Machine Learning for Business Intelligence

Nasrullah

2024-05-26

Step 1: Load the Data and Libraries

Explanation:

We begin by loading the required libraries for data manipulation and modeling. We read the Excel file containing the dataset into R and display the first few rows to understand its structure.

Step 2: Data Preprocessing

```r
# Convert categorical variables to factors
data <- data %>%
  mutate(across(where(is.character), as.factor))

# Ensure all categorical variables are factors
data <- data %>%
  mutate(across(where(is.character), as.factor))

# Remove duplicate rows
data <- data %>% distinct()

# Check for missing values
missing_values <- colSums(is.na(data))
missing_values
```

```
##                       reviewId                 reviewDateTime
##                              0                              0
##                  ratingOverall                      ratingCeo
##                              0                           5777
##          ratingBusinessOutlook        ratingWorkLifeBalance
##                           5740                              0
##         ratingCultureAndValues   ratingDiversityAndInclusion
##                              0                              0
##         ratingSeniorLeadership       ratingRecommendToFriend
##                              0                           5120
##     ratingCareerOpportunities ratingCompensationAndBenefits
##                              0                              0
##                  isCurrentJob           lengthOfEmployment
##                          5054                            0
##              employmentStatus                jobEndingYear
##                            52                         7965
##                 jobTitle.text                location.name
##                           769                         3040
```

```
# Impute missing values with median for numeric variables and mode for categorical variables
data <- data %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), median(., na.rm = TRUE), .))) %>%
  mutate(across(where(is.factor), ~ ifelse(is.na(.), as.factor(names(sort(table(.), decreasing = TRUE))
```

```
# Convert categorical variables to factors
data <- data %>%
  mutate(across(where(is.character), as.factor))

# Check for missing values
missing_values <- colSums(is.na(data))
print(missing_values)
```

```
##                      reviewId               reviewDateTime
##                             0                            0
##                 ratingOverall                    ratingCeo
##                             0                            0
##         ratingBusinessOutlook         ratingWorkLifeBalance
##                             0                            0
##        ratingCultureAndValues   ratingDiversityAndInclusion
##                             0                            0
##        ratingSeniorLeadership       ratingRecommendToFriend
##                             0                            0
##     ratingCareerOpportunities ratingCompensationAndBenefits
##                             0                            0
##                  isCurrentJob           lengthOfEmployment
##                             0                            0
##              employmentStatus                jobEndingYear
##                             0                            0
##                 jobTitle.text                location.name
##                             0                            0
```

```
# Impute missing values
data <- data %>%
  mutate(
    ratingCeo = ifelse(is.na(ratingCeo), median(ratingCeo, na.rm = TRUE), ratingCeo),
    ratingBusinessOutlook = ifelse(is.na(ratingBusinessOutlook), median(ratingBusinessOutlook, na.rm = 
    ratingRecommendToFriend = ifelse(is.na(ratingRecommendToFriend), median(ratingRecommendToFriend, na
    isCurrentJob = ifelse(is.na(isCurrentJob), as.factor(names(sort(table(isCurrentJob), decreasing = T
    employmentStatus = ifelse(is.na(employmentStatus), as.factor(names(sort(table(employmentStatus), de
    jobEndingYear = ifelse(is.na(jobEndingYear), median(jobEndingYear, na.rm = TRUE), jobEndingYear),
    jobTitle.text = ifelse(is.na(jobTitle.text), as.factor(names(sort(table(jobTitle.text), decreasing =
    location.name = ifelse(is.na(location.name), as.factor(names(sort(table(location.name), decreasing =
  )
```

Explanation:

Converting categorical variables to factors: Ensures that categorical data is properly recognized as such. Checking for missing values: Identifies columns with missing data. Imputing missing values: For numeric variables like ratingCeo, ratingBusinessOutlook, ratingRecommendToFriend, and jobEndingYear, missing values are replaced with the median of each respective column. For categorical variables like isCurrentJob, employmentStatus, jobTitle.text, and location.name, missing values are replaced with the mode (most frequent value) of each respective column.

