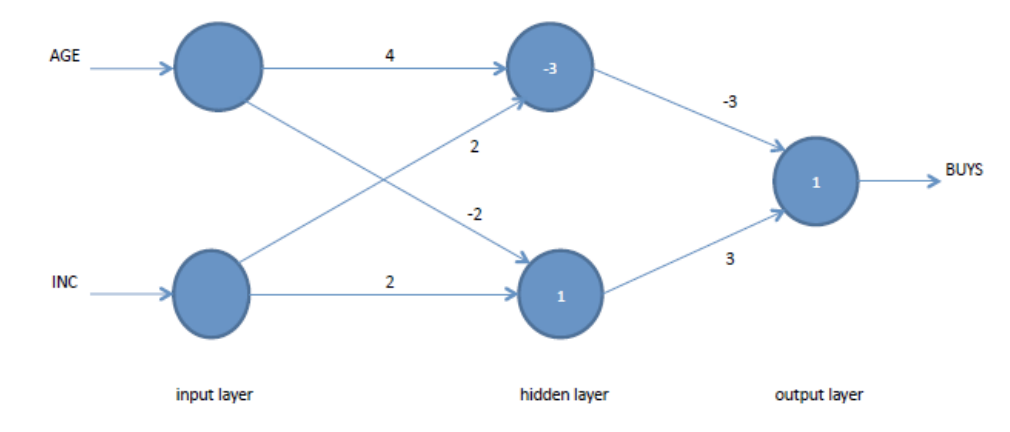
Homework 4

**Rohan GUPTA  
PAVAN PALVE  
MOHAK BEHAL**

2018

1. Consider the problem of predicting whether a customer responds favorably to a coupon offering $1 off the purchase price of Billy Beans. Customers can be of three types: “No” (N) customers, who will not buy Billy Beans at all, “Maybe” (M) customers, who will buy Billy Beans only with the coupon, and “Yes” (Y) customers who will buy Billy Beans whether or not a coupon is offered but will use a coupon if given one. We will try to predict the type of customer just from knowledge of AGE and INCOME The continuous inputs AGE and INC have been normalized to the range 0-1 by Min-max normalization. The range of **AGE** is **20-80**, and the range of **income** is **10K-110K**. The output BUYS is put in the range 0-1, where 0 means NO, 0.5 means MAYBE, and 1 means YES. Operationally, any **output** from **0−0.2499 is NO**, **0.25−0.7499 is MAYBE**, and **0.75−1 is YES**.



**Given:**

1. For input y, output is a sigmoid function
2. Un-normalized inputs: Income = 50K; Age =70

**To Find:**

1. Normalized Inputs
2. Output of Neural Network

**Solution:**

1. Normalized Inputs; where inputs

Age Range

Income

Normalizing Age (Input = 70) = (70-20) / (80 -20)

**Age’ = 0.833**

Normalizing Income (Input = 50) = (50 -10)/ (110-10)

**Income’ = 0.40**

1. Inputs

Income = 0.40; Age = 0.833

Calculating output from the hidden layer y1

= 1.132

Applying transfer function to the input;

= = 0.756

**y1 = 0.756**

Calculating output from hidden layer y2

= 0.134

Applying transfer function to the input;

= = 0.533

**y2 = 0.533**

y1 and y2 will become the inputs to the output of neural network

= 0.332

Applying transfer function to the input;

= = 0.583

**y= 0.583**

The final output of the neural network lies within the range of **0.25−0.7499** which gives the representation of class **Maybe.**

*“We can conclude that the person with age = 70 and income = 50K will buy Billy Beans only with a coupon”*

1. Assume x1, x2 and x3 are three Boolean input variables (i.e. x1, x2, x3 ∈ {0, 1}). For each of the following functions construct a neural network (draw the network) that produce the associated out-puts.

a. (x1 AND x2) AND x3

b. (x1 AND x2) OR x3

c. (x1 OR x2) AND (x1 OR x3)

You need to determine the number of **units**, **weights** and **activation functions** to construct a neural network. I do not expect you to optimally compute the weights. Use any weights that you think it will give you correct results.

