K Nearest

Eoin Flynn

18 March 2018

Bond University
Data Science

Contents

Functions

This section will hold all of the functions that will be used throughout this markdown.

```
# Change rows to factors
setRowAsFactor <- function(dataset, columns) {
    for (column in columns) {
        dataset[, column] <- as.factor(dataset[, column])
    }
    return(dataset)
}



# Get the models predictions
getPredictions <- function(model, dataset, outcomeColumn, type = "prob") {
    suppressMessages(library(ROCR))

    probability <- as.numeric(predict(model, newdata = customerDataset,
        type = "prob")[, 2])
    predictions <- prediction(probability, outcomeColumn)

    return(predictions)
}

# Get the model AUC and return it
getModelAUC <- function(model, dataset, outcomeColumn, type = "prob") {
    suppressMessages(library(ROCR))



    prediction <- getPredictions(model = model, dataset = dataset,
        outcomeColumn = outcomeColumn)
    auc <- performance(prediction, "auc")@y.values


    return(auc)
}


# Plot ROC curves
plotROCCurves <- function(model1Prediction, model2Prediction,
    main, model1Colour = "#009900", model2Colour = "#FF8000",
    model1Name, model2Name, legendLocation = "bottomright") {
    suppressMessages(library(ROCR))

    model1Performance <- performance(model1Prediction, "tpr",
        "fpr")
    model2Performance <- performance(model2Prediction, "tpr",
        "fpr")

    plot(model1Performance, main = main, col = model1Colour,
        print.auc = TRUE)
    plot(model2Performance, add = T, col = model2Colour)
```

```r
    legend(legendLocation, legend = paste(rep(c(model1Name, model2Name))),
        col = c(model1Colour, model2Colour), cex = 0.8, fill = c(model1Colour,
            model2Colour))
}

# Calculate AUC. Returns as a decimal
getAUC <- function(outcomeColumn, dataset, model, oneClass, zeroClass) {
    suppressMessages(library(ModelMetrics))
    prediction_df <- createPrediction_df(model, dataset, oneClass = oneClass,
        zeroClass = zeroClass)
    auc <- auc(outcomeColumn, prediction_df$classification)

    return(auc)
}

# Create a confusion matrix. Returns a confusion matrix
createConfusionMatrix <- function(model, dataset, outcomeColumn,
    oneClass, zeroClass) {
    suppressMessages(library(caret))
    prediction_df <- createPrediction_df(model, dataset, oneClass = oneClass,
        zeroClass = zeroClass)
    userConfusionMatrix <- table(prediction_df$classification,
        outcomeColumn)

    return(userConfusionMatrix)
}

# Create a new customer for predicition. Returns a dataframe
createCustomer <- function(originalDataset, gender, SeniorCitizen,
    Partner, Dependents, tenure, PhoneService, MultipleLines,
    InternetService, OnlineSecurity, OnlineBackup, DeviceProtection,
    TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling,
    PaymentMethod, MontlyCharges, TotalCharges, Churn) {
    # Create a copy of the original dataset and keep one row that
    # will be overridden with the new data.
    newCustomer <- customerDataset[1, ]

    newCustomer$gender <- gender
    newCustomer$SeniorCitizen <- SeniorCitizen
    newCustomer$Partner <- Partner
    newCustomer$Dependents <- Dependents
    newCustomer$tenure <- tenure
    newCustomer$PhoneService <- PhoneService
    newCustomer$MultipleLines <- MultipleLines
    newCustomer$InternetService <- InternetService
    newCustomer$OnlineSecurity <- OnlineSecurity
    newCustomer$OnlineBackup <- OnlineBackup
    newCustomer$DeviceProtection <- DeviceProtection
    newCustomer$TechSupport <- TechSupport
    newCustomer$StreamingTV <- StreamingTV
    newCustomer$StreamingMovies <- StreamingMovies
    newCustomer$Contract <- Contract
    newCustomer$PaperlessBilling <- PaperlessBilling
```

```r
    newCustomer$PaymentMethod <- PaymentMethod
    newCustomer$MonthlyCharges <- MontlyCharges
    newCustomer$TotalCharges <- TotalCharges
    newCustomer$Churn <- Churn

    # Convert fields that are factors
    newCustomer <- setRowAsFactor(newCustomer, c("gender", "SeniorCitizen",
        "Partner", "Dependents", "PhoneService", "MultipleLines",
        "InternetService", "OnlineSecurity", "OnlineBackup",
        "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies",
        "Contract", "PaperlessBilling", "PaymentMethod", "Churn"))

    return(newCustomer)
}

# Gets a dataframe from a locally hosted MySQL server.
# Returns a dataframe
loadDataframeFromMySQL <- function(user, password, host = "localhost",
    dbname, statement, port = 3306) {
    suppressMessages(library(RMySQL))

    # Connect to the server
    dataBase <- dbConnect(MySQL(), user = user, password = password,
        host = host, dbname = dbname, port = port)
    # Retrieve the info the from the specified server
    dataframe <- dbGetQuery(dataBase, statement = statement)
    # Close the connection to the server
    dbDisconnect(dataBase)

    return(dataframe)

}

# Returns a predictive model
predictiveModel <- function(formula, dataset, method, neighbours = 1:10,
    metric = "Accuracy", trControl) {
    suppressMessages(library(caret))
    model <- train(formula, data = dataset, method = method,
        tuneGrid = expand.grid(.k = neighbours), metric = metric,
        trControl = trControl)

    return(model)
}

# Returns a train control object to be fed into a predictive
# model
trainControlObject <- function(method = "repeatedcv", number = 10,
    repeats) {
    suppressMessages(library(caret))
    object <- trainControl(method = method, number = number,
        repeats = repeats)

    return(object)
```

```
}

# Create and return a dataframe of the classifications and
# their probabilities
createPrediction_df <- function(model, dataset, predictionType = "prob",
    oneClass, zeroClass) {
    # Run the prediction
    prediction <- suppressWarnings(predict(model, dataset, type = predictionType))
    # Convert to a dataframe
    prediction_df <- data.frame(prediction)
    # Rename the column to reference easier
    colnames(prediction_df) <- c("NegativeProb", "PositiveProb")
    # Add a row for the classification
    prediction_df$classification <- rep(zeroClass, nrow(prediction_df))
    # Convert all probabilites above 0.5 to be the affirmative
    # class
    prediction_df$classification[prediction_df$PositiveProb >
        0.5] <- oneClass
    prediction_df$classification <- as.factor(prediction_df$classification)


    return(prediction_df)
}
```

