Akanksha Bansal	Assignment 3
Data Mining	17,March 2014

Problem 1 Part a)

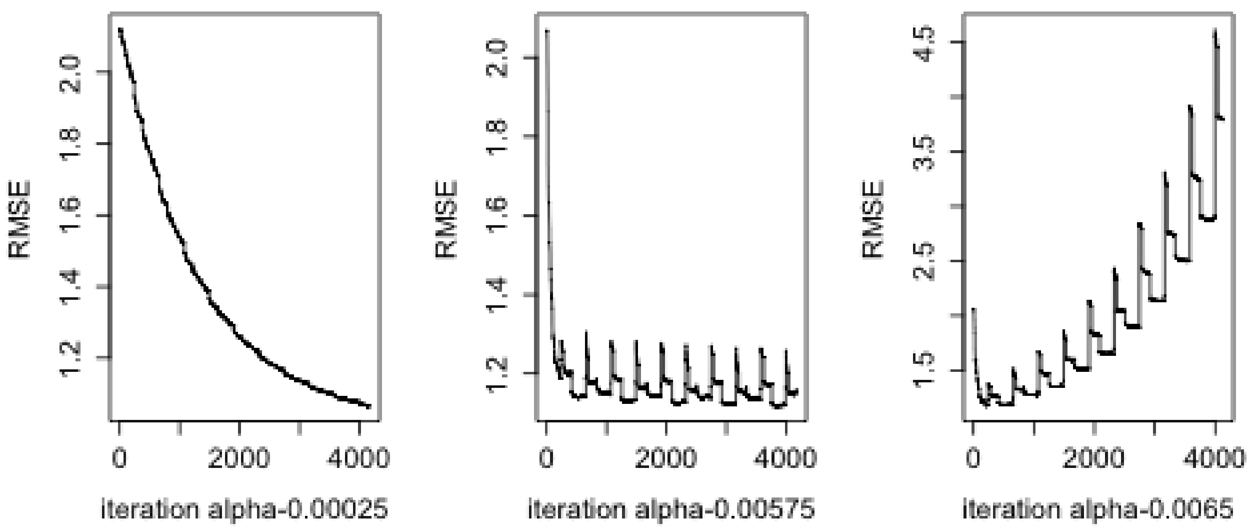


Image indicating the Error values for the various values of Alpha

Training Number	
α	Iteration Number
0.00025	4170
0.00575	4137
0.0065	4133

Error Values	
Model	MSE
MLR	1564
$SGD\alpha = 0.00025$	41.6
$SGD\alpha = 0.00575$	69.2
$SGD\alpha = 0.0065$	69.8

The SGD model has been able to reduce the amount of error by a huge factor. This is because now the coefficients are being adjusted in a manner which reduces error value. The smaller the value of α the smaller the MSE. This is because α determines the step size , and in SGD, the smaller the step size, the better the approximation. However, as the value of α reduces the number of iterations increases, thus increasing the runtime. When the value of the threshold is reduced, from 0.1 to 0.01, the number of iterations needed for the training to complete jumps from the range to 4000 to the range of 40000. The behavior of SGD depends a lot on the choice of gradient function and α. Therefor for different gradient functions, for same value of α we might get different iteration numbers when the solution would converge.

Problem 2 For a n-D Space lets say that the convex hulls intersect and thus there exists a z for two such set of points. The convex hull of such a set of points can be represented as :
$\sum_i \alpha_i x_i$ where $\alpha_i > 0$ and $\sum_i \alpha_i = 0$

