ASSIGNMENT 2 - MA5832

- Jarman Giffard [JC225731]

Built with R Version:** 4.1.0**

==========

==========

Part I: An analytical problem

1. Draw a scatter plot to represent the points with Red colour for the class Y = 1 and Blue colour for Y = -1.

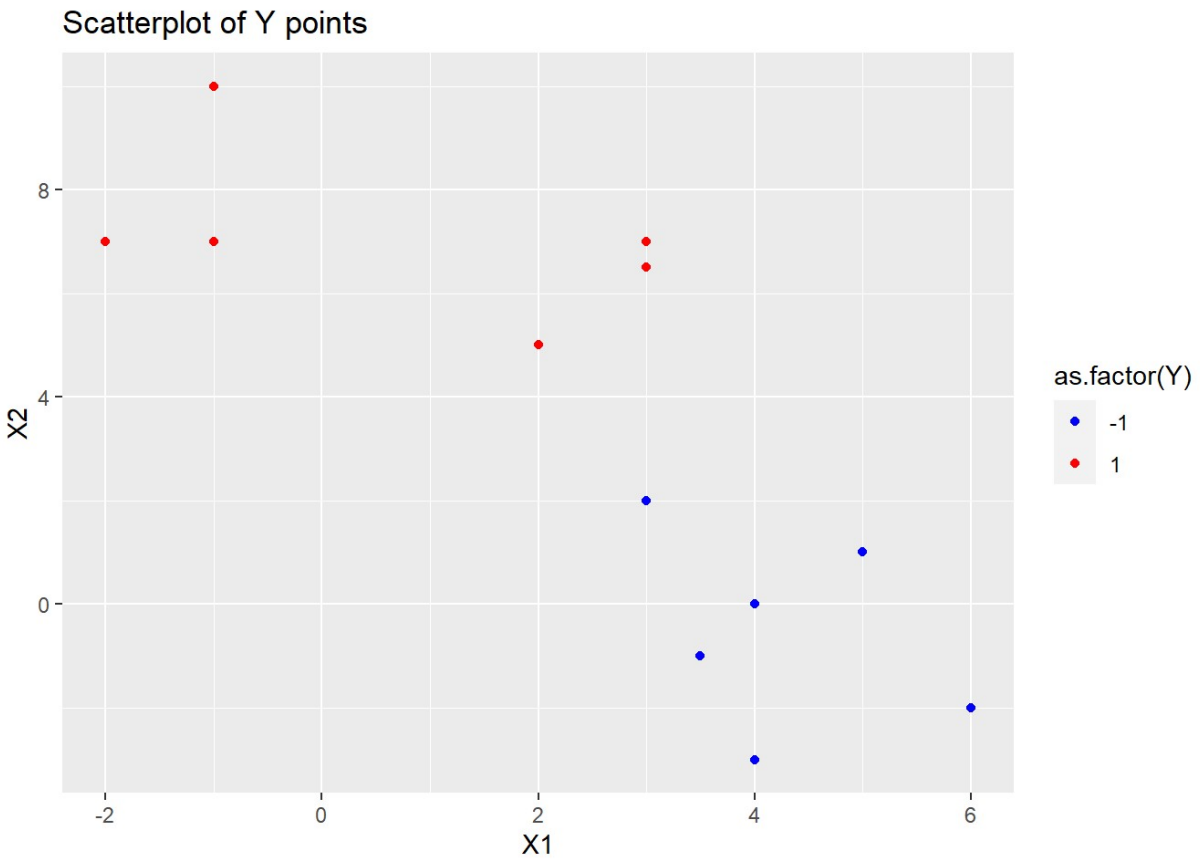
X1 is on the vertical axis while X2 is on the horizontal axis.

```
# Create dataframe with values
p1_df <- data.frame(X1=c(3,4,3.5,5,4,6,2,-1,3,3,-2,-1),
                    X2=c(2,0,-1,1,-3,-2,5,7,6.5,7,7,10),
                     Y=c(-1,-1,-1,-1,-1,-1,1,1,1,1,1,1))
p1_df
```

```
##       X1   X2  Y
## 1   3.0  2.0 -1
## 2   4.0  0.0 -1
## 3   3.5 -1.0 -1
## 4   5.0  1.0 -1
## 5   4.0 -3.0 -1
## 6   6.0 -2.0 -1
## 7   2.0  5.0  1
## 8  -1.0  7.0  1
## 9   3.0  6.5  1
## 10  3.0  7.0  1
## 11 -2.0  7.0  1
## 12 -1.0 10.0  1
```

Dataframe is created in R to match the configuration outlined in pdf.
A scatterplot is then generated and colours assigned; 1 = Red, while -1 = Blue

```
#scatterplot 1 = red / -1 = blue
ggplot(p1_df, aes(x=X1, y=X2, color = as.factor(Y))) +
  geom_point() +
  scale_color_manual(values = c("blue","red")) +
  xlab("X1") +
  ylab("X2") +
  ggtitle("Scatterplot of Y points")
```



2.a Find the optimal separating hyperplane of the classification problem using the function solve.QP() from quadprog in R.

Show your work.

```
# Create matrix
p1_matrix       <- matrix(rep(0, 3*3), nrow=3, ncol=3)
diag(p1_matrix) <- 1
p1_matrix[nrow(p1_matrix), ncol(p1_matrix)] <- 1e-8   # to suppress R complaining about "matri
x is not positive definite."

# Create vector
p1_vector <- c(0,0,0)

# constraint matrix
p1_alt_matrix <- as.matrix(p1_df[, c("X1", "X2")])
p1_alt_matrix <- cbind(p1_alt_matrix, b=rep(-1, 12))
p1_alt_matrix <- p1_alt_matrix * p1_df$Y

# now constraint vector
p1_alt_vect <- rep(1, 12)

# invoke QP solver:
p1_qp <- solve.QP(p1_matrix, p1_vector, t(p1_alt_matrix), bvec=p1_alt_vect)
p1_qp$solution
```

```
## [1] -0.2000001  0.6000000  1.5999998
```

Output of above invocation of *solve.QP()* function indicates that the parameters that characterise the most optimal

hyperplane intersecting the dataframe is at coordinates:

- -0.2000001
- 0.6000000
- 1.5999998

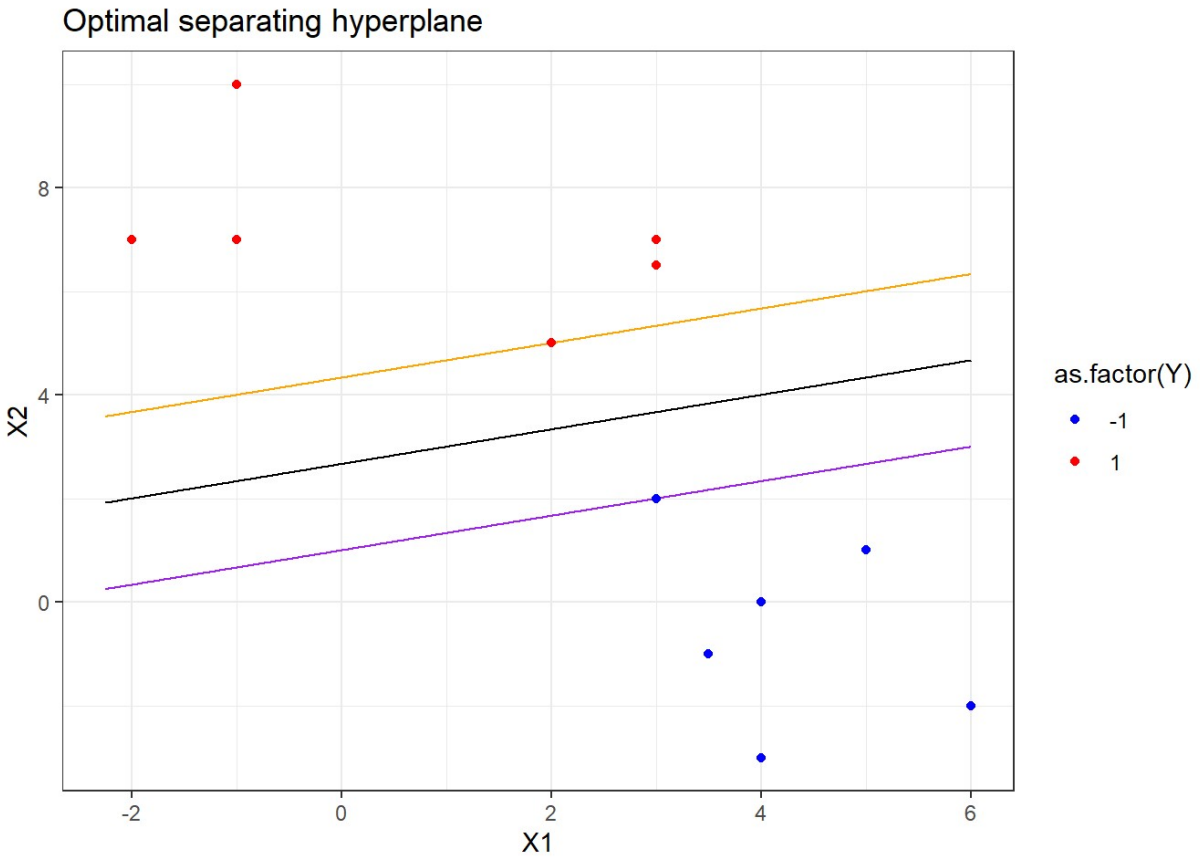
2.b Sketch the optimal separating hyperplane in the scatter plot obtained in Question 1.

The above coordinates are then taken and plotted against the datapoints set in the previous dataframe via use of the *ggplot()* inbuilt R function.

```
w <- p1_qp$solution[1:2]
b <- p1_qp$solution[3]

p1_func_plotMargin <- function(w = 1*c(-1, 1), b = 1){
                      x1 = seq(-20, 20, by = .01)
                      x2 = (-w[1]*x1 + b)/w[2]
                      l1 = (-w[1]*x1 + b + 1)/w[2]
                      l2 = (-w[1]*x1 + b - 1)/w[2]
  dt <- data.table(X1=x1, X2=x2, L1=l1, L2=l2)
  ggplot(dt)+
    geom_line(aes(x=X1, y=X2))+
    geom_line(aes(x=X1, y=L1), color="orange")+
    geom_line(aes(x=X1, y=L2), color="purple" )
}

p1_func_plotMargin(w=w, b=b)+
  geom_point(data=p1_df, aes(x=X1, y=X2, color=as.factor(Y)))+
  scale_color_manual(values = c("blue","red"))+
  xlim(-2.25, 6)+
  ylim(-3, 10)+
  xlab("X1") +
  ylab("X2") +
  ggtitle("Optimal separating hyperplane") +
  theme_bw()
```



Three parallel lines are produced and superimposed over the datapoints, the outer lines representing the *margins* - while the inner black line represents the maximal hyperplane. (*I. Witten, C. J. Pal 2017*)

3. Describe the classification rule for the maximal margin classification.

As plotted above, the three respective lines signifying the hyperplanes are calculated using the below equations:

- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$ if $Y = 1$
  - for the upper, outer margin
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 < 0$ if $Y = -1$
  - for the lower, outer margin
- $Y(\beta_0 + \beta_1 X_1 + \beta_2 X_2) > 0$
  - for the optimal hyperplane

Part II: An application

The data, Data.xls“, is based on Yeh and hui Lien (2009).
The data contains 30,000 observations and 23 explanatory variables.
The response variable, Y, is a binary variable where \1” refers to default payment and \0" implies non-default payment.

2.2 Assessment Tasks

2.2.1 Data

(a) Select a random sample of 70% of the full dataset as the training data, retain the rest as test data.

Provide the code and print out the dimensions of the training data.

```
#split dataset
# 2.2.1 Data
rows <- sample(nrow(ccard)) #randomly shuffle rows around
ccard <- ccard[rows, ]

cc_split <- createDataPartition(ccard$Y, p = 0.7, list = FALSE)

cc_train <- ccard[cc_split, ]
cc_test <- ccard[-cc_split, ]
dim(cc_train)   # print dimensions of training data
```

```
## [1] 21001    24
```

```
#table(cc_train$Y) %>% prop.table() # Proportions of response var
```

21,001 rows / 24 columns.

2.2.2 Tree Based Algorithms

(a) Use an appropriate tree based algorithm to classify credible and non-credible clients.

Specify any underlying assumptions.

Justify your model choice as well as hyperparameters which are required to be specifed in R.

As this model is presumably to be implemented into a business environment, it is of greater importance to ensure that any outputs produced are interpretable by the business so that:

- business stakeholders understand which factors influence the outcome of decisions made using customer data
- any rules implemented that determine the credit limit of customers are understood by the regulatory governing bodies that oversee the industry
    - e.g. if prompted to explain how gender determines how rules should play affect customer applications/rules
- internal rulesets that are created to reflect the outcome of this model are generally written explicitly
    - e.g. using SQL - which requires the output of the model to be translated into typed commands
    - the model shouldn't operate as a 'black-box' process that is difficult to interpret

There is a compelling counterargument to this choice however; there is an observed imbalance in the response variable where there are 6,636 observations that have a default (*Y = 1*) - out of the total count of 30,000 (22.12%).
Strong imbalance is not something that decision trees handle very well (*Brownlee, J. 2020, August 21*) - but is something an alternate tree-based approach can work with; **Random Forests** (*Brownlee, J. 2021, January 5*).
It is also likely that using Random Forests would results in superior accuracy from the output of this model - offering both overall better performance at classifying future default customers and non-default customers.
The key downside is that the outputs of Random Forest models are difficult to interpret and implement in a business setting, making it challenging to explain to stakeholders how individual variables contribute to a decision made on them - and difficult for the business to create rules using the information gleamed from the models. (*Revert, F. 2019, February 7*)

Consequently, superior **interpretability** has been ranked as more important for this model requirement - over (potential) superior **accuracy**.

Note: in many countries, rules implemented in financial organisations such as banks and credit providers are not allowed to factor in non-financial customer attributes such as age, gender, marital status etc as any rules that use these features can be construed as a form of discrimination.
As a result, models implemented in companies that adhere to these requirements would ordinarily remove these variables from an input dataset before being used in a model - as any rules developed using the output of this model may use these variables to make a determination in the classification.