Data

In this section we will load in our data and do some basic data exploration.

```
customerDataset <- loadDataframeFromMySQL(user = "root", password = "A13337995",
    dbname = "world", statement = "Select * from world.customerChurn")

# Loop through and change all relevant rows to factors and
# returns the dataset post modification
customerDataset <- setRowAsFactor(customerDataset, c("gender",
    "SeniorCitizen", "Partner", "Dependents", "PhoneService",
    "MultipleLines", "InternetService", "OnlineSecurity", "OnlineBackup",
    "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies",
    "Contract", "PaperlessBilling", "PaymentMethod", "Churn"))

# Drop the columns that will not be needed
customerDataset <- customerDataset[, -which(names(customerDataset) %in%
    c("customerID"))]
```

Model

We will create two models, one using all variables from the customer dataset, the other using the top three variables outlined in our decision tree paper. The top three variables are the type on contract, the type of internet service, and the customer's tenure. We will test our models between 3 and 45 neighbours as 1 and 2 neighbours typically have little value and more than 45 may make the model prone to overfitting. After the models have been created we will examine their overall accuracy and how accurate they are when they predict that a customer will churn. Once we have established which model has the greatest accuracy for the business's intended use we will examine if any other number of neighbours close to the one chosen by R would provide a better model for the intended use case.

Model Creation

We will now create out two models and explore their results.