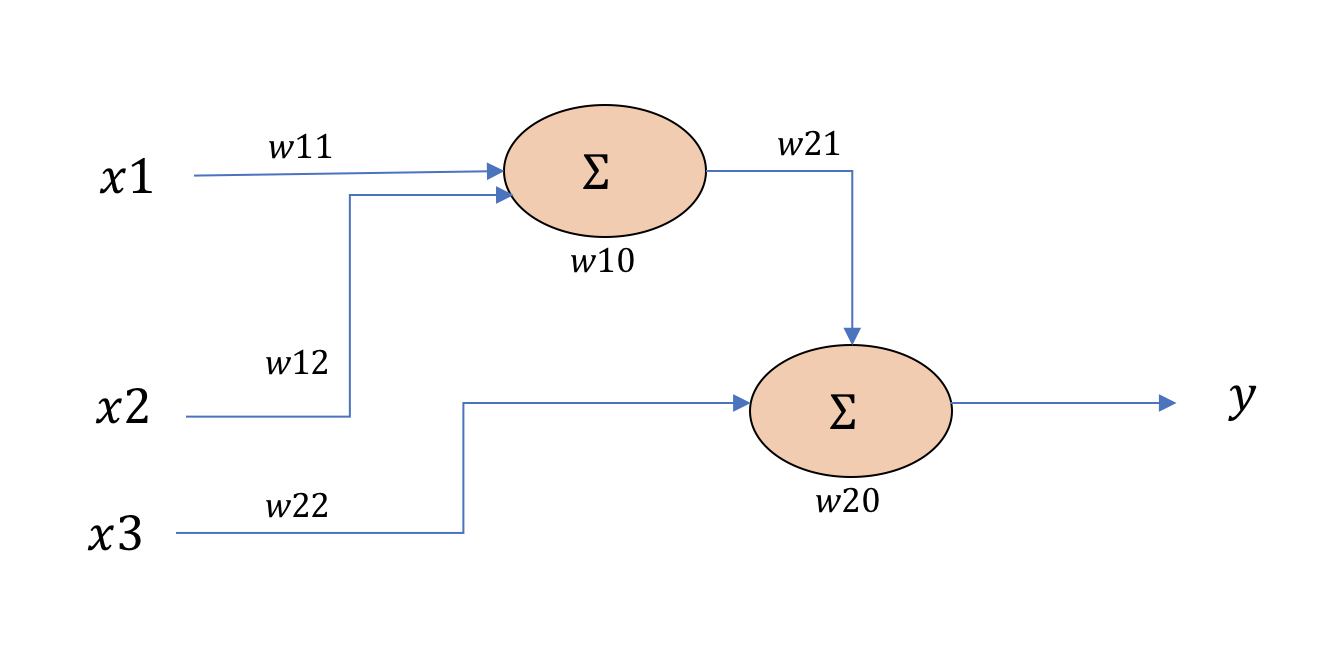
**Inputs**

|  |  |  |
| --- | --- | --- |
| X1 | X2 | X3 |
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

1. (x1 AND x2) AND x3

**Solution:**

Neural Network



Activation Function- **Step Function**

Output

**w10** = -0.8; **w11** = 0.5; **w12** = 0.5

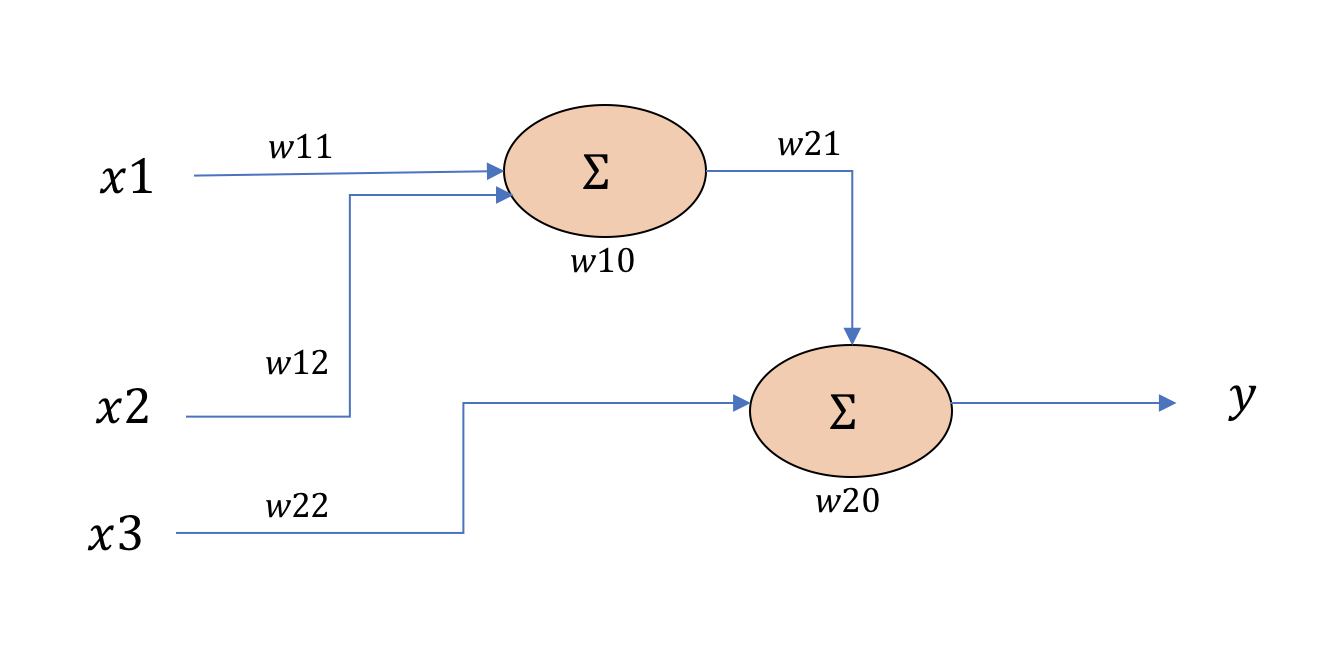
**w20** = -0.8; **w21** = 0.5; **w22** = 0.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (x1 AND x2) | (x1 AND x2) AND x3 | y’ | Step Function  f(Y= y’) | y | Step Function  f(Y= y) |
| 0 | 0 | -0.8 | 0 | -0.8 | 0 |
| 0 | 0 | -0.8 | 0 | -0.3 | 0 |
| 0 | 0 | -0.3 | 0 | -0.8 | 0 |
| 0 | 0 | -0.3 | 0 | -0.3 | 0 |
| 0 | 0 | -0.3 | 0 | -0.8 | 0 |
| 0 | 0 | -0.3 | 0 | -0.3 | 0 |
| 1 | 0 | 0.2 | 1 | -0.3 | 0 |
| 1 | 1 | 0.2 | 1 | 0.2 | 1 |