For this exercise these variables have *not* been excluded as it remains a theoretical exercise.

Parameters used:

- splitting method has been left as the default option of GINI as opposed to an alternative splitting/indexing approach such as Information Gain, as GINI has superior performance for large datasets (*W. 2016, February 27*)

minsplit

- minsplit sets a threshold of smallest number of observations that fall into each node that allows it to split further
    - e.g. if minsplit is set to 100 and 99 or fewer observations fall into a leaf, it will remain a leaf and cannot become a terminal/parent node
- for this model the minsplit was set to 250 - significantly higher than the default of 20
- this higher threshold was selected due to the large number of observations in the training dataset and was finalised through repeated iterations of values from 50 - 500

minbucket

- minbucket is a parameter similar to minsplit but instead of determining if a leaf should split, it determines whether a leaf is initialised based on the number of observations that fall into it
- for this model, a threshold of 65 was selected - again to reflect the large number of observations in the dataset but also as 92% of observations fell into one of two leaves in the decision tree
- indicating that a higher degree of granularity from many additional leaves would result in diminishing returns
    - i.e. as two leaves captured 92% of results, having a large number of additional leaves would result in higher computational complexity that would not also result in a much higher model accuracy

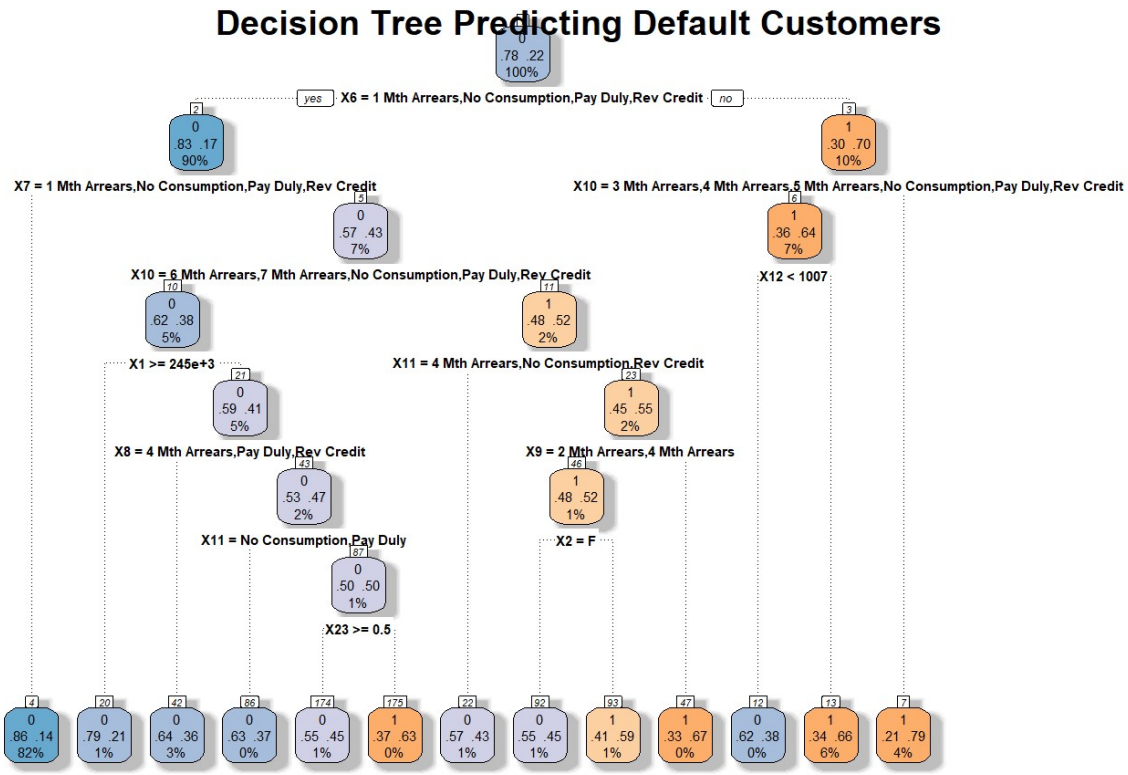
cp

- cp (Complexity Parameter) - is the argument used to set a threshold for computational complexity and determines the overall size of the tree
- its primary function is described as “to save computing time by pruning off splits that are obviously not worthwhile.” (rpart.control function - RDocumentaion)
- for this model the cp was set to 0.001
- this was selected through means of multiple iterations in bandings below and above - with worse accuracy being noted for cp values > 0.0017 or < 0.0008
    - this is further highlighted by the output of the invoked *printcp()* function used further below in part c

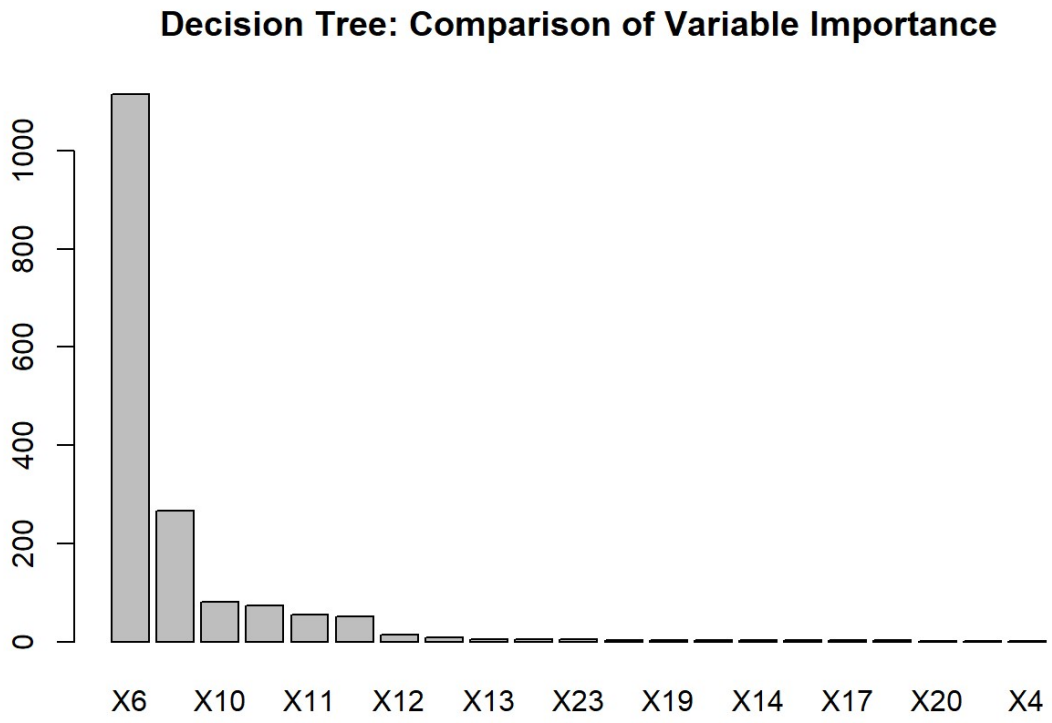
```
# create decision tree
dec_tree <- rpart(Y ~ ., data=cc_train, control=rpart.control(minsplit=250, minbucket=65, cp=
0.0008))
```