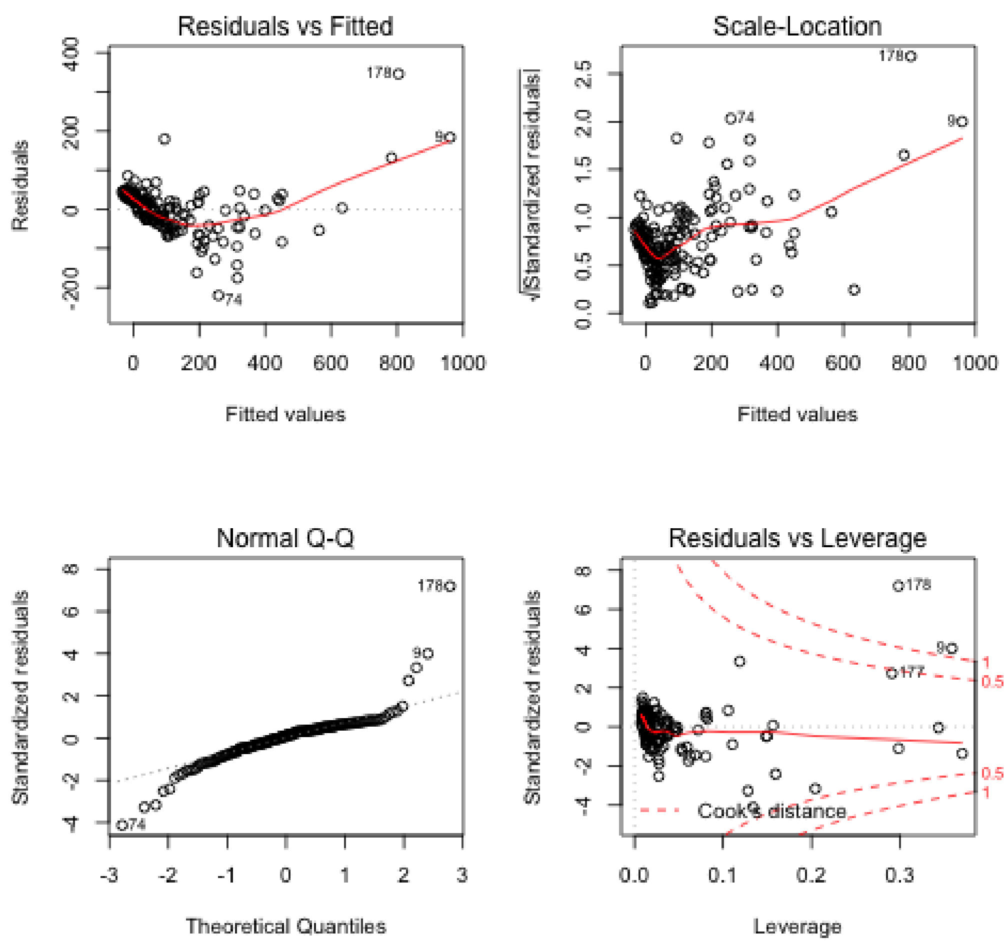
Thus z can be represented as
$z = \sum_i \alpha_i x_i = \sum_j \gamma_j y_j$
and $\sum_i \alpha_i = \sum_j \gamma_j = 1$

If these two sets of points are linearly separable, there exists a separating hyper-plane determined by w, w_0 such that
$w^T x_i + w_0 > 0 \forall i$
$w^T y_j + w_0 < 0 \forall j$
$w^T z + w_0 = w^T \sum_i \alpha_i x_i + w_0$
$= \sum_i \alpha_i w^T x_i + w_0 \sum_i \alpha_i$ as $\sum_i \alpha_i = 1$
$= \sum_i \alpha_i (w^T x_i + w_0)$
Similarly
$w^T z + w_0 = \sum_i \gamma_i (w^T y_i + w_0)$
$\sum_i \alpha_i (w^T x_i + w_0) = \sum_i \gamma_i (w^T y_i + w_0)$
$\sum_i \alpha_i (w^T x_i + w_0) = \sum_i \gamma_i (w^T y_i + w_0)$
This implies $\alpha_i = \gamma_j = 0 \forall i, j$

