BA_HW2_Group_Tech_Document

Mayank Singh

2024-10-25

Setup Libraries

```
### Setup Libraries
library(stats)
library(dplyr)
library(cluster)
library(ggplot2)
library(scales)
library(cluster)
library(rpart)
library(rpart.plot)
library(class)
library(caret)
library(rsample)
library(ROSE)
library(smotefamily)
library(tidyr)
library(corrplot)
library(pROC)
options(warn = -1)
```

Reading Data

```
# import dataset
music = read.csv("XYZData.csv")
music = music[,2:27]

# making the flag columns as factors
music$male <- as.factor(music$male)
music$good_country <- as.factor(music$good_country)
music$delta_good_country <- as.factor(music$delta_good_country)
music$adopter <- as.factor(music$adopter)
```

Checking Class Imbalance

Data provided is heavily imbalanced where only 4% have positive labels(the customers of interest in our case). Therefore, we need to apply over/under sampling techniques to fix this issue.

```
#check table
table(music$adopter)
```

```
##
##     0     1
## 40000  1540
```

```
#check classes distribution
prop.table(table(music$adopter))
```

```
##
##         0         1
## 0.9629273 0.0370727
```

Train-Validation-Test Split

We are splitting the data into three sets - Training + Validation + Testing

We are taking out validation set from the 80% training data.

```
# taking 80% data for training and 20% for testing
music_rows = createDataPartition(y = music$adopter, p = 0.8, list = FALSE)
music_train_valid = music[music_rows,]
music_test = music[-music_rows,]

# splitting for train(80%) and validation(20%)
music_train_rows = createDataPartition(y = music_train_valid$adopter, p = 0.8, list = FALSE)
music_train = music_train_valid[music_train_rows,]
music_valid = music_train_valid[-music_train_rows,]
```

Oversampling

We tried oversampling, undersampling and SMOTE. However, oversampling provides the best results

```r
# we tried oversampling, undersampling and SMOTE - oversampling provides best
# results

# bringing the proportion of minority class(adopter = 1) to 45% using over-
# sampling
music_over_sampled <- ovun.sample(adopter~., data = music_train, method = "over"
                                  , p = 0.45, seed = 111)$data
table(music_over_sampled$adopter)
```

```
##
##     0     1
## 25600 21006
```