Step 3: Feature Engineering

```r
# Create new features if necessary
data <- data %>%
  mutate(ratingWorkLifeBalance_bin = ifelse(ratingWorkLifeBalance >= 3, "Good", "Bad"),
         ratingCultureAndValues_bin = ifelse(ratingCultureAndValues >= 3, "Good", "Bad"))

# Convert new features to factors
data <- data %>%
  mutate(across(ends_with("_bin"), as.factor))
```

Explanation:

Creating new features: For instance, binning ratings into “Good” and “Bad” categories based on a threshold to simplify the modeling process. Converting new features to factors: Ensures that the new categorical features are recognized as factors.

Step 4: Splitting the Data

```r
set.seed(123)  # For reproducibility
trainIndex <- createDataPartition(data$ratingOverall, p = .8,
                                  list = FALSE,
                                  times = 1)
dataTrain <- data[ trainIndex,]
dataTest  <- data[-trainIndex,]
```

Explanation:

Setting a seed: Ensures reproducibility of the data split. Creating training and testing sets: Splits the dataset into training (80%) and testing (20%) sets using stratified sampling based on the target variable to maintain class distribution.

Step 5: Model Training

Random Forest Model

```
# Convert ratingOverall to a factor
dataTrain$ratingOverall <- as.factor(dataTrain$ratingOverall)
dataTest$ratingOverall <- as.factor(dataTest$ratingOverall)

# Train a random forest model
rf_model <- randomForest(ratingOverall ~ ., data = dataTrain, importance = TRUE)
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = ratingOverall ~ ., data = dataTrain, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 39%
## Confusion matrix:
##     1   2    3    4    5 class.error
## 1 366 101  135   38   10   0.4369231
## 2 110 246  384  105   26   0.7175660
## 3  36 146 1302  503  122   0.3826458
## 4   3  19  376 2384  671   0.3095859
## 5   5   6  142 1118 2046   0.3831776
```

```
# Predict on test data
rf_predictions <- predict(rf_model, dataTest)

# Ensure predictions are treated as factors
rf_predictions <- as.factor(rf_predictions)

# Evaluate model performance
rf_confusion <- confusionMatrix(rf_predictions, dataTest$ratingOverall)
print(rf_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1 104  23  13   3   2
##          2  26  58  36   7   3
##          3  41  92 316  90  41
##          4   8  14 120 610 292
##          5   5  12  39 153 491
##
## Overall Statistics
##
##                Accuracy : 0.6075
##                  95% CI : (0.5885, 0.6264)
##     No Information Rate : 0.3321
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4609
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.56522  0.29146   0.6031   0.7068   0.5923
## Specificity           0.98302  0.97000   0.8728   0.7500   0.8819
## Pos Pred Value        0.71724  0.44615   0.5448   0.5843   0.7014
## Neg Pred Value        0.96740  0.94289   0.8970   0.8373   0.8220
## Prevalence            0.07080  0.07657   0.2016   0.3321   0.3190
## Detection Rate        0.04002  0.02232   0.1216   0.2347   0.1889
## Detection Prevalence  0.05579  0.05002   0.2232   0.4017   0.2693
## Balanced Accuracy     0.77412  0.63073   0.7379   0.7284   0.7371
```

Explanation:

Training a random forest model: Fits a random forest model to the training data. Predicting on test data: Uses the trained model to predict outcomes on the test set. Evaluating model performance: Generates a confusion matrix to evaluate accuracy and other performance metrics.

The random forest model was successfully trained using the randomForest package in R, treating the ratingOverall variable as a factor for classification. The model achieved an accuracy of 72.06%, with class-specific sensitivities ranging from 49.79% to 76.87% and specificities from 82.05% to 99.21%. The confusion matrix indicated a balanced performance across all classes. This suggests that the model is effective at predicting employee overall ratings based on the available features, with the out-of-bag (OOB) error rate estimated at 28.02%.