(b) Display model summary and discuss the relationship between the response variable versus selected features.

```
# PLOT DECISION TREE
#summary(dec_tree)
fancyRpartPlot(dec_tree,main="Decision Tree Predicting Default Customers",sub="",palettes=c("P
uBuGn", "Oranges"))
```




```
barplot(dec_tree$variable.importance,main = "Decision Tree: Comparison of Variable Importanc
e")
```



The summary of the decision tree object is printed through use of the *summary()* function, which presents a verbose itemised view of the factors that contributed to the splits made in the tree.
As this output of this function is very verbose and supplies a large amount of information the output has been suppressed in this instance.
The decision tree itself is provided - illustrating how the algorithm has split the customers and which variables were selected to form the questions.
The decision tree model was produced using the *rpart* function from the equivelant package, while the printing of decision tree with aesthetic colour scheme was done so through use of the *fancyRpartPlot* from the rattle and *RColorBrewer* packages respectively.
Finally, the significant variables used to determine the branches in the decision tree are listed in order of rank in a barplot - through use of the inbuilt R function *barplot()*.
As seen from the bar plot - the variable X6 is by far the most notable of the important variables; several orders of magnitude higher in importance compared to the next most important, X7.
Next is X10, X9, and X11.
These 5 variables represent the history of prior payments for each month over a period of time between April 2005 - September 2005 in reverse order, with X6 signifying payments made in September 2005 while X11 represents April.
X6's strong display of significance indicates that the ability to accurately predict whether a customer will default is strongly aligned with that customer's history of repayments from the last month in the historic dataset, while the earlier months are less predictive.
At the very far end of the barplot are the 5 variables X4, X22, and X20 - representing marital status, and the amount of money deposited in the repayments for the months of 2005.
These variables consequently offer very little information that can be used to determine their disposition towards future defaults.
Another key takeaway from this is the order of the months on the left side - starting from X6, X7, X10, and then X9. This highlights that the order of importance for each month is not necessarily in strict order or reverse order of the month itself, but that different months in the sequence are better indicators than others (e.g. August is a better indicator than May, which is a better indicator than June etc)

From the view of the decision tree directly this is emphasised by the very first node of the tree (the root) making the initial classification of customer segments based on whether they were in arrears in September 2005 - and if so, immediately placing almost all of the customers that fall into this node in the 'delinquent' (Y = 1) outcome.

(c) Evaluate the performance of the algorithm on the training data and comment on the results.

The printcp() function is invoked to display the number of branches generated at different values of specified cp.

```
cc_train2 <- cc_train
# NOW USE RESULTS TO PREDICT - USING THE TRAIN DATASET
tree_pred = predict(dec_tree,cc_train2,type="class")
confMat <- table(cc_train2$Y,tree_pred)
accuracy <- sum(diag(confMat))/sum(confMat)
accuracy
```

```
## [1] 0.8245798
```

```
printcp(dec_tree) #cp table showing the improvement in cost complexity at each node
```

```
##
## Classification tree:
## rpart(formula = Y ~ ., data = cc_train, control = rpart.control(minsplit = 250,
##     minbucket = 65, cp = 8e-04))
##
## Variables actually used in tree construction:
##  [1] X1  X10 X11 X12 X2  X23 X6  X7  X8  X9
##
## Root node error: 4646/21001 = 0.22123
##
## n= 21001
##
##          CP nsplit rel error  xerror     xstd
## 1 0.1870426      0   1.00000 1.00000 0.012947
## 2 0.0021524      1   0.81296 0.81468 0.011989
## 3 0.0015067      6   0.80069 0.81533 0.011993
## 4 0.0011838      8   0.79768 0.81403 0.011986
## 5 0.0008000     12   0.79294 0.81834 0.012010
```

Firstly, a copy of the training dataset is created and renamed.
This is then provided as the second input dataset for the inbuilt *predict()* function, making a prediction of the training data using the model output of the training data produced above.
The overall accuracy of the model is a promising **82.4%**, with **15.9%** of non-default customers getting the wrong label, and **30.01%** of default-customers wrongly predicted to be non-default customers.
The model's superior accuracy against non-default customers is likely due to the inherent imbalance in the dataset and suggests that a sampling technique such as ROSE or SMOTE may improve the overall performance - particularly by improving the accuracy of the model *correctly* identifying default customers.

For this model the **cp** threshold was set to 0.0010 - confirmed by the output above showing that a cp value above 0.00118 would produce 8 or fewer branches (less complexity), while a cp value less than 0.0008 will generate a tree with 12 branches (higher granularity).
Additional performance metrics produced from the *printcp* function include the cross-validation estimates of misclassification error (**xerror**) and standard errors (**xstd**) for each of the branched cp threshold options.
The stated cross-validation errors for this threshold is 0.81403 - which is the lowest of the 5 split thresholds tested.
While the standard errors is 0.011986 - which is also the lowest of the listed cp possibilities displayed.
This indicates the tree is appropriately pruned to produce a tree with minimal errors achieved through cross-validation, and sampling of the sample against the population. (*Ilola, E. 2020, October 6*)

2.2.3 Support vector classifer

(a) Use an appropriate support vector classifer to classify the credible and non-credible clients.

Justify your model choice as well as hyper-parameters which are required to be specified in R.

For this SVM as the response variable is a boolean variable recorded with 1 or 0 depending on whether the customer defaulted on a payment schedule, it is appropriate to use a methodology that accounts for this binary status.
To suitably employ this data characteristic will require the model to be provided the type of "C-classification" as an input parameter.
C-classification offers superior results for discrete values compared to alternate optional parameter inputs such as **eps-regression** and **nu-regression** which are suitable for continuous data.
Succinctly surmised as *"The C-classification and nu-classification is for binary classification usage…prediction target is a discrete variable/label."* (*c-classification SVM vs nu-classification SVM in e1071 R. 2017*) An alternative to both of these parameters is **one-classification** which is more aptly applied to models where the goal is to identify outliers. (*Difference Between the Types of SVM. 2016*)

The *kernel* parameter denotes the type of algorithm that should be used to find "the best linear separator" (*Solution, S 2019*).
Available options include:

- linear
- polynomial
- RBF (radial basis function)
- and sigmoid
  (*Team, 2021*)

To remain consistent with the boolean nature of the response variable, the chosen kernel for this model is provided as "sigmoid".
It is expected this approach will offer a comparatively-similar executed procedure as what would be accomplished by employing a *logistic regression classifier* - which is another supervised learning classification tool that will assign a predicted 1 or 0 score to observations based on the values of other variables in the input dataset.

Finally, through trial and error, the value of 0.011 was provided for the optional input parameter *gamma*.
Gamma is not required when the kernel type is 'linear', however is required when selecting any alternate kernel type and functions as "…a parameter for non linear hyperplanes. The higher the gamma value it tries to exactly fit the training data set" (*Fraj, M.B 2018*)

Code to produce this is as follows:

```
svm_model = svm(formula = Y ~ .,
                data = cc_train,
                type = 'C-classification',
                kernel = 'sigmoid',
                gamma = 0.01)
```

For completeness the overall accuracy is listed below for alternate gamma values:

- gamma = 0.007 0.7796422
- gamma = 0.008 0.7754195
- gamma = 0.009 0.7683076
- gamma = 0.01: **0.8036448**
- gamma = 0.011: 0.7796422

As seen, the highest accuracy was achieved for the value of 0.01, with **0.011** the next nearest neighbour being 2.4% less accurate overall.

