Logistic Regression & Classification Tree for Predicting Match in Dating App

Clementine Surya & 22200226

2023-06-26

Introduction

In this part, the company developing a new speed dating app wants to build a machine learning system that can match individuals based on the input variables in the data. The task requires fitting a logistic regression model and a classification tree using the data and then comparing their predictive performance using the data_test.

First, we load and prepare the dataset:

```
# Prepare the data running the following lines of code.
# Note that the data are split into two sets: one used to fit the models, the other used for predictions and assessment
data_speed <- read.csv("data_speed_dating.csv", sep = ";")
n <- nrow(data_speed)
set.seed(2023)
set <- sample(1:n, 2290)
data_test <- data_speed[set,]    # for assessing the predictive performance
data <- data_speed[-set,]        # for fitting the models
```

After that, we create the model.

Model creation - Logistic Regression

When evaluating the performance of the models, the company is primarily interested in the ability of the models to correctly identify positive matches between users. Therefore, we identify the optimal threshold τ related to the precision/recall curve and the F_1 score. In this case, the optimal value of τ is the one that maximizes the F_1 score.

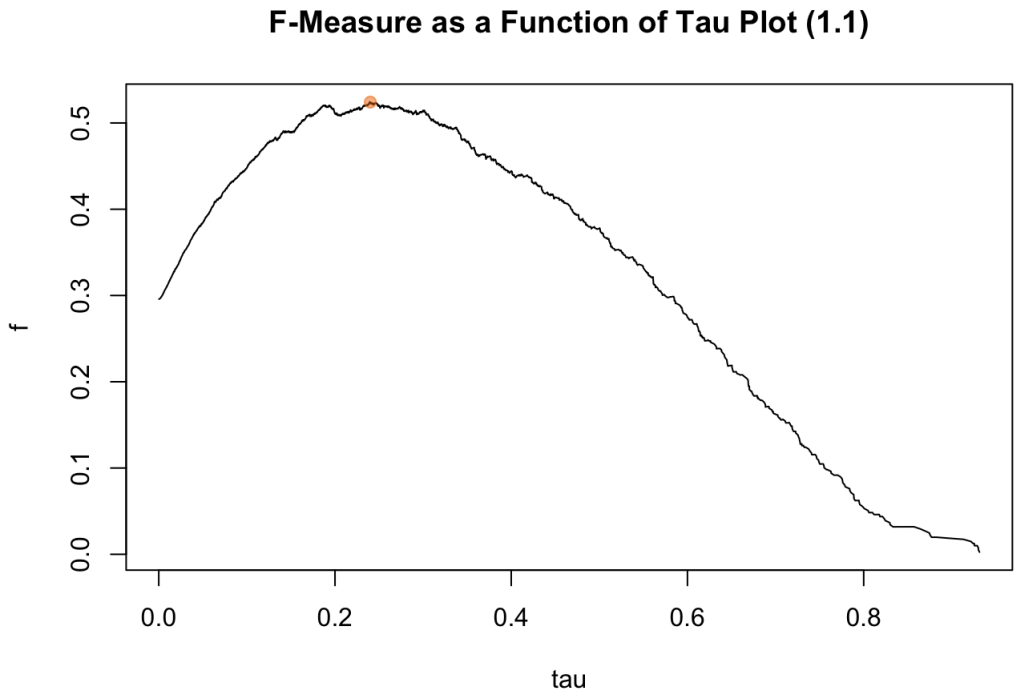
```
# Change "match" column in the train and test sets to binary 1/0 values
data$match <-  ifelse(data$match == "yes", 1,0)
data_test$match <- ifelse(data_test$match == "yes", 1,0)
# Change "gender" column in the train and test sets to factor
data$gender <- factor(data$gender, labels = c("f", "m"))
data_test$gender <- factor(data_test$gender, labels = c("f", "m"))
```

