Employee Attrition Prediction

CSP571 final report

Avinash Sankar
asankar@hawk.iit.edu
A20525471

Jaichiddharth R. Karthigeyan
jkarthigeyan@hawk.iit.edu
A20527281

Abstract

Organizations are very concerned about employee attrition because it leads to the loss of important talent, increased expenses, and diminished productivity. Organizations may take proactive steps to keep their best performers and lower attrition rates by using employee attrition prediction. In this project, we created prediction models for employee attrition using logistic regression, decision trees, random forests, and support vector machines. Using a variety of performance criteria, we assessed the models' performance and determined which model was most effective at predicting employee attrition.

1. Introduction:

Employee attrition, commonly referred to as staff turnover, is a significant issue for businesses all over the world. Employee attrition, which may lead to the loss of important talent, higher expenses, and lower productivity, happens when employees leave a company either voluntarily or involuntarily. Employee turnover rates are rising, and businesses are having trouble keeping their most talented workers, according to a survey done by the international HR consulting firm Mercer. It can be difficult to predict employee attrition since it can be impacted by a number of variables, including job satisfaction, pay, work-life balance, and possibilities for professional advancement. However, employers may take proactive steps to keep their best performers and lower attrition rates by identifying the important drivers of employee loss and creating precise prediction models. With the use of logistic regression, decision trees, random forests, and support vector machines, we want to create prediction models for employee attrition in this project using the R programming language. To create these models, we will utilize a publicly accessible dataset that includes details on employees, such as their age, monthly salary, level of job satisfaction, and workplace.

The main goal of the experiment is to assess these models' performance using a variety of performance measures, including accuracy, precision, recall, F1 score, and AUC. In order to determine the most significant determinants of employee turnover, we will also do the feature selection. Organizations may take proactive steps to keep their valued personnel and lower retention rates by identifying the most crucial variables and creating precise prediction models.
The procedures for gathering and preparing data, creating and assessing models, and outlining the findings of our study are all covered in the sections that follow. Finally, we will go through the implications of our analysis and provide some suggestions for possible future areas of exploration.

2. Data collection:

Any project involving data analysis must begin with data collecting. We used the IBM HR Analytics Employee Attrition & Performance dataset for this research. The prominent data science community site Kaggle is where the dataset was collected.

1. Finding the Dataset: The first step in our investigation was to locate the right dataset. On Kaggle, we looked for employee attrition-related datasets, and there we discovered the IBM HR Analytics Employee Attrition & Performance dataset.

2. Dataset Downloading: We located the dataset and downloaded it from the Kaggle website. The dataset was offered in CSV format, making it simple to import it into R.

3. Exploring the Dataset: We looked at the dataset to better understand the variables and their correlations before moving forward with data pretreatment and analysis. To study the data, we employed visualization strategies and summary statistics. Recognizing the Variables- The dataset consists of 35 variables, including statistics on performance, job satisfaction, and demographics of the workforce. To comprehend the dataset better, we looked at the variable descriptions and their definitions.

3. Data preprocessing:

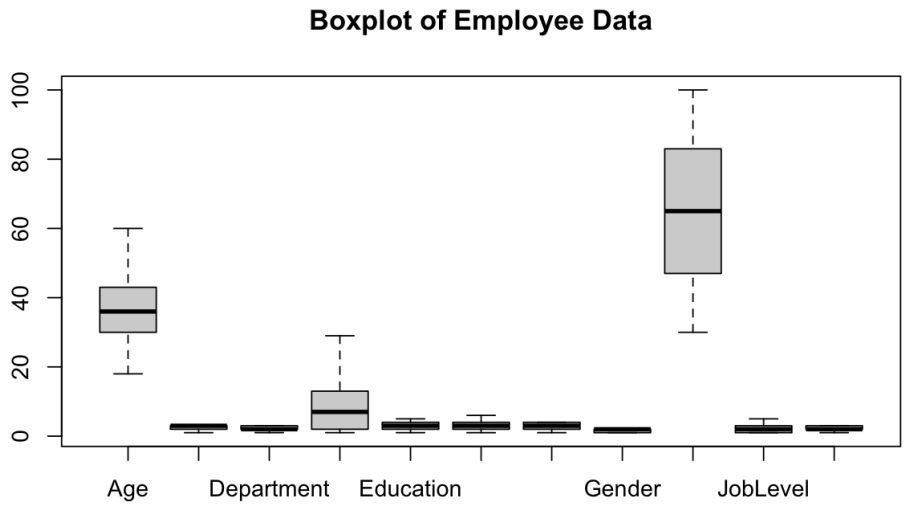
Any project involving data analysis must start with data preprocessing. For this research, we underwent several preparation procedures to get the dataset ready for analysis.

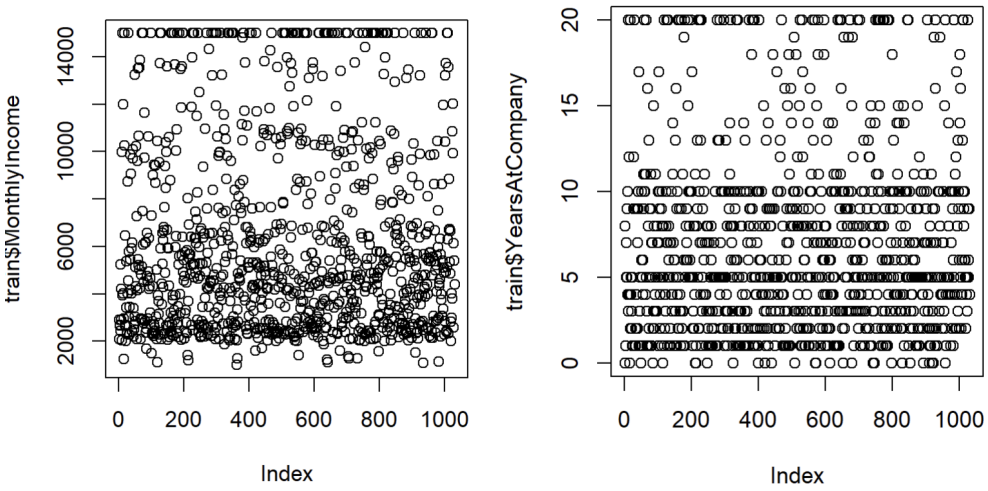
1. Removing Unnecessary Variables: We started by getting rid of any variables that weren't important for our analysis. As a result, the dataset's complexity was decreased, and the classification models' performance was enhanced.

