Supervised Machine Learning Algorithms

Coffy Andrews-Guo

2023-04-20

## Support Vector Machine (SVM) performance comparison to Decision Trees and Random Forest algorithms

[GitHub Source Code](https://github.com/candrewxs/D622)

### Introduction

Support Vector Machine (SVM), Decision Trees, and Random Forests are popular machine-learning algorithms for classification and regression tasks. These algorithms are used to understand [Kaggle’s Shop Customer Data](https://www.kaggle.com/datasets/datascientistanna/customers-dataset?select=Customers.csv), dataset by predicting customer behavior, identifying patterns, and making data-driven decisions as a binary classification problem.

### Aim of Analysis

The primary objective of this analysis is to evaluate and compare different customer behavior prediction models based on the available dataset. To achieve this, these are the methods:

• Conduct an analysis of the dataset and the features within the dataset to predict whether gender affects product sales.  
• Conduct descriptive data analysis on the Shop’s Customer Data.  
• Select the optimal model that provides the highest level of accuracy.

### Approach

**Overview of the Proposed Models**

The Kaggle dataset, Shop Customer Data, was modeled involving these outlined steps:

• First, the data was downloaded from the Kaggle website, prepared, and preprocessed.  
• Exploratory data analysis was performed to determine hidden relationships and patterns. These preprocessing techniques involved the imputation of null, label encoding, data transformation, and feature selection.  
• After explanatory analysis, the splitting of data into train and test sets were done on the ratio of 3:1. Assigning 75% of the dataset for training purposes and 25% for testing purposes.  
• The attempted ML models, including support vector machine, decision trees, and random forests, were formed and loaded with the testing dataset to categorize them into female and male. Supervised classifiers have been used for the modeling.

**Methods and Methodologies**

The Kaggle website provides data sets on an open data platform for machine learning. The Shop Customer Data analyzes a hypothetical shop’s ideal customers in this project. The customer data is collected through membership cards to provide valuable insights to help a business better understand its customers. The dataset includes 2000 observations and 8 variables. Each variable provides insights into customer behavior, purchasing habits, and other variables that impact their decisions. These are the attributes description:

• Customer ID: A unique identifier assigned to each customer in the dataset.  
• Gender: The gender of the customer, male or female.  
• Age: The age of the customer, usually measured in years.  
• Annual Income: The annual income of the customer.  
• Spending Score: A score the shop assigns based on the customer’s behavior and spending nature.  
• Profession: The occupation or profession of the customer.  
• Work Experience: The number of years of work experience of the customer.  
• Family Size: The size of the customer’s family.

**Discussion of the Methodology**

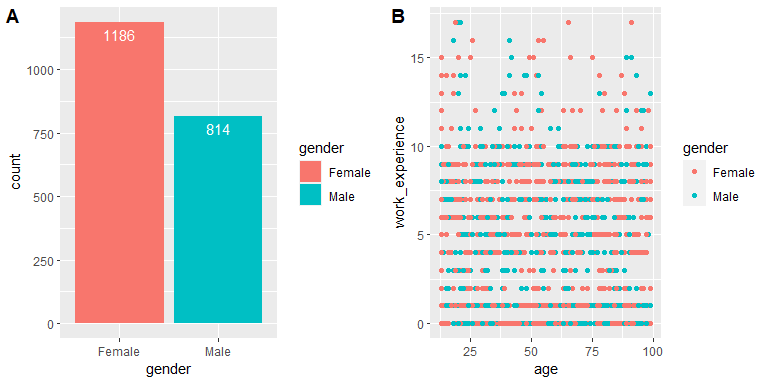
The proposed machine learning methods to predict customer purchasing behavior using the data set features led to a particular essential class, gender. The prediction has been classified into two classes, female and male. This analysis applied the support vector machine (SVM), decision trees, and random forests algorithms to classify the features/attributes into the mentioned classes.