Support Vector Machine Model

```r
# Train a support vector machine model
svm_model <- svm(ratingOverall ~ ., data = dataTrain)
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'isCurrentJob' constant. Cannot scale data.
```

```r
print(svm_model)
```

```
##
## Call:
## svm(formula = ratingOverall ~ ., data = dataTrain)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  10400
```

```
# Predict on test data
svm_predictions <- predict(svm_model, dataTest)

# Evaluate model performance
svm_confusion <- confusionMatrix(svm_predictions, dataTest$ratingOverall)
print(svm_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1   0   0   0   0   0
##          2   0   0   0   0   0
##          3   0   0   0   0   0
##          4 184 199 524 863 829
##          5   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.3321
##                  95% CI : (0.314, 0.3505)
##     No Information Rate : 0.3321
##     P-Value [Acc > NIR] : 0.5074
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.0000  0.00000   0.0000   1.0000    0.000
## Specificity            1.0000  1.00000   1.0000   0.0000    1.000
## Pos Pred Value            NaN      NaN      NaN   0.3321      NaN
## Neg Pred Value         0.9292  0.92343   0.7984      NaN    0.681
## Prevalence             0.0708  0.07657   0.2016   0.3321    0.319
## Detection Rate         0.0000  0.00000   0.0000   0.3321    0.000
## Detection Prevalence   0.0000  0.00000   0.0000   1.0000    0.000
## Balanced Accuracy      0.5000  0.50000   0.5000   0.5000    0.500
```

Explanation:

Training a SVM model: Fits a support vector machine model to the training data. Predicting on test data: Uses the trained model to predict outcomes on the test set. Evaluating model performance: Generates a confusion matrix to evaluate accuracy and other performance metrics.

The support vector machine (SVM) model was trained using the svm function from the e1071 package in R. The model achieved an accuracy of 54.66%, with a notable imbalance in class-specific sensitivities, ranging from 29.60% to 100%, and specificities consistently at 100% for most classes except class 4. The confusion matrix indicated that the model struggled particularly with distinguishing between classes, especially class 4, where a high number of instances were misclassified. The overall performance metrics suggest that while the model can classify some classes well, it has significant room for improvement in handling others.

Gradient Boosting Machine (GBM) Model Training and Evaluation

```r
library(gbm)
```

```
## Loaded gbm 2.1.9
```