Here, I fit the logistic regression model and find the optimal threshold for classification. Then, I show the confusion matrix to show the predictive performance and show the accuracy, sensitivity, specificity, and precision.

```
# Fit logistic regression model with `match` as the response variable and all other variables as predictors
fit_logit <- glm(match ~ ., data = data, family = binomial)
summary(fit_logit) # Print summary statistics of the logistic regression model
```

```
##
## Call:
## glm(formula = match ~ ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3156  -0.5815  -0.3445  -0.1543   2.9770
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -8.994265   0.370233 -24.294  < 2e-16 ***
## genderm          -0.019451   0.091073  -0.214   0.8309
## age_diff         -0.021760   0.009577  -2.272   0.0231 *
## attractive        0.236224   0.032150   7.348 2.02e-13 ***
## sincere          -0.033484   0.040373  -0.829   0.4069
## intelligent       0.033156   0.048262   0.687   0.4921
## funny             0.187472   0.037365   5.017 5.24e-07 ***
## ambitious        -0.061765   0.037624  -1.642   0.1007
## shared_interests  0.193951   0.029070   6.672 2.52e-11 ***
## like              0.479711   0.032227  14.885  < 2e-16 ***
## guess_liked       0.125160   0.023676   5.286 1.25e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4227.5  on 4579  degrees of freedom
## Residual deviance: 3248.6  on 4569  degrees of freedom
## AIC: 3270.6
##
## Number of Fisher Scoring iterations: 6
```

```
pred_obj <- prediction(fitted(fit_logit), data$match) # Create prediction object with the fitted values from
the logistic regression model and the `match` variable
perf_f <- performance(pred_obj, "f") # Compute the performance of the prediction object in terms of f-measur
e
tau <- perf_f@x.values[[1]] # Extract tau from the performance object
f <- perf_f@y.values[[1]] # Extract f-measure from the performance object
best_pr <- which.max(f) # Find the threshold that maximizes the f-measure

# Plot the f-measure as a function of tau, and mark the threshold that maximizes the f-measure
plot(tau, f, type = "l", main = "F-Measure as a Function of Tau Plot (1.1)")
points(tau[best_pr], f[best_pr], pch = 19, col = adjustcolor("darkorange2", 0.5))
```




```
tau[best_pr] # Return the threshold that maximizes the f-measure
```

```
##      1733
## 0.2400755
```

```
# Classification for optimal tau
# Predict the response variable using the fitted model
pred_logit <- predict(fit_logit, data_test, type="response")

# If the predicted value is greater optimal assign 1, if else 0
pred_logit_out <- ifelse(pred_logit > tau[best_pr], 1, 0)

# Create a confusion matrix of predicted vs actual values
logit_mat <- table(Actualvalue = data_test$match, Predictedvalue=pred_logit_out)
logit_mat # show the confusion matrix
```

```
##            Predictedvalue
## Actualvalue    0    1
##           0 1573  314
##           1  137  266
```

```
# Calculate the accuracy of the model using the confusion matrix
logit_acc <- sum(diag(logit_mat)) / sum(logit_mat)

# Calculate the sensitivity of the model using the confusion matrix
logit_sens <- logit_mat[2, 2] / sum(logit_mat[2, ])

# Calculate the specificity of the model using the confusion matrix
logit_spec <- logit_mat[1, 1] / sum(logit_mat[1, ])

# Calculate the precision of the model using the confusion matrix
logit_prec <- logit_mat[2,2]/sum(logit_mat[,2])

# Print the accuracy, sensitivity, specificity, and precision of the model
cat("Accuracy:", logit_acc)
```

```
## Accuracy: 0.8030568
```

```
cat("\nSensitivity:", logit_sens)
```

```
##
## Sensitivity: 0.6600496
```

