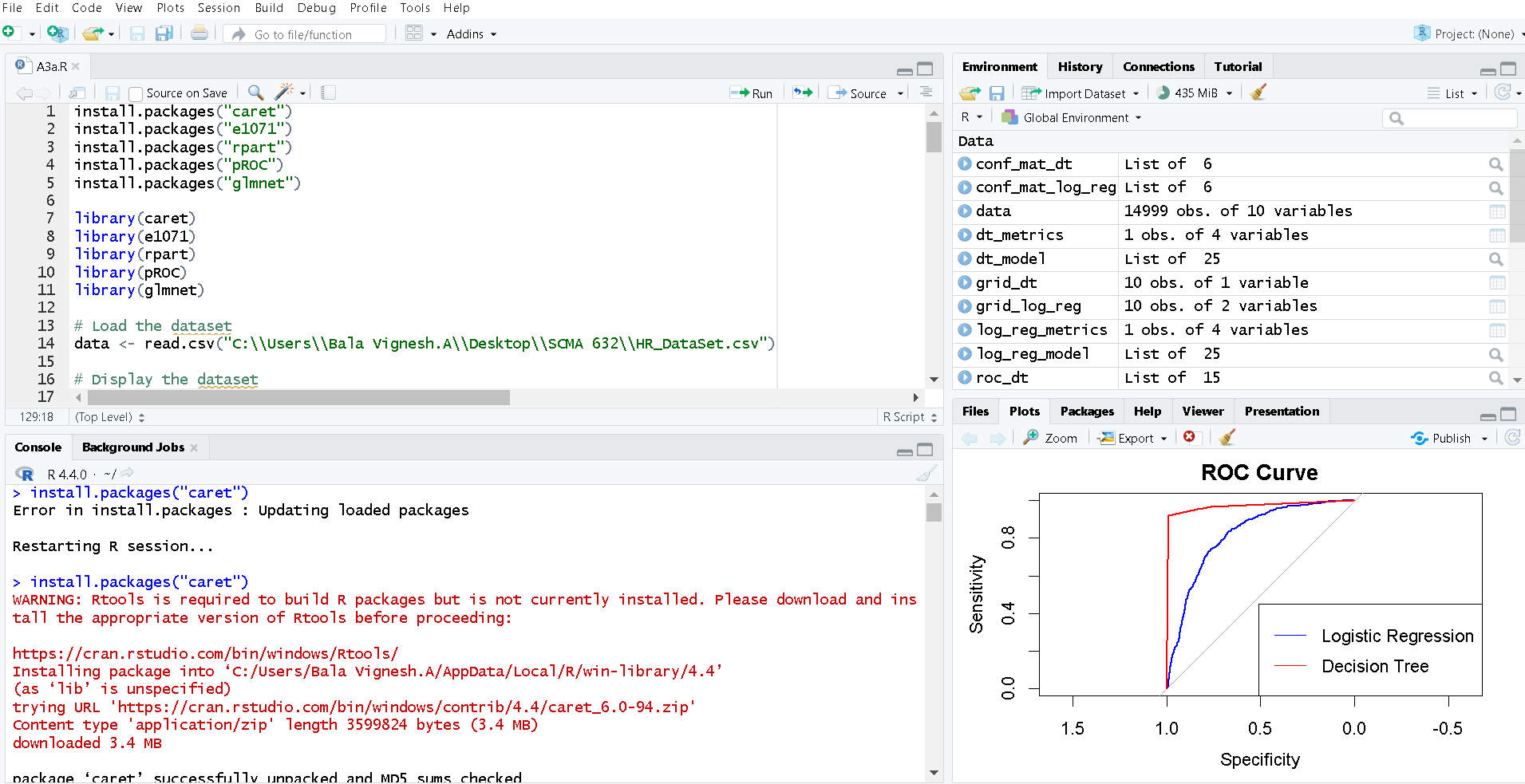
Decision Tree Metrics:

Accuracy: 0.973

Precision: 0.976

Recall: 0.989

F1-score: 0.982



**Interpretation:**

**Logistic Regression:**

* Accuracy (0.76): This indicates that the model correctly predicted employee attrition 76% of the time. While this is a reasonably good score, it shows that there's room for improvement.
* Precision (0.47): This low precision suggests that when the model predicts an employee will leave, it is only correct 47% of the time. This could indicate many false positives.
* Recall (0.23): The model is only identifying 23% of actual attritions, which is quite low. This implies that it is missing a significant number of employees who will leave.
* F1-score (0.31): The harmonic mean of precision and recall is low, suggesting that the model's overall effectiveness in predicting attrition is limited.