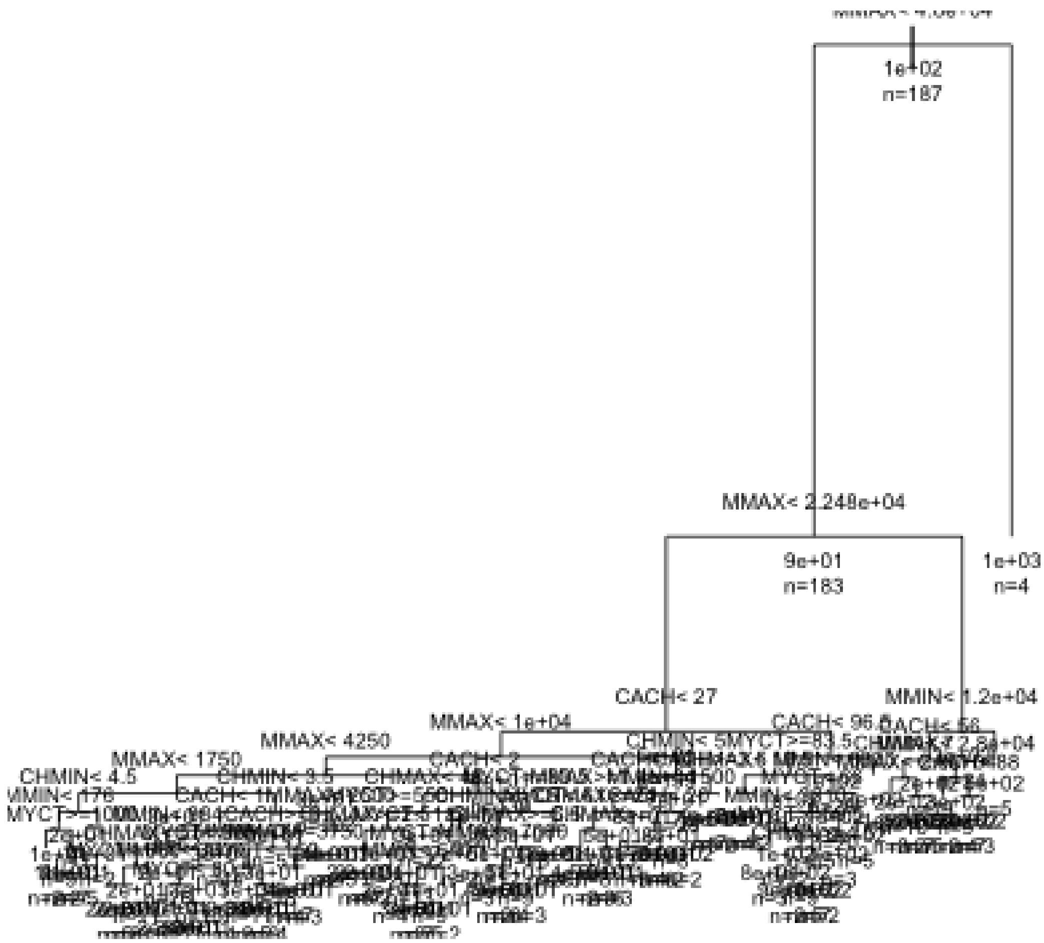
However this contradicts our previous assumption . Thus assuming the two sets of points are linearly separable leads us to a contradiction. Thus if the convex hulls of two separate sets of points intersect $\Rightarrow$ the two sets are not linearly separable.

Problem 3 Part a) A Regression tree is first generated with cp = 0. This creates an overfitted model. And then the tree is pruned so that the overfitting can be removed. The pruned tree is used for the computations needed in the rest of the parts.