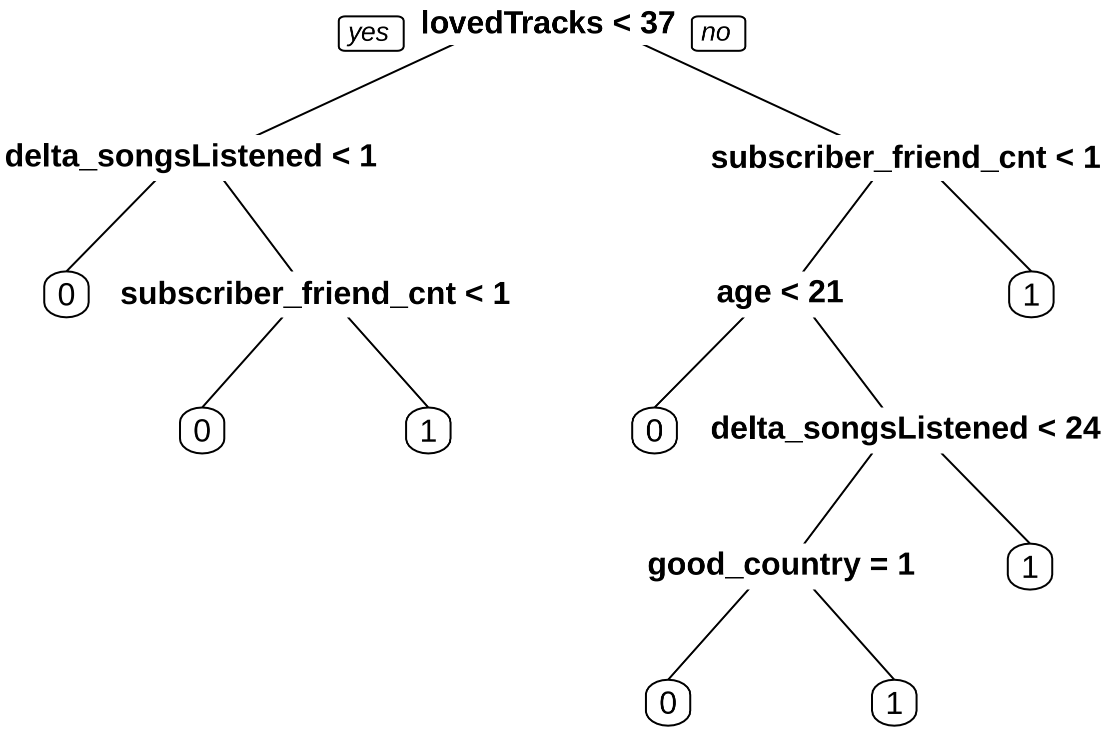
Baseline Modeling - Decision Tree

We are going ahead with Decision Tree as it is performing the better than Naive Bayes model. For model evaluation, we are choosing Recall as our primary performance metric as we want to maximise the model's ability to predict all the potential customers who can purchase the premium subscription

Training

```r
# training Decision Tree
tree = rpart(adopter ~ ., data = music_over_sampled,
             method = "class",
             parms = list(split = "information"),
             control = list(minsplit = 3,
                            maxdepth = 5,
                            cp = 0.01)
             )

# print out the tree
prp(tree, varlen = 0)
```