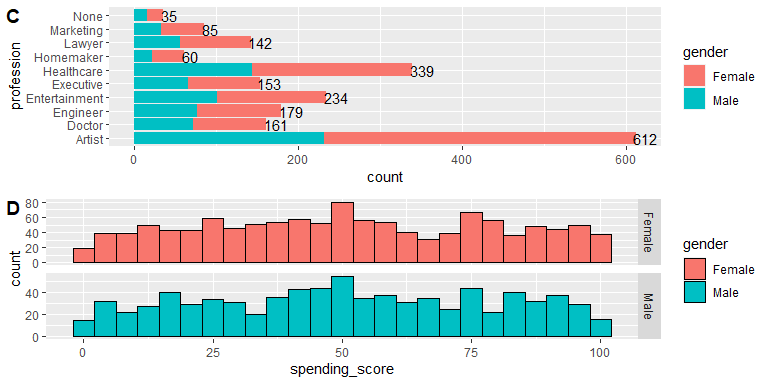
Let’s view the Shop Customer Data data frame (only the first six rows):

| customerid | gender | age | annual\_income | spending\_score | profession | work\_experience | family\_size |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Male | 19 | 15,000 | 39 | Healthcare | 1 | 4 |
| 2 | Male | 21 | 35,000 | 81 | Engineer | 3 | 3 |
| 3 | Female | 20 | 86,000 | 6 | Engineer | 1 | 1 |
| 4 | Female | 23 | 59,000 | 77 | Lawyer | 0 | 2 |
| 5 | Female | 31 | 38,000 | 40 | Entertainment | 2 | 6 |
| 6 | Female | 22 | 58,000 | 76 | Artist | 0 | 2 |

**Visualizations**

A visual representation of the data before applying the machine learning models:





The output column in the dataset (see Plots above) is a categorical column mapped to females or males.

**Plot A** indicates **1186** female customers, significantly higher than the number of male customers at **814**. It indicates a bias towards the female gender and can potentially impact the performance of the machine learning models trained on this dataset.

**Plot B** examines the relationship between age distribution and work experience in a scatterplot. It shows a dense population of customers with work experience ranging between 0 – 10, indicating an overlapping issue.

**Plot C** shows the distribution of customers across professions in the dataset, showing that artists represent the highest number of customers (total of 612).

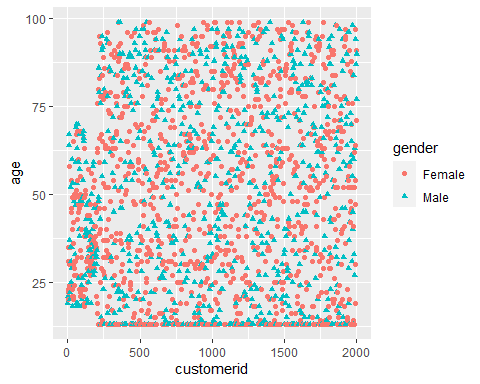
**Plot D** histogram of the spending score between gender indicates that the frequency counts are roughly uniform. However, each class has a notable peak around the 50 range, but the maximum frequency count is higher among females than males.

### Analysis and Discussion of Results

**Building the Models**

The model has multiple covariates. The plot() will only run automatically if the data = argument has exactly three columns (one of which is a response).

We begin by generating the observations, which belong to two classes, and checking whether the classes are linearly separable.



The plot shows that the observations are non-linear.

Next, we fit the support vector classifier. For the *svm()* function to perform classification, the response variable, *gender*, was encoded as a factor variable, and missing variables were imputed.

**Split Data into Training and Test Set**

The reason for selecting SVM is that its kernel can transform low-dimensional input space into high-dimensional space, effectively converting a non-separable problem into a separable one. The partitioned dataset has a ratio of 3:1, where 75 percent is in the training set (2440 observations of 8 variables), and the remaining 25 percent is in the test set (500 observations of 8 variables). To minimize bias and improve accuracy for modeling, the training dataset was balanced using the SMOTE procedure, which stands for *Synthetic Minority Oversampling TEchnique*.

#### Support Vector Machine

The SVM classifier employed a linear kernel to achieve a linear separation of the data using a hyper-plane. The parallel hyper-planes ensured that each data class was separated with as significant a distance as possible to identify the best-fitting model with the cost parameter set to **C = 10.** With the fine-tuned method, the best performance was **C = 0.1.** The confusion matrix represents the classification performance on a test dataset, where the accurate or true values are already known, shown in Fig.2.

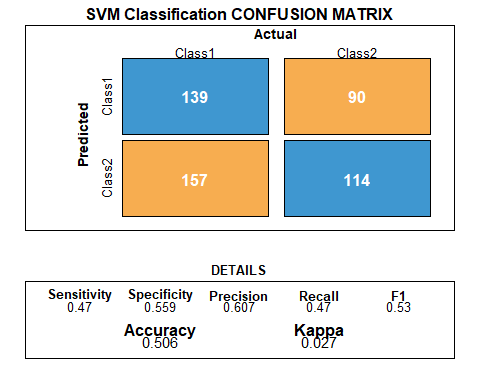


Fig.2: SVM Classification

Fig.2 shows that the SVM model accuracy on the test data is approximately **50.6** percent.