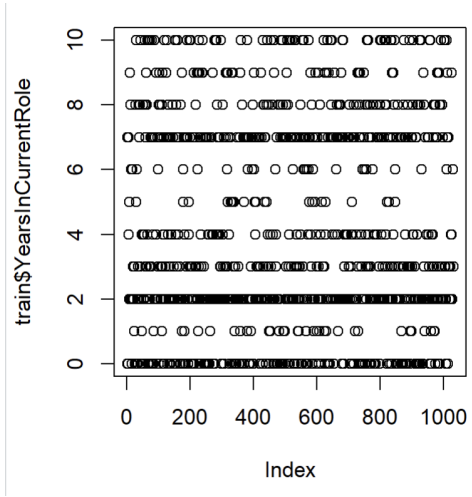
```
#Removing unnecessary variables
employee_data <- select(employee_data,-c(EmployeeNumber, Over18, StandardHours, EmployeeCount))

#Checking for missing values
sum(is.na(employee_data))
```

```
[1] 0
```

2. Identifying Missing Values: We looked for any missing values in the dataset. Since missing values might affect the analysis, we must manage them correctly. To find any missing values in the dataset, we utilized R's is. na () function.
3. Looking for Outliers: We looked for any outliers in the dataset. We had to recognize the outliers and deal with them correctly because they could affect the analysis. To find any outliers in the dataset, we employed summary statistics and box plots.



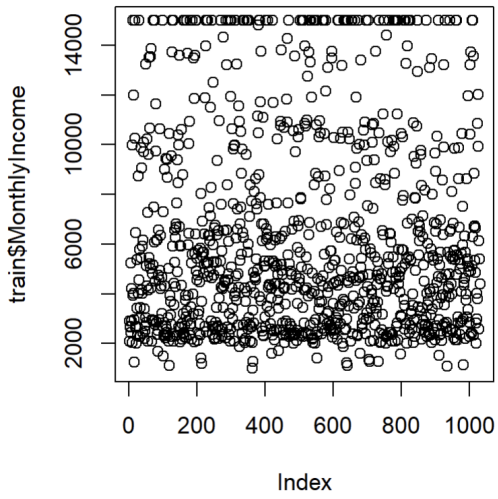


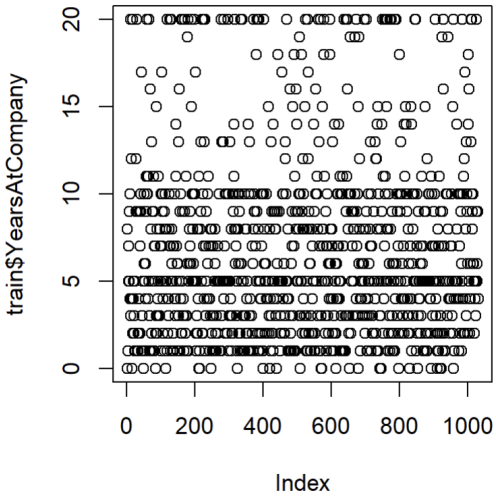


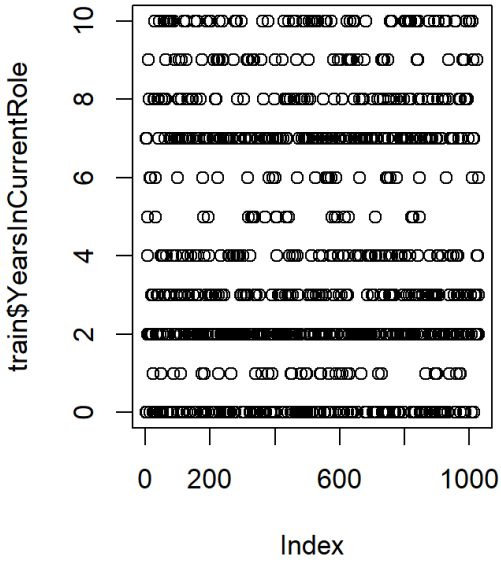
4. Handling Outliers: After locating the outliers, we took the proper action. For numerical variables, we made the decision to substitute the median for the outliers. As a result, the influence of the outliers on the analysis was not observed as there are no outliers.

```
#Handling outliers
train$MonthlyIncome[train$MonthlyIncome > 15000] <- 15000
plot(train$MonthlyIncome)

train$YearsAtCompany[train$YearsAtCompany > 20] <- 20
plot(train$YearsAtCompany)
train$YearsInCurrentRole[train$YearsInCurrentRole > 10] <- 10
plot(train$YearsInCurrentRole)
```

5.







