Anticipating Attrition: A Machine Learning Approach to Employee Attrition

Rani Misra and Cheyenne Airington

Spring 2024: STA 4143, Data Mining

Professor: Dengdeng Yu, Ph.D.

May 10, 2024

Contents

1 Introduction

A company's success can be attributed to the products or services they provide and the range and magnitude of the employees they hire. However, every business faces challenges retaining their employees at one point or another due to various circumstances. Employee attrition, or the action of an employee leaving an organization and not being replaced, is a fundamental concern for any HR department.
According to Suzanne Lucas at the Academy to Innovate HR (AIHR), attrition often results in a decrease in the size of an organization's or department's workforce because positions aren't refilled when employees leave. Attrition is often interchangeably used with the term 'turnover'. However, employee turnover measures all employment terminations, including those positions that are refilled by new employees, whereas employee attrition focuses on long-term vacancies or permanent position eliminations.
There are two general types of attrition: voluntary and involuntary. Voluntary attrition occurs when employees choose to leave the company on their own accord, such as for personal reasons or finding better opportunities, and the employer decides not to replace them or cannot find a replacement. When the company makes the decision to part ways with an employee and eliminates their position, it is defined as involuntary attrition. This typically occurs during reorganization or layoffs, or could occur due to poor performance or misconduct in singular cases.
High attrition rates can lead to a lack of continuity, gaps in training, insufficient institutional knowledge, employee burnout as workloads increase for remaining employees, lower productivity, harm to employee reputation regarding retention rates, and most importantly, a loss of overall business knowledge and resources. However, there are some cases in which attrition is beneficial. Redundant positions can be eliminated when the employee leaves, allowing for better resource management. Financial struggles can be minimized by not filling vacant roles. In some cases, distributing the role of the vacant position to other team members can elicit professional growth and development. There are various factors that contribute to attrition at business practices. Low unemployment rates may lead to difficulties in filling positions as employees leave. Workforce demographics such as multiple employees retiring or resigning at the same time can result in a skills gap which could increase attrition as it would take longer to fill vacant roles and train new employees. Voluntary resignations and restructuring within companies through mergers and acquisitions can also lead to higher attrition rates. Financial difficulties or shifts in business focus can also result in higher attrition as roles are reconsidered for their importance and are potentially eliminated. Technological advancements can result in a decrease of overall employment within a firm, thereby also increasing attrition. Lastly, outsourcing, or hiring outside services, is another factor that leads to attrition as firms employ third parties instead of retaining specialized employees in efforts to cut down costs.
With attrition being a major concern to a business's welfare depending on their circumstances, it is crucial to investigate which factors specifically lead employees to attrition and what can be done to reduce the impact of these factors.

1.1 Data Dictionary

This fictional dataset was sourced through Kaggle and created by IBM data scientists to offer a comprehensive and varied analysis of an organization's employees, focusing on areas such as employee attrition, personal and job-related factors, and financials. It includes numerous parameters such as:

- Age

- Gender
- Marital Status
- Business Travel Frequency
- Daily Rate of Pay
- Departmental Information such as Distance From Home Office or Education Level obtained by the employee in question
- Job Involvement (level)
- Job Level (relative to similar roles within the same organization)
- Job Role specifically meant for that individual (function/task)
- Total working hours in a week/month/year be it overtime or standard hours for a given role
- Percent Salary Hike during their tenure with the company from promotion or otherwise
- Performance Rating based on specific criteria established by leadership
- Relationship Satisfaction among peers at workplace but also taking into account outside family members that can influence stress levels in varying capacities
- Monthly Income considered at its starting point once hired then compared against their monthly pay rate with overtime hours included if applicable
- Number Companies Worked before if any
- Attrition; covering whether there was an intent to stay with one employer through retirement age or if attrition took place for reasons beyond ones control earlier than expected.

2 Objective

The main objective was to build a model that can accurately predict whether or not an employee will terminate their job (i.e. reach attrition). To achieve this goal, we used statistical and machine learning methods to train various classification models to predict the Attrition class, and chose the most accurate model based on various evaluation metrics.

3 Methodology

3.1 Data Preparation and Summary

Prior to training and testing classification models, we first cleansed our data by addressing any null values and constant variables. During this process, we removed constant variables ”EmployeeCount”, ”Over18”, and ”StandardHours” from the data to reduce model redundancy. We then removed the variable, ”EmployeeID”, as it is a unique identifier for each employee and would not add meaningful variability to the data. There were no nulls in the data, so, we proceeded with a forward selection

process.