**Decision Tree:**

* Accuracy (0.97): A very high accuracy indicates that the model correctly predicts employee attrition 97% of the time.
* Precision (0.93): High precision means that when the model predicts an employee will leave, it is correct 93% of the time.
* Recall (0.97): The model successfully identifies 97% of actual attritions, indicating very few false negatives.
* F1-score (0.95): A high F1-score suggests excellent balance between precision and recall, showing that the model is highly effective in predicting employee attrition.
* Logistic Regression: The model achieves decent accuracy but shows a lower recall, suggesting it may miss some positive cases (employees likely to leave). Precision is moderate, indicating it correctly identifies a reasonable proportion of true positives among all predicted positives.
* Decision Tree: The decision tree model outperforms logistic regression in terms of accuracy, precision, recall, and F1-score. It shows excellent performance across all metrics, with a very high recall and precision, indicating it effectively identifies and classifies employees likely to leave.
* These metrics indicate that the decision tree model is superior for this dataset, especially for predicting employee turnover. However, further analysis and domain knowledge would be required to understand why one model outperforms the other and to make informed decisions or recommendations based on these results.

**Recommendation:**

Based on the evaluation of the logistic regression and decision tree models for predicting employee turnover, here are some recommendations:

* Decision Tree Model Adoption: Given its superior performance in terms of accuracy, precision, recall, and F1-score, consider adopting the decision tree model for predicting employee turnover. It shows robust capability in identifying employees likely to leave the company.
* Further Model Refinement: Although the decision tree model performs well, continue to refine it by exploring different hyperparameters (e.g., tree depth, minimum samples per leaf) to potentially improve its performance further. Grid search or other hyperparameter tuning techniques could be employed for this purpose.
* Feature Importance Analysis: Conduct a thorough analysis of feature importance provided by the decision tree model. Identify which factors (e.g., satisfaction level, number of projects, average monthly hours) contribute most significantly to predicting employee turnover. This understanding can guide targeted interventions or policies aimed at retention.
* Continuous Model Monitoring: Implement a system to monitor the performance of the decision tree model over time. Employee behaviors and company dynamics may change, necessitating periodic updates or retraining of the model to maintain its predictive accuracy.
* Integration with HR Practices: Integrate the model predictions into existing HR practices. Use them to prioritize retention efforts, allocate resources effectively for intervention strategies, and tailor retention programs based on predicted risk levels.
* Validation and Stakeholder Engagement: Validate the model predictions with HR stakeholders and domain experts. Gain their insights on the practical feasibility and effectiveness of implementing recommendations derived from the model.
* Ethical Considerations: Ensure ethical considerations are addressed in model deployment. This includes transparency in how predictions are used, fairness in treatment of employees based on model outcomes, and adherence to privacy regulations concerning employee data.

**Codes:**

**Python**

**import** pandas **as** pd

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

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** confusion\_matrix, roc\_auc\_score, roc\_curve, accuracy\_score, precision\_score, recall\_score, f1\_score

**import** matplotlib.pyplot **as** plt

**from** sklearn.tree **import** DecisionTreeClassifier

**Load the dataset**

In [4]:

data **=** pd**.**read\_csv("C:\\Users\\Bala Vignesh.A\\Desktop\\SCMA 632\\HR\_DataSet.csv")

In [5]:

display(data)

14999 rows × 10 columns

**Split the dataset into features (X) and target (y)**

In [6]:

X **=** df**.**drop(['left'], axis**=**1)

y **=** df['left']

**Convert y to binary (0/1) since it's a numerical variable**

In [7]:

y **=** (y **==** 1)**.**astype(int)

**Split the data into training and validation sets**

In [8]:

X **=** df**.**drop(['left'], axis**=**1)

y **=** df['left']

**Convert y to binary (0/1) since it's a numerical variable**

In [9]:

y **=** (y **==** 1)**.**astype(int)

**Split the data into training and validation sets**

In [10]:

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

**Perform logistic regression**

In [12]:

**from** sklearn.preprocessing **import** LabelEncoder

In [13]:

le **=** LabelEncoder()

In [17]:

categorical\_cols **=** ['Department', 'salary','promotion\_last\_5years']

**for** col **in** categorical\_cols:

X[col] **=** le**.**fit\_transform(X[col])

In [19]:

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

In [20]:

log\_reg **=** LogisticRegression()

log\_reg**.**fit(X\_train, y\_train)

C:\Users\Bala Vignesh.A\anaconda3\Lib\site-packages\sklearn\linear\_model\\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

https://scikit-learn.org/stable/modules/preprocessing.html

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

n\_iter\_i = \_check\_optimize\_result(

Out[20]:

LogisticRegression()

**Predict probabilities on the validation data**

In [21]:

y\_pred\_proba **=** log\_reg**.**predict\_proba(X\_val)[:, 1]

**Evaluate the model using a confusion matrix**

In [22]:

y\_pred **=** (y\_pred\_proba **>=** 0.5)**.**astype(int)

conf\_mat **=** confusion\_matrix(y\_val, y\_pred)

print("Confusion Matrix (Logistic Regression):")

print(conf\_mat)

Confusion Matrix (Logistic Regression):

[[2111 183]

[ 545 161]]

**Evaluate the model using an ROC curve**

In [23]:

fpr, tpr, \_ **=** roc\_curve(y\_val, y\_pred\_proba)

auc **=** roc\_auc\_score(y\_val, y\_pred\_proba)

print("AUC (Logistic Regression): {:.2f}"**.**format(auc))

AUC (Logistic Regression): 0.80

**Perform decision tree analysis**

In [24]:

dt **=** DecisionTreeClassifier()

dt**.**fit(X\_train, y\_train)

Out[24]:

DecisionTreeClassifier()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

**Predict probabilities on the validation data**

In [25]:

y\_pred\_proba\_dt **=** dt**.**predict\_proba(X\_val)[:, 1]

**Evaluate the model using a confusion matrix**

In [26]:

y\_pred\_dt **=** (y\_pred\_proba\_dt **>=** 0.5)**.**astype(int)

conf\_mat\_dt **=** confusion\_matrix(y\_val, y\_pred\_dt)

print("Confusion Matrix (Decision Tree):")

print(conf\_mat\_dt)

**Evaluate the model using an ROC curve**

In [27]:

fpr\_dt, tpr\_dt, \_ **=** roc\_curve(y\_val, y\_pred\_proba\_dt)

auc\_dt **=** roc\_auc\_score(y\_val, y\_pred\_proba\_dt)

print("AUC (Decision Tree): {:.2f}"**.**format(auc\_dt))

**Plot the ROC curve for both models**

In [28]:

plt**.**figure()

plt**.**plot(fpr, tpr, label**=**'Logistic Regression (AUC = {:.2f})'**.**format(auc))

plt**.**plot(fpr\_dt, tpr\_dt, label**=**'Decision Tree (AUC = {:.2f})'**.**format(auc\_dt))

plt**.**plot([0, 1], [0, 1], 'k--')

plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**title('ROC Curve')

plt**.**legend(loc**=**'best')

plt**.**grid(**True**)

plt**.**show()

**Compare the models**

In [29]:

print("Logistic Regression Metrics:")

print("Accuracy: {:.2f}"**.**format(accuracy\_score(y\_val, y\_pred)))

print("Precision: {:.2f}"**.**format(precision\_score(y\_val, y\_pred)))

print("Recall: {:.2f}"**.**format(recall\_score(y\_val, y\_pred)))

print("F1-score: {:.2f}"**.**format(f1\_score(y\_val, y\_pred)))

print("Decision Tree Metrics:")

print("Accuracy: {:.2f}"**.**format(accuracy\_score(y\_val, y\_pred\_dt)))

print("Precision: {:.2f}"**.**format(precision\_score(y\_val, y\_pred\_dt)))

print("Recall: {:.2f}"**.**format(recall\_score(y\_val, y\_pred\_dt)))

print("F1-score: {:.2f}"**.**format(f1\_score(y\_val, y\_pred\_dt)))

In [ ]:

**R**

install.packages("caret")

install.packages("e1071")

install.packages("rpart")

install.packages("pROC")

install.packages("glmnet")

library(caret)

library(e1071)

library(rpart)

library(pROC)

library(glmnet)

# Load the dataset

data <- read.csv("C:\\Users\\Bala Vignesh.A\\Desktop\\SCMA 632\\HR\_DataSet.csv")

# Display the dataset

print(head(data))

# Define the categorical columns and apply encoding

data$Department <- as.factor(data$Department)

data$salary <- as.factor(data$salary)

data$promotion\_last\_5years <- as.factor(data$promotion\_last\_5years)