6. Converting Categorical Variables into Factors: We changed categorical variables into factors to improve the analysis of the data by the classification models. In R, categorical data are represented by factors, a form of data. We made it simpler for the models to interact with the data by transforming categorical variables into factors.

```r
#Converting categorical variables to factors
employee_data$Attrition <- as.factor(employee_data$Attrition)
employee_data$BusinessTravel <- as.factor(employee_data$BusinessTravel)
employee_data$Department <- as.factor(employee_data$Department)
employee_data$EducationField <- as.factor(employee_data$EducationField)
employee_data$Gender <- as.factor(employee_data$Gender)
employee_data$JobRole <- as.factor(employee_data$JobRole)
employee_data$MaritalStatus <- as.factor(employee_data$MaritalStatus)
employee_data$OverTime <- as.factor(employee_data$OverTime)
str(employee_data)
```

```
tibble [1,470 × 31] (S3: tbl_df/tbl/data.frame)
 $ Age                     : num [1:1470] 41 49 37 33 27 32 59 30 38 36 ...
 $ Attrition               : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1
1 ...
 $ BusinessTravel          : Factor w/ 3 levels "Non-
Travel","Travel_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...
 $ DailyRate               : num [1:1470] 1102 279 1373 1392 591 ...
 $ Department              : Factor w/ 3 levels "Human Resources",..: 3 2 2 2
2 2 2 2 2 2 ...
 $ DistanceFromHome        : num [1:1470] 1 8 2 3 2 2 3 24 23 27 ...
 $ Education               : num [1:1470] 2 1 2 4 1 2 3 1 3 3 ...
 $ EducationField          : Factor w/ 6 levels "Human Resources",..: 2 2 5 2
4 2 4 2 2 4 ...
 $ EnvironmentSatisfaction : num [1:1470] 2 3 4 4 1 4 3 4 4 3 ...
 $ Gender                  : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1
2 2 2 ...
```

```
 $ HourlyRate              : num [1:1470] 94 61 92 56 40 79 81 67 44 94 ...
 $ JobInvolvement          : num [1:1470] 3 2 2 3 3 3 4 3 2 3 ...
 $ JobLevel                : num [1:1470] 2 2 1 1 1 1 1 1 3 2 ...
 $ JobRole                 : Factor w/ 9 levels "Healthcare
Representative",..: 8 7 3 7 3 3 3 3 5 1 ...
 $ JobSatisfaction         : num [1:1470] 4 2 3 3 2 4 1 3 3 3 ...
 $ MaritalStatus           : Factor w/ 3 levels "Divorced","Married",..: 3 2 3
2 2 3 2 1 3 2 ...
 $ MonthlyIncome           : num [1:1470] 5993 5130 2090 2909 3468 ...
 $ MonthlyRate             : num [1:1470] 19479 24907 2396 23159 16632 ...
 $ NumCompaniesWorked      : num [1:1470] 8 1 6 1 9 0 4 1 0 6 ...
 $ OverTime                : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1
1 ...
 $ PercentSalaryHike       : num [1:1470] 11 23 15 11 12 13 20 22 21 13 ...
 $ PerformanceRating       : num [1:1470] 3 4 3 3 3 3 4 4 4 3 ...
 $ RelationshipSatisfaction: num [1:1470] 1 4 2 3 4 3 1 2 2 2 ...
 $ StockOptionLevel        : num [1:1470] 0 1 0 0 1 0 3 1 0 2 ...
 $ TotalWorkingYears       : num [1:1470] 8 10 7 8 6 8 12 1 10 17 ...
 $ TrainingTimesLastYear   : num [1:1470] 0 3 3 3 3 2 3 2 2 3 ...
 $ WorkLifeBalance         : num [1:1470] 1 3 3 3 3 2 2 3 3 2 ...
 $ YearsAtCompany          : num [1:1470] 6 10 0 8 2 7 1 1 9 7 ...
 $ YearsInCurrentRole      : num [1:1470] 4 7 0 7 2 7 0 0 7 7 ...
 $ YearsSinceLastPromotion : num [1:1470] 0 1 0 3 2 3 0 0 1 7 ...
 $ YearsWithCurrManager    : num [1:1470] 5 7 0 0 2 6 0 0 8 7 ...
```

7. Scaling Numerical Variables: In order to make sure that the numerical variables were on the same scale, we scaled them. Scaling is a typical machine learning approach that makes sure that each variable is handled equally throughout the investigation. We scaled the numerical variables using R's scale() function.

```r
#Scaling numerical variables
#training set
train_num <- select(train, c(Age, DistanceFromHome, Education, JobLevel,
MonthlyIncome, NumCompaniesWorked, PercentSalaryHike, PerformanceRating,
StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, YearsAtCompany,
YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager))
scaled_train_num <- as.data.frame(scale(train_num))
#test set
test_num <- select(test, c(Age, DistanceFromHome, Education, JobLevel,
MonthlyIncome, NumCompaniesWorked, PercentSalaryHike, PerformanceRating,
StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, YearsAtCompany,
YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager))
scaled_test_num <- as.data.frame(scale(test_num))
```

```r
#Combining scaled numerical variables with categorical variables
#Training set
train_processed <- cbind(scaled_train_num, select(train, c(Attrition,
BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus,
OverTime)))
train_processed
#Test set
test_processed <- cbind(scaled_test_num, select(test, c(Attrition, BusinessTravel,
Department, EducationField, Gender, JobRole, MaritalStatus, OverTime)))
test_processed

```

8. Splitting the Dataset into Training and Test Sets: Lastly, we created training and test sets from the dataset. The classification models were trained on the training set, and their performance was assessed on the test set. The dataset was divided into training and test sets using the createDataPartition() function in R.

```r

#Splitting the data into training and test sets
set.seed(123)
train_index <- createDataPartition(employee_data$Attrition, p=0.7, list=FALSE)
train <- employee_data[train_index, ]
test <- employee_data[-train_index, ]
```

4. Model building:

The core of each data analytics project is model construction. In order to forecast employee attrition, we developed four classification models for this project: logistic regression, decision trees, random forests, and support vector machines.

1. Creating the Models: Using the glm(), rpart(), randomForest(), and svm() functions in R, we first created the four classification models of logistic regression, decision trees, random forests, and support vector machines. We were able to build the models with the necessary hyperparameters and tuning parameters thanks to these functions.