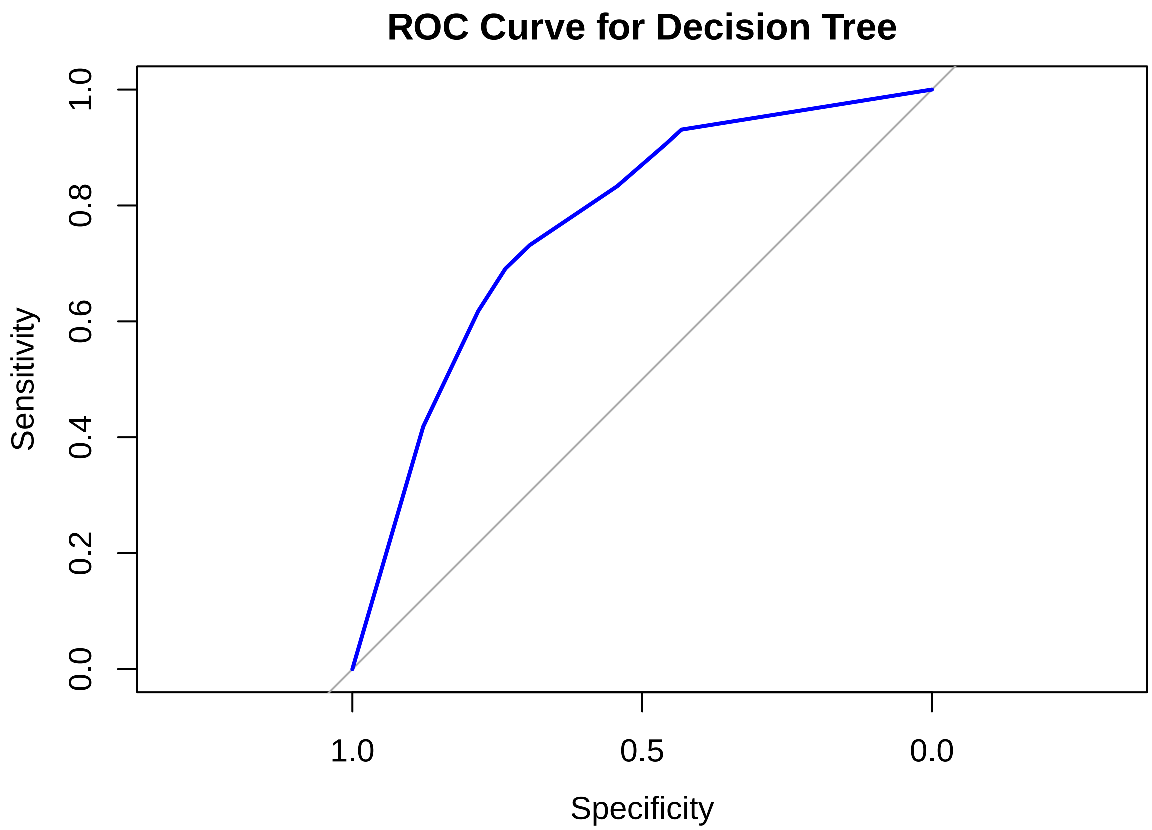


Validation

```
# Predict probabilities for the validation set (needed for ROC/AUC)
pred_prob_tree <- predict(tree, music_valid, type = "prob")[, 2]

# Create ROC curve and calculate AUC
roc_curve <- roc(music_valid$adopter, pred_prob_tree)

# Plot ROC curve
plot(roc_curve, col = "blue", main = "ROC Curve for Decision Tree")
```




```r
# Calculate AUC value
auc_value <- auc(roc_curve)
print(paste("AUC Value: ", auc_value))
```

```
## [1] "AUC Value:  0.768361598069106"
```

```r
# Print the confusion matrix using thresholded predictions
# We are using a threshold of 0.3 to maximize the model's ability to predict
# customers who have high propensity to adopt the premium model
pred_tree_class <- ifelse(pred_prob_tree > 0.3, "1", "0")

confusionMatrix(data = as.factor(pred_tree_class),
                reference = music_valid$adopter,
                mode = "prec_recall",
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##            0 3475   41
##            1 2925  205
##
##                 Accuracy : 0.5537
##                   95% CI : (0.5417, 0.5657)
```

```
##     No Information Rate : 0.963
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0567
##
##  Mcnemar's Test P-Value : <2e-16
##
##               Precision : 0.06550
##                  Recall : 0.83333
##                      F1 : 0.12145
##              Prevalence : 0.03701
##          Detection Rate : 0.03085
##    Detection Prevalence : 0.47096
##       Balanced Accuracy : 0.68815
##
##        'Positive' Class : 1
##
```

Cross-Validation

As we got a decent Recall from the baseline model. Let's proceed with a k - fold validation.We would want to be confident with our model performance, therefore, we are using a 5-fold cross-validation to check average performance of the model.

```r
# Create 5-fold cross-validation
cv = createFolds(y = music_train_valid$adopter, k = 5)

# Initialize empty vectors to store metrics
auc_values  = c()
recall_values = c()
precision_values = c()
f1_score_values = c()

for (test_rows in cv) {

  # Train-validaton split
  music_train = music_train_valid[-test_rows,]
  music_valid = music_train_valid[test_rows,]

  # Oversampling
  music_over_sampled <- ovun.sample(adopter ~ ., data = music_train,
                                    method = "over", p = 0.45, seed = 111)$data

  # Fit Decision Tree
  tree = rpart(adopter ~ ., data = music_over_sampled,
               method = "class",
               parms = list(split = "information"),
```

```r
                control = list(minsplit = 3,
                               maxdepth = 5,
                               cp = 0.01)
                )
  # we are getting best results with the above parameters mentioned for
  # minimum splits, max depth and cp

  # Get class probabilities on validation data
  pred_prob_tree <- predict(tree, music_valid, type = "prob")[, 2]

  # Create ROC curve and calculate AUC
  roc_curve <- roc(music_valid$adopter, pred_prob_tree)
  auc_value <- auc(roc_curve)

  # Store AUC value
  auc_values <- c(auc_values, auc_value)

  # Get class predictions (threshold = 0.3)
  pred_tree_class <- ifelse(pred_prob_tree > 0.3, "1", "0")

  # Confusion Matrix
  cm <- confusionMatrix(data = as.factor(pred_tree_class),
                        reference = as.factor(music_valid$adopter),
                        mode = "prec_recall",
                        positive = '1')

  # Extract and store metrics from confusion matrix
  recall_values <- c(recall_values, cm$byClass["Recall"])
  precision_values <- c(precision_values, cm$byClass["Precision"])
  f1_score_values <- c(f1_score_values, cm$byClass["F1"])
}

# Calculate and print mean of each metric
print(paste("Mean AUC:", round(mean(auc_values),2)))
```

```
## [1] "Mean AUC: 0.74"
```

```r
print(paste("Mean Recall:", round(mean(recall_values, na.rm = TRUE),2)* 100,"%"))
```

```
## [1] "Mean Recall: 81 %"
```

```r
print(paste("Mean Precision:", round(mean(precision_values, na.rm = TRUE),2)* 100,"%"))
```

```
## [1] "Mean Precision: 7 %"
```

```r
print(paste("Mean F1-Score:", round(mean(f1_score_values, na.rm = TRUE),2)))
```

```
## [1] "Mean F1-Score: 0.12"
```

Final Train and Test

We are training model on all the training data and we will test the model finally on the unseen test data