RT	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	2053.3	28.2	0.924	45.3

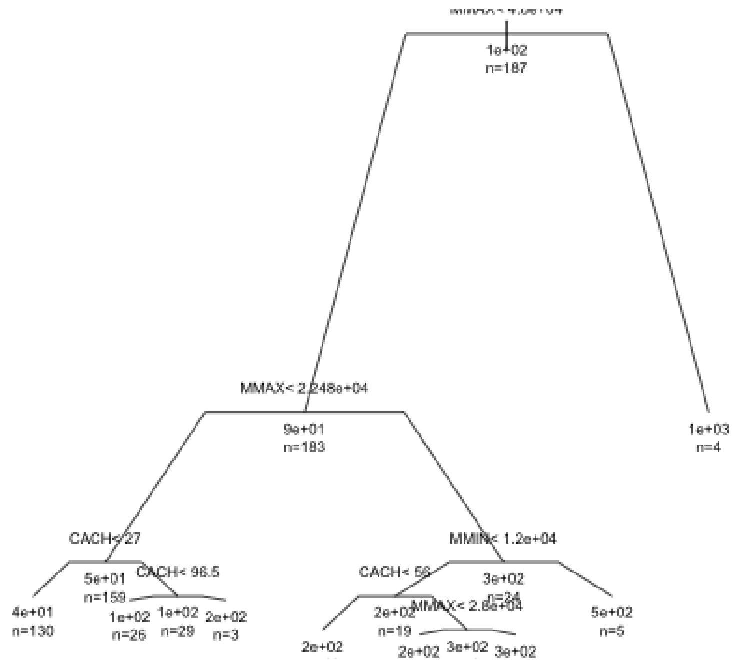


MLR PLot

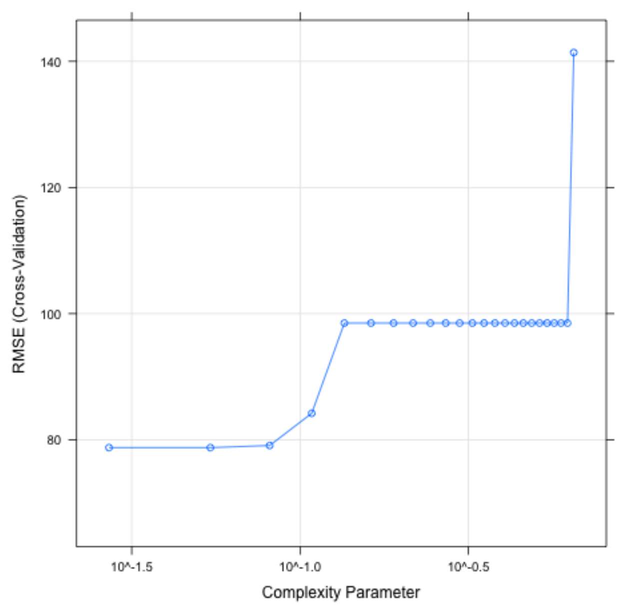


OverFit Regression tree

Part b)



Pruned Regression tree to overcome the Overfitting done in previous step after choosing the value of cp which corresponds to minimum error value.

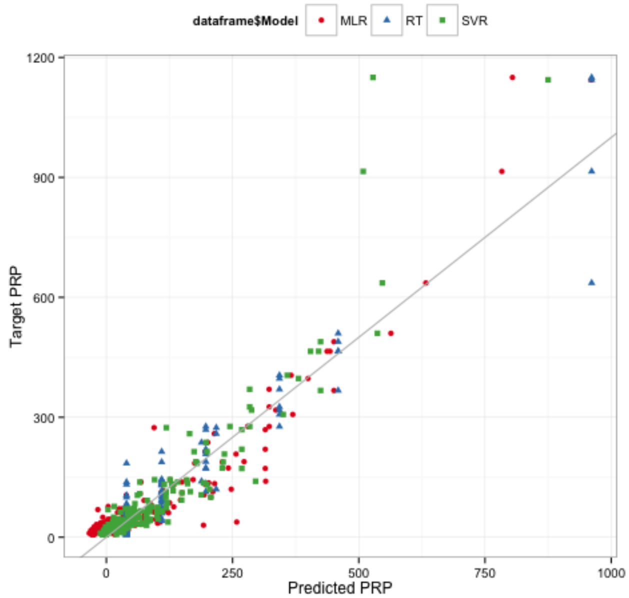


5-fold Cross Validation results for the Regression tree plotted for the various values of complexity parameter

Part c)

Error Values				
SVR	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	4640	32.6	0.85	68.1
$Train = 1, Test = test$	3938.7	33.0	0.761	62.8
$Train = 2, Test = test$	321276	1136	0.742	567

Part d)



ggPlot showing the combined plots of the three models

With the addition of outliers the overall error values increased for all the three models. However out of the three models, MLR was affected the most with the addition of outliers which is expected. Because SVR makes use of Huber loss function the increase in the error values for SVR is not huge. Regression trees have the least recorded error values for all three types of test cases out of the three models. Thus, Regression trees had the best performance with given tuning. The choice of complexity of the model, will affect the performance of this model.

Error Values				
MLR	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	3166.4	37.9	0.882	56.3
$Train = 1, Test = test$	5475.4	44.1	0.69	74.0
$Train = 2, Test = test$	394011	1247	0.622	628
RT	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	2053.3	28.2	0.924	45.3
$Train = 1, Test = test$	3054.8	40.4	0.817	55.3
$Train = 2, Test = test$	401037	1114	0.565	633
SVR	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	4640	32.6	0.85	68.1
$Train = 1, Test = test$	3938.7	33.0	0.761	62.8
$Train = 2, Test = test$	321276	1136	0.742	567

Part f)

Error Values				
Random Forest	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	793.1	14.1	0.973	28.3
RT	MSE	MAE	R^2	RMSE
$Train = 1, Test = 1$	2053.3	28.2	0.924	45.3

The performance of Random Forests is even more promising than that of the Regression Tress.