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
set.seed(12216)


allVariableModel <- predictiveModel(formula = Churn ~ ., dataset = customerDataset,
    method = "knn", neighbours = 3:45, trControl = trainControlObject(repeats = 5))

topThreeModel <- predictiveModel(formula = Churn ~ Contract +
    InternetService + tenure, dataset = customerDataset, method = "knn",
    neighbours = 3:45, trControl = trainControlObject(repeats = 5))
allVariableModel
```

```
## k-Nearest Neighbors
##
## 7032 samples
##   19 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 6329, 6329, 6329, 6329, 6329, 6329, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.7512522  0.3332301
##    4  0.7495158  0.3265884
##    5  0.7636237  0.3495568
##    6  0.7629710  0.3468718
##    7  0.7714460  0.3628864
##    8  0.7711625  0.3617007
##    9  0.7791257  0.3774556
##   10  0.7770776  0.3691445
##   11  0.7814279  0.3769666
##   12  0.7803761  0.3714237
##   13  0.7825934  0.3727722
##   14  0.7821668  0.3697731
##   15  0.7828208  0.3706669
##   16  0.7827635  0.3693149
##   17  0.7826787  0.3668893
##   18  0.7831342  0.3660530
##   19  0.7846132  0.3698433
##   20  0.7841578  0.3680674
```

```
##   21  0.7843281  0.3677020
##   22  0.7843851  0.3662550
##   23  0.7843569  0.3646770
##   24  0.7847263  0.3653045
##   25  0.7852386  0.3651236
##   26  0.7852093  0.3639776
##   27  0.7851246  0.3633728
##   28  0.7840721  0.3593145
##   29  0.7839874  0.3580086
##   30  0.7832193  0.3549772
##   31  0.7832469  0.3545445
##   32  0.7827063  0.3518629
##   33  0.7821095  0.3502417
##   34  0.7819674  0.3483380
##   35  0.7827932  0.3499306
##   36  0.7817686  0.3458876
##   37  0.7824518  0.3473010
##   38  0.7827645  0.3474467
##   39  0.7827927  0.3466259
##   40  0.7830206  0.3464548
##   41  0.7831633  0.3465271
##   42  0.7825365  0.3427540
##   43  0.7827927  0.3438432
##   44  0.7823087  0.3414474
##   45  0.7814273  0.3376478
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 25.
```

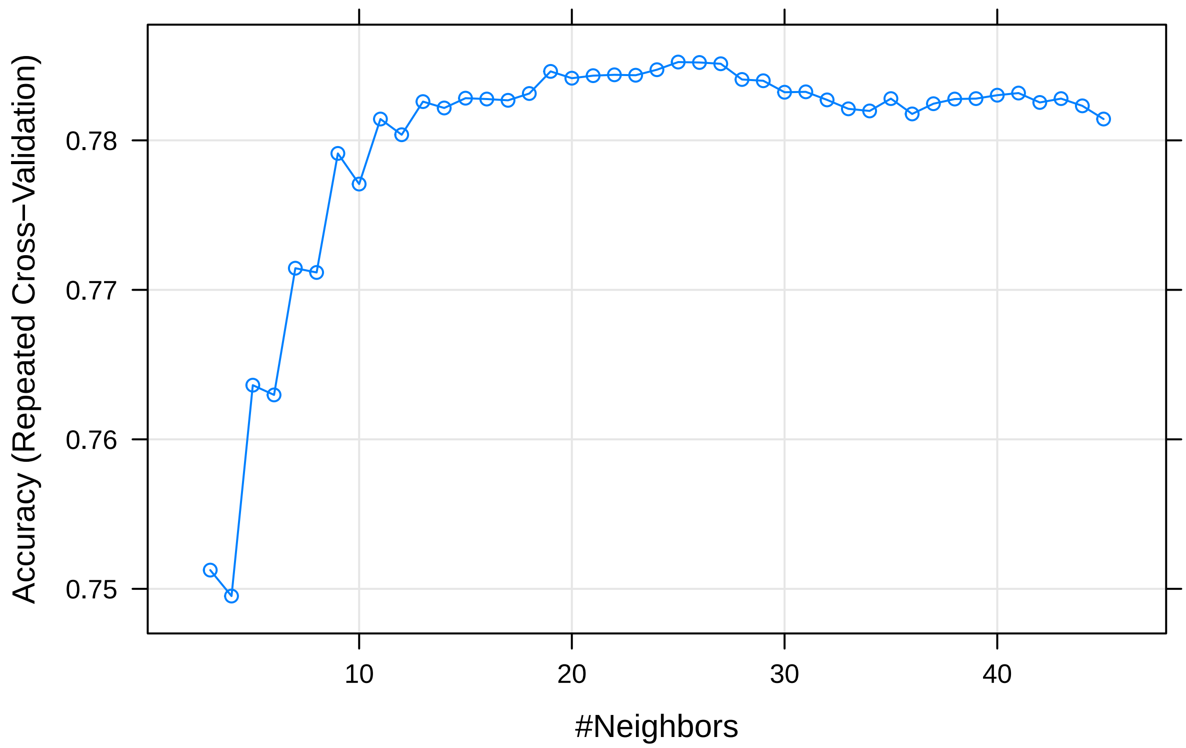
```
topThreeModel
```