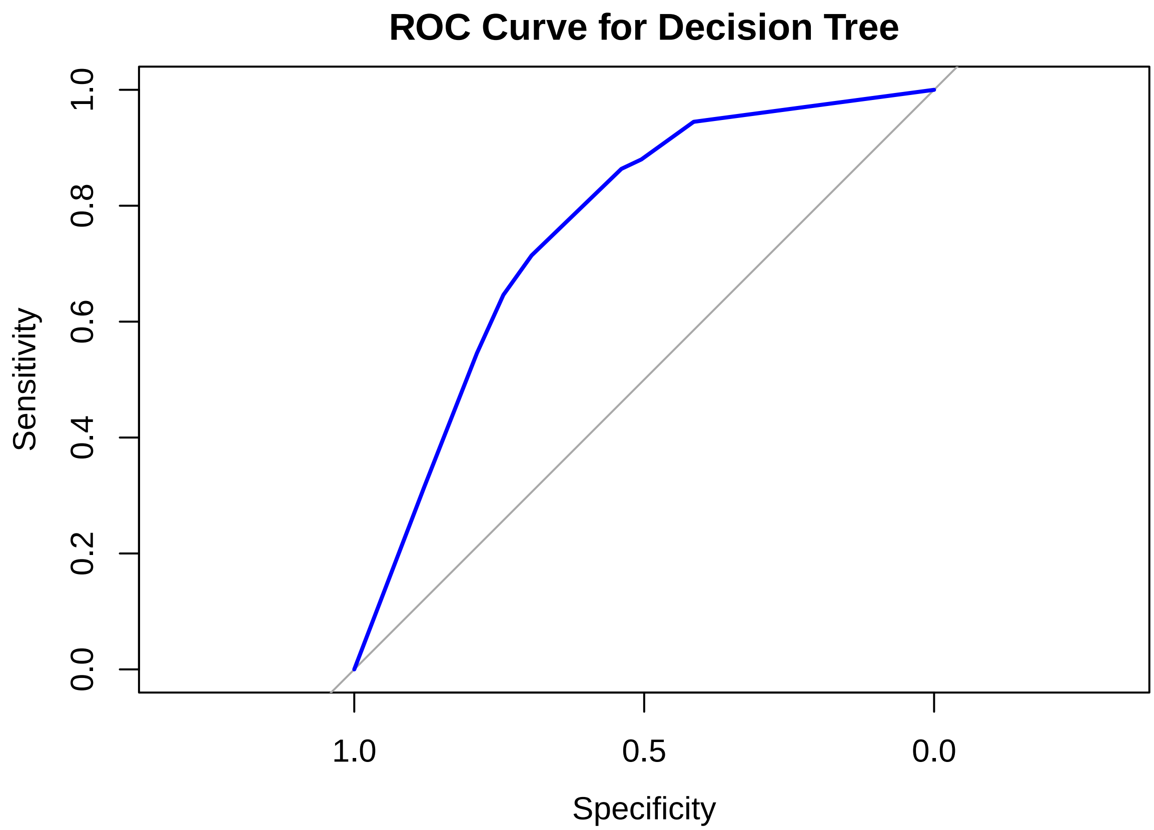
```r
# over-sampling
music_over_sampled <- ovun.sample(adopter~., data = music_train_valid,
                                  method = "over",
                                  p = 0.45, seed = 111)$data

# training Decision Tree
tree = rpart(adopter ~ ., data = music_over_sampled,
             method = "class",
             parms = list(split = "information"),
             control = list(minsplit = 3,
                            maxdepth = 5,
                            cp = 0.01)
             )

# Predict probabilities for the test set (needed for ROC/AUC)
pred_prob_tree <- predict(tree, music_test, type = "prob")[, 2]

# Create ROC curve and calculate AUC
roc_curve <- roc(music_test$adopter, pred_prob_tree)

# Plot ROC curve
plot(roc_curve, col = "blue", main = "ROC Curve for Decision Tree")
```




```r
# Calculate AUC value
auc_value <- auc(roc_curve)
print(paste("AUC Value: ", auc_value))
```

```
## [1] "AUC Value:  0.756074066558442"
```

```r
# Print the confusion matrix
pred_tree_class <- ifelse(pred_prob_tree > 0.3, "1", "0")

confusionMatrix(data = as.factor(pred_tree_class),
                reference = music_test$adopter,
                mode = "prec_recall",
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3317   17
##          1 4683  291
##
##                Accuracy : 0.4343
##                  95% CI : (0.4236, 0.445)
##     No Information Rate : 0.9629
##     P-Value [Acc > NIR] : 1
```