Problem 4 $P(C_1|X) = \frac{P(C_1)P(X|C_1)}{P(X)}$

The division line occurs where $P(C_1|X) = P(C_2|X)$

$P(C_1) = 1/5$ and $P(C_2) = 4/5$

$P(X|C_1) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X-\mu)(X-\mu)^T}{2\sigma^2}}$ where σ = variance of C_1 which is 1 (because the covariance matrix for C_1 and C_2 is an identity matrix) μ= [2 0]

Similarly $P(X|C_2) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(X-\mu)(X-\mu)^T}{2}}$ where σ = variance of C_2 which is 1 μ = [0 2]

Thus $\frac{1}{5} * \frac{1}{\sqrt{2\pi}} e^{-\frac{(X-\mu_1)(X-\mu_1)^T}{2}} = \frac{4}{5} \frac{1}{\sqrt{2\pi}} e^{-\frac{(X-\mu_2)(X-\mu_2)^T}{2}}$

$e^{-\frac{(X-\mu_1)(X-\mu_1)^T}{2}} = 4 * e^{-\frac{(X-\mu_2)(X-\mu_2)^T}{2}}$ Taking log of both sides.

$-\frac{(X-\mu_1)(X-\mu_1)^T}{2} = 2\log 2 + -\frac{(X-\mu_2)(X-\mu_2)^T}{2}$

$-(X-\mu_1)(X-\mu_1)^T = 4\log 2 + -(X-\mu_2)(X-\mu_2)^T$

Replacing the value of X by x,y and values of μ_1 and μ_2

$(x-2)^2 + y^2 = -4\log 2 + x^2 + (y-2)^2$

$= x^2 - 4x + 4 + y^2 = -4\log 2 + x^2 + y^2 - 4y + 4$

$-x = -\log 2 - y$

$\log 2 = x - y$

Part b)

Now because the cost function has changed , the division line will occurs where $2P(C_2|X) - P(C_1|X) = 0$ This is because the total cost for misclassification is : $2k * P(C_2|X) + k * P(C_1|X)$.

We need to minimize this value. Thus it is ok to misclassify a C_2 value as C_1 unless the penalty of misclassifying is not twice the cost.

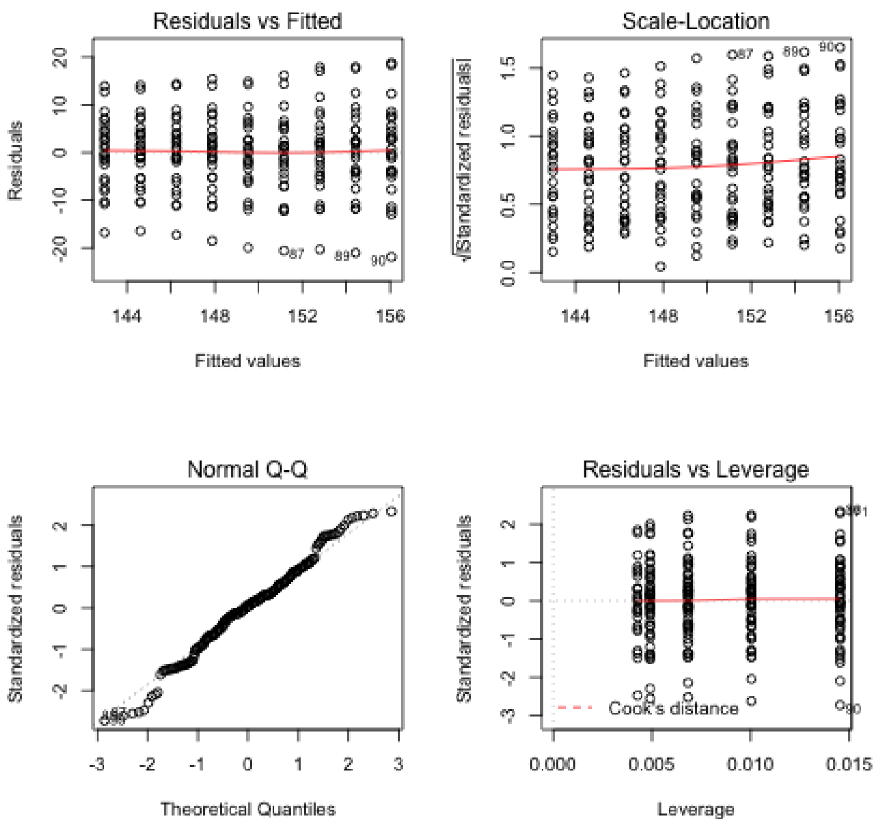
Using the values given above and the above equation we get.
$\frac{1}{5} * \frac{1}{\sqrt{2\pi}} e^{-\frac{(X-\mu_1)(X-\mu_1)^T}{2}} = 2 * \frac{4}{5} \frac{1}{\sqrt{2\pi}} e^{-\frac{(X-\mu_2)(X-\mu_2)^T}{2}}$
$e^{-\frac{(X-\mu_1)(X-\mu_1)^T}{2}} = 8 * e^{-\frac{(X-\mu_2)(X-\mu_2)^T}{2}}$ Taking log of both sides.
$-\frac{(X-\mu_1)(X-\mu_1)^T}{2} = \log 8 - \frac{(X-\mu_2)(X-\mu_2)^T}{2}$
$(X-\mu_1)(X-\mu_1)^T = -2\log 8 + (X-\mu_2)(X-\mu_2)^T$
Replacing the value of X by x,y and values of μ_1 and μ_2
$(x-2)^2 + y^2 = -2\log 8 + x^2 + (y-2)^2$
$x^2 - 4x + 4 + y^2 = -2\log 8 + x^2 + y^2 - 4y + 4$
$-2x = -\log 8 - 2y$
$\log 8 = 2x - 2y$