```
## k-Nearest Neighbors
##
## 7032 samples
##    3 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 6330, 6329, 6329, 6329, 6328, 6328, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.7810877  0.4101355
##    4  0.7818552  0.4122577
##    5  0.7816849  0.4112815
##    6  0.7826802  0.4132486
##    7  0.7835338  0.4140237
##    8  0.7843868  0.4151292
##    9  0.7846713  0.4154510
##   10  0.7849275  0.4161003
##   11  0.7849271  0.4159485
##   12  0.7851549  0.4166810
##   13  0.7849273  0.4160800
##   14  0.7854392  0.4169988
```

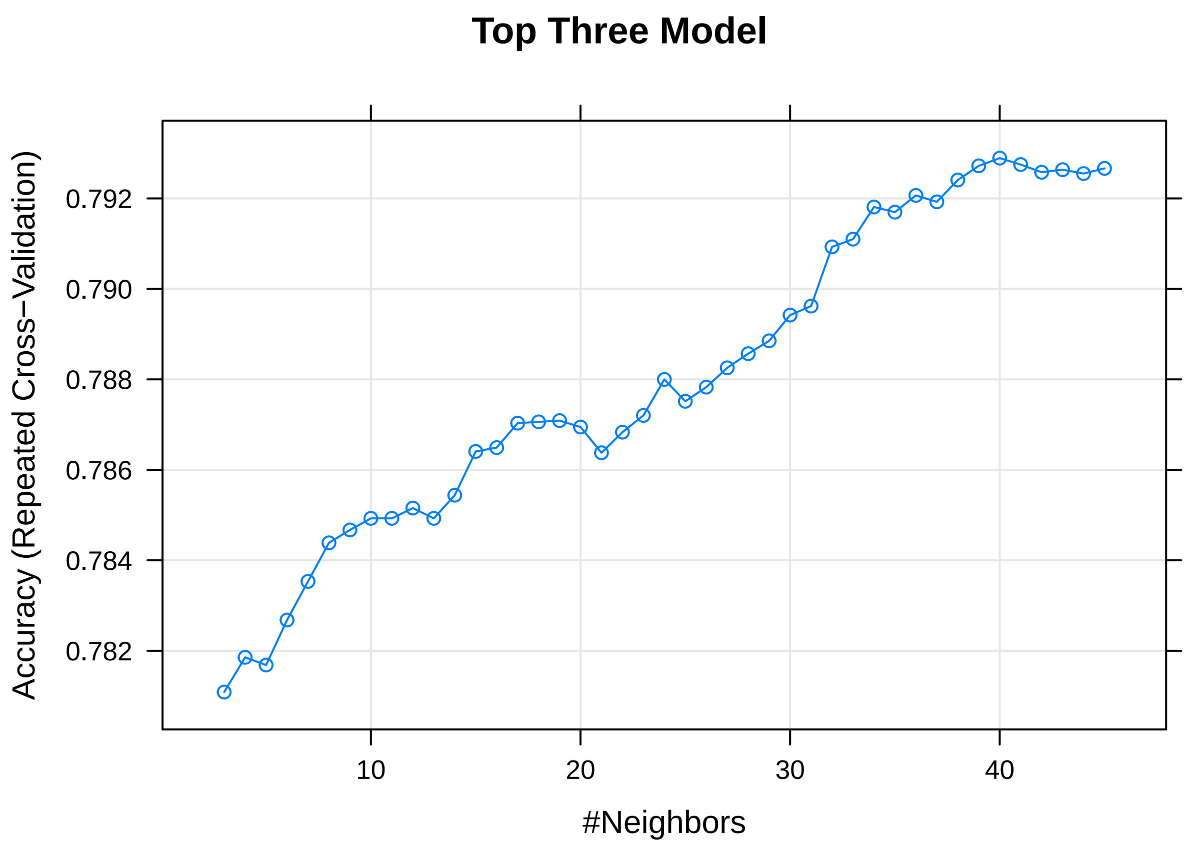
```
##   15  0.7864066  0.4188277
##   16  0.7864915  0.4182480
##   17  0.7870320  0.4185549
##   18  0.7870605  0.4168363
##   19  0.7870886  0.4149821
##   20  0.7869462  0.4131081
##   21  0.7863777  0.4105335
##   22  0.7868327  0.4101205
##   23  0.7872025  0.4092336
##   24  0.7879987  0.4100534
##   25  0.7875155  0.4075588
##   26  0.7878281  0.4080975
##   27  0.7882550  0.4088138
##   28  0.7885680  0.4094095
##   29  0.7888520  0.4095492
##   30  0.7894213  0.4106512
##   31  0.7896203  0.4099117
##   32  0.7909286  0.4115054
##   33  0.7910994  0.4107424
##   34  0.7918103  0.4119777
##   35  0.7916964  0.4108626
##   36  0.7920662  0.4118698
##   37  0.7919239  0.4119186
##   38  0.7924073  0.4137182
##   39  0.7927202  0.4150493
##   40  0.7928910  0.4155434
##   41  0.7927487  0.4150273
##   42  0.7925782  0.4144677
##   43  0.7926349  0.4145292
##   44  0.7925496  0.4141824
##   45  0.7926635  0.4145938
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 40.
```