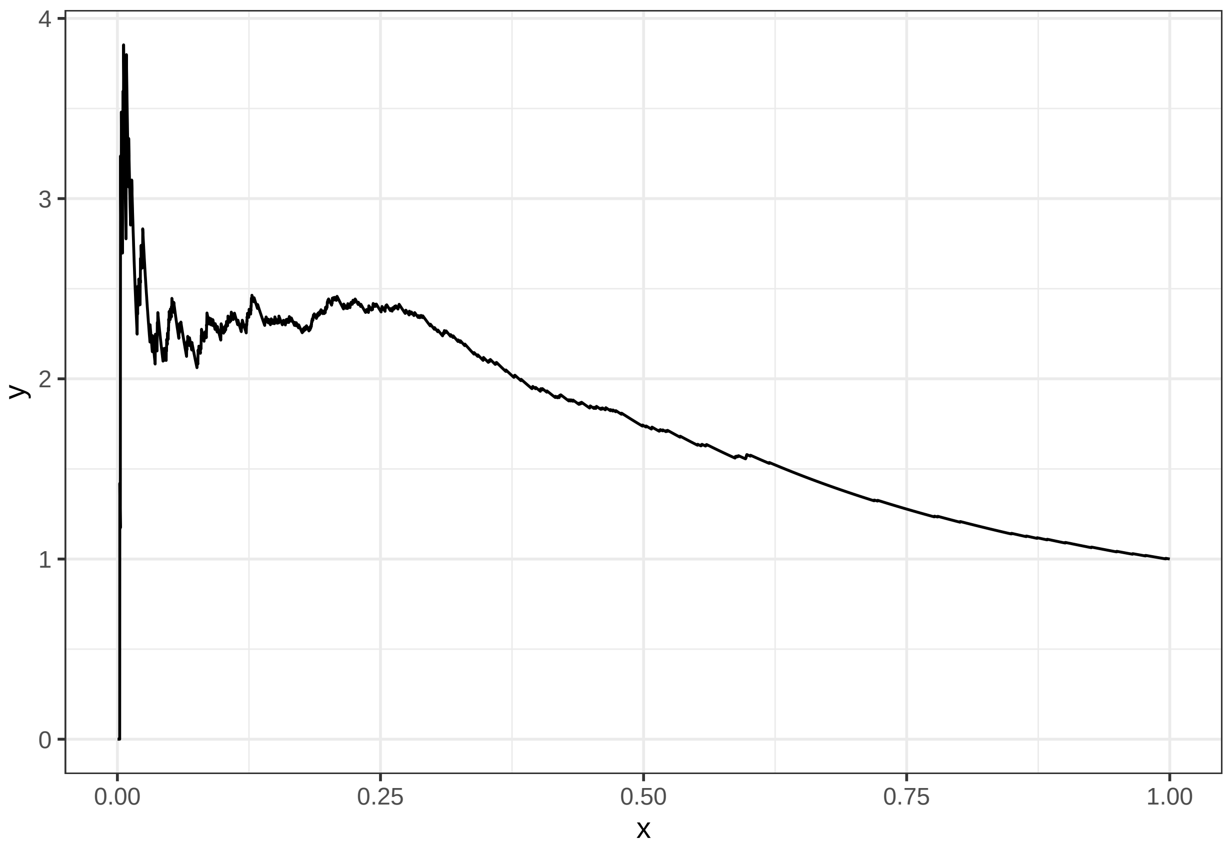
```
##
##                  Kappa : 0.0434
##
##  Mcnemar's Test P-Value : <2e-16
##
##              Precision : 0.05850
##                 Recall : 0.94481
##                     F1 : 0.11019
##             Prevalence : 0.03707
##         Detection Rate : 0.03503
##   Detection Prevalence : 0.59870
##      Balanced Accuracy : 0.67972
##
##       'Positive' Class : 1
##
```

Lift Curve

The lift remains above 2 for a substantial portion of the curve, particularly for the top 10-20% of the customers. This means that targeting up to the top 20% of customers will still result in significant improvement over random guessing.

```
music_test_lift = music_test %>%
  mutate(prob = pred_prob_tree) %>%
  arrange(desc(prob)) %>%
  mutate(adopter_yes = ifelse(adopter==1,1,0)) %>%
  # the following two lines make the lift curve
  mutate(x = row_number()/nrow(music_test),
         y = (cumsum(adopter_yes)/sum(adopter_yes))/x)

ggplot(data = music_test_lift, aes(x = x, y = y)) +
  geom_line() +
  theme_bw()
```



Feature Importance

We want to check what features is the model using to make predictions.