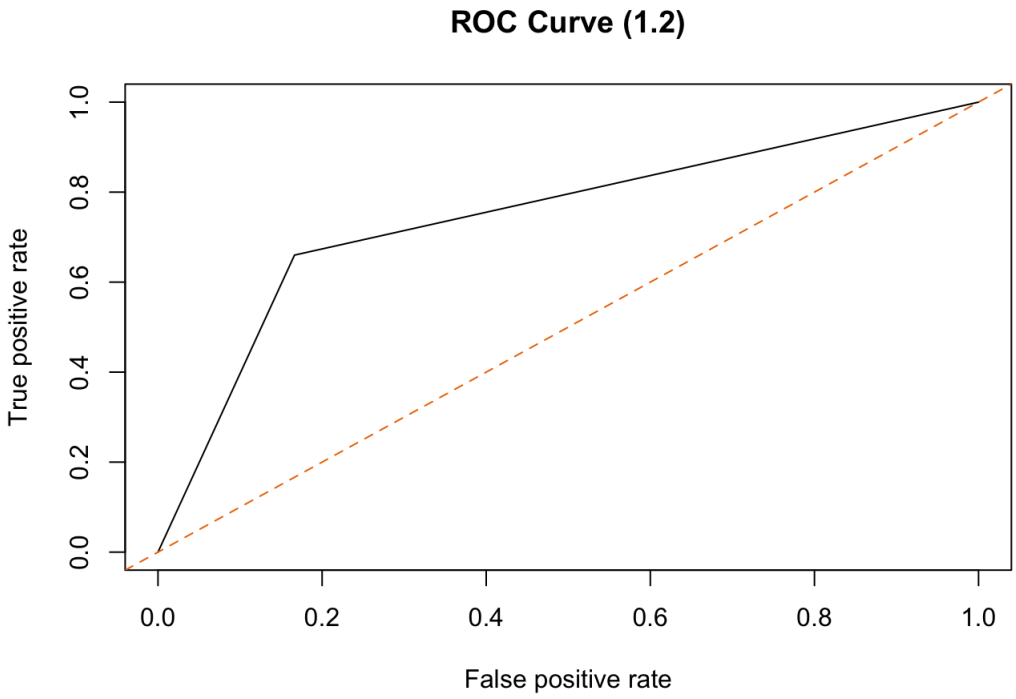
```
cat("\nSpecificity:", logit_spec)
```

```
##
## Specificity: 0.8335983
```

```
cat("\nPrecision:", logit_prec)
```

```
##
## Precision: 0.4586207
```

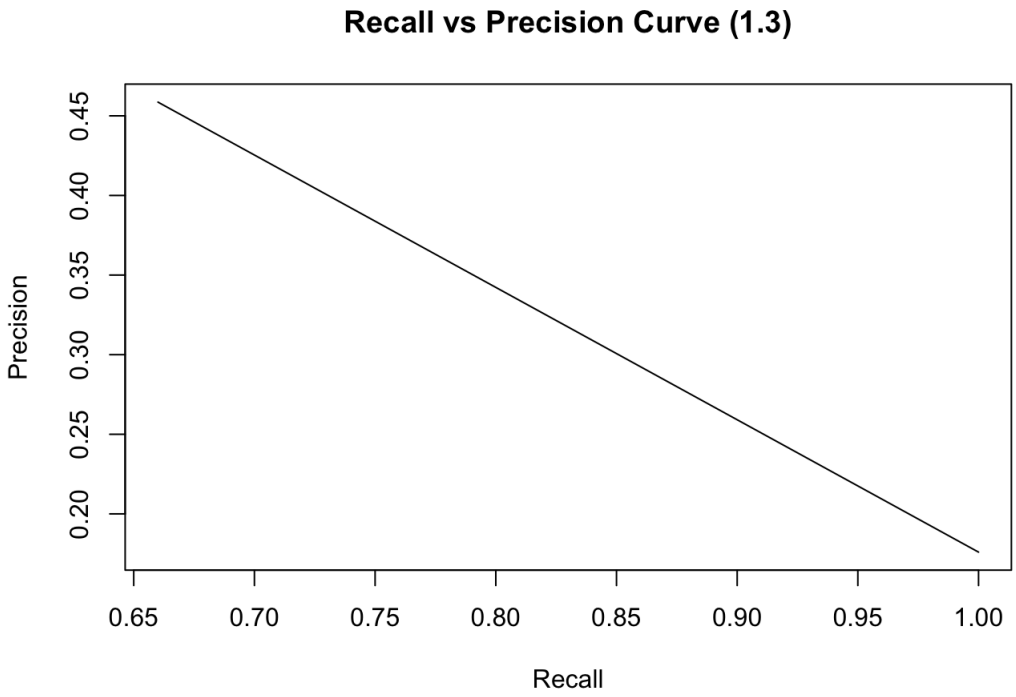
```
# Create a prediction object using the predicted response and actual values of the test data
pred_obj_logit <- prediction(pred_logit_out, data_test$match)

# Calculate the TPR and FPR values for the ROC curve
roc_logit <- performance(pred_obj_logit,'tpr','fpr')

# Plot the ROC curve
plot(roc_logit, main = "ROC Curve (1.2)")

# Add the bisecting line to the plot
abline(0, 1, col = "darkorange2", lty = 2)
```




```
# Calculate the area under the ROC curve (AUC)
auc_logit <- performance(pred_obj_logit, "auc")

# Print the AUC value
auc_logit@y.values
```

```
## [[1]]
## [1] 0.746824
```