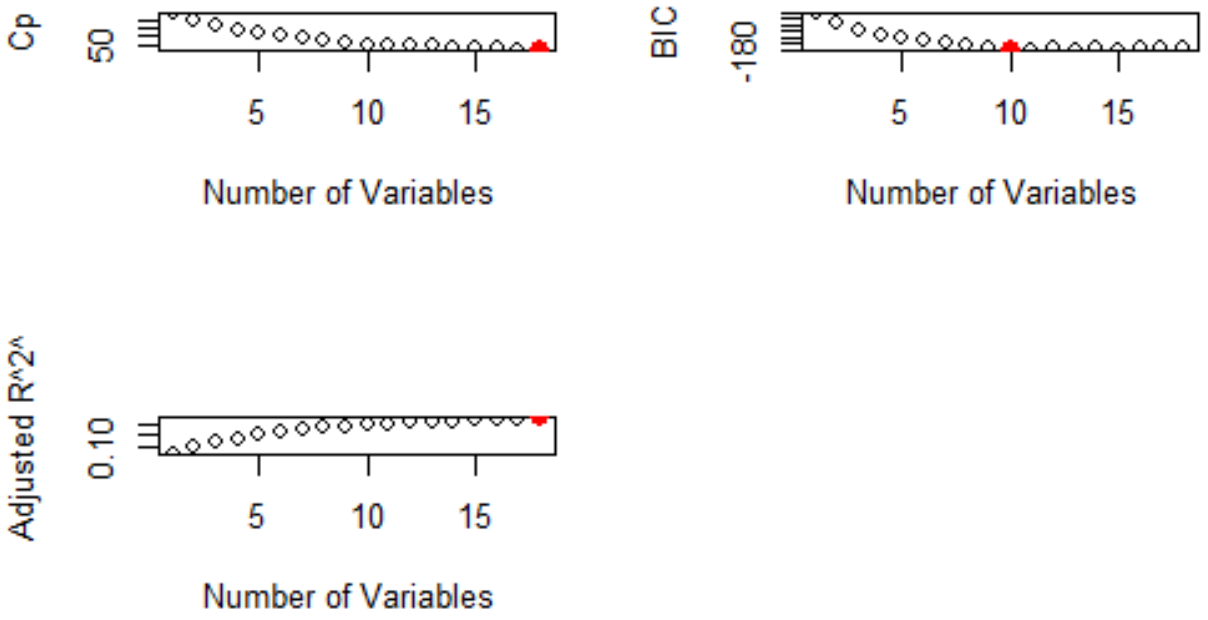

Figure 1: Forward Selection Metrics

Observing the metrics in Figure 1, we see that a minimum BIC (Bayesian Information Criterion) occurs at 10 features. This is an indication that at 10 features, our final model has a preferable balance between complexity and fit. Choosing this specific subset allows us to focus on the most important variables in the model. The final selected features are as follows:

- BusinessTravelTravelFrequently
- EnvironmentSatisfaction
- JobInvolvement
- JobRoleSales Representative
- JobSatisfaction
- MaritalStatusSingle
- NumCompaniesWorked
- OverTimeYes
- TotalWorkingYears

- WorkLifeBalance

Note that during the forward selection process, some variables chosen are binary, dummy versions of the original. This occurs in variables, "BusinessTravelFrequently", "MaritalStatusSingle","JobRoleSales Representative" and "OverTimeYes". The remaining variables are selected as is, and are included in model development. In Figure 2, we can see that the selected variables are uncorrelated.