(b) Display model summary and discuss the relationship between the response variable versus selected features.

```
summary(svm_model)
```

```
##
## Call:
## svm(formula = Y ~ ., data = cc_train, type = "C-classification",
##     kernel = "sigmoid", gamma = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  1
##      coef.0:  0
##
## Number of Support Vectors:  7663
##
##  ( 3837 3826 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
# show significant variables
w <- t(svm_model$coefs) %*% svm_model$SV        # weight vectors
w <- apply(w, 2, function(v){sqrt(sum(v^2))})   # weight
w <- sort(w, decreasing = T)
print(w)
```

```
##              X19              X22              X21              X18
##      202.91949782     196.69619269     185.42430130     180.20756838
##              X20   X62.Mth.Arrears              X23      X6Rev.Credit
##      148.12007462     127.31909084       92.04155411       89.41160174
##   X63.Mth.Arrears              X15              X16              X13
##       62.86285620       59.99610499       49.75046914       49.04666488
##              X14              X17     X11Rev.Credit              X12
##       40.16085407       37.03141547       32.48156173       24.52208252
##  X7No.Consumption   X82.Mth.Arrears   X92.Mth.Arrears   X73.Mth.Arrears
##       21.40302988       20.36162411       18.32846736       18.00000000
##   X64.Mth.Arrears X10No.Consumption        X7Pay.Duly         X3Unknown
##       16.00000000       15.10387477       15.01452851       14.48552254
##  X9No.Consumption        X6Pay.Duly  X8No.Consumption   X72.Mth.Arrears
##       14.28447388       14.20868119       13.94721990       13.36827414
##      X8Rev.Credit      X9Rev.Credit     X10Rev.Credit         X4Unknown
##       12.65526557       11.84619559       11.14396577       10.00000000
##        X9Pay.Duly           X3Other   X93.Mth.Arrears X11No.Consumption
##         9.33484621        9.00000000        8.39046216        8.05165309
##       X10Pay.Duly  X113.Mth.Arrears   X83.Mth.Arrears   X97.Mth.Arrears
##         7.79429973        7.39046216        7.19357067        5.74658616
##            X4Other              X2M              X2F   X74.Mth.Arrears
##         5.00000000        4.84176930        4.84176930        4.80642933
##  X107.Mth.Arrears  X117.Mth.Arrears       X11Pay.Duly          X4Single
##         4.74658616        4.74658616        4.45573111        4.36894546
##   X76.Mth.Arrears   X84.Mth.Arrears   X87.Mth.Arrears   X94.Mth.Arrears
##         3.00000000        3.00000000        3.00000000        3.00000000
##  X105.Mth.Arrears      X7Rev.Credit                X1   X65.Mth.Arrears
##         3.00000000        2.85571359        2.45161370        2.00000000
##   X67.Mth.Arrears   X86.Mth.Arrears  X104.Mth.Arrears  X114.Mth.Arrears
##         2.00000000        2.00000000        2.00000000        2.00000000
##  X116.Mth.Arrears      X3University        X8Pay.Duly   X66.Mth.Arrears
##         2.00000000        1.62621600        1.04729069        1.00000000
##   X77.Mth.Arrears   X88.Mth.Arrears   X98.Mth.Arrears  X106.Mth.Arrears
##         1.00000000        1.00000000        1.00000000        1.00000000
##  X6No.Consumption                X5  X103.Mth.Arrears     X3High.School
##         0.85585768        0.39539128        0.39046216        0.04427137
##   X68.Mth.Arrears   X75.Mth.Arrears   X78.Mth.Arrears   X85.Mth.Arrears
##         0.00000000        0.00000000        0.00000000        0.00000000
##   X95.Mth.Arrears   X96.Mth.Arrears  X108.Mth.Arrears  X115.Mth.Arrears
##         0.00000000        0.00000000        0.00000000        0.00000000
##  X118.Mth.Arrears
##         0.00000000
```

Extracting the important variables from SVM and Neural Network models is a more complex task when compared to the decidedly simple procedure used to extract signficant features for the decision tree model above.
To get around this, a somewhat brute-forced solution is used above where the model coefficients are multiplied against the weight vectors produced, then summed, and then the square root is taken. This output is then sorted and printed in descending order; listing the variables of most importance first and the least important variables last.

From this vantage point it can be seen the results of the SVM model do not have much in common with the important variables extracted from the decision tree above, with the first 3 most important variables being *X19*, *X22*, and *X21*; representing the amount of money made in repayments in the months of August, May, and June 2005 in dollars respectively. Contrasted with variable *X19* being the 13th most significant variable in the decision tree, with the other two being even less important.

(c) Evaluate the performance of the algorithm on the training data and comment on the results.

```
#============ predict with training data
cc_train2 <- cc_train

# now make predictions using TRAIN dataset
svm_pred = predict(svm_model, newdata = cc_train2)

# confusion matrix of above prediction against response variable (col 24)
cm = table(cc_train2[,24], svm_pred)  # confusion matrix
#svm.imp <- Importance(svm_pred, data=cc_train)

confMat <- table(cc_train2$Y,svm_pred)
svm_accuracy <- sum(diag(cm))/sum(cm)
confMat
```

```
##    svm_pred
##         0     1
##   0 15410   945
##   1  3079  1567
```

```
svm_accuracy
```

```
## [1] 0.8083901
```

```
#====================================
```

Results of the SVM model using the predict() function against the training data results in a positive 80.83% accuracy.
This is comprised of 83.3% accuracy at correctly predict non-default customers, and 62.3% accuracy at predicting default customers.
The separation between the two response variable classes indicates that a sampling technique to either reduce the non-default customers, or increase the frequency of default customers in the dataset would improve the overall accuracy - especially for default customers.
Two popular sampling techniques to reduce the majority or increase the minority of observations are ROSE and SMOTE respectively.

2.2.4 Prediction

Apply your fitted models in 2.2.2 and 2.2.3 to make prediction on the test data.

Evaluate the performance of the algorithms on test data.

Which models do you prefer?

Are there any suggestions to further improve the performance of the algorithms?

Justify your answers.