Problem 5 a)
$P(B = good, F = empty, G = empty, S = yes) = P(B = good) * P(F = empty) * P(G = empty|B = good, F = empty) * P(S = yes|B = good, F = empty)$
$= 0.9 * 0.2 * 0.8 * 0.2 = 0.00576$

b)
$P(B = bad, F = empty, G = notempty, S = no) = P(B = bad) * P(F = empty) * P(G = notempty|B = bad, F = empty) * P(S = no|B = bad, F = empty)$
$= 0.1 * 0.2 * 0.1 * 1.0 = 0.002$

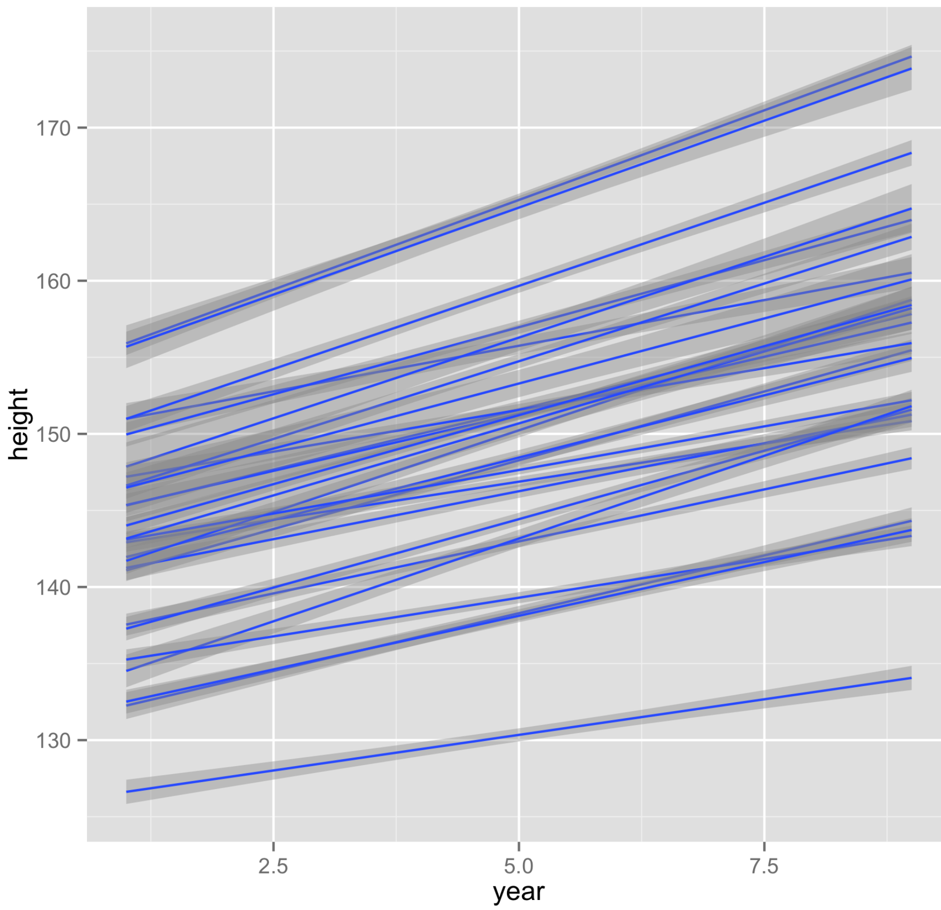
c) Given that the battery is bad, compute the probability that the car will start. =
$P(S = yes|B = Bad) = \sum_{g,f} P(S = yes, B = Bad, G = g, F = f)/P(B = Bad)$
$= (P(S = yes, B = bad, F = empty, G = empty) + P(S = yes, B = bad, F = empty, G = notempty) + P(S = yes, B = bad, F = notempty, G = empty) + P(S = yes, B = bad, F = notempty, G = notempty))/P(B = Bad)$
Another way to calculate this can be
$P(S = yes|B = Bad) = P(F = empty) * P(S = yes|B = bad, F = empty) + P(F = notempty) * P(S = yes|B = bad, F = notempty)$
$= (0.2 * 0 + 0.8 * 0.1) = 0.08$