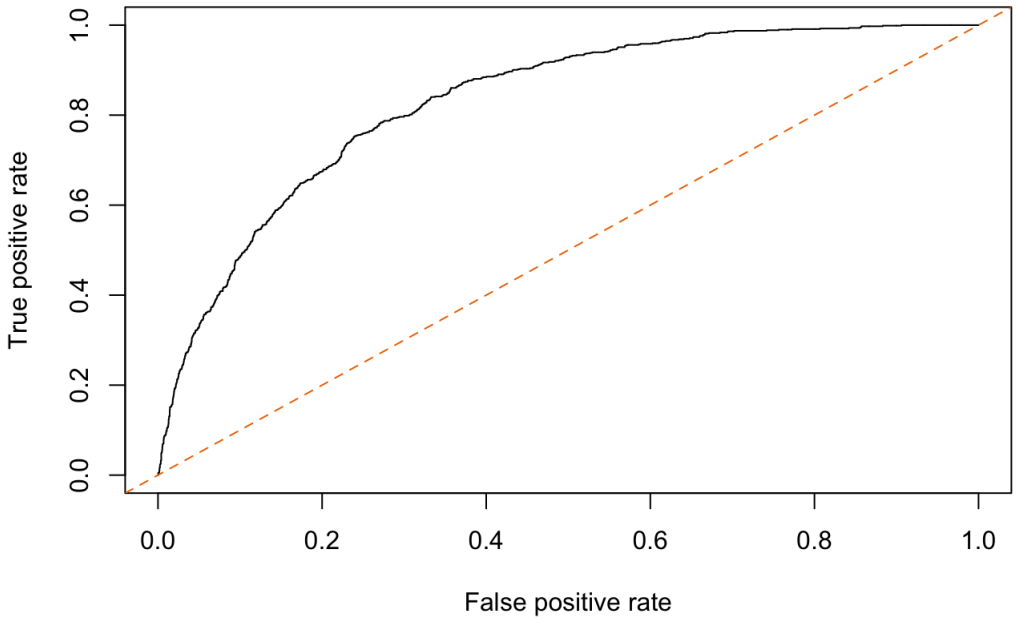
```
# produce precision/recall curve
pr_logit <- performance(pred_obj_logit, "prec", "rec")
plot(pr_logit, main = "Recall vs Precision Curve (1.3)") # show the precision and recall curve
```




```
# compute area under the PR curve
aucpr_logit <- performance(pred_obj_logit, "aucpr")
aucpr_logit@y.values
```

```
## [[1]]
## [1] 0.4805105
```