1. (x1 AND x2) OR x3

**Solution:**

Neural Network



Activation Function- **Step Function**

Output

**w10** = -0.8; **w11** = 0.5; **w12** = 0.5

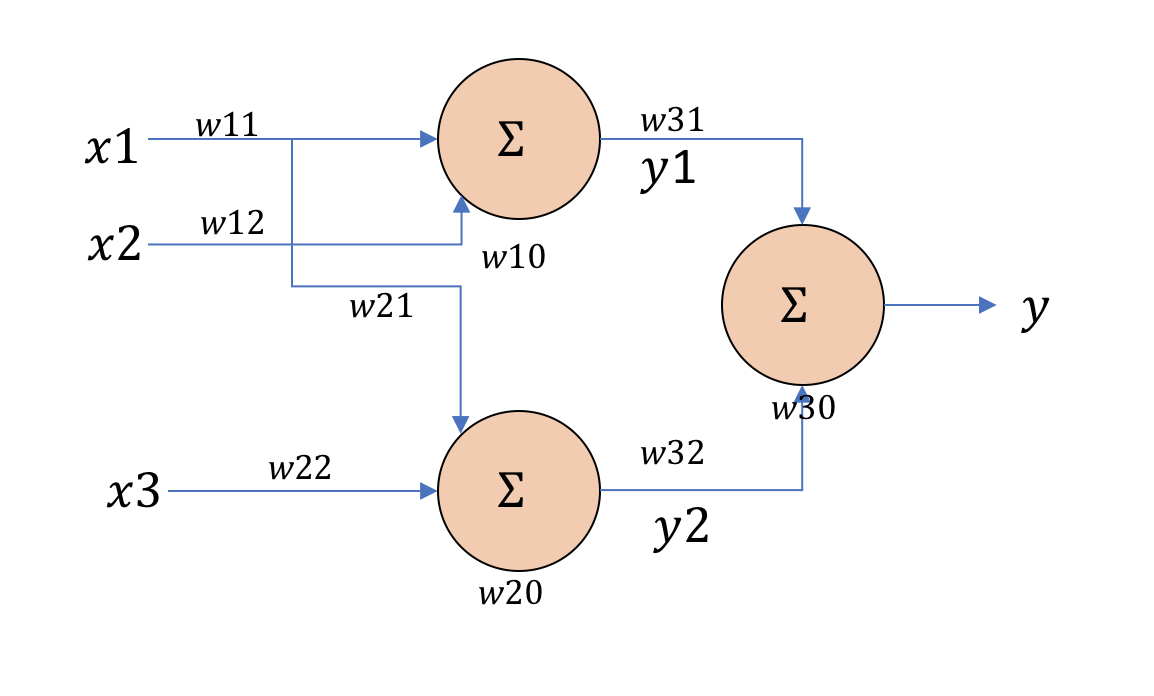
**w20** = -0.4; **w21** = 0.5; **w22** = 0.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (x1 AND x2) | (x1 AND x2) OR x3 | y’ | Step Function  f(Y= y’) | y | Step Function  f(Y= y) |
| 0 | 0 | -0.8 | 0 | -0.4 | 0 |
| 0 | 1 | -0.8 | 0 | 0.1 | 1 |
| 0 | 0 | -0.3 | 0 | -0.4 | 0 |
| 0 | 1 | -0.3 | 0 | 0.1 | 1 |
| 0 | 0 | -0.3 | 0 | -0.4 | 0 |
| 0 | 1 | -0.3 | 0 | 0.1 | 1 |
| 1 | 1 | 0.2 | 1 | 0.1 | 1 |
| 1 | 1 | 0.2 | 1 | 0.6 | 1 |

1. (x1 OR x2) AND (x1 OR x3)

**Solution:**

Neural Network



Activation Function- **Step Function**

Output

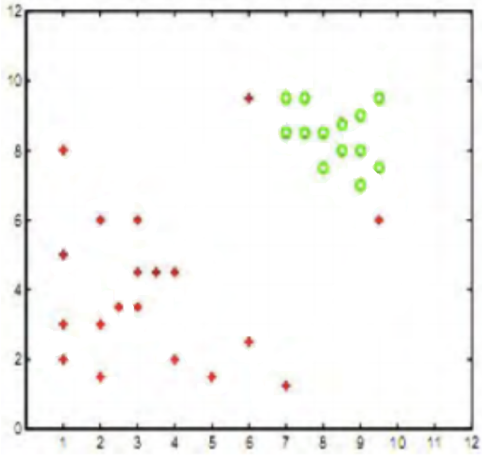
**w10** = -0.3; **w11** = 0.5; **w12** = 0.5

**w20** = -0.3; **w21** = 0.5; **w22** = 0.5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (x1 OR x2) | (x1 OR  x3) | (x1 OR x2) AND (x1 OR x3) | y1 | Step Function  f(Y= y1) | y2 | Step Function  f(Y= y2) | y | Step Function  f(Y= y) |
| 0 | 0 | 0 | -0.3 | 0 | -0.3 | 0 | -0.8 | 0 |
| 0 | 1 | 0 | -0.3 | 0 | 0.2 | 1 | -0.3 | 0 |
| 1 | 0 | 0 | 0.2 | 1 | -0.3 | 0 | -0.3 | 0 |
| 1 | 1 | 1 | 0.2 | 1 | 0.2 | 1 | 0.2 | 1 |
| 1 | 1 | 1 | 0.2 | 1 | 0.2 | 1 | 0.2 | 1 |
| 1 | 1 | 1 | 0.2 | 1 | 0.7 | 1 | 0.2 | 1 |
| 1 | 1 | 1 | 0.7 | 1 | 0.2 | 1 | 0.2 | 1 |
| 1 | 1 | 1 | 0.7 | 1 | 0.7 | 1 | 0.2 | 1 |

**w30** = -0.8; **w31** = 0.5; **w32** = 0.5

1. The original SVM proposed was a linear classier. As discussed in class, in order to make SVM non-linear we map the training data on to a higher dimensional feature space and then use a linear classier in the that space. This mapping can be done with the help of kernel functions. For this question assume that we are training an SVM with a quadratic kernel - i.e. our kernel function is a polynomial kernel of degree 2. This means the resulting decision boundary in the original feature space may be parabolic in nature. The dataset on which we are training is given below:



The slack penalty C will determine the location of the separating parabola. Please answer the following questions qualitatively.

For a non-linear SVM we map the classifier to higher dimension: -

= 0; i is correctly classified beyond margin

> 0; i is correctly classified. However, distance is less than 1, so we are allowing it to be inside the margin.

< 0; i is misclassified.

**- > Should be as small as possible for non-linear SVM**

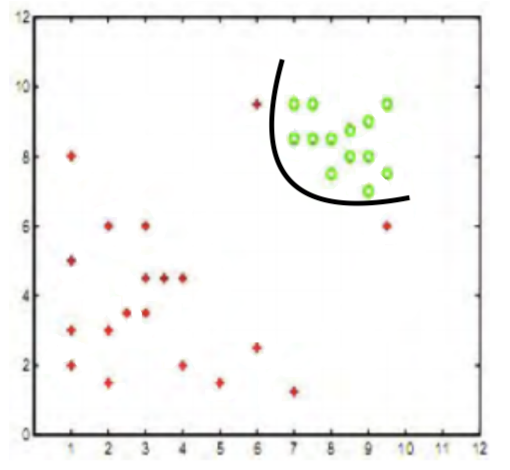
**Large C means less error**

**Small C means more error**

1. Where would the decision boundary be for very large values of C? (Remember that we are using a quadratic kernel). Justify your answer in one sentence and then draw the decision boundary in the figure below.