Problem 6 Part a)

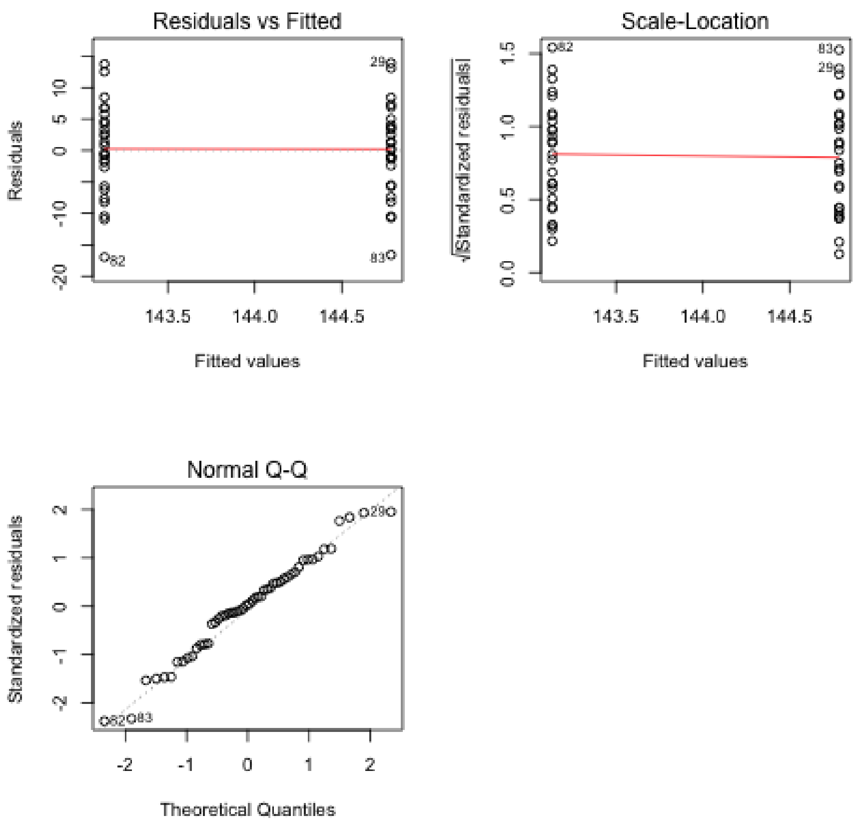


MLR where the model is ignoring the variable id

Part b) This image is able to capture all the multiple relations which can be used to define the given height and year relation. This model is grouping the values of the variable id.