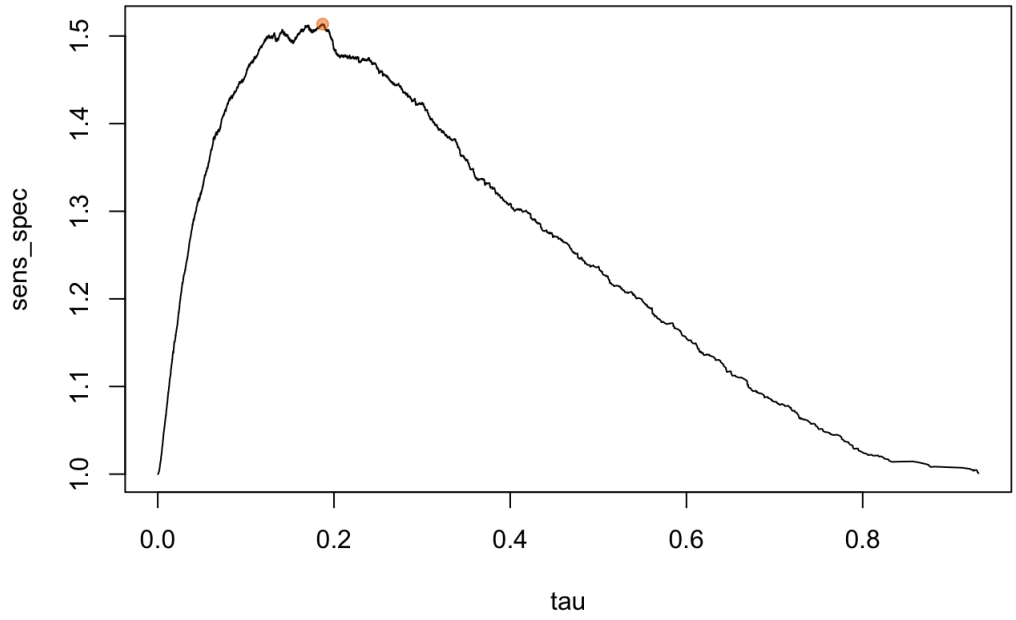
```
library(ROCR)
roc <- performance(pred_obj, "tpr", "fpr")
plot(roc)
abline(0, 1, col = "darkorange2", lty = 2) # add bisect line
```



```
# compute the area under the ROC curve
auc <- performance(pred_obj, "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.8299092
```

```
# note that we access the value of auc in the list with '@'
# this is because the object-list is of class S4

sens <- performance(pred_obj, "sens")
spec <- performance(pred_obj, "spec")
tau <- sens@x.values[[1]]
sens_spec <- sens@y.values[[1]] + spec@y.values[[1]]
best_roc <- which.max(sens_spec)
plot(tau, sens_spec, type = "l")
points(tau[best_roc], sens_spec[best_roc], pch = 19, col = adjustcolor("darkorange2", 0.5))
```




```
tau[best_roc] # optimal tau according to the ROC curve
```

```
##      3392
## 0.1871524
```