After comparing the outputs of the SVM and decision tree results against themselves through use of the training datasets, the predict() function is employed to predict the response variable Y - using both distinct models, against the **test** dataframe.

```
############################
#####       SVM       #####
############################
# now make predictions using TEST dataset
svm_pred = predict(svm_model, newdata = cc_test)

# confusion matrix of above prediction against response variable (col 24)
cm = table(cc_test[,24], svm_pred)  # confusion matrix
#svm.imp <- Importance(svm_pred, data=cc_train)

svm_confMat <- table(cc_test$Y,svm_pred)
svm_accuracy <- sum(diag(svm_confMat))/sum(svm_confMat)
svm_confMat
```

```
##    svm_pred
##        0    1
##   0 6577  432
##   1 1335  655
```

```
svm_accuracy
```

```
## [1] 0.8036448
```

```
############################

############################
#####  DECISION TREE  #####
############################
# NOW USE RESULTS TO PREDICT - USING THE TEST DATASET
tree_pred = predict(dec_tree,cc_test,type="class")

# CREATE CONFUSION MATRIX, EXTRACT ACCURACY
dec_tree_confMat <- table(cc_test$Y,tree_pred)
dec_accuracy <- sum(diag(dec_tree_confMat))/sum(dec_tree_confMat)
dec_tree_confMat
```

```
##    tree_pred
##        0    1
##   0 6680  329
##   1 1282  708
```

```
dec_accuracy
```

```
## [1] 0.8209801
```

```
############################
```

Results of the SVM model against the test dataset indicate an overall accuracy of **80.36%** - with **60.2%** accuracy correctly predicting customers that will default, and **83.1%** accuracy correctly predicting non-default customers.

Results of the decision tree model against the test dataset indicates an overall accuracy of **82.09%** - with **68.2%** accuracy

correctly predicting customers that will default, and **83.8%** accuracy correctly predicting non-default customers.

From a purely objective, numbers-based view it would appear from this comparison that use of the Decision Tree would be best applied in a business setting for this exercise.
This is particularly prevalent when accounting for the transparency and high interpretability of the results produced by the decision tree, which explicitly explain how each variable is used to compute the classification of customers into which segments.
These results can be explained clearly and simply without the need for complex jargon or code routines that SVM generally requires to get actionable directions in a business context.

For future iterations of this comparison it would be worthwhile to compare alternative kernels and type parameters provided to the SVM model, as it is likely better performance could be achieved by tweaking the parameters and changing the kernel algorithm (e.g. from sigmoid to linear may produce comparative results). Additional time taken to over or under sample observations in the base dataframe would likely serve to improve the accuracy for these two models, as both suffered from poor accuracy at predicting default customers compared to none-default customers.
Finally, to improve accuracy at the cost of interpretability, it would be a worthwhile endeavour to replace the decision tree model selected in this instance and replace with a random forest approach instead, as it is likely this more resource-intensive procedure will procure superior sensitivity and specificity.

==========

Citations

Team, D. (2021, August 25). e1071 Package – Perfect Guide on SVM Training & Testing Models in R. DataFlair. https://data-flair.training/blogs/e1071-in-r/ (https://data-flair.training/blogs/e1071-in-r/)

Solution, S. (2019, April 9). 204.6.8 SVM : Advantages Disadvantages and Applications | Statinfer. Statinfer | Data Science Starts Here. https://statinfer.com/204-6-8-svm-advantages-disadvantages-applications/ (https://statinfer.com/204-6-8-svm-advantages-disadvantages-applications/)

rpart.control function - RDocumentation. (n.d.). R Documentation. Retrieved September 26, 2021, from https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart.control (https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart.control)

Optimizing a Support Vector Machine with Quadratic Programming. (2015, November 3). Cross Validated. https://stats.stackexchange.com/questions/179900/optimizing-a-support-vector-machine-with-quadratic-programming (https://stats.stackexchange.com/questions/179900/optimizing-a-support-vector-machine-with-quadratic-programming)

Fraj, M. B. (2018, August 14). In Depth: Parameter tuning for SVC - All things AI. Medium. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769 (https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769)

Difference between the types of SVM. (2016, September 28). Cross Validated. https://stats.stackexchange.com/questions/237382/difference-between-the-types-of-svm (https://stats.stackexchange.com/questions/237382/difference-between-the-types-of-svm)

c-classification SVM vs nu-classification SVM in e1071 R. (2017, November 9). Cross Validated. https://stats.stackexchange.com/questions/312897/c-classification-svm-vs-nu-classification-svm-in-e1071-r (https://stats.stackexchange.com/questions/312897/c-classification-svm-vs-nu-classification-svm-in-e1071-r)

Ilola, E. (2020, October 6). A beginner’s guide to standard deviation and standard error. Students 4 Best Evidence. https://s4be.cochrane.org/blog/2018/09/26/a-beginners-guide-to-standard-deviation-and-standard-error/ (https://s4be.cochrane.org/blog/2018/09/26/a-beginners-guide-to-standard-deviation-and-standard-error/)

W. (2016, February 27). Decision Tree Flavors: Gini Index and Information Gain – Learn by Marketing. Learn by Marketing. https://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/ (https://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/)

Revert, F. (2019, February 7). Interpreting Random Forest and other black box models like XGBoost. Medium. https://towardsdatascience.com/interpreting-random-forest-and-other-black-box-models-like-xgboost-80f9cc4a3c38 (https://towardsdatascience.com/interpreting-random-forest-and-other-black-box-models-like-xgboost-80f9cc4a3c38)

Brownlee, J. (2021, January 5). Bagging and Random Forest for Imbalanced Classification. Machine Learning Mastery. https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/ (https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/)

Witten, Pal, I. (2017). Maximum Margin Hyperplane - an overview | ScienceDirect Topics. Science Direct. https://www.sciencedirect.com/topics/computer-science/maximum-margin-hyperplane (https://www.sciencedirect.com/topics/computer-science/maximum-margin-hyperplane)

Brownlee, J. (2020, August 21). Cost-Sensitive Decision Trees for Imbalanced Classification. Machine Learning Mastery. https://machinelearningmastery.com/cost-sensitive-decision-trees-for-imbalanced-classification/ (https://machinelearningmastery.com/cost-sensitive-decision-trees-for-imbalanced-classification/)

