Employee Attrition prediction

Code

Jaichiddharth Rangasamy Karthigeyan, Avinash Sankar

Hide

```
#Importing the necessary libraries
library(readr)
library(dplyr)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(e1071)
library(ROCR)
library(pROC)
library(ggplot2)

#Loading the data
employee_data <- read_delim("WA_Fn-UseC_-HR-Employee-Attrition.csv")

dim(employee_data)

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

Hide

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

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

summary(employee_data)
```

Hide

```
#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)
```

Hide

```
#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, ]
```

Hide

```
#Checking for outliers
boxplot(train[, c(1, 3, 5, 6, 7, 8, 11, 12, 13, 15, 18)], main="Boxplot of Employee Data")

plot(train$MonthlyIncome)
plot(train$YearsAtCompany)
plot(train$YearsInCurrentRole)

#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)
```

Hide

```
#Scaling numerical variables
#training set
train_num <- select(train, c(Age, DistanceFromHome, Education, JobLevel, MonthlyIncome, NumCo
mpaniesWorked, PercentSalaryHike, PerformanceRating, StockOptionLevel, TotalWorkingYears, Tra
iningTimesLastYear, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCur
rManager))
scaled_train_num <- as.data.frame(scale(train_num))
#test set
test_num <- select(test, c(Age, DistanceFromHome, Education, JobLevel, MonthlyIncome, NumComp
aniesWorked, PercentSalaryHike, PerformanceRating, StockOptionLevel, TotalWorkingYears, Train
ingTimesLastYear, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrM
anager))
scaled_test_num <- as.data.frame(scale(test_num))
```

Hide

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

Hide

```
#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))
```

Hide

```
#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)
```

Hide

```
#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)
```

Hide

```
#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
```

Hide

```
# 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 = "infor
mation"))
summary(tree_tuned)

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

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

Hide

```


# 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, mode="everything", positive
= "Yes")
logit_cm
```

Hide

```
#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, mode= "everything",positive = "Ye
s")
rf_cm
```

Hide

```
#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, mode= "everything",positive =
"Yes")
svm_cm
```

Hide

```
#Performance Metrics
#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)

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

#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
```

Hide

```
#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)


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

#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
```

Hide

```
#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)


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

#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
```

Hide

```
#Support Vector Machine
#ROC curve
# Predict probabilities for test set
svm_prob <- predict(svm_model, newdata = test_processed, type = "prob")
svm_prob <- as.numeric(svm_prob)
# Create ROC curve object

#length(test_processed$Attrition)
svm_prob <- matrix(svm_prob, ncol = 2)
#length(svm_prob)
#table(test_processed$Attrition)
#table(svm_prob)
#levels(test_processed$Attrition)
svm_roc <- roc(test_processed$Attrition, svm_prob)
# 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)

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

#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
```