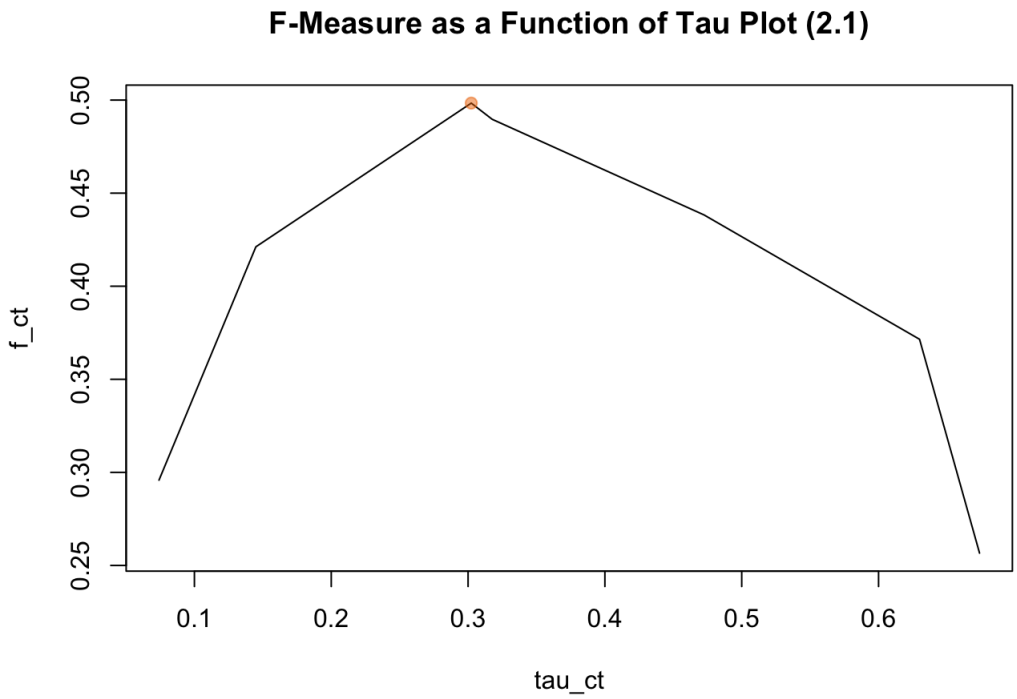
Model creation - Classification Tree

First, I convert target variable match to factor

```
# Change "match" column in the train and test sets factor
data$match <- factor(data$match , labels = c("no","yes"))
data_test$match <- factor(data_test$match , labels = c("no","yes"))
```

Here, I fit the logistic regression model and find the optimal threshold for classification. Then, I show the confusion matrix to show the predictive performance and show the accuracy, sensitivity, specificity, and precision.

```r
# Load the rpart and rpart.plot packages to create and visualize decision trees
library(rpart)
library(rpart.plot)

# Fit a decision tree using the rpart function
fit_ct <- rpart(match~., data = data, method = 'class')

# Predict the response variable using the fitted decision tree
phat <- predict(fit_ct, type = "prob")

pred_obj_ct <- prediction(phat[,2], data$match) # Create prediction object with the fitted values from the l
ogistic regression model and the `match` variable
perf_f_ct <- performance(pred_obj_ct, "f") # Compute the performance of the prediction object in terms of f-
measure
tau_ct <- perf_f_ct@x.values[[1]] # Extract tau from the performance object
f_ct <- perf_f_ct@y.values[[1]] # Extract f-measure from the performance object
best_pr_ct <- which.max(f_ct) # Find the threshold that maximizes the f-measure
# Plot the f-measure as a function of tau, and mark the threshold that maximizes the f-measure
plot(tau_ct, f_ct, type = "l", main = "F-Measure as a Function of Tau Plot (2.1)")
points(tau_ct[best_pr_ct], f_ct[best_pr_ct], pch = 19, col = adjustcolor("darkorange2", 0.5))
```




```r
tau_ct[best_pr_ct] # Return the threshold that maximizes the f-measure
```

```
##      6689
## 0.3023256
```

```r
# Classification for optimal tau
# Predict the probability using the fitted model
pred_ct <- predict(fit_ct, data_test, type = "prob")[,2]

# If the predicted value is greater then optimal assign 1, if else 0
pred_ct_out <- ifelse(pred_ct > tau_ct[best_pr_ct], 1, 0)

# Create a confusion matrix of predicted vs actual values
ct_mat <- table(Actualvalue = data_test$match, Predictedvalue=pred_ct_out)
ct_mat # show the confusion matrix
```

```
##            Predictedvalue
## Actualvalue    0    1
##         no  1630  257
##         yes  184  219
```

```
# Calculate the accuracy of the model using the confusion matrix
ct_acc <- sum(diag(ct_mat)) / sum(ct_mat)

# Calculate the sensitivity of the model using the confusion matrix
ct_sens <-ct_mat[2, 2] / sum(ct_mat[2, ])

# Calculate the specificity of the model using the confusion matrix
ct_spec <- ct_mat[1, 1] / sum(ct_mat[1, ])

# Calculate the precision of the model using the confusion matrix
ct_prec <- ct_mat[2,2]/sum(ct_mat[,2])

# Print the accuracy, sensitivity, specificity, and precision of the model
cat("Accuracy:", ct_acc)
```

```
## Accuracy: 0.8074236
```

```
cat("\nSensitivity:", ct_sens)
```

```
##
## Sensitivity: 0.5434243
```