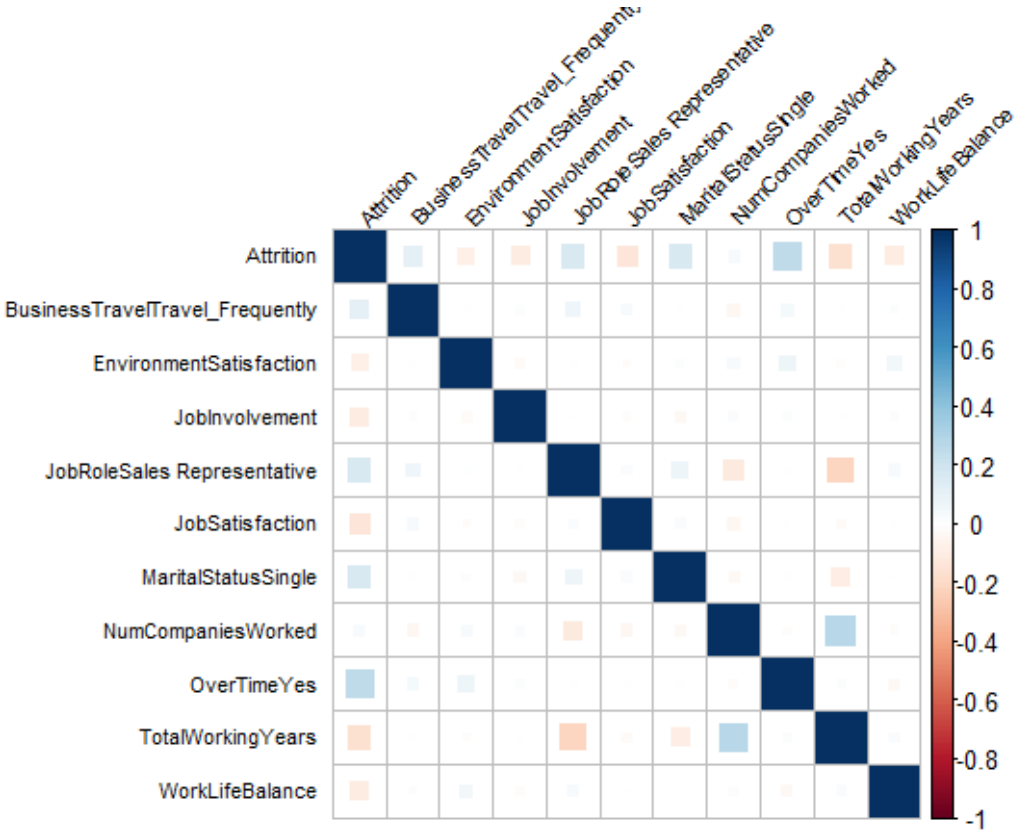

Figure 2: Correlation Matrix

3.2 Analysis

After conducting feature selection, we proceeded to train three different classification models: logistic regression, linear discriminant analysis, and quadratic discriminant analysis. Each model was trained and evaluated using a $75\% - 25\%$ train-test split on the selected features, with further validation through 10-fold cross-validation. The following are the evaluation metrics obtained post-training:

Model	Accuracy	Recall	AUC Value	p Value
LOG	0.8694470	0.9740260	0.6518175	0.001939485
LDA	0.8603808	0.9610390	0.6509105	0.021153476
QDA	0.8377153	0.9350649	0.6351302	0.519942645