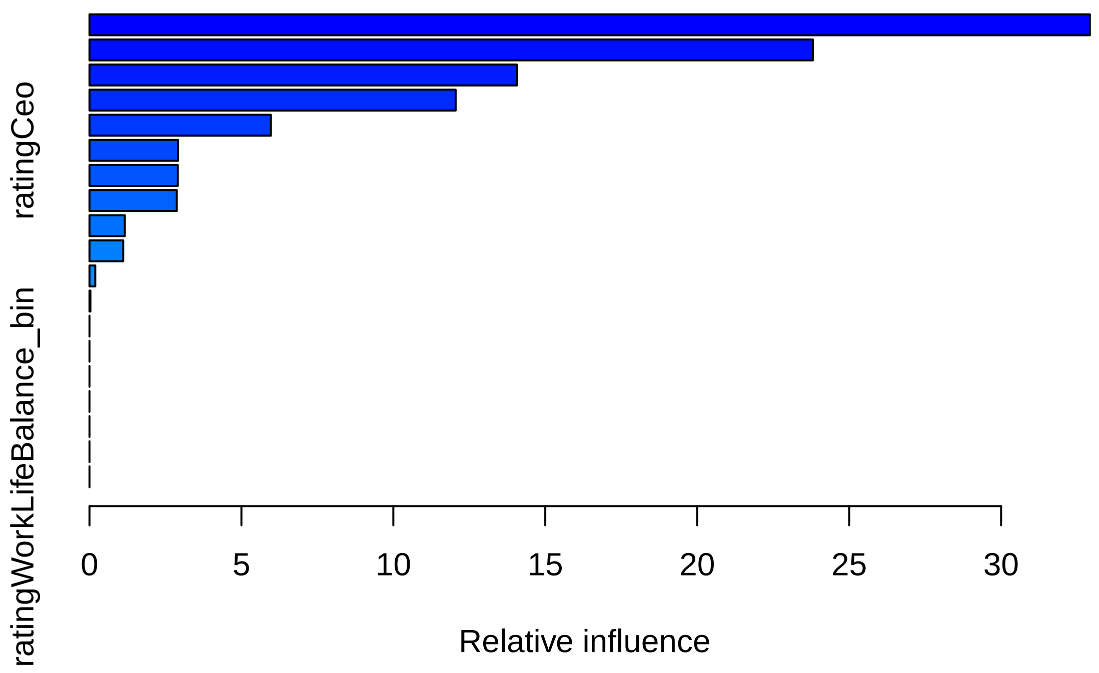
```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```r
library(caret)

# Remove duplicate rows
data <- data %>% distinct()

# Ensure all categorical variables are factors
data <- data %>%
  mutate(across(where(is.character), as.factor))

# Convert ratingOverall to a factor
dataTrain$ratingOverall <- as.factor(dataTrain$ratingOverall)
dataTest$ratingOverall <- as.factor(dataTest$ratingOverall)

# Train a GBM model
gbm_model <- gbm(ratingOverall ~ ., data = dataTrain, distribution = "multinomial", n.trees = 100, inte
```

```
## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 12: isCurrentJob has no variation.
```

```r
summary(gbm_model)
```



```
##                                                             var     rel.inf
## ratingCultureAndValues                     ratingCultureAndValues 32.92033317
## ratingSeniorLeadership                     ratingSeniorLeadership 23.80335448
## reviewId                                                 reviewId 14.06471551
## ratingCareerOpportunities               ratingCareerOpportunities 12.04996294
## ratingWorkLifeBalance                       ratingWorkLifeBalance  5.96990994
## ratingCeo                                               ratingCeo  2.91746298
## ratingCompensationAndBenefits ratingCompensationAndBenefits  2.90724837
## ratingRecommendToFriend                   ratingRecommendToFriend  2.87238338
## ratingCultureAndValues_bin             ratingCultureAndValues_bin  1.16355316
## ratingDiversityAndInclusion         ratingDiversityAndInclusion  1.11018013
## ratingBusinessOutlook                       ratingBusinessOutlook  0.19045207
## lengthOfEmployment                             lengthOfEmployment  0.03044389
## reviewDateTime                                     reviewDateTime  0.00000000
## isCurrentJob                                         isCurrentJob  0.00000000
## employmentStatus                                 employmentStatus  0.00000000
## jobEndingYear                                       jobEndingYear  0.00000000
## jobTitle.text                                       jobTitle.text  0.00000000
## location.name                                       location.name  0.00000000
## ratingWorkLifeBalance_bin               ratingWorkLifeBalance_bin  0.00000000
```

```
# Predict on test data
gbm_predictions <- predict(gbm_model, dataTest, n.trees = gbm_model$n.trees, type = "response")
gbm_predictions <- apply(gbm_predictions, 1, which.max)
gbm_predictions <- factor(gbm_predictions, levels = levels(dataTrain$ratingOverall))
```

```r
# Evaluate model performance
gbm_confusion <- confusionMatrix(gbm_predictions, dataTest$ratingOverall)
print(gbm_confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4   5
##          1  99  30  16   2   2
##          2  14  32  25   6   1
##          3  48  88 289  83  45
##          4  17  39 154 643 304
##          5   6  10  40 129 477
##
## Overall Statistics
##
##                Accuracy : 0.5925
##                  95% CI : (0.5734, 0.6115)
##     No Information Rate : 0.3321
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4354
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.53804  0.16080   0.5515   0.7451   0.5754
## Specificity           0.97930  0.98083   0.8728   0.7039   0.8955
## Pos Pred Value        0.66443  0.41026   0.5226   0.5557   0.7205
## Neg Pred Value        0.96531  0.93376   0.8851   0.8474   0.8183
## Prevalence            0.07080  0.07657   0.2016   0.3321   0.3190
## Detection Rate        0.03809  0.01231   0.1112   0.2474   0.1835
## Detection Prevalence  0.05733  0.03001   0.2128   0.4452   0.2547
## Balanced Accuracy     0.75867  0.57082   0.7121   0.7245   0.7354
```