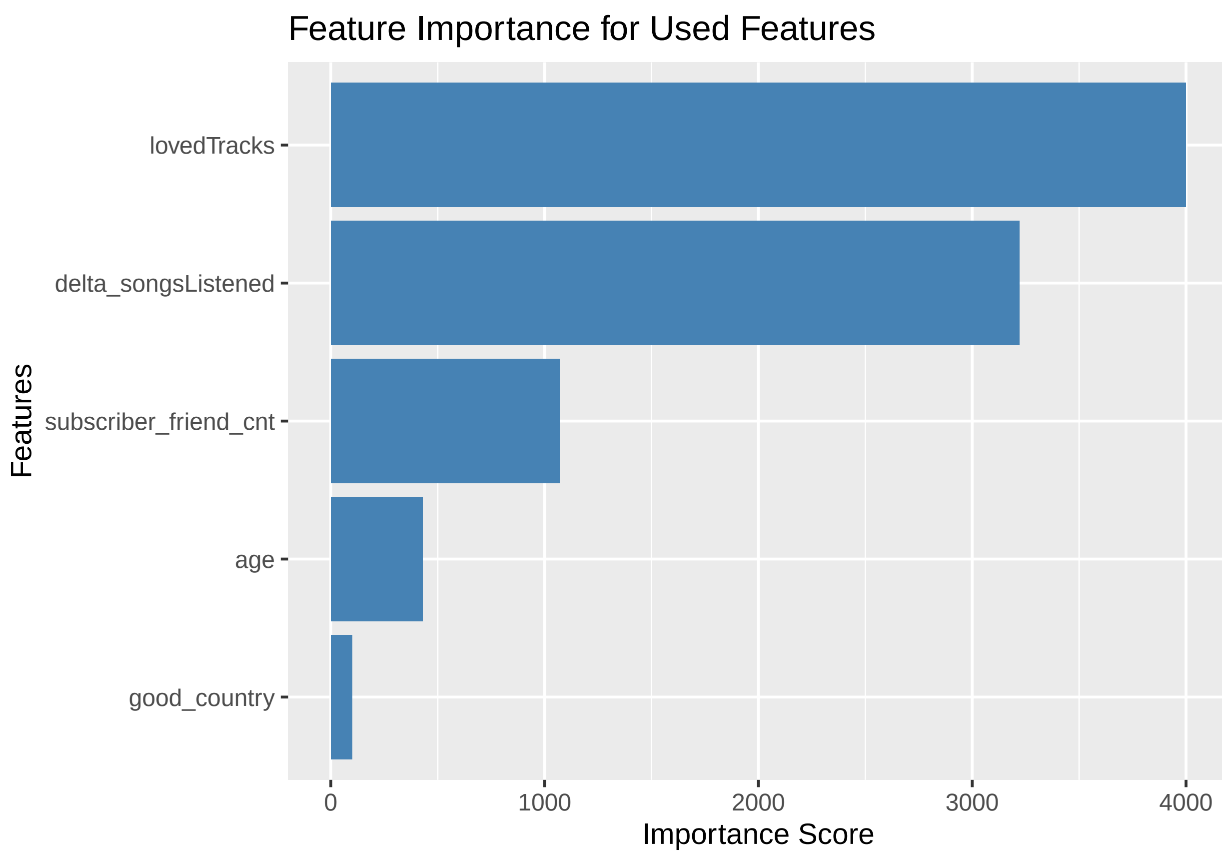
```
# Extract the names of the features used in the splits
used_features <- unique(tree$frame$var[tree$frame$var != "<leaf>"])

# Subset the variable importance to only include used features
importance_used <- tree$variable.importance[used_features]

# Convert to a data frame for easier plotting
importance_df_used <- as.data.frame(importance_used)
colnames(importance_df_used) <- c("Importance")

# Add feature names as a separate column
importance_df_used$features <- rownames(importance_df_used)

ggplot(importance_df_used, aes(x = reorder(features, Importance),
                               y = Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Feature Importance for Used Features", x = "Features",
       y = "Importance Score")
```


Feature Importance for Used Features
lovedTracks
delta_songsListened
subscriber_friend_cnt
age
good_country
Features
0
1000
2000
3000
4000
Importance Score