```
plot(allVariableModel, main = "All Variable Model")
```

All Variable Model




```r
plot(topThreeModel, main = "Top Three Model")
```



###Model Outputs
We can see that each model uses a largely different value of K. Where the All Variable Model uses only 24 neighbours to achieve it's peak accuracy, model 2 uses 40. The plots demonstrate the increases and decreases in accuracy with the change in the number of neighbours.

Model Comparison

The two models produce similar guideline accuracy results being withing 2% of one another however the best strategy for testing their true accuracy is to create an AUC curve and compare the results.

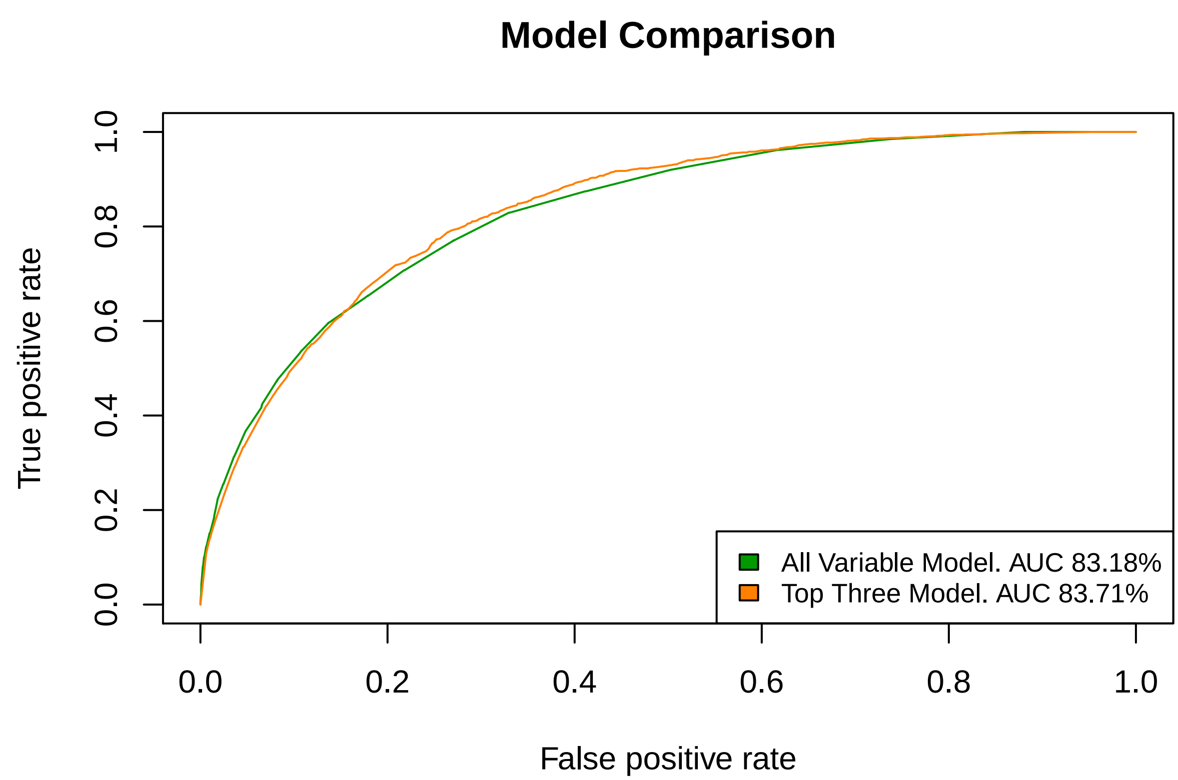
```
# Create the prediction statistics that are used to claculate
# the AUC
model1Predicitions <- getPredictions(allVariableModel, customerDataset,
    customerDataset$Churn)
model2Predicitions <- getPredictions(topThreeModel, customerDataset,
    customerDataset$Churn)