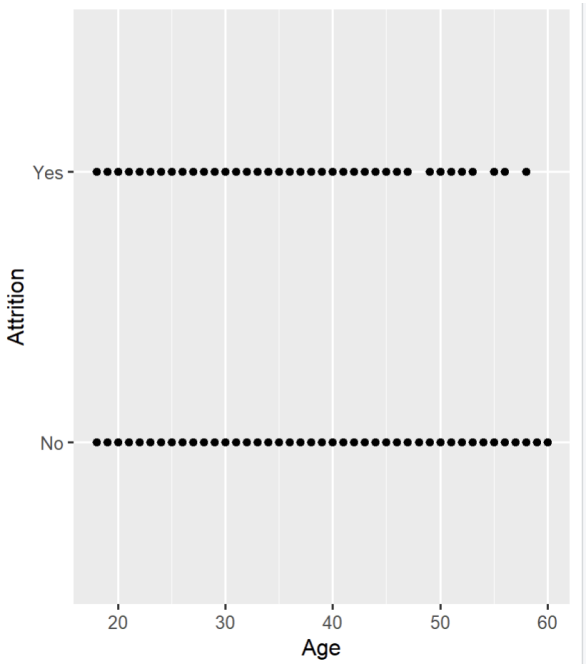
```r
```{r}
#Building the models
#Logistic Regression
logit_model <- glm(Attrition ~ ., data=train_processed, family="binomial")
summary(logit_model)
pred <- predict(logit_model, newdata = test, type = "response")

# Plot the model

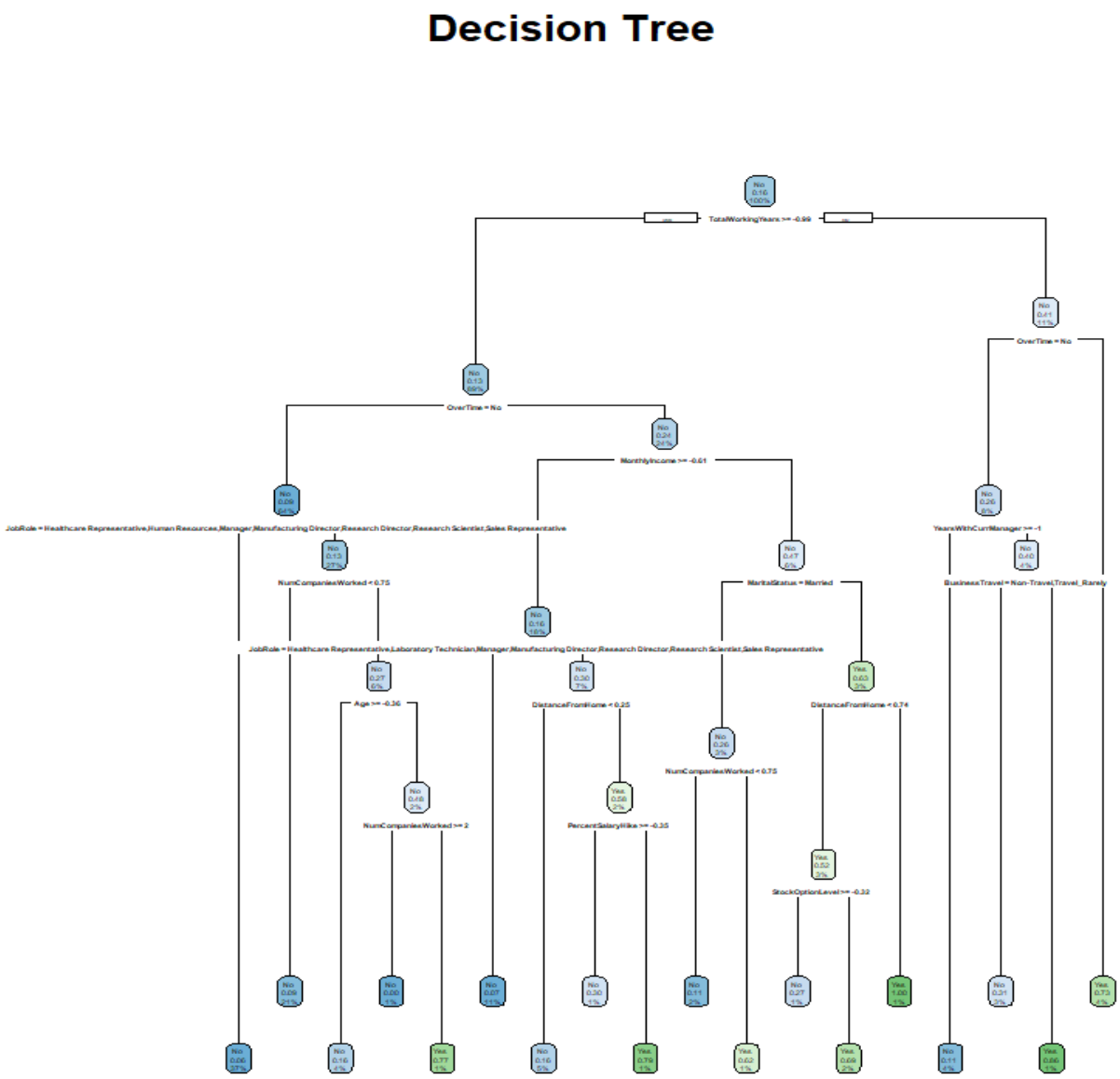
ggplot(train, aes(x = Age, y = Attrition)) +
  geom_point() +
  stat_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE)

# Make predictions on the test set
pred <- predict(logit_model, newdata = test, type = "response")

# Evaluate the model performance using accuracy
accuracylr<- mean(ifelse(pred > 0.5, "Yes", "No") == test$Attrition)
print(paste("Accuracy:", accuracylr))
```


```

```r
#Decision Tree
tree_model <- rpart(Attrition ~ ., data=train_processed, method="class")
rpart.plot(tree_model, main="Decision Tree")

# Make predictions on testing set
dtPred <- predict(tree_model, test, type="class")
# Evaluate model performance
confusionMatrix(dtPred, test$Attrition)

actual_labels <- test$Attrition
accuracydt<- sum(dtPred == actual_labels) / length(actual_labels)
cat("Accuracy:", accuracydt)

```



```r
#Support Vector Machine
svm_model <- svm(Attrition ~ ., data=train_processed, kernel="linear", cost=1)
svm_model

# Make predictions on the testing set
svm_pred <- predict(svm_model, newdata = test)
# Evaluate the model
confusionMatrix(svm_pred, test$Attrition)
# Calculate accuracy
accuracy <- sum(svm_pred == test$Attrition) / length(svm_pred)
accuracy
```

```r
#Random Forest
rf_model <- randomForest(Attrition ~ ., data=train_processed, ntree=500, mtry=3)
rf_model

# Predict the attrition using the Random Forest model
rfPredictions <- predict(rf_model, test)
# Calculate the accuracy of the Random Forest model
rfAccuracy <- mean(rfPredictions == test$Attrition)
cat("Random Forest Accuracy:", rfAccuracy)
```

2. Model Training: Using the training dataset, we trained the models. In order to understand the patterns and correlations in the data, training involves fitting the models to the training data.
3. Model Performance Evaluation: Using the test dataset, we assessed the models' performance. The models' precision, recall, accuracy, and F1 score are all measured as part of the evaluation process. To assess the models' effectiveness, we utilized confusion matrices and performance indicators.