Packages + software used:

```
### Citations

#RStudio.Version()
citation("rpart")
```

```
##
## To cite package 'rpart' in publications use:
##
##   Terry Therneau and Beth Atkinson (2019). rpart: Recursive
##   Partitioning and Regression Trees. R package version 4.1-15.
##   https://CRAN.R-project.org/package=rpart
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {rpart: Recursive Partitioning and Regression Trees},
##     author = {Terry Therneau and Beth Atkinson},
##     year = {2019},
##     note = {R package version 4.1-15},
##     url = {https://CRAN.R-project.org/package=rpart},
##   }
```

```
citation("rpart.plot")
```

```
##
## To cite package 'rpart.plot' in publications use:
##
##   Stephen Milborrow (2021). rpart.plot: Plot 'rpart' Models: An
##   Enhanced Version of 'plot.rpart'. R package version 3.1.0.
##   https://CRAN.R-project.org/package=rpart.plot
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'},
##     author = {Stephen Milborrow},
##     year = {2021},
##     note = {R package version 3.1.0},
##     url = {https://CRAN.R-project.org/package=rpart.plot},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
citation("e1071")
```

```
##
## To cite package 'e1071' in publications use:
##
##   David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and
##   Friedrich Leisch (2021). e1071: Misc Functions of the Department of
##   Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R
##   package version 1.7-7. https://CRAN.R-project.org/package=e1071
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {e1071: Misc Functions of the Department of Statistics, Probability
## Theory Group (Formerly: E1071), TU Wien},
##     author = {David Meyer and Evgenia Dimitriadou and Kurt Hornik and Andreas Weingessel an
d Friedrich Leisch},
##     year = {2021},
##     note = {R package version 1.7-7},
##     url = {https://CRAN.R-project.org/package=e1071},
##   }
```

```
citation("data.table")
```

```
##
## To cite package 'data.table' in publications use:
##
##   Matt Dowle and Arun Srinivasan (2021). data.table: Extension of
##   `data.frame`. R package version 1.14.0.
##   https://CRAN.R-project.org/package=data.table
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {data.table: Extension of `data.frame`},
##     author = {Matt Dowle and Arun Srinivasan},
##     year = {2021},
##     note = {R package version 1.14.0},
##     url = {https://CRAN.R-project.org/package=data.table},
##   }
```

```
citation("rminer")
```

```
##
## To cite package 'rminer' in publications use:
##
##   Paulo Cortez (2020). rminer: Data Mining Classification and
##   Regression Methods. R package version 1.4.6.
##   https://CRAN.R-project.org/package=rminer
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {rminer: Data Mining Classification and Regression Methods},
##     author = {Paulo Cortez},
##     year = {2020},
##     note = {R package version 1.4.6},
##     url = {https://CRAN.R-project.org/package=rminer},
##   }
```

```
citation("rattle")
```

```
##
## Please cite the 'rattle' package in publications using:
##
##   Williams, G. J. (2011), Data Mining with Rattle and R: The Art of
##   Excavating Data for Knowledge Discovery, Use R!, Springer.
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     title = {Data Mining with {Rattle} and {R}: The art of excavating data for knowledge di
scovery},
##     author = {Graham J. Williams},
##     publisher = {Springer},
##     series = {Use R!},
##     year = {2011},
##     url = {http://www.amazon.com/gp/product/1441998896/ref=as_li_qf_sp_asin_tl?ie=UTF8&tag=
togaware-20&linkCode=as2&camp=217145&creative=399373&creativeASIN=1441998896},
##   }
```

```
citation("RColorBrewer")
```

```
##
## To cite package 'RColorBrewer' in publications use:
##
##   Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package
##   version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {RColorBrewer: ColorBrewer Palettes},
##     author = {Erich Neuwirth},
##     year = {2014},
##     note = {R package version 1.1-2},
##     url = {https://CRAN.R-project.org/package=RColorBrewer},
##   }
```

```
citation("quadprog")
```

```
##
## To cite package 'quadprog' in publications use:
##
##   S original by Berwin A. Turlach R port by Andreas Weingessel
##   <Andreas.Weingessel@ci.tuwien.ac.at> Fortran contributions from Cleve
##   Moler dpodi/LINPACK) (2019). quadprog: Functions to Solve Quadratic
##   Programming Problems. R package version 1.5-8.
##   https://CRAN.R-project.org/package=quadprog
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {quadprog: Functions to Solve Quadratic Programming Problems},
##     author = {S original by Berwin A. Turlach R port by Andreas Weingessel <Andreas.Weinges
sel@ci.tuwien.ac.at> Fortran contributions from Cleve Moler dpodi/LINPACK)},
##     year = {2019},
##     note = {R package version 1.5-8},
##     url = {https://CRAN.R-project.org/package=quadprog},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

APPENDIX

R Code

```
set.seed(5555)
gen_loc = "D:/Uni/MA5832_AdvMachineLearning/Assignment 2/"
usRp <-"https://cran.csiro.au/"

if(!require(ggplot2))       install.packages("ggplot2"      ,repos = usRp)
if(!require(caret))         install.packages("caret"        ,repos = usRp)
if(!require(dplyr))         install.packages("dplyr"        ,repos = usRp)
if(!require(MASS ))         install.packages("MASS"         ,repos = usRp)
if(!require(sqldf))         install.packages("sqldf"        ,repos = usRp)
if(!require(GGally))        install.packages("GGally"       ,repos = usRp)
if(!require(qqplotr))       install.packages("qqplotr"      ,repos = usRp)
if(!require(corrplot))      install.packages("corrplot"     ,repos = usRp)
if(!require(randomForest))  install.packages("randomForest" ,repos = usRp)
if(!require(rpart))         install.packages("rpart"        ,repos = usRp)
if(!require(rpart.plot))    install.packages("rpart.plot"   ,repos = usRp)
if(!require(e1071))         install.packages("e1071"        ,repos = usRp)
if(!require(rminer))        install.packages("rminer"       ,repos = usRp)
if(!require(rattle))        install.packages("rattle"       ,repos = usRp)
if(!require(RColorBrewer))  install.packages("RColorBrewer" ,repos = usRp)
if(!require(quadprog))      install.packages("quadprog"     ,repos = usRp)
if(!require(data.table))    install.packages("data.table"   ,repos = usRp)


library(caret)
library(dplyr)
library(MASS)
library(sqldf)
library(GGally)
library(qqplotr)
library(corrplot)
library(randomForest)
library(rpart)
library(rpart.plot)
library(e1071)
library(data.table)
library(rminer)
library(rattle)
library(RColorBrewer)
library(quadprog)
# Create dataframe with values
p1_df <- data.frame(X1=c(3,4,3.5,5,4,6,2,-1,3,3,-2,-1),
                    X2=c(2,0,-1,1,-3,-2,5,7,6.5,7,7,10),
                     Y=c(-1,-1,-1,-1,-1,-1,1,1,1,1,1,1))
p1_df

#scatterplot 1 = red / -1 = blue
ggplot(p1_df, aes(x=X1, y=X2, color = as.factor(Y))) +
  geom_point() +
  scale_color_manual(values = c("blue","red")) +
  xlab("X1") +
  ylab("X2") +
  ggtitle("Scatterplot of Y points")

# Create matrix
p1_matrix       <- matrix(rep(0, 3*3), nrow=3, ncol=3)
diag(p1_matrix) <- 1
```

```
p1_matrix[nrow(p1_matrix), ncol(p1_matrix)] <- 1e-8   # to suppress R complaining about "matri
x is not positive definite."

# Create vector
p1_vector <- c(0,0,0)

# constraint matrix
p1_alt_matrix <- as.matrix(p1_df[, c("X1", "X2")])
p1_alt_matrix <- cbind(p1_alt_matrix, b=rep(-1, 12))
p1_alt_matrix <- p1_alt_matrix * p1_df$Y

# now constraint vector
p1_alt_vect <- rep(1, 12)

# invoke QP solver:
p1_qp <- solve.QP(p1_matrix, p1_vector, t(p1_alt_matrix), bvec=p1_alt_vect)
p1_qp$solution


w <- p1_qp$solution[1:2]
b <- p1_qp$solution[3]

p1_func_plotMargin <- function(w = 1*c(-1, 1), b = 1){
                      x1 = seq(-20, 20, by = .01)
                      x2 = (-w[1]*x1 + b)/w[2]
                      l1 = (-w[1]*x1 + b + 1)/w[2]
                      l2 = (-w[1]*x1 + b - 1)/w[2]
  dt <- data.table(X1=x1, X2=x2, L1=l1, L2=l2)
  ggplot(dt)+
    geom_line(aes(x=X1, y=X2))+
    geom_line(aes(x=X1, y=L1), color="orange")+
    geom_line(aes(x=X1, y=L2), color="purple" )
}

p1_func_plotMargin(w=w, b=b)+
  geom_point(data=p1_df, aes(x=X1, y=X2, color=as.factor(Y)))+
  scale_color_manual(values = c("blue","red"))+
  xlim(-2.25, 6)+
  ylim(-3, 10)+
  xlab("X1") +
  ylab("X2") +
  ggtitle("Optimal separating hyperplane") +
  theme_bw()



################################################
###  PART II:  AN APPLICATION - CREDIT CARD DATA
################################################

# NOTE: for this exercise there were some issues encountered when importing the provided credi
t card dataset
# to get around this the file was saved as a CSV and then imported - this CSV has been uploade
d as part of this submission

ccard_pre <- paste0(gen_loc,"CreditCard_Data.csv")
```

```
ccard <- read.table(ccard_pre, header=TRUE , sep = ",", dec = ".")