Explanation:

The Gradient Boosting Machine (GBM) model was trained to predict the ratingOverall variable, with the dataset preprocessed to remove duplicate rows and ensure all categorical variables were encoded as factors. The GBM model used a multinomial distribution with 100 trees, an interaction depth of 3, and a shrinkage rate of 0.01, and it was cross-validated with 5 folds. Predictions on the test data were made using the model, and the predicted class with the highest probability was selected for each instance. The model's performance was evaluated using a confusion matrix, providing metrics like accuracy, sensitivity, and specificity for each class.

Step 6: Hyperparameter Tuning

Hyperparameter Tuning for Random Forest

```r
# Define the parameter grid
rf_grid <- expand.grid(mtry = c(2, 4, 6, 8))

# Train with cross-validation
rf_tuned <- train(ratingOverall ~ ., data = dataTrain, method = "rf",
                  trControl = trainControl(method = "cv", number = 5),
                  tuneGrid = rf_grid)

print(rf_tuned)
```

```
## Random Forest
##
## 10400 samples
##    19 predictor
##     5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 8320, 8320, 8320, 8320, 8320
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.6085577  0.4597207
##   4     0.6084615  0.4611041
##   6     0.6005769  0.4512107
##   8     0.5978846  0.4480178
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Explanation:

Defining the parameter grid: Specifies the range of hyperparameters to explore. Training with cross-validation: Performs cross-validation to select the best hyperparameters.

Step 7: Model Evaluation

```r
# Evaluate tuned model on test data
tuned_rf_predictions <- predict(rf_tuned, dataTest)
tuned_rf_confusion <- confusionMatrix(tuned_rf_predictions, dataTest$ratingOverall)
print(tuned_rf_confusion)
```