**Solution: a**

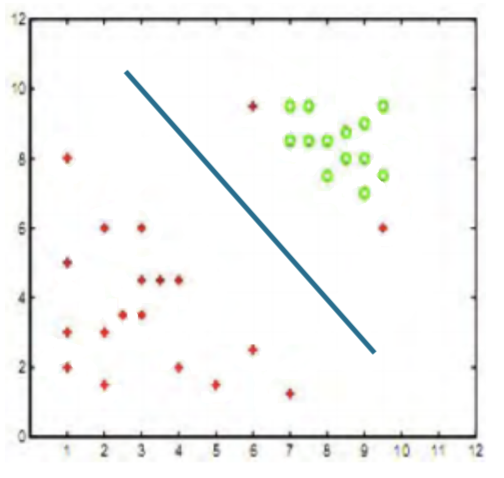
As seen above; for a large value of C, the possibility of misclassification is minimal which gives a better separation between the data points.



1. Where would the decision boundary be for C nearly equal to 0? Justify your answer in one sentence and then draw the decision boundary in the figure below.

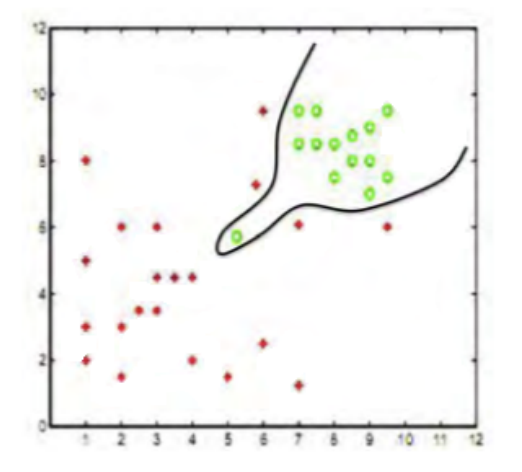
**Solution: b**

If C is equal to zero, the minimization equation will be as same as the equation of linearly separable support vector machine. The cost of misclassification is way too small and hence the decision boundary will be linear.



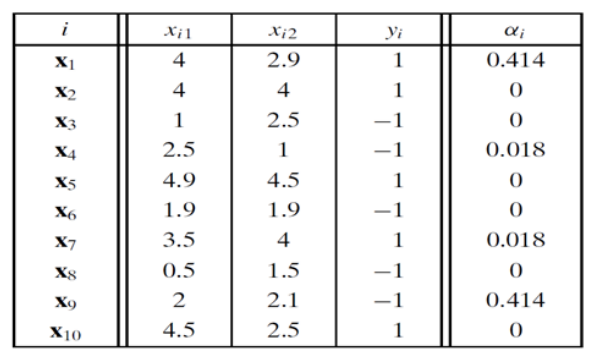
1. Now suppose we add three more data points as shown in figure below. Now the data are not quadratic ally separable, therefore we decide to use a degree-5 kernel and find the following decision boundary. Most probably, our SVM suffers from a phenomenon which will cause wrong classification of new data points. Name that phenomenon, and in one sentence, explain what it is.

**Solution: c**



This kind of classification makes the model more complex. Model fits more training data of polynomial degree kernel. This leads to overfitting in test data.

1. In the table below, 10 points with their corresponding labels and Lagrangian multipliers (αi) are provided.



1. What is the equation of the SVM hyper-plane?

**Solution:**

Equation of SVM Hyperplane is denoted by:

Or,

To calculate **w**,

wi = 0.414\*1\* [] - 0.018\*1\* [] + 0.018\*1\* [] - 0.414\*1\* []

wi = []

To calculate **b**,

Substituting back w and b in equation of SVM hyperplane, we get,

1. What is the distance of x6 from the hyper-plane? Is it within the margin of the classifier?

**Solution:**

Distance formulae,

D = | | = = 1.237

Distance x6 from hyper-plane is 1.237 and it does not lie within the margin of the classifier (2.152 / 2 = 1.07) as its distance is greater than distance of the margin.

1. Classify the point (3, 3) using the SVM model from above.

**Solution:**

D = | | = = 0.2061

The distance of the point is 0.2061. The point (3,3) lies within the margin and is towards the positive class, therefore, the point will be classified as positive.

1. (Support vector machines)
2. Support vector machines maximize the width of the margin which separates the examples of 2 classes. What advantage does a classifier with a wide margin have over a classifier that has a much smaller margin?

**Solution: a**

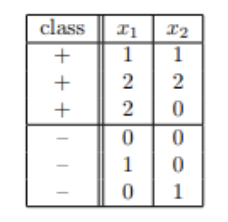
Effect of margin has an inverse relation with noise. Large margin has least effect of noise in data. In case of unseen data and higher margin, there are higher chances that the point will be classified correctly as compared to smaller margin.

1. Non-linear support vector machine which use kernels which map a dataset into a higher dimensional space are quite popular. What advantages you see in using non-linear support vector machine over linear support vector machines?