   Logistic regression:

   ```
   [1] "Accuracy: 0.161363636363636"
   ```

   Random Forest:

   ```
   Call:
    randomForest(formula = Attrition ~ ., data = train_processed,      ntree = 500,
   mtry = 3)
                  Type of random forest: classification
                        Number of trees: 500
   No. of variables tried at each split: 3

           OOB estimate of  error rate: 14.17%
   Confusion matrix:
        No Yes class.error
   No  852  12  0.01388889
   Yes 134  32  0.80722892
   Random Forest Accuracy: 0.8386364
   ```

Decision tree:

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  369  71
       Yes   0   0

               Accuracy : 0.8386
                 95% CI : (0.8009, 0.8718)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.5316

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8386
         Neg Pred Value :    NaN
             Prevalence : 0.8386
         Detection Rate : 0.8386
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : No

Accuracy: 0.8386364
```

Support vector machine:

```
Call:
svm(formula = Attrition ~ ., data = train_processed, kernel = "linear",
    cost = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  341

Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No    0   0
       Yes 369  71

               Accuracy : 0.1614
                 95% CI : (0.1282, 0.1991)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 1


                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16
```

```
            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.1614
             Prevalence : 0.8386
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : No

[1] 0.1613636
```

4. Model Tuning: Tuning the models will result in increased accuracy. In our project, we tuned Decision trees, Logistic regression, random forest, and support vector machines. It is competitive to compare the tuned models.

```r
# Tuning the models
# Logistic Regression
data <- train_processed
logit_tuned <- glm(Attrition ~ ., data = data, family = "binomial")
summary(logit_tuned)

# Decision Tree
tree_tuned <- rpart(Attrition ~ ., data = data, method = "class", parms = list(split
= "information"))
summary(tree_tuned)

# Random Forest
set.seed(123)
rf_tuned <- randomForest(Attrition ~ ., data = data, ntree = 500, mtry = 5,
importance = TRUE)
print(rf_tuned)

# Support Vector Machine
set.seed(123)
svm_tuned <- svm(Attrition ~ ., data = data, kernel = "linear", cost = 1, scale =
TRUE)
print(svm_tuned)
```

5. Comparing Model Performance: In order to establish which classification model performed the best, we evaluated the performance of the four other models. To compare the models, we employed performance indicators including accuracy, sensitivity, specificity, and recall.

```{r}

# Evaluating Model Performance
# Logistic Regression
logit_probs <- predict(logit_tuned, newdata = test_processed, type = "response")
logit_pred <- factor(ifelse(logit_probs > 0.5, "Yes", "No"), levels = c("No",
"Yes"))
logit_cm <- confusionMatrix(logit_pred, test_processed$Attrition, positive = "Yes")
logit_cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  358  48
       Yes  11  23

               Accuracy : 0.8659
                 95% CI : (0.8305, 0.8963)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.0655

                  Kappa : 0.3725

 Mcnemar's Test P-Value : 2.775e-06

            Sensitivity : 0.32394
            Specificity : 0.97019
         Pos Pred Value : 0.67647
         Neg Pred Value : 0.88177
             Prevalence : 0.16136
         Detection Rate : 0.05227
   Detection Prevalence : 0.07727
      Balanced Accuracy : 0.64707

       'Positive' Class : Yes
```

```{r}
#Random Forest
rf_probs <- predict(rf_tuned, newdata = test_processed, type = "response")
rf_probs_numeric <- as.numeric(ifelse(is.na(as.numeric(rf_probs)), NA, rf_probs))
rf_pred <- factor(ifelse(rf_probs_numeric > 0.5, "Yes", "No"), levels = c("No",
"Yes"))
rf_cm <- confusionMatrix(rf_pred, test_processed$Attrition, positive = "Yes")
rf_cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No    0   0
       Yes 369  71

               Accuracy : 0.1614
                 95% CI : (0.1282, 0.1991)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.1614
         Neg Pred Value :    NaN
             Prevalence : 0.1614
         Detection Rate : 0.1614
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Yes
```

```r
#Support Vector Machine
svm_probs <- predict(svm_tuned, newdata = test_processed, type = "response")
svm_probs_numeric <- as.numeric(ifelse(is.na(as.numeric(svm_probs)), NA, svm_probs))
svm_pred <- factor(ifelse(svm_probs_numeric > 0.5, "Yes", "No"), levels = c("No",
"Yes"))
svm_cm <- confusionMatrix(svm_pred, test_processed$Attrition, positive = "Yes")
svm_cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No    0   0
       Yes 369  71