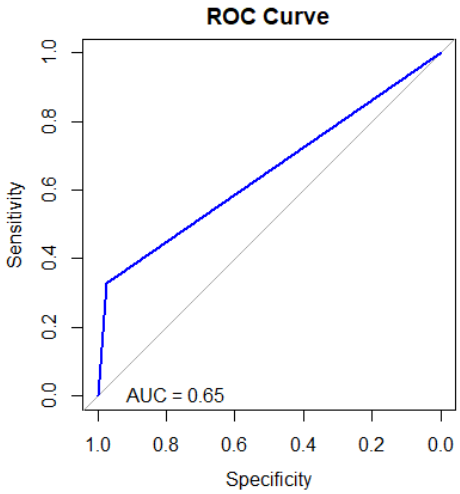

Figure 3: Logistic Regression ROC/AUC

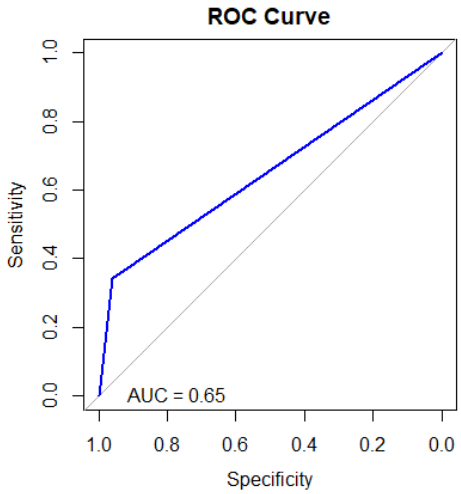

Figure 4: Linear Discriminant Analysis ROC/AUC

H

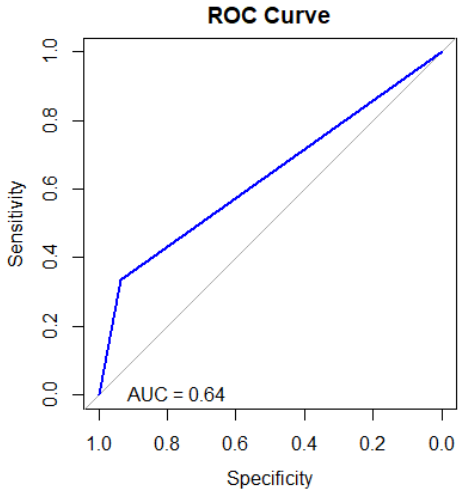

Figure 5: Quadratic Discriminant Analysis ROC/AUC

Out of the models trained, the logistic regression model is the most significant with a low p value of $< .002$ and obtained the best accuracy, recall, and AUC. Although the overall AUC leaves more to be desired in this model, we decided that the importance lies primarily in the model's true positive rate and overall significance for this particular use case. Since the logistic model performs well in accurately predicting the "Yes" attrition class (termination), we chose this as our final model. For future analyses, we may want to consider different models or include different features to better predict the "No" attrition class (not terminated).

4 Generalization

As the analysis is conducted on simulated data, these applications can be reproduced to fit data regarding hiring practices at any firm. As variable characteristics might differ among data sets, interpretations are subject to change depending on the type of data collected and the objective at hand.

5 Conclusion

The forward variable selection analysis found that Environmental Satisfaction, Job Involvement, Job Satisfaction, Number of Companies Worked, Total Working Year, Work-Life Balance, Traveling Frequently, being a Sales Representative, being Single, and Working Overtime are all variables that are more likely to lead employees to attrition. Using optimization and evaluation metrics such as accuracy, recall/sensitivity, and ROC/AUC, a logistic model was determined to be the best at classifying the data with 87% accuracy, a 97% recall rate, and 65% performance in the classification model.

Using the results found in this analysis, firms can create business plans that aim to reduce the significance of variables that are more likely to lead employees to attrition. Once these practices are drafted and implemented, businesses can reproduce this analysis with latest data to test whether employee attrition was actually reduced.

6 Acknowledgment

The data, along with other useful information used to inform this project can be found at the following public references:

1. Karanth, M. (2020). Tabular Summary of HR Analytics Dataset. Zenodo. doi:10.5281/zenodo.4088439.

2. Lucas, Suzanne. (2023, October 3). Employee Attrition: Meaning, Impact & Attrition Rate Calculation. *Academy to Innovate HR (AIHR)*. Retrieved from https://www.aihr.com/blog/employee-attrition/.

3. TheDevastator. Employee Attrition and Factors. Kaggle. Retrieved March 14, 2024, from https://www.kaggle.com/datasets/thedevastator/employee-attrition-and-factors.