**Solution: b**

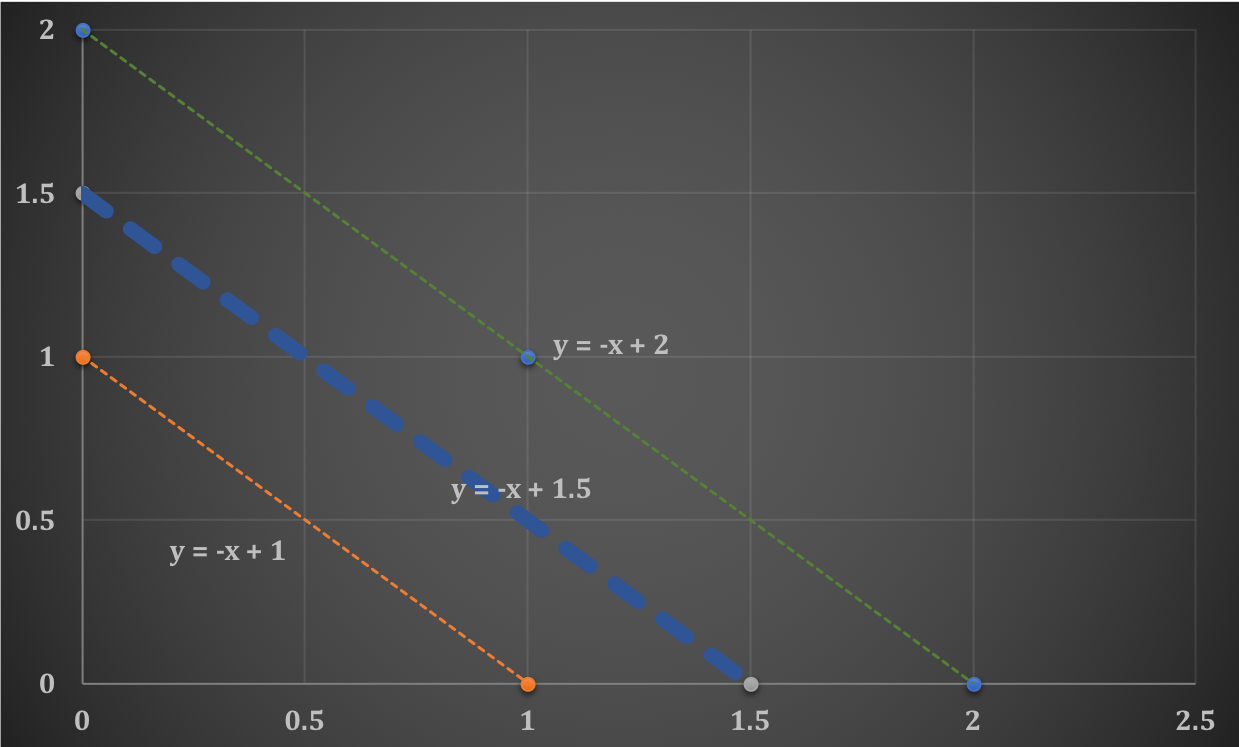
There is a high probability to find a hyper plane in higher dimension which linear separate two classes as compared to linear SVM. Even if the linear separation is not completely accomplished; only a few examples will be on wrong side of hyper margin.

1. Consider the following training data,



(a) Plot these six training points. Are the classes {+, −} linearly separable?

**Solution:**

****

Yes, these points are linearly separable. Orange line and below points classifies into a negative class whereas, and green line and above is classified in positive class.

(b) Construct the weight vector of the maximum margin hyperplane by inspection and identify the support vectors.

**Solution:**

The decision boundary or the perpendicular bisector pass the mid-point of the data.

The equation of the line is: -

Hence weights of will be (1,1)

**Scaling of W:**

We will select a support vector and put it on the above equation,

Selecting point (0,1) and the scale factor

C = 2; Hence the actual weights are (2,2),

And the value of intercept b = -3.

The weight vector of the maximum margin hyperplane is **(2,2)** and support vectors are:

**(1, 0), (0,1), (2,0), (1,1)**

(c) If you remove one of the support vectors does the size of the optimal margin decrease, stay the same, or increase?

**Solution:**

Cases in removing support vectors,

1. Remove (2, 0); Margin remains the same
2. Remove (0, 1); Margin remains the same
3. Remove (1,0); Margin increases
4. Remove (1,1); Margin increases
5. Please do the case “Predicting Customer Churn at QWE Inc”. You need to purchase this case from the Harvard Business Publishing website (link is available in the syllabus). This is an open-ended case study. Feel free to apply whatever we have learned in class to find the best possible model. You need to clean this data first and explore any methods to construct a good model to predict Churn.

Preparing Data:

R Code:

qweData <- data.frame(caseData)

str(qweData)

rownames(qweData) <- qweData[, 1]

qweData$ID <- NULL

colnames(qweData) <- c("Customer.Age", "Churn", "CHI.Score\_Dec", "CHI.Score\_NovDec",

"Support.Cases\_Dec", "Support.Cases\_NovDec", "Support.Priority\_Dec",

"Support.Priority\_NovDec", "Login\_NovDec", "Blog.Articles\_NovDec",

"Views\_NovDec", "DaysSinceLogin\_NovDec")

qweData$Churn <- as.factor(as.numeric(qweData$Churn))

table(qweData$Churn)

summary(qweData$Customer.Age)

qweData$subscTime<-ifelse(qweData$Customer.Age <=6, "New Customers",

ifelse(qweData$Customer.Age > 6 & qweData$Customer.Age <= 14, "Mid Customers", "Old Customers"))

1. Is Wall’s belief about the dependence of churn rates on customer age supported by the data? To get some intuition, try visualizing this dependence (Hint: You can answer this question based on graphs. No need to run any statistical tests).