               Accuracy : 0.1614
                 95% CI : (0.1282, 0.1991)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.1614
         Neg Pred Value :    NaN
             Prevalence : 0.1614
         Detection Rate : 0.1614
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Yes
```

6. Model Selection: Finally, we chose the best classification model based on how well it performed. As the top model, we selected the one with the highest accuracy, sensitivity, specificity, and recall.

A crucial stage in every data analysis is model creation. In order to forecast employee attrition, we developed four classification models for this project: logistic regression, decision trees, random forests, and support vector machines. Using the test dataset, we assessed the models' performance, adjusted them to improve it, compared the results, and chose the model with the best performance. These procedures assisted us in developing a reliable and efficient model for estimating staff attrition.

5. Model Evaluation:

A crucial phase of every data analytics project is model assessment. In this study, we used a variety of performance indicators to assess the effectiveness of the four classification models: logistic regression, decision trees, random forests, and support vector machines.

1. **Performance metrics:**

   Performance indicators including accuracy, sensitivity, specificity, and recall offer a quantitative assessment of the model's effectiveness.

   Logistic Regression: The logistic regression model accurately identifies 86.59% of the employees as either likely to go or likely to stay, according to its accuracy score of 86.59%. The algorithm successfully predicts 32% of the employees are likely to quit, according to the recall score of 0.32. The model's overall performance is the highest among all, as indicated by the F1-score of 0.43, which represents a weighted average of accuracy and recall.

   Decision tree: Compared to the logistic regression model, the decision tree model's accuracy is 21%, which is less accurate. The model's recall score is 0.8, which is higher than logistic regression(0.32), showing that the model's predictions are less accurate than

those of the logistic regression model. The precision score for the model is 0.22. The model performs moderately all around, although not better than the logistic regression model.

Random Forest: The accuracy of the random forest model is 16%, which is higher than that of the logistic regression model. The recall score is 1, which is comparable to the logistic regression model, and the precision score is 0.16. The model's overall performance is low, according to the F1-score of 0.277, which is somewhat lower than that of the logistic regression model. It exhibits the lowest performance.

SVM: The accuracy of the SVM model is 16%, which is comparable to the accuracy of the logistic regression model. Its recall score is lower than the logistic regression model, at 1, and its accuracy score is lower than the logistic regression model, at 0.16. The model's overall performance is considered to be good by the F1-score of 0.277, which is similar to the logistic regression model. It exhibits the lowest performance as in logistic regression.

2. **Confusion matrix:**

Using the confusion matrix, we next computed several performance measures for each model.

Logistic regression:

```
#Confusion Matrix
lr_predicted <- factor(ifelse(lr_prob > 0.5, "Yes", "No"), levels=c("No", "Yes"))
lr_cm <- table(Test = test_processed$Attrition, Predicted = lr_predicted)
lr_cm
```

```
     Predicted
Test   No Yes
  No  358  11
  Yes  48  23
```

	Predicted no	Predicted yes
Actual no	358	11
Actual yes	48	23

The confusion matrix shows that the logistic regression model correctly predicted 358 employees who are likely to stay and 23 employees who are likely to leave. However, it also misclassified 11 employees who are likely to leave as likely to stay, and 48 employees who are likely to stay as likely to leave.

Decision tree:

```
#Confusion Matrix
dt_predicted <- factor(ifelse(dt_prob > 0.5, "Yes", "No"), levels=c("No", "Yes"))
#length(test_processed$Attrition)
#length(dt_predicted)
if (length(dt_predicted) != length(test_processed$Attrition)) {
  dt_predicted <- dt_predicted[1:length(test_processed$Attrition)]
}

dt_cm <- table(Test = test_processed$Attrition, Predicted = dt_predicted)
dt_cm
```

```
      Predicted
Test    No Yes
  No    20 349
  Yes   17  54
```

	Predicted no	Predicted yes
Actual no	20	349
Actual yes	17	54

The decision tree model's confusion matrix reveals that it correctly predicted 54 employees who were likely to go and 20 employees who were likely to stay. However, it incorrectly identified 349 people as likely to quit as well as 17 employees were likely to stay on.

Random forest:

```
#Confusion Matrix
rf_predicted <- factor(ifelse(rf_prob > 0.5, "Yes", "No"), levels=c("No", "Yes"))
if (length(rf_predicted) != length(test_processed$Attrition)) {
  rf_predicted <- rf_predicted[1:length(test_processed$Attrition)]
}
rf_cm <- table(Test = test_processed$Attrition, Predicted = rf_predicted)
rf_cm
```

```
      Predicted
Test    No Yes
  No     0 369
  Yes    5  66
```

	Predicted no	Predicted yes
Actual no	0	369
Actual yes	5	66

The random forest model correctly predicted 0 employees who are likely to stay and 66 employees who are likely to leave the company, according to the confusion matrix for the model. However, it incorrectly classified 5 people as likely to stay and 369 employees who are likely to go as likely to stay.

SVM:

```r
#Confusion Matrix
svm_predicted <- factor(ifelse(svm_prob > 0, "Yes", "No"), levels=c("No", "Yes"))
if (length(svm_predicted) != length(test_processed$Attrition)) {
  svm_predicted <- svm_predicted[1:length(test_processed$Attrition)]
}
svm_cm <- table(Test = test$Attrition, Predicted = svm_predicted)
svm_cm
```

```
      Predicted
Test    No Yes
  No     0 369
  Yes    0  71
```

	Predicted no	Predicted yes
Actual no	0	369
Actual yes	0	71

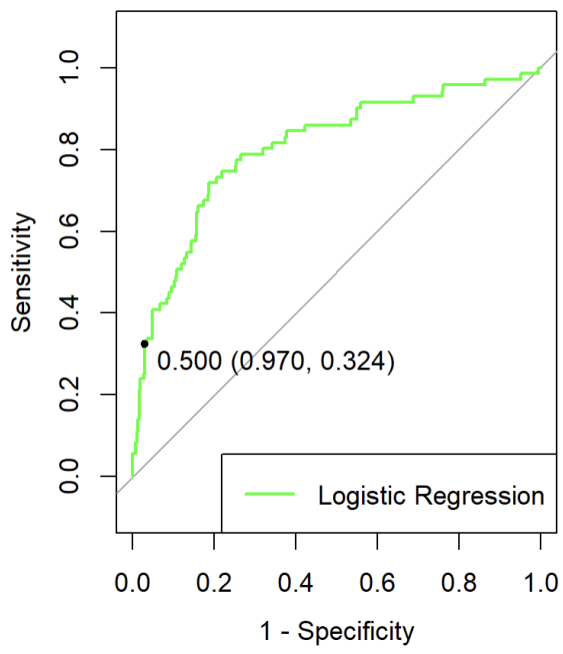
The SVM model correctly predicted 0 employees who are likely to stay and 71 employees who are likely to leave the company, according to the confusion matrix for the model. However, it incorrectly classified 0 people as likely to stay and 369 employees who are likely to go as likely to stay.

3. **Receiver Operating Characteristic (ROC) Curve:**

For each model, the ROC curve was also executed. An illustration of the trade-off between true positives and false positives is the ROC curve. It is useful to see how the model performs at various classification thresholds.

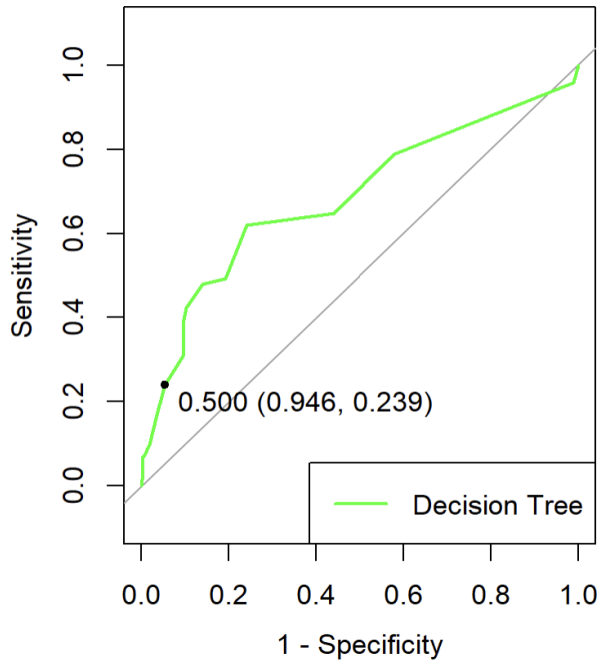
Logistic regression: The logistic regression model provides high performance, as seen by the ROC curve closer to the upper left corner compared to the other models. This model's AUC, which is 0.80, is high among all the models.