A R Code

```
# data clean up
# Check the structure of the dataset
str(data)
head(data)

# Check for nulls and redundant constant fields, constant variables mess up the
↪  selection process since it's based on variability

# Check for null values
null_columns <- sapply(data, function(x) any(is.na(x)))
if (any(null_columns)) {
  print("Null values detected.")
  print("Columns with null values:")
  print(names(data)[null_columns])
} else {
  print("No null values detected.")
}

# Check for constant fields
constant_columns <- sapply(data, function(x) length(unique(x)) == 1)
if (any(constant_columns)) {
  print("Constant fields detected.")
  print("Columns with constant values:")
  print(names(data)[constant_columns])
} else {
  print("No constant fields detected.")
}

# Drop constant fields and "EmployeeNumber" variable, I'm dropping
↪  EmployeeNumber since this alone can't be a good indicator of whether an
↪  employee will term as it's unique for every employee and only differentiates
↪  employees from one another

data <- data %>%
  select(-where(~length(unique(.)) == 1), -EmployeeNumber)

#This step may not be necessary but it shouldn't affect model performance
# Convert character variables to factors
data <- data %>%
  mutate_if(is.character, as.factor)

# Check the structure of the dataset
str(data)

set.seed(1)
test_sample <- sample(1:nrow(data), nrow(data)/4)
train <- data[-test_sample, ]
test <- data[test_sample, ]

library(leaps)

fwd <- regsubsets(Attrition ~ ., data=train, nvmax=17, method='forward')
fwd_sum <- summary(fwd)
```

```
par(mfrow=c(2,2))
plot(fwd_sum$cp ,xlab="Number of Variables ", ylab="Cp",
type="b")
points(which.min(fwd_sum$cp), fwd_sum$cp[which.min(fwd_sum$cp)], col="red",
       cex=2, pch=20)

plot(fwd_sum$bic ,xlab="Number of Variables ",
ylab="BIC",type="b")
points(which.min(fwd_sum$bic), fwd_sum$bic[which.min(fwd_sum$bic)], col="red",
       cex=2, pch=20)

plot(fwd_sum$adjr2 ,xlab="Number of Variables ",
ylab="Adjusted R^2^",type="b")
points(which.max(fwd_sum$adjr2), fwd_sum$adjr2[which.max(fwd_sum$adjr2)],
       col="red", cex=2, pch=20)

which.min(fwd_sum$cp)
which.min(fwd_sum$bic)
which.max(fwd_sum$adjr2)

test_matrix <- model.matrix(Attrition~., data=test)

val.errors <- rep(NA,17)
for(i in 1:17){
 coefi <- coef(fwd,id=i)
 pred <- test_matrix[,names(coefi)]%*%coefi
 pred_binary <- ifelse(pred >= 0.5, 1, 0)
 val.errors[i] <- mean((test$Attrition_binary - pred_binary)^2)
}

val.errors <- rep(NA,17)
for(i in 1:17){
  coefi <- coef(fwd,id=i)
  pred <- test_matrix[,names(coefi)]%*%coefi
  pred_binary <- ifelse(pred >= 1.5, 2, 1)
  val.errors[i] <- mean((as.numeric(test$Attrition) - pred_binary)^2)
}

which.min(val.errors)

plot(val.errors, type='b')
points(which.min(val.errors), val.errors[12], col='red', pch=20, cex=2)

fwd_full <- regsubsets(Attrition ~ ., data=data, nvmax=17, method='forward')
coef(fwd_full, 8)

##Here is how I handled the dummy variable thing, since we don't need the
↪  original data atp I just manually one hot encoded the needed variables in
↪  the train and test sets

# Get the indices of the selected variables
selected_indices <- which.min(fwd_sum$bic)

# Extract the names of the selected variables
selected_variables <- names(which(fwd_sum$which[selected_indices, ] != 0,
↪  arr.ind = TRUE))
```

```
sel_var <-selected_variables[-1]
# Print the selected variables
print(sel_var)

# Identify the columns that need one-hot encoding
columns_to_encode <- c("MaritalStatus", "BusinessTravel","JobRole","OverTime")

# One-hot encode selected columns
encoded_train <- model.matrix(~ . - 1, data = train[columns_to_encode])
encoded_test<-model.matrix(~ . - 1, data = test[columns_to_encode])

# Combine encoded columns with original dataset
encoded_train <- cbind(train, encoded_train)
encoded_test <- cbind(test, encoded_test)

# Filter out columns from the final dataset
train_fnl <- encoded_train[, c("Attrition", sel_var)]
test_fnl <- encoded_test[, c("Attrition", sel_var)]



#Log model

# Fit logistic regression model using train_fnl
cv <- trainControl(method='cv', number=10, savePredictions = T)
logistic_model <- train(Attrition ~ ., data = train_fnl, method='glm', family =
↪  binomial, trControl=cv)
logistic_model

# Calculate precision, recall, and F1-score using the confusion matrix
predicted <- logistic_model$pred$pred