**Solution:**

R Code:

qweData$subscTime <- as.factor(qweData$subscTime)

table <- table(qweData$Churn, qweData$subscTime)

table

propTable <- prop.table(table)

propTable

barplot(table, main = "Churn Categories", xlab = "Time Category (Levels)",

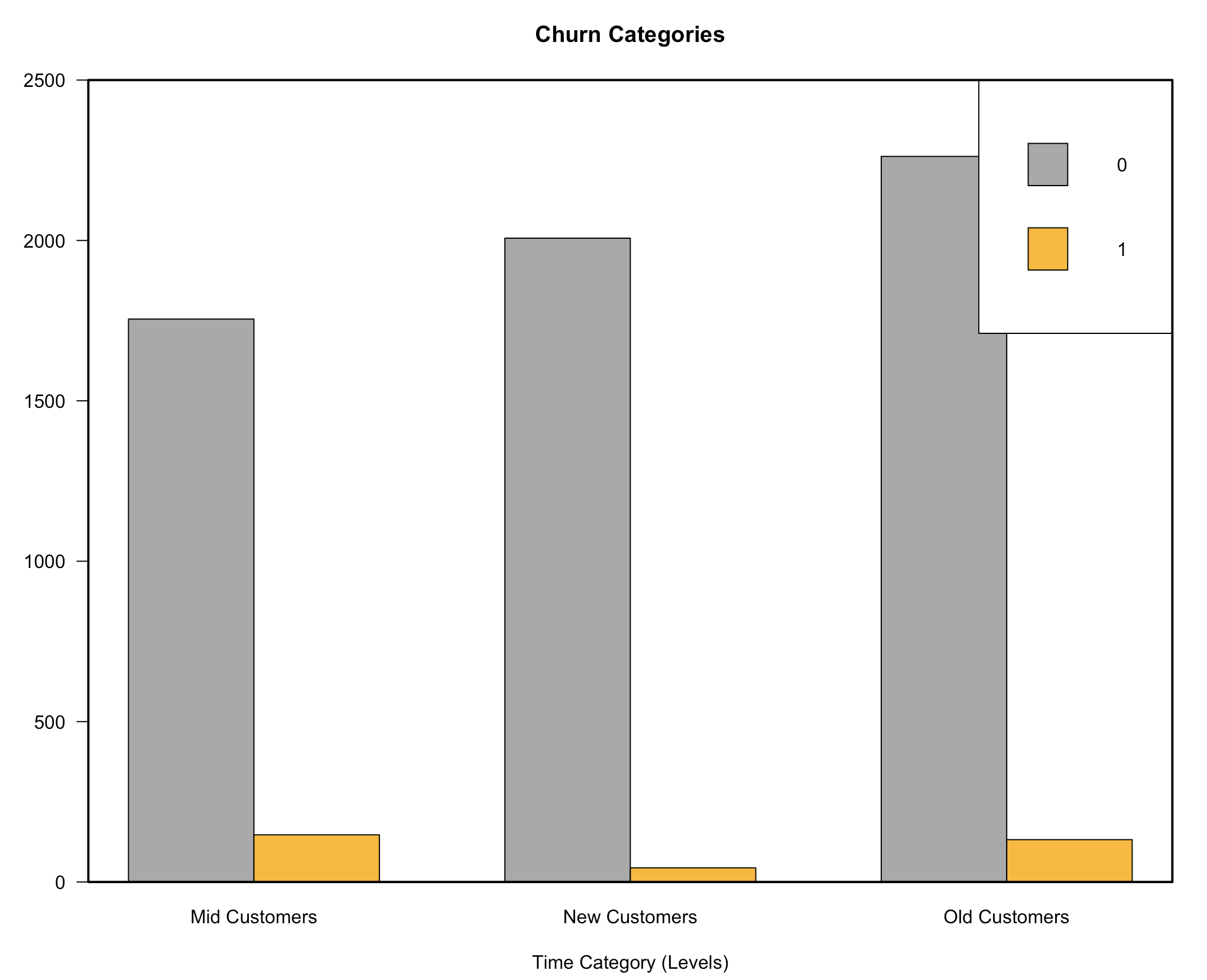
col = c("darkgrey", "darkgoldenrod1"), las = 1, ylim=c(0,2500), beside=T)

legend("topright", legend = c("0", "1"), fill = c("darkgrey", "darkgoldenrod1"))

box(lwd = 2)

highPercentageAttr <- 132/(2262+132) # 5.5%

mediumPercentageAttr <- 147/(1755+147) # 7.7%



|  |  |
| --- | --- |
| Customer Age (months) | Churn Rate |
| Greater than 14 (Old) | 5.5% |
| Greater than 6 and less than 14 (Mid) | 7.7% |
| Less than 6 (New) | 2.1% |

***Conclusion:***

The difference between the churn rates is 2.2% between old customers (>14 months) and fairly old customers (6<age<=14 (months)). Although, this satisfy Wall’s hypothesis, but we cannot rely on just this data and 2 months of analysis to reach to a conclusion, as there is significantly large churn rate with old customers. There may be other factors with age which may be affecting churn of customers.

1. Construct the best model to predict customers churn. You can try different classification models such as Logistic Regression, Neural Network and Decision Trees. You need to find the best settings for these models. Explain how you evaluate your models i.e. what is your evaluation measure? Accuracy? Precision? Recall? etc.

Follow two different approaches to construct a good model: (1) Try to see if your observation in part (a) can help improving your predictions. (2) If you notice, the number of customers who churn is significantly smaller than the number of customers who do not churn. Therefore, this data set is imbalanced. You could try to determine if different level of balancing effects the performance of your models. If you decide to balance your data, please explain the changes you may observe and provide a short reasoning. You can use the ovun.sample() function to balance your data. To learn about balancing data, you can read the following link:

**Solution:**

Sampling test and train data:

index <- sample(1:nrow(qweData),round(0.6\*nrow(qweData)))

trainData<- qweData[index,]

testData <- qweData[-index,]

1. ***Logistic Regression***

# Lets apply the step function for variable selection

null = glm(Churn~1, data= trainData, family = "binomial") # Includes only the intercept

full = glm(Churn~., data= trainData, family = "binomial")

step(null, scope=list(lower=null, upper=full), direction="forward")

1. **UNBALANCED APPROACH**

logit.main <- glm(formula = Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, family = "binomial", data = trainData)

logit.prob <- predict(logit.main,type = 'response',newdata = testData)

logit.pred <- ROCR::prediction(logit.prob, testData$Churn)

logit.perf <- performance(logit.pred,"tpr","fpr")

#finding best cutoff.

opt.cut <- function(perf, pred){

cut.ind <- mapply(FUN = function(x,y,p){d=(x-0)^2+(y-1)^2

ind<- which(d==min(d))

c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])},perf@x.values, perf@y.values,perf@alpha.values)