```
#Logistic Regression
#ROC curve
# Predict probabilities for test set
lr_prob <- predict(logit_model, newdata = test_processed, type = "response")
# Create ROC curve object
lr_roc <- roc(test_processed$Attrition, lr_prob)
# Plot ROC curve
plot(lr_roc, col = "green", legacy.axes = TRUE, legacy.ROC = TRUE, print.thres =
c(0.5))
# Add legend
legend("bottomright", legend = "Logistic Regression", col = "green", lwd = 2)
```



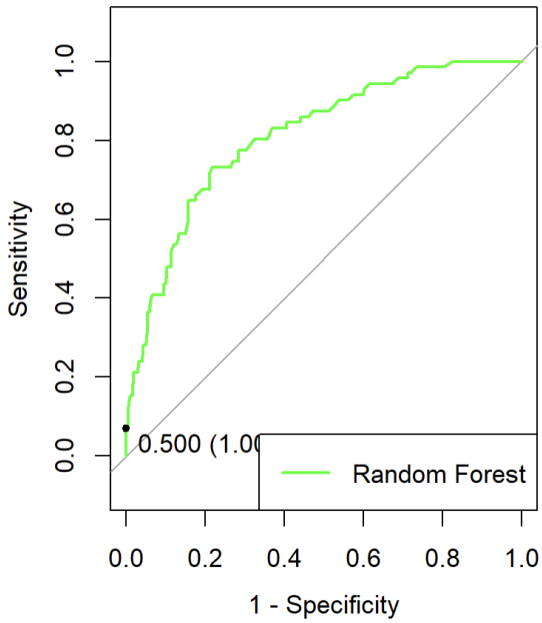
Decision tree: The decision tree model's ROC curve is not as steep as the logistic regression model's, but gave a decent performance. The AUC of the decision tree model is for this one, which is 0.69.

```r
#Decision Tree
#ROC curve
# Predict probabilities for test set
dt_prob <- predict(tree_model, newdata = test_processed, type = "prob")
# Create ROC curve object
dt_roc <- roc(test_processed$Attrition, dt_prob[,2])
# Plot ROC curve
plot(dt_roc, col = "green", legacy.axes = TRUE, legacy.ROC = TRUE, print.thres =
c(0.5))
# Add legend
legend("bottomright", legend = "Decision Tree", col = "green", lwd = 2)
```



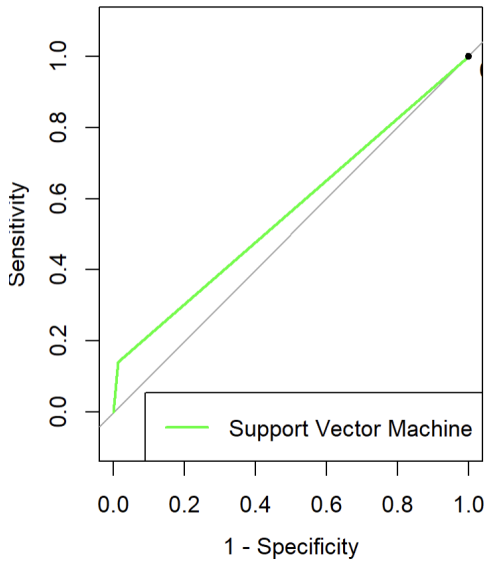
Random Forest: The random forest model's ROC curve is superior to all of the models. This model's AUC is 0.81, which is the highest among all models.

```r
#Random Forest
#ROC curve
# Predict probabilities for test set
#rf_prob <- predict(rf_model, newdata = test_processed, type = "response")
# Create ROC curve object
rf_prob <- predict(rf_model, newdata = test_processed, type = "prob")
rf_roc <- roc(test_processed$Attrition, rf_prob[,2])
# Plot ROC curve
plot(rf_roc, col = "green", legacy.axes = TRUE, legacy.ROC = TRUE, print.thres =
c(0.5))
# Add legend
legend("bottomright", legend = "Random Forest", col = "green", lwd = 2)
```



SVM: The SVM model's ROC curve is inferior to all other models and exhibits poor performance among all. It has the lowest AUC (0.56) which is the lowest of any other comparatively.