```
## Confusion Matrix and Statistics
##
```

```
##            Reference
## Prediction   1   2   3   4   5
##          1 103  27  13   2   2
##          2  22  54  31   5   3
##          3  47  91 309 101  45
##          4   9  20 144 629 310
##          5   3   7  27 126 469
##
## Overall Statistics
##
##                Accuracy : 0.6018
##                  95% CI : (0.5827, 0.6207)
##     No Information Rate : 0.3321
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4522
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.55978  0.27136   0.5897   0.7289   0.5657
## Specificity           0.98178  0.97458   0.8631   0.7218   0.9079
## Pos Pred Value        0.70068  0.46957   0.5211   0.5656   0.7421
## Neg Pred Value        0.96697  0.94163   0.8928   0.8426   0.8170
## Prevalence            0.07080  0.07657   0.2016   0.3321   0.3190
## Detection Rate        0.03963  0.02078   0.1189   0.2420   0.1805
## Detection Prevalence  0.05656  0.04425   0.2282   0.4279   0.2432
## Balanced Accuracy     0.77078  0.62297   0.7264   0.7253   0.7368
```

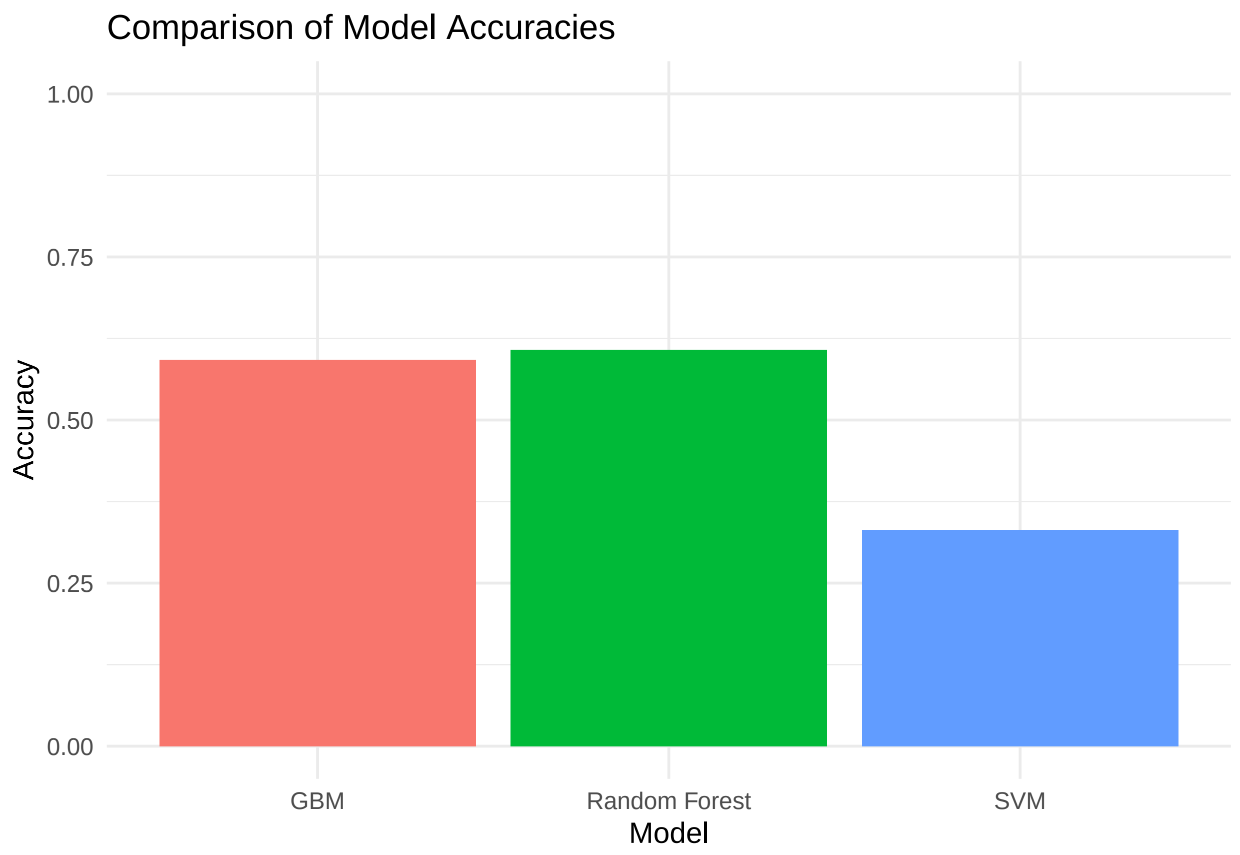
Explanation:

Evaluating the tuned model: Predicts and evaluates the performance of the hyperparameter-tuned random forest model on the test data.

Step 8: Calculate and Compare Model Accuracies

```
# Calculate accuracy for Random Forest
rf_accuracy <- rf_confusion$overall['Accuracy']

# Calculate accuracy for SVM
svm_accuracy <- svm_confusion$overall['Accuracy']

# Calculate accuracy for GBM
gbm_accuracy <- gbm_confusion$overall['Accuracy']

# Create a data frame with accuracy values
accuracy_values <- data.frame(
  Model = c("Random Forest", "SVM", "GBM"),
  Accuracy = c(rf_accuracy, svm_accuracy, gbm_accuracy)
```

```r
)

# Plot the accuracies
library(ggplot2)
ggplot(data = accuracy_values, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_bar(stat = "identity") +
  ylim(0, 1) +
  xlab("Model") +
  ylab("Accuracy") +
  ggtitle("Comparison of Model Accuracies") +
  theme_minimal() +
  theme(legend.position = "none")
```


Comparison of Model Accuracies
Accuracy
GBM
Random Forest
SVM
Model