observed <- logistic_model$pred$obs
conf_matrix <- confusionMatrix(predicted, observed)

accuracy <- sum(conf_matrix$table[1,1], conf_matrix$table[2,2]) /
↪  sum(conf_matrix$table)
precision <- conf_matrix$byClass["Pos Pred Value"]
recall <- conf_matrix$byClass["Sensitivity"]
f1_score <- 2 * (precision * recall) / (precision + recall)

# Calculate AUC
roc_curve <- roc(as.numeric(observed), as.numeric(predicted))
auc_value <- auc(roc_curve)

# Display evaluation metrics
print(paste("Accuracy:", round(accuracy, 4)))
print(paste("Precision:", round(precision, 4)))
print(paste("Recall:", round(recall, 4)))
print(paste("F1-score:", round(f1_score, 4)))
print(paste("AUC:", round(auc_value, 4)))

# Plot ROC curve
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
text(0.95, 0, paste("AUC =", round(auc(roc_curve), 2)), adj = c(1, 0), pos = 4)
```

```r
##LDA

# Fit LDA model using train_fnl
cv <- trainControl(method = 'cv', number = 10, savePredictions = TRUE)
lda_model <- train(Attrition ~ ., data = train_fnl, method = 'lda', trControl =
↪  cv)
lda_model

# Calculate precision, recall, and F1-score using the confusion matrix
lda_predicted <- lda_model$pred$pred
lda_observed <- lda_model$pred$obs
lda_conf_matrix <- confusionMatrix(lda_predicted, lda_observed)

lda_accuracy <- sum(lda_conf_matrix$table[1,1], lda_conf_matrix$table[2,2]) /
↪  sum(lda_conf_matrix$table)
lda_precision <- lda_conf_matrix$byClass["Pos Pred Value"]
lda_recall <- lda_conf_matrix$byClass["Sensitivity"]
lda_f1_score <- 2 * (lda_precision * lda_recall) / (lda_precision + lda_recall)

# Calculate AUC
lda_roc_curve <- roc(as.numeric(lda_observed), as.numeric(lda_predicted))
lda_auc_value <- auc(lda_roc_curve)

# Display evaluation metrics
print(paste("Accuracy:", round(lda_accuracy, 4)))
print(paste("Precision:", round(lda_precision, 4)))
print(paste("Recall:", round(lda_recall, 4)))
print(paste("F1-score:", round(lda_f1_score, 4)))
print(paste("AUC:", round(lda_auc_value, 4)))

# Plot ROC curve
plot(lda_roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
text(0.95, 0, paste("AUC =", round(auc(lda_roc_curve), 2)), adj = c(1, 0), pos =
↪  4)


#QDA
# Fit QDA model using train_fnl
cv <- trainControl(method = 'cv', number = 10, savePredictions = TRUE)
qda_model <- train(Attrition ~ ., data = train_fnl, method = 'qda', trControl =
↪  cv)
qda_model

# Calculate precision, recall, and F1-score using the confusion matrix
qda_predicted <- qda_model$pred$pred
qda_observed <- qda_model$pred$obs
qda_conf_matrix <- confusionMatrix(qda_predicted, qda_observed)

qda_accuracy <- sum(qda_conf_matrix$table[1,1], qda_conf_matrix$table[2,2]) /
↪  sum(qda_conf_matrix$table)
qda_precision <- qda_conf_matrix$byClass["Pos Pred Value"]
qda_recall <- qda_conf_matrix$byClass["Sensitivity"]
qda_f1_score <- 2 * (qda_precision * qda_recall) / (qda_precision + qda_recall)

# Calculate AUC
qda_roc_curve <- roc(as.numeric(qda_observed), as.numeric(qda_predicted))
```

```
qda_auc_value <- auc(qda_roc_curve)

# Display evaluation metrics
print(paste("Accuracy:", round(qda_accuracy, 4)))
print(paste("Precision:", round(qda_precision, 4)))
print(paste("Recall:", round(qda_recall, 4)))
print(paste("F1-score:", round(qda_f1_score, 4)))
print(paste("AUC:", round(qda_auc_value, 4)))

# Plot ROC curve
plot(qda_roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
text(0.95, 0, paste("AUC =", round(auc(qda_roc_curve), 2)), adj = c(1, 0), pos =
↪  4)

# Parse out P values
log_p<-conf_matrix$overall["AccuracyPValue"]
lda_p<-lda_conf_matrix$overall["AccuracyPValue"]
qda_p<-qda_conf_matrix$overall["AccuracyPValue"]

# Create vectors for accuracy, recall, and AUC values for different models
model_accuracy <- c(accuracy, lda_accuracy, qda_accuracy)
model_recall <- c(recall, lda_recall, qda_recall)
model_auc_value <- c(auc_value, lda_auc_value, qda_auc_value)
model_pvalue <- c(log_p, lda_p, qda_p)

# Create a dataframe
comparison_table <- data.frame(Model = c("LOG", "LDA", "QDA"),
                               Accuracy = model_accuracy,
                               Recall = model_recall,
                               AUC_Value = model_auc_value,
                               P_Value = model_pvalue)

# Gives us a general look at some metrics we can compare, maybe something like
↪  this can be included in the slide deck
print(comparison_table)
```