}

print(opt.cut(logit.perf,logit.pred)) #0.06576468 cutoff.

logit.cut<- rep('0',nrow(testData))

logit.cut[logit.prob < 0.05]<- 1

accuracy.meas(testData$Churn, logit1.pred)

auc <- performance(logit.pred, "auc")

auc <- unlist(slot(auc, "y.values"));auc

**AUC:** 0.662

**Accuracy:** 44.7%

**Precision:** 2.1%

**Recall:** 23.8%

1. **BALANCED APPROACH**
2. *Over sampling*

data\_balanced\_over <- ovun.sample(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, ,data = trainingData, method = "over", seed = 1)$data

logit2\_over <- glm(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, = data\_balanced\_over, family = "binomial")

score <- predict(logit2\_over, type="response", newdata = testData)

pred <- ROCR::prediction(score, testData$Churn)

perf <- performance(pred, "tpr", "fpr")

print(opt.cut(perf,pred)) #0.4869081

logit2\_prob\_over.pred <- rep(0, nrow(testData))

logit2\_prob\_over.pred[score < 0.4869081] <- 1

accuracy.meas(logit2\_prob\_over.pred, testData$Churn)

roc.curve(testData$Churn, logit2\_prob\_over.pred, plotit = F) # 0.671 – AUC

**AUC:** 0.671

**Accuracy:** 34.97%

**Precision:** 2.2%

**Recall:** 29.5%

1. *Under sampling*

data\_balanced\_under <- ovun.sample(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, ,data = trainingData, method = "under", seed = 1)$data

logit2\_under <- glm(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, = data\_balanced\_under, family = "binomial")

score <- predict(logit2\_under, type="response", newdata = testData)

pred <- ROCR::prediction(score, testData$Churn)

perf <- performance(pred, "tpr", "fpr")

print(opt.cut(perf,pred)) # 0.5145652

logit2\_prob\_under.pred <- rep(0, nrow(testData))

logit2\_prob\_under.pred[score < 0.5145652] <- 1

table(logit2\_prob\_over.pred)

confusionMatrix(logit2\_prob\_over.pred, testData$Churn)

roc.curve(testData$Churn, logit2\_prob\_under.pred, plotit = F) # 0.661 – AUC

**AUC:** 0.671

**Accuracy:** 32.77%

**Precision:** 2.5%

**Recall:** 34.4%

1. *Both*

data\_balanced\_both <- ovun.sample(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec ,data = trainingData, method = "both", p=0.5, seed = 1)$data

logit2\_both <- glm(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec, data = data\_balanced\_both, family = "binomial")

score <- predict(logit2\_both, type="response", newdata = testData)

pred <- ROCR::prediction(score, testData$Churn)

perf <- performance(pred, "tpr", "fpr")

print(opt.cut(perf,pred)) # 0.4723422

logit2\_prob\_both.pred <- rep(0, nrow(testData))

logit2\_prob\_both.pred[score < 0.4723422] <- 1

table(logit2\_prob\_both.pred)

accuracy.meas(testData$Churn, logit2\_prob\_both.pred)

confusionMatrix(logit2\_prob\_both.pred, testData$Churn)

roc.curve(testData$Churn, logit2\_prob\_both.pred, plotit = F)

**AUC:** 0.678

**Accuracy:** 34.66%

**Precision:** 2.2%

**Recall:** 34.91%

1. ***Decision Trees (Random Forest)***
   1. **UNBALANCED APPROACH**

rf.mod<- randomForest(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec,, data = trainData, sampsize = nrow(trainData))

rf.pred <- predict(rf.mod ,newdata = testData,type = 'class')

#confusion matrix

table(testData$Churn,rf.pred)

#roc.curve

roc.curve(testData$Churn,rf.pred)

**AUC:** 0.502

**Accuracy:** 94.84%

**Precision:** 9.09%

**Recall:** 1.2%

* 1. **BALANCED APPROACH**

1. *Over sampling*

rf\_train\_over <- randomForest(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec,

data = data\_balanced\_over, sampsize = nrow(data\_balanced\_over))

rf\_pred <- predict(rf\_train\_over, testData, type = "class")

confusionMatrix(rf\_pred, testData$Churn)

accuracy.meas(testData$Churn, rf\_pred)

roc.curve(testData$Churn, rf\_pred, plotit = F)

**AUC:** 0.656

**Accuracy:** 91.45%

**Precision:** 13.97%

**Recall:** 55.94%

1. *Under sampling*

rf\_train\_under <- randomForest(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec,

data = data\_balanced\_under, sampsize = nrow(data\_balanced\_under))

rf\_pred\_under <- predict(rf\_train\_under, testData, type = "class")

confusionMatrix(rf\_pred\_under, testData$Churn)

accuracy.meas(testData$Churn, rf\_pred\_under)

roc.curve(testData$Churn, rf\_pred\_under, plotit = F)

**AUC:** 0.621

**Accuracy:** 61.2%

**Precision:** 7.64%

**Recall:** 62.18%

1. *Both*

rf\_train\_both <- randomForest(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec,

data = data\_balanced\_both, sampsize = nrow(data\_balanced\_both))

rf\_pred\_both <- predict(rf\_train\_both, newdata = testData, type = "class")

confusionMatrix(rf\_pred\_both, testData$Churn)

accuracy.meas(testData$Churn, rf\_pred\_both)

roc.curve(testData$Churn, rf\_pred\_both, plotit = T)

**AUC:** 0.586

**Accuracy:** 94.84%

**Precision:** 13.40%

**Recall:** 25.21%

1. ***Neural Network***
   1. **UNBALANCED APPROACH**

scaled <- as.data.frame(scale(qweData[-c(2,13)], center = mins, scale = maxs - mins))

scaled$Churn <- qweData$Churn

scaled$subscTime <- qweData$subscTime

# breaking testdata in trained and scaled.

index <- sample(1:nrow(qweData),round(0.6\*nrow(qweData)))

train\_scaled <- scaled[index,]

test\_scaled <- scaled[-index,]

names(train\_scaled)

nn<- nnet(Churn ~ DaysSinceLogin\_NovDec + subscTime + CHI.Score\_Dec + Views\_NovDec,

data=train\_scaled,linout=F,size = 10,decay =0.01,maxit=1000)

nn.preds <- predict(nn,test\_scaled,type = 'class')

table(test\_scaled$Churn,nn.preds)