#### Decision Trees

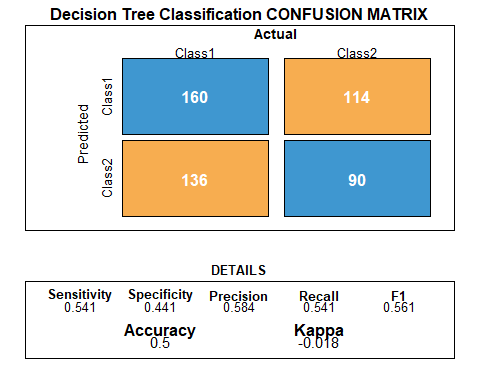


Fig.3 - Decision Tree Classification

Fig.3 shows that the Decision Tree model accuracy on the test data is approximately **50.0** percent.

### Random Forest

Random forest trees are a random subset of the features built on multiple decision trees.

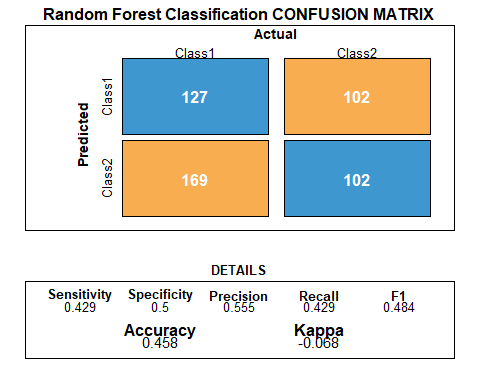


Fig.4 - Random Forest Classification

Fig.4 shows that the Random Forest Tree model accuracy on the test data is approximately **45.8** percent.

### Conclusion: Tabulation of classifiers’ performances

The evaluation metric used to determine which algorithm had the more accurate results were accuracy and kappa because they are easy to interpret.

Precision Kappa  
SVM (Kernal: Linear) 0.6069869 0.02703810  
Decision Tree 0.5839416 -0.01798163  
Random Forest 0.5545852 -0.06750071

Table 2 shows SVM model has a kappa value of 0.0270381, suggesting that the model’s performance is not good and requires improvement. The decision tree kappa value of -0.018 and the Random Forest kappa value of -0.068 suggests that their model performance are inferior and requires significant improvement. **Therefore, the SVM model is the best algorithm for more accurate results.**

The SVM algorithm is better suited for classification tasks rather than regression tasks. While SVM can be applied to classification and regression problems, its primary strength lies in solving them. SVMs work by finding the hyperplane that best separates the different classes, which makes them especially useful for problems where the classes are not linearly separable. However, in regression problems, the goal is to predict a continuous output variable, which may not be well-suited for the binary classification approach used by SVMs. Therefore, while SVMs can be applied to classification and regression tasks, they are generally more effective and commonly used for classification problems.

## References:

* [An Introduction to Statistical Learning with Applications in R](https://hastie.su.domains/ISLR2/ISLRv2_website.pdf)
* [Error in plot, formula missing when using svm](https://stackoverflow.com/questions/25716998/error-in-plot-formula-missing-when-using-svm)
* [How to Plot SVM Object in R (With Example)](https://www.statology.org/plot-svm-in-r/)
* [Support Vector Machine In R: Using SVM To Predict Heart Disease](https://www.edureka.co/blog/support-vector-machine-in-r/)
* [How to Evaluate Machine Learning Algorithms with R](https://machinelearningmastery.com/evaluate-machine-learning-algorithms-with-r/)
* [R Graphics Cookbook](https://r-graphics.org/recipe-bar-graph-labels)

## Appendix: All code for this report

knitr::opts\_chunk$set(echo = FALSE, comment = NA, warning = FALSE, message = FALSE)  
#Load required libraries  
suppressPackageStartupMessages({  
 library("tidyverse")  
 library("e1071") # svm  
 library("caTools") # data partitioning  
 library("DMwR") # imbalance data  
 library("caret") # data preprocessing and transform  
 library("rpart") # decision trees  
 library("randomForest")  
 library("kableExtra") # tidy table  
 library("flextable")  
 library("cowplot") # arrange multiple plots  
 })  
  
#Load the data  
dfhw<- read\_csv("https://raw.githubusercontent.com/candrewxs/D622/main/Customers.csv")  
  
#view dataframe column names  
colnames(dfhw)  
#Change column names to lower case  
names(dfhw) <- tolower(names(dfhw))  
  