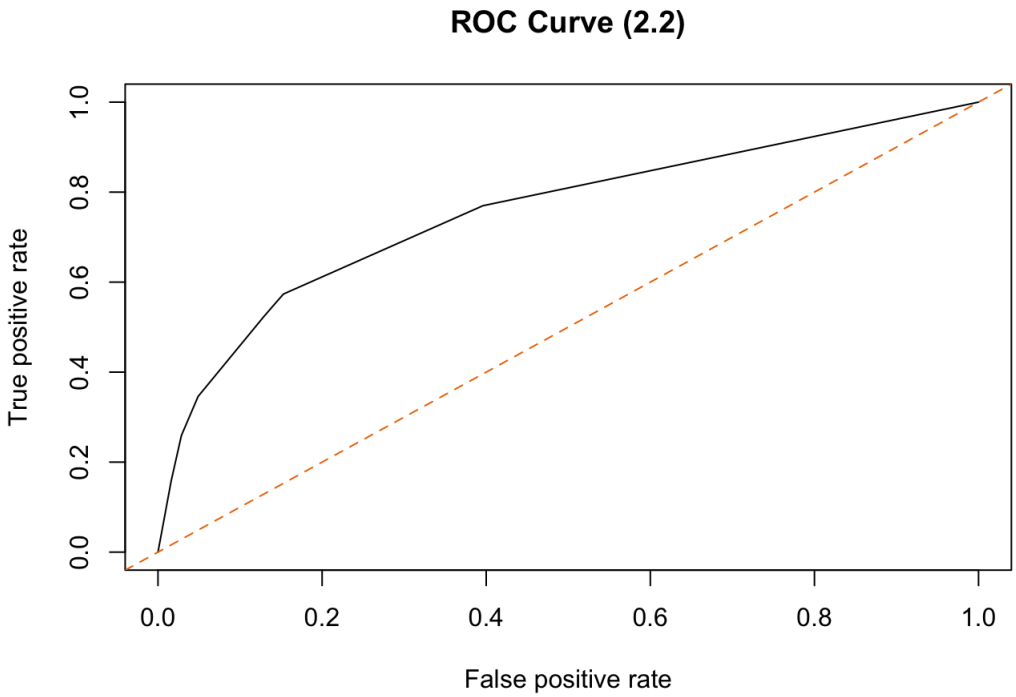
```
cat("\nSpecificity:", ct_spec)
```

```
##
## Specificity: 0.863805
```

```
cat("\nPrecision:", ct_prec)
```

```
##
## Precision: 0.460084
```