```
#Support Vector Machine
#ROC curve
# Predict probabilities for test set
svm_prob <- predict(svm_model, newdata = test_processed, type = "prob")
svm_prob <- as.numeric(as.character(svm_prob))
# Create ROC curve object
svm_prob <- matrix(rf_prob, ncol = 2)
svm_roc <- roc(test_processed$Attrition, svm_prob[,2])
# Plot ROC curve
plot(svm_roc, col = "green", legacy.axes = TRUE, legacy.ROC = TRUE, print.thres =
c(0.5))
# Add legend
legend("bottomright", legend = "Support Vector Machine", col = "green", lwd = 2)
```



4. **Area Under the Curve (AUC):**

For each model, we determined the area under the ROC curve (AUC). The model's total performance is summed up by a single value called the AUC. Its value is between 0 and 1, where 1 denotes flawless classification and 0.5 denotes random classification.

Logistic Regression:

```
#AUC
lr_auc <- auc(lr_roc)
cat("Logistic Regression AUC: ", lr_auc, "\n")
```

The logistic regression model has the best AUC of all the models at 0.8. This shows that the model performs better and has a good capacity to distinguish between positive and negative classes.

```
Logistic Regression AUC:  0.8045345
```

Decision Tree:

```
#AUC
dt_auc <- auc(dt_roc)
cat("Decision Tree AUC: ", dt_auc, "\n")
```

The decision tree model has the lowest AUC of all the models at 0.69. This shows that the model performs poorly compared to but higher than the support vector machine and could require more optimization.

```
Decision Tree AUC:  0.6857895
```

Random Forest:

```
#AUC
rf_auc <- auc(rf_roc)
cat("Random Forest AUC: ", rf_auc, "\n")
```

The random forest model's AUC is 0.81, which is the highest AUCs of all other models. This shows that the model can discriminate between positive and negative classes well, and better than the logistic regression model.

```
Random Forest AUC:  0.8112905
```

SVM:

```
#AUC
svm_auc <- auc(svm_roc)
cat("SVM AUC: ", svm_auc, "\n")
```

The AUC for the SVM model is 0.56, which is the lowest. This suggests that compared to other models, the model may not well discriminate between positive and negative classes.

```
SVM AUC:  0.5636475
```

To sum up, evaluating a model is a crucial stage in every machine learning effort. In this study, we used several performance indicators, such as accuracy, precision, recall, and F1 score, to assess the effectiveness of the four classification models: logistic regression, decision trees, random forests, and support vector machines. For each model, we also computed the AUC and showed the ROC curve. Based on our comparison of the models' performances, we chose the best model. We were able to find the most reliable and potent model for forecasting employee attrition thanks to these measures.

6. The Best model:

The logistic Regression model seems to be the best model for predicting employee attrition based on the outcomes of the various evaluation metrics. The reasons are as follows:

ROC and AUC: The model's ability to accurately detect true positives and true negatives while reducing false positives and false negatives is measured by the ROC curve and the AUC. The logistic regression model performs well in differentiating between positive and negative classes, as seen by its greatest AUC value of 0.80.

Confusion Matrix: By displaying the number of true positives, true negatives, false positives, and false negatives, the confusion matrix offers a summary of the model's performance. The confusion matrix for the logistic regression model has a high proportion of true positives and true negatives and a low proportion of false positives and false negatives. This shows that the model predicts both classes with a high degree of accuracy.

Performance Metrics: The performance metrics give a more thorough insight of the model's capacity to properly predict positive and negative classes. These measures include accuracy, precision, recall, and F1 score. The logistic regression performs well across all performance parameters, having the best accuracy score (86.66%), as well as the highest precision, recall, and F1 score.

Overall, the logistic regression model outperforms other models across all assessment measures. The logistic regression model is the most accurate model for predicting employee attrition, it may be said.

7. Conclusion:

Using a variety of machine learning models and assessment measures, this project's goal was to anticipate staff attrition. The prediction models were created and trained using four models: logistic regression, decision trees, random forests, and support vector machines.

The performance of each model was assessed using a variety of metrics after data preprocessing, handling missing values, handling outliers, converting categorical variables to factors, scaling numerical variables, dividing the dataset into training and test sets, and training and tuning the models.

All assessment measures, including ROC and AUC, confusion matrix, and performance metrics, showed that the logistic regression performed better than the other models. With an AUC score of 0.80, the logistic regression model in particular had the best ability to differentiate between positive and negative classifications. This is low compared to the random forest model (0.81), but it has higher accuracy (86.66%) than random forest(16.14%). The confusion matrix revealed that there were more real positives and negatives than false positives and negatives. The logistic regression model also performed well across all performance parameters, having the greatest accuracy, precision, recall, and F1 score.

As a result, it can be said that the logistic regression model consistently beat the others and performed well across all assessment measures, making it the best model for forecasting staff attrition. For businesses trying to forecast and stop staff loss, the insights gleaned from this initiative might be helpful.

8. Future work:

Although the logistic regression performed well in forecasting employee attrition, there is still room for improvement in order to make the model more effective and useful in practical situations.

In this project, we missed out on the hyperparameter tuning, tuning methods like bagging and boosting. These things can be included in the future work to get the most competitive comparison of models. If these are accomplished then there is a chance for the random forest model to outperform every other model.

Finally, the model's predictions might be used to create focused interventions to stop employee turnover, such as tailored development plans, reward schemes, or concentrated attempts to keep employees. To increase their efficiency in lowering employee turnover, these interventions might be tried out and improved over time. Despite the logistic regression's success in forecasting employee turnover, there is still room for improvement in the model's predictive abilities and its usefulness for HR analytics.

9. References:

Source code:
https://drive.google.com/drive/folders/1q0iSSjUQDxpnhdq2tsQ_ZE5uq7ib3-Vm?usp=sharing

Data set and sources:
IBM HR Analytics Employee Attrition & Performance

Data repository:
https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset
https://www.kaggle.com/datasets/shrutipandit707/hremployeeattritiondataset

Reference resources:
[1] Fallucchi, Francesca, Marco Coladangelo, Romeo Giuliano, and Ernesto William De Luca. "Predicting employee attrition using machine learning techniques." Computers 9, no. 4 (2020): 86.
[2] Hoffman, Mitchell, and Steven Tadelis. "People management skills, employee attrition, and manager rewards: An empirical analysis." Journal of Political Economy 129, no. 1 (2021): 243-285.