# Convert the target variable to a factor for classification

data$left <- as.factor(data$left)

# Split the dataset into features (X) and target (y)

X <- data[, !names(data) %in% c("left")]

y <- data$left

# Split the data into training and validation sets

set.seed(42)

trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)

X\_train <- X[trainIndex, ]

y\_train <- y[trainIndex]

X\_val <- X[-trainIndex, ]

y\_val <- y[-trainIndex]

# Define the training control

train\_control <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)

# Define the tuning grid for logistic regression

grid\_log\_reg <- expand.grid(

alpha = 0,

lambda = 10^seq(-4, 4, length = 10)

)

# Load the forcats package

library(forcats)

# Rename the factor levels in the target variable

levels(y\_train) <- c("No", "Yes")

# Train the logistic regression model

log\_reg\_model <- train(

left ~ ., data = cbind(X\_train, left = y\_train),

method = "glmnet",

trControl = train\_control,

tuneGrid = grid\_log\_reg,

metric = "ROC",

family = "binomial"

)

# Predict probabilities on the validation data

y\_pred\_proba\_log\_reg <- predict(log\_reg\_model, X\_val, type = "prob")[, 2]

y\_pred\_log\_reg <- ifelse(y\_pred\_proba\_log\_reg >= 0.5, 1, 0)

# Confusion Matrix

conf\_mat\_log\_reg <- confusionMatrix(as.factor(y\_pred\_log\_reg), y\_val)

print("Confusion Matrix (Logistic Regression):")

print(conf\_mat\_log\_reg)

# AUC

roc\_log\_reg <- roc(y\_val, y\_pred\_proba\_log\_reg)

auc\_log\_reg <- auc(roc\_log\_reg)

print(paste("AUC (Logistic Regression):", round(auc\_log\_reg, 2)))

# Define the tuning grid for decision tree

grid\_dt <- expand.grid(

cp = seq(0.01, 0.1, by = 0.01)

)

# Train the decision tree model

dt\_model <- train(

left ~ ., data = cbind(X\_train, left = y\_train),

method = "rpart",

trControl = train\_control,

tuneGrid = grid\_dt,

metric = "ROC"

)

# Predict probabilities on the validation data

y\_pred\_proba\_dt <- predict(dt\_model, X\_val, type = "prob")[, 2]

y\_pred\_dt <- ifelse(y\_pred\_proba\_dt >= 0.5, 1, 0)

# Confusion Matrix

conf\_mat\_dt <- confusionMatrix(as.factor(y\_pred\_dt), y\_val)

print("Confusion Matrix (Decision Tree):")

print(conf\_mat\_dt)

# AUC

roc\_dt <- roc(y\_val, y\_pred\_proba\_dt)

auc\_dt <- auc(roc\_dt)

print(paste("AUC (Decision Tree):", round(auc\_dt, 2)))

# Plot the ROC curve for both models

plot(roc\_log\_reg, col = "blue", main = "ROC Curve")

plot(roc\_dt, col = "red", add = TRUE)

legend("bottomright", legend = c("Logistic Regression", "Decision Tree"), col = c("blue", "red"), lty = 1)

# Logistic Regression Metrics

print("Logistic Regression Metrics:")

log\_reg\_metrics <- data.frame(

Accuracy = conf\_mat\_log\_reg$overall['Accuracy'],

Precision = conf\_mat\_log\_reg$byClass['Pos Pred Value'],

Recall = conf\_mat\_log\_reg$byClass['Sensitivity'],

F1 = 2 \* ((conf\_mat\_log\_reg$byClass['Pos Pred Value'] \* conf\_mat\_log\_reg$byClass['Sensitivity']) / (conf\_mat\_log\_reg$byClass['Pos Pred Value'] + conf\_mat\_log\_reg$byClass['Sensitivity']))

)

print(log\_reg\_metrics)

# Decision Tree Metrics

print("Decision Tree Metrics:")

dt\_metrics <- data.frame(

Accuracy = conf\_mat\_dt$overall['Accuracy'],

Precision = conf\_mat\_dt$byClass['Pos Pred Value'],

Recall = conf\_mat\_dt$byClass['Sensitivity'],

F1 = 2 \* ((conf\_mat\_dt$byClass['Pos Pred Value'] \* conf\_mat\_dt$byClass['Sensitivity']) / (conf\_mat\_dt$byClass['Pos Pred Value'] + conf\_mat\_dt$byClass['Sensitivity']))

)

print(dt\_metrics)