#Change column names  
names(dfhw)[4] <- "annual\_income"  
names(dfhw)[5] <- "spending\_score"  
names(dfhw)[7] <- "work\_experience"  
names(dfhw)[8] <- "family\_size"  
  
set\_flextable\_defaults(  
 font.size = 8, theme\_fun = theme\_booktabs,  
 padding = 6,  
 background.color = "#EFEFEF"  
)  
  
flextable(head(dfhw)) %>% align(align = "center", part = "all") %>% autofit()  
  
#find total NA values by column   
sapply(dfhw, function(x) sum(is.na(x)))  
  
#replace missing values in "profession" column with string "None"  
dfhw <- dfhw %>%  
 mutate(profession = replace(profession, is.na(profession), "None"))  
   
#check for NA's  
sapply(dfhw, function(x) sum(is.na(x)))  
  
#Change variables, gender and profession, data type to factor  
#Response variable, gender, encoded as a factor  
dfhw <- dfhw %>%  
 mutate(gender = as.factor(gender)) %>%  
 mutate(profession = as.factor(profession)) %>%  
 mutate(age = ifelse(age <13, 13, age))  
  
#view the statistical summary of the values in the dataframe  
summary(dfhw)  
  
p1 <- ggplot(dfhw, aes(x = gender, fill = gender)) +   
 geom\_bar() +  
 geom\_text(aes(label = ..count..), stat = "count", vjust = 1.5, colour ="white")  
  
#p1  
  
p2 <- ggplot(dfhw, aes(x = age, y = work\_experience)) +  
 geom\_point(aes(colour = gender))  
  
#p2  
  
p3 <- ggplot(dfhw, aes(y = profession)) +  
 geom\_bar(aes(fill = gender)) +  
 geom\_text(aes(label = ..count..), stat = "count", hjust = 0.1, colour ="black")  
  
#p3  
  
p4 <- ggplot(dfhw, aes(x = spending\_score, color = gender, fill = gender)) +  
 geom\_histogram(bins = 25, colour = "black") +  
 facet\_grid(gender ~ ., scales = "free")  
  
#p4  
  
plot\_grid(p1, p2, labels = c('A', 'B'))  
  
plot\_grid(p3, p4, ncol = 1, labels = c("C", "D"))  
  
#creating new object for the models   
dat <- dfhw  
  
ggplot(dat, aes(customerid, age, color = gender, shape = gender)) + geom\_point()  
#create a test and training set   
set.seed(1234)  
split <- sample.split(dat$gender, SplitRatio = 0.75)  
  
training\_set <- subset(dat, split == TRUE)  
test\_set <- subset(dat, split == FALSE)  
  
print(dim(training\_set)); print(dim(test\_set))  
  
#View the target distribution (gender) in the dataset  
round(prop.table(table(select(dat, gender), exclude = NULL)), 4) \* 100  
  
#View the target distribution (gender) in the training set  
round(prop.table(table(select(training\_set, gender), exclude = NULL)), 4) \* 100  
  
#View the target (gender) distribution in the testing set  
round(prop.table(table(select(test\_set, gender), exclude = NULL)), 4) \* 100  
  
#Use SMOTE function to resolve imbalance data (gender - response variable)  
#in the training set  
set.seed(1234)  
training\_set <- SMOTE(gender ~ ., data.frame(training\_set), perc.over = 100, perc.under = 200)  
  
#View balance data (response variable: gender)  
round(prop.table(table(select(training\_set, gender), exclude = NULL)), 4) \* 100  
  
classifier <- svm(gender ~ ., data = training\_set, kernel = "linear", scale = TRUE)  
classifier  
  
set.seed(1234)  
tune\_out <- tune(svm, gender ~ ., data = training\_set, kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 50)))  
  
summary(tune\_out)  
  
bestmod <- tune\_out$best.model  
summary(bestmod)  
  
bestmod\_pred <- predict(bestmod, test\_set[-2])  
(cm\_svm <- confusionMatrix(table(bestmod\_pred, test\_set$gender)))  
  
bestmod\_pred\_table <- table(Prediction = bestmod\_pred, Actual = test\_set$gender)  
bestmod\_pred\_table  
  
draw\_confusion\_matrix <- function(cm\_svm) {  
  
 layout(matrix(c(1,1,2)))  
 par(mar=c(2,2,2,2))  
 plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')  
 title('SVM Classification CONFUSION MATRIX', cex.main=1.5)  
  