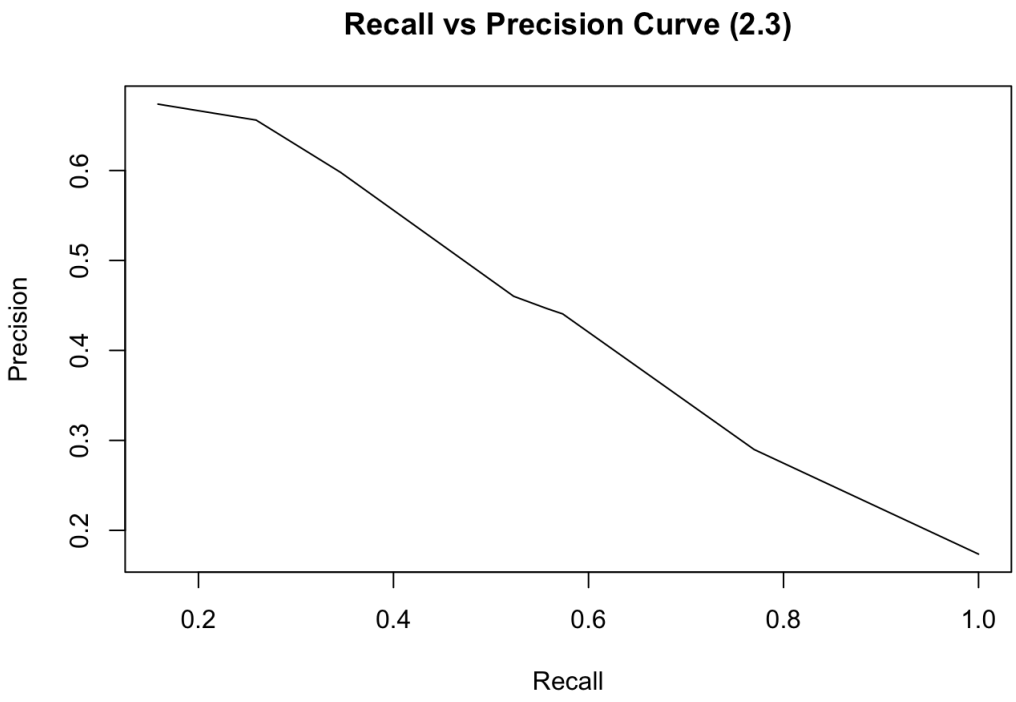
```
# Calculate the TPR and FPR values for the ROC curve
roc_ct <- performance(pred_obj_ct,'tpr','fpr')

# Plot the ROC curve
plot(roc_ct, main = "ROC Curve (2.2)")

# Add the bisecting line to the plot
abline(0, 1, col = "darkorange2", lty = 2)
```




```
# Calculate the area under the ROC curve (AUC)
auc_ct <- performance(pred_obj_ct , "auc")
auc_ct@y.values
```

```
## [[1]]
## [1] 0.7557191
```

```
# produce precision/recall curve
pr_ct <- performance(pred_obj_ct, "prec", "rec")
plot(pr_ct, main = "Recall vs Precision Curve (2.3)") # show the precision and recall curve
```




```
# compute area under the PR curve
aucpr_ct <- performance(pred_obj_ct, "aucpr")
aucpr_ct@y.values
```

```
## [[1]]
## [1] 0.4580235
```

The task was to assess and compare the predictive performance of two classifiers using data_test. The company developing the app is primarily interested in the ability of the models to correctly identify positive matches between users, so both sensitivity and precision are of particular importance. The performance metrics of both models indicate that they do not achieve high predictive sensitivity and precision in detecting positive matches between users.

The logistic regression classifier achieved an accuracy of 80%, sensitivity of 66%, specificity of 83%, and precision of 46%. On the other hand, the classification tree classifier achieved an accuracy of 81%, sensitivity of 54%, specificity of 86%, and precision of 46%. Both models performed similarly in terms of accuracy, specificity, and precision, but there were differences in sensitivity. **Logistic regression had a higher sensitivity compared to classification tree, which is desirable for correctly identifying positive matches between users**.

In addition, the ROC-AUC and PR-AUC values for both models were also evaluated. The logistic regression model had a ROC-AUC of 0.75 and PR-AUC of 0.48, while the classification tree model had a ROC-AUC of 0.76 and PR-AUC of 0.46. Overall, both models had similar ROC-AUC values whereas based on the PR-AUC values, **the logistic regression model had a slightly higher value of 0.48 compared to the classification tree model's value of 0.46**. **This suggests that the logistic regression model slightly better at accurately identify positive matches while minimizing the number of incorrect matches compared to classification tree model**.