# Create a dataframe of the prediciton results. This df will
# allow us to access the classification results
model1Predicition_df <- createPrediction_df(allVariableModel,
    customerDataset, predictionType = "prob", oneClass = "Yes",
    zeroClass = "No")
model2Predicition_df <- createPrediction_df(topThreeModel, customerDataset,
    predictionType = "prob", oneClass = "Yes", zeroClass = "No")

# Calculate the AUC for each model
```

```
allVariableModelAUC <- as.numeric(getModelAUC(model = allVariableModel,
    dataset = customerDataset, outcomeColumn = customerDataset$Churn,
    type = "prob"))
topThreeModelAUC <- as.numeric(getModelAUC(model = topThreeModel,
    dataset = customerDataset, outcomeColumn = customerDataset$Churn,
    type = "prob"))

# Plot the ROC curves on the same graph for comparison
plotROCCurves(model1Prediction = model1Predicitions, model2Prediction = model2Predicitions,
    main = "Model Comparison", model1Name = sprintf("All Variable Model. AUC %.2f%%",
        allVariableModelAUC * 100), model2Name = sprintf("Top Three Model. AUC %.2f%%",
        topThreeModelAUC * 100))
```




```
# Create a confusion matrix for each model
model1ConfusionMatrix <- createConfusionMatrix(allVariableModel,
    customerDataset, customerDataset$Churn, oneClass = "Yes",
    zeroClass = "No")

model2ConfusionMatrix <- createConfusionMatrix(topThreeModel,
    customerDataset, customerDataset$Churn, oneClass = "Yes",
    zeroClass = "No")
model1ConfusionMatrix
```

```
##      outcomeColumn
##         No  Yes
##   No  4826 1086
```

```
##   Yes  337  783
```

```
model2ConfusionMatrix
```

```
##      outcomeColumn
##         No  Yes
##   No  4736 1016
##   Yes  427  853
```

```
# Calculate the difference in the AUC
aucDifference <- (topThreeModelAUC - allVariableModelAUC) * 100
# Calculate the accuracy of accurately predicting yes
model1YesAccuracy <- (model1ConfusionMatrix[2, 2]/(model1ConfusionMatrix[2,
    1] + model1ConfusionMatrix[2, 2])) * 100
model2YesAccuracy <- (model2ConfusionMatrix[2, 2]/(model2ConfusionMatrix[2,
    1] + model2ConfusionMatrix[2, 2])) * 100
```

Model Comparison Results

Looking at the model's AUC we can see that the top three model is still the most accurate overall, beating the all variable model by 0.53%. Where the top three model begins to faulter is how accurately it predidicts yes as it is only accurate 66.64% where the All Variable Model is correct when it predicts that a customer will churn 69.91%.