 # create the matrix   
 rect(150, 430, 240, 370, col='#3F97D0')  
 text(195, 435, 'Class1', cex=1.2)  
 rect(250, 430, 340, 370, col='#F7AD50')  
 text(295, 435, 'Class2', cex=1.2)  
 text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)  
 text(245, 450, 'Actual', cex=1.3, font=2)  
 rect(150, 305, 240, 365, col='#F7AD50')  
 rect(250, 305, 340, 365, col='#3F97D0')  
 text(140, 400, 'Class1', cex=1.2, srt=90)  
 text(140, 335, 'Class2', cex=1.2, srt=90)  
  
 # add in the cm results   
 res <- as.numeric(cm\_svm$table)  
 text(195, 400, res[1], cex=1.6, font=2, col='white')  
 text(195, 335, res[2], cex=1.6, font=2, col='white')  
 text(295, 400, res[3], cex=1.6, font=2, col='white')  
 text(295, 335, res[4], cex=1.6, font=2, col='white')  
  
 # add in the specifics   
 plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')  
 text(10, 85, names(cm\_svm$byClass[1]), cex=1.2, font=2)  
 text(10, 70, round(as.numeric(cm\_svm$byClass[1]), 3), cex=1.2)  
 text(30, 85, names(cm\_svm$byClass[2]), cex=1.2, font=2)  
 text(30, 70, round(as.numeric(cm\_svm$byClass[2]), 3), cex=1.2)  
 text(50, 85, names(cm\_svm$byClass[5]), cex=1.2, font=2)  
 text(50, 70, round(as.numeric(cm\_svm$byClass[5]), 3), cex=1.2)  
 text(70, 85, names(cm\_svm$byClass[6]), cex=1.2, font=2)  
 text(70, 70, round(as.numeric(cm\_svm$byClass[6]), 3), cex=1.2)  
 text(90, 85, names(cm\_svm$byClass[7]), cex=1.2, font=2)  
 text(90, 70, round(as.numeric(cm\_svm$byClass[7]), 3), cex=1.2)  
  
 # add in the accuracy information   
 text(30, 35, names(cm\_svm$overall[1]), cex=1.5, font=2)  
 text(30, 20, round(as.numeric(cm\_svm$overall[1]), 3), cex=1.4)  
 text(70, 35, names(cm\_svm$overall[2]), cex=1.5, font=2)  
 text(70, 20, round(as.numeric(cm\_svm$overall[2]), 3), cex=1.4)  
}  
  
draw\_confusion\_matrix(cm\_svm)  
  
#model the decision tree algorithm on train set  
cust\_model <- rpart(  
 gender ~ .,   
 method = "class",  
 data = training\_set)  
  
#Predict Decision Tree Model - response variable: gender  
cust\_pred <- predict(cust\_model, test\_set[-2], type = "class")  
head(cust\_pred, 10)  
  
#Confusion matrix for Decision Tree  
cust\_pred\_table <- table(Prediction = cust\_pred, Actual = test\_set$gender)  
cust\_pred\_table  
  
#Caret package confusion matrix  
cm <- confusionMatrix(table(Prediction = cust\_pred, Actual = test\_set$gender), mode = "prec\_recall")  
  
cm  
draw\_confusion\_matrix2 <- function(cm) {  
  
 layout(matrix(c(1,1,2)))  
 par(mar=c(2,2,2,2))  
 plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')  
 title('Decision Tree Classification CONFUSION MATRIX', cex.main=1.5)  
  
 # create the matrix   
 rect(150, 430, 240, 370, col='#3F97D0')  
 text(195, 435, 'Class1', cex=1.2)  
 rect(250, 430, 340, 370, col='#F7AD50')  
 text(295, 435, 'Class2', cex=1.2)  
 text(125, 370, 'Predicted', cex=1.3, srt=90, font=1.5)  
 text(245, 450, 'Actual', cex=1.3, font=2)  
 rect(150, 305, 240, 365, col='#F7AD50')  
 rect(250, 305, 340, 365, col='#3F97D0')  
 text(140, 400, 'Class1', cex=1.2, srt=90)  
 text(140, 335, 'Class2', cex=1.2, srt=90)  
  
 # add in the cm results   
 res <- as.numeric(cm$table)  
 text(195, 400, res[1], cex=1.6, font=2, col='white')  
 text(195, 335, res[2], cex=1.6, font=2, col='white')  
 text(295, 400, res[3], cex=1.6, font=2, col='white')  
 text(295, 335, res[4], cex=1.6, font=2, col='white')  
  