**AUC:** 0.50

**Accuracy:** 95.35%

**Precision:** 0%

**Recall:** 0%

* 1. **BALANCED APPROACH**

1. *Over sampling*

maxs <- apply(data\_balanced\_over[-c(1,5)], 2, max)

mins <- apply(data\_balanced\_over[-c(1,5)], 2, min)

data\_balanced\_over

scaled <- as.data.frame(scale(data\_balanced\_over[-c(1,5)], center = mins, scale = maxs - mins))

scaled$Churn <- data\_balanced\_over$Churn

scaled$subscTime <- data\_balanced\_over$subscTime

train\_scaled <- scaled

maxs <- apply(data\_balanced\_over[, c("CHI.Score\_Dec", "Views\_NovDec", "DaysSinceLogin\_NovDec")], 2, max)

mins <- apply(data\_balanced\_over[, c("CHI.Score\_Dec", "Views\_NovDec", "DaysSinceLogin\_NovDec")], 2, min)

scaled <- as.data.frame(scale(testData[,c("CHI.Score\_Dec", "Views\_NovDec", "DaysSinceLogin\_NovDec")], center = mins, scale = maxs - mins))

scaled$Churn <- testData$Churn

scaled$subscTime <- testData$subscTime

test\_scaled <- scaled

names(data\_balanced\_over)

nn\_over<- nnet(Churn ~ .,

data=data\_balanced\_over, linout=F,size = 10,decay =0.01,maxit=1000)

nn.preds\_over = predict(nn\_over,test\_scaled,type = 'class')

table(test\_scaled$Churn,nn.preds\_over)

**AUC:** 0.603

**Accuracy:** 74.83%

**Precision:** 8.64%

**Recall:** 44.26%

1. *Under Sampling*

maxs <- apply(data\_balanced\_under[-c(1,5)], 2, max)

mins <- apply(data\_balanced\_under[-c(1,5)], 2, min)

data\_balanced\_under

scaled <- as.data.frame(scale(data\_balanced\_under[-c(1,5)], center = mins, scale = maxs - mins))

scaled$Churn <- data\_balanced\_under$Churn

scaled$subscTime <- data\_balanced\_under$subscTime

train\_scaled <- scaled

nn\_under<- nnet(Churn ~ .,data=data\_balanced\_under, linout=F,size = 10,decay =0.01,maxit=1000)

nn.preds\_under = predict(nn\_under,test\_scaled,type = 'class')

table(test\_scaled$Churn,nn.preds\_under)

roc.curve(test\_scaled$Churn, nn.preds\_under)

**AUC:** 0.582

**Accuracy:** 69.98%

**Precision:** 7.35%

**Recall:** 45.08%

1. *Both*

maxs <- apply(data\_balanced\_both[-c(1,5)], 2, max)

mins <- apply(data\_balanced\_both[-c(1,5)], 2, min)

data\_balanced\_both

scaled <- as.data.frame(scale(data\_balanced\_both[-c(1,5)], center = mins, scale = maxs - mins))

scaled$Churn <- data\_balanced\_both$Churn

scaled$subscTime <- data\_balanced\_both$subscTime

train\_scaled <- scaled

nn\_both<- nnet(Churn ~ .,data=data\_balanced\_both, linout=F,size = 10,decay =0.01,maxit=1000)

nn.preds\_both = predict(nn\_both, test\_scaled, type = 'class')

table(test\_scaled$Churn, nn.preds\_both)

roc.curve(test\_scaled$Churn, nn.preds\_both)

**AUC:** 0.619

**Accuracy:** 36.39%

**Precision:** 6.42%

**Recall:** 90.16%

***Conclusion:***

We have 4% of the customers who churned in February. As per this data, we must target the population with highest **precision**; so that we can give offers to the relevant audience. Evaluation measure will be precision in our case.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Imbalanced | | Balanced | | | | | |
| Precision | AUC | Over | | Under | | Both | |
| Precision | AUC | Precision | AUC | Precision | AUC |
| Logistic | 2.1% | 0.662 | 2.2% | 0.671 | 2.5% | 0.671 | 2.2% | 0.678 |
| Random Forest | 9.09% | 0.502 | **13.97%** | **0.656** | 7.64% | 0.612 | 13.40% | 0.586 |
| Neural Net | 0 | 0.50 | 8.64% | 0.603 | 7.35% | 0.582 | 6.42% | 0.619 |

*“The best model to predict customer churn rate of random forest”*

1. Answer Well’s “ultimate question”: provide the list of 100 customers with highest churn probabilities and the top three drivers of churn for each customer (You do not need to include a list of 100 customers. Just mention how you get this list – you can write the code for example).

**Solution:**

rf\_pred\_over.prob <- predict(rf\_train\_over, qweData, type = "prob")

top100CustomerIds <- rownames(head(rf\_pred\_over.prob[order(-rf\_pred\_over.prob[,2]),], 100))

top100CustomerWhoAreLikelyToChurn <- qweData[top100CustomerIds, ]

View(top100CustomerWhoAreLikelyToChurn)

varImp(rf\_train\_over)

Overall

DaysSinceLogin\_NovDec 1325.9576

subscTime 447.8872

CHI.Score\_Dec 2007.2593

Views\_NovDec 1639.2625

***Conclusion:***

We can conclude that the top drivers of customer retention are:

1. DaysSinceLogin\_NovDec (DaysSinceLogin\_0\_1)
2. subscTime (Customer\_Age)
3. CHI.Score\_Dec (Customer Happiness Index\_0)

Libraries used:

library(tidyverse)

library(ggplot2)

library(randomForest)

library(rpart)

library(car)

library(ROCR)

library(neuralnet)

library(ROSE)

library(caret)

library(nnet)

library(ISLR)