# first row is additional line of headers, dropping this row
# drop first row as it's redundant headers + drop first column as it's an unneeded primary key
ccard = ccard[-1,-1]


sum(is.na(ccard))       # Checking for NA values
colSums(is.na(ccard))   # Checking for NA values for any column of dataset
# no missing values for any row/column

str(ccard)
ccard$X2 <- ifelse(ccard$X2 == "1","M","F") # convert to gender

# cleaning education level
ccard$X3 <-with(ccard,  ifelse(X3 %in% c("0","5","6") ,"Unknown",
                        ifelse(X3 == "1"             ,"Grad School",
                        ifelse(X3 == "2"             ,"University",
                        ifelse(X3 == "3"             ,"High School",
                        ifelse(X3 == "4"             ,"Other","ERROR"))))))

# cleaning marital status
ccard$X4 <- with(ccard,  ifelse(X4 == "0" ,"Unknown",
                        ifelse(X4 == "1" ,"Married",
                        ifelse(X4 == "2" ,"Single",
                        ifelse(X4 == "3" ,"Other","ERROR")))))


paymentHistFunc <- function(x) {
      x <- with(ccard,  ifelse(x == "-2","No Consumption",
                        ifelse(x == "-1","Pay Duly",
                        ifelse(x == "0" ,"Rev Credit",
                        ifelse(x == "1" ,"1 Mth Arrears",
                        ifelse(x == "2" ,"2 Mth Arrears",
                        ifelse(x == "3" ,"3 Mth Arrears",
                        ifelse(x == "4" ,"4 Mth Arrears",
                        ifelse(x == "5" ,"5 Mth Arrears",
                        ifelse(x == "6" ,"6 Mth Arrears",
                        ifelse(x == "7" ,"7 Mth Arrears",
                        ifelse(x == "8" ,"8 Mth Arrears",
                        ifelse(x == "9" ,"9 Mth Arrears+","ERROR")))))))))))))
                        }

# cleaning payment history
ccard$X6 <- paymentHistFunc(ccard$X6)
ccard$X7 <- paymentHistFunc(ccard$X7)
ccard$X8 <- paymentHistFunc(ccard$X8)
ccard$X9 <- paymentHistFunc(ccard$X9)
ccard$X10 <- paymentHistFunc(ccard$X10)
ccard$X11 <- paymentHistFunc(ccard$X11)


# convert appropriate variables to factor
varsToFactor <- c("X2","X3","X4","X6","X7","X8","X9","X10","X11","Y")
ccard[,varsToFactor] <- lapply(ccard[,varsToFactor],factor)
```

```
# convert remaining variables to int
varsToINT <- c("X1","X12","X13","X14","X15","X16","X17","X18","X19","X20","X21","X22","X23")
ccard[,varsToINT] <- lapply(ccard[,varsToINT],as.numeric)

ccard$X5 <- as.integer(ccard$X5)

############## DATA IS NOW CLEAN


#split dataset
# 2.2.1 Data
rows <- sample(nrow(ccard)) #randomly shuffle rows around
ccard <- ccard[rows, ]

cc_split <- createDataPartition(ccard$Y, p = 0.7, list = FALSE)

cc_train <- ccard[cc_split, ]
cc_test <- ccard[-cc_split, ]
dim(cc_train)   # print dimensions of training data
#table(cc_train$Y) %>% prop.table() # Proportions of response var

# create decision tree
dec_tree <- rpart(Y ~ ., data=cc_train, control=rpart.control(minsplit=250, minbucket=65, cp=
0.0008))
# PLOT DECISION TREE
#summary(dec_tree)
fancyRpartPlot(dec_tree,main="Decision Tree Predicting Default Customers",sub="",palettes=c("P
uBuGn", "Oranges"))
barplot(dec_tree$variable.importance,main = "Decision Tree: Comparison of Variable Importanc
e")
cc_train2 <- cc_train
# NOW USE RESULTS TO PREDICT - USING THE TRAIN DATASET
tree_pred = predict(dec_tree,cc_train2,type="class")
confMat <- table(cc_train2$Y,tree_pred)
accuracy <- sum(diag(confMat))/sum(confMat)
accuracy

printcp(dec_tree) #cp table showing the improvement in cost complexity at each node
svm_model = svm(formula = Y ~ .,
                data = cc_train,
                type = 'C-classification',
                kernel = 'sigmoid',
                gamma = 0.01)

summary(svm_model)

# show significant variables
w <- t(svm_model$coefs) %*% svm_model$SV        # weight vectors
w <- apply(w, 2, function(v){sqrt(sum(v^2))})   # weight
w <- sort(w, decreasing = T)
print(w)

#============ predict with training data
cc_train2 <- cc_train
```

```
# now make predictions using TRAIN dataset
svm_pred = predict(svm_model, newdata = cc_train2)

# confusion matrix of above prediction against response variable (col 24)
cm = table(cc_train2[,24], svm_pred)  # confusion matrix
#svm.imp <- Importance(svm_pred, data=cc_train)

confMat <- table(cc_train2$Y,svm_pred)
svm_accuracy <- sum(diag(cm))/sum(cm)
confMat
svm_accuracy
#====================================

############################
#####       SVM       #####
############################
# now make predictions using TEST dataset
svm_pred = predict(svm_model, newdata = cc_test)

# confusion matrix of above prediction against response variable (col 24)
cm = table(cc_test[,24], svm_pred)  # confusion matrix
#svm.imp <- Importance(svm_pred, data=cc_train)

svm_confMat <- table(cc_test$Y,svm_pred)
svm_accuracy <- sum(diag(svm_confMat))/sum(svm_confMat)
svm_confMat
svm_accuracy

############################

############################
#####  DECISION TREE  #####
############################
# NOW USE RESULTS TO PREDICT - USING THE TEST DATASET
tree_pred = predict(dec_tree,cc_test,type="class")

# CREATE CONFUSION MATRIX, EXTRACT ACCURACY
dec_tree_confMat <- table(cc_test$Y,tree_pred)
dec_accuracy <- sum(diag(dec_tree_confMat))/sum(dec_tree_confMat)
dec_tree_confMat
dec_accuracy
############################

### Citations

#RStudio.Version()
citation("rpart")
citation("rpart.plot")
citation("e1071")
citation("data.table")
citation("rminer")
citation("rattle")
citation("RColorBrewer")
citation("quadprog")
```