 # add in the specifics   
 plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')  
 text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)  
 text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)  
 text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)  
 text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)  
 text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)  
 text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)  
 text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)  
 text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)  
 text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)  
 text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)  
  
 # add in the accuracy information   
 text(30, 35, names(cm$overall[1]), cex=1.5, font=2)  
 text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)  
 text(70, 35, names(cm$overall[2]), cex=1.5, font=2)  
 text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)  
}   
draw\_confusion\_matrix2(cm)  
postResample(pred =cust\_pred, obs = test\_set$gender )  
set.seed (1234)  
rf\_model <- randomForest(gender ~ ., data = training\_set, importance = TRUE, proximity = TRUE)  
  
rf\_model  
  
  
rf\_pred <- predict(rf\_model, test\_set[-2])  
(rf\_cm <- confusionMatrix(table(rf\_pred, test\_set$gender)))  
  
draw\_confusion\_matrix3 <- function(rf\_cm) {  
  
 layout(matrix(c(1,1,2)))  
 par(mar=c(2,2,2,2))  
 plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')  
 title('Random Forest Classification CONFUSION MATRIX', cex.main=1.5)  
  
 # create the matrix   
 rect(150, 430, 240, 370, col='#3F97D0')  
 text(195, 435, 'Class1', cex=1.2)  
 rect(250, 430, 340, 370, col='#F7AD50')  
 text(295, 435, 'Class2', cex=1.2)  
 text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)  
 text(245, 450, 'Actual', cex=1.3, font=2)  
 rect(150, 305, 240, 365, col='#F7AD50')  
 rect(250, 305, 340, 365, col='#3F97D0')  
 text(140, 400, 'Class1', cex=1.2, srt=90)  
 text(140, 335, 'Class2', cex=1.2, srt=90)  
  
 # add in the cm results   
 res <- as.numeric(rf\_cm$table)  
 text(195, 400, res[1], cex=1.6, font=2, col='white')  
 text(195, 335, res[2], cex=1.6, font=2, col='white')  
 text(295, 400, res[3], cex=1.6, font=2, col='white')  
 text(295, 335, res[4], cex=1.6, font=2, col='white')  
  
 # add in the specifics   
 plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')  
 text(10, 85, names(rf\_cm$byClass[1]), cex=1.2, font=2)  
 text(10, 70, round(as.numeric(rf\_cm$byClass[1]), 3), cex=1.2)  
 text(30, 85, names(rf\_cm$byClass[2]), cex=1.2, font=2)  
 text(30, 70, round(as.numeric(rf\_cm$byClass[2]), 3), cex=1.2)  
 text(50, 85, names(rf\_cm$byClass[5]), cex=1.2, font=2)  
 text(50, 70, round(as.numeric(rf\_cm$byClass[5]), 3), cex=1.2)  
 text(70, 85, names(rf\_cm$byClass[6]), cex=1.2, font=2)  
 text(70, 70, round(as.numeric(rf\_cm$byClass[6]), 3), cex=1.2)  
 text(90, 85, names(rf\_cm$byClass[7]), cex=1.2, font=2)  
 text(90, 70, round(as.numeric(rf\_cm$byClass[7]), 3), cex=1.2)  
  
 # add in the accuracy information   
 text(30, 35, names(rf\_cm$overall[1]), cex=1.5, font=2)  
 text(30, 20, round(as.numeric(rf\_cm$overall[1]), 3), cex=1.4)  
 text(70, 35, names(rf\_cm$overall[2]), cex=1.5, font=2)  
 text(70, 20, round(as.numeric(rf\_cm$overall[2]), 3), cex=1.4)  
}   
draw\_confusion\_matrix3(rf\_cm)  
library("MLmetrics")  
names(cm$byClass)  
names(cm$overall)  
pref\_svm <- c(cm\_svm$byClass[5], cm\_svm$overall[2])  
pref\_dt <- c(cm$byClass[5], cm$overall[2])  
pref\_rf <- c(rf\_cm$byClass[5], rf\_cm$overall[2])  
tab\_pref <- rbind(pref\_svm, pref\_dt, pref\_rf)  
rownames(tab\_pref) <- c( "SVM (Kernal: Linear)", "Decision Tree", "Random Forest")  
(tab.pref <- data.frame(tab\_pref))  
  
#tab.pref <- data.frame(tab\_pref,row.names = c("SVM (Kernal: Linear)", "Decision Tree", "Random Forest"))