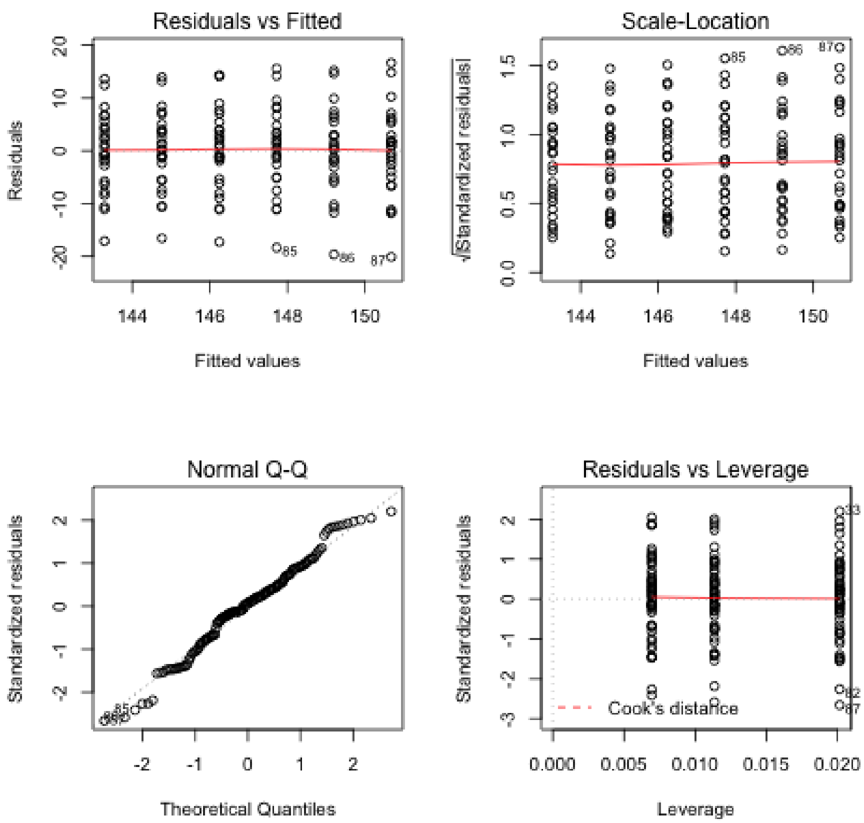


Part c)



Error Values	
Model	MSE
Global	68.7
Local	328
MLM	4.23

Part d)



Error Values	
Model	MSE
Global	80.1
Local	68.7
MLM	2.72

Part e) Looking at these values we can say that MLM can perform better only when there some amount of data for the various classes. When the Amount of training data is less, MLM are not ale to perform well. The Error value for the local model indicates similar detail that when the training value is small the models like MLM are unable to predict correctly and become unreliable. However, they are able to provide very satisfactory results with increase in a small amount of training data.

R - Code

Problem 1

```
rm(list = ls())
setwd("/Users/bansal/Google Drive/Spring2014/DataMining/HW3/")
library(MASS)
library(hydroGOF)
library(ggplot2)
test = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/forestfire-test.csv
    ",header=TRUE)
train = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/forestfire-train.
    csv",header=TRUE)

for (j in 1:length(train))
{
  train[,j] = (train[,j] - mean(train[,j]))/sd(train[,j])
  test[,j] = (test[,j] - mean(train[,j]))/sd(train[,j])
}

# squared error cost function
cost <- function(x, y, theta) {
  return (rmse(as.vector(x * theta) , y))
}

#Gradient Function
gradFun <- function(xn, yn, theta) {
  return (xn * (yn - xn * theta) )
}

# learning rate and iteration limit
#alphas <- c(0.00025)

alphas <- c(0.00025, 0.00575, 0.0065)
num.iterations <- 10
threshold <- 0.1

temp <- as.matrix(train)
x <- temp[,1:9]
x[,9] <- x[,9]-x[,9] +1
y <- temp[,c="area"]

temp <- as.matrix(test)
xtest <- temp[,1:9]
xtest[,9] <- xtest[,9]-xtest[,9] +1
ytest <- temp[,c="area"]


# gradient descent
gradient.descent <- function(X, Y, alpha=0.1, num.iterations=1000, threshold=0.1) {

  # initialize coefficients
  xn = X[1,]
  yn = Y[1]
  theta1 <- rep(0.5,9)
  gradient <- gradFun(xn, yn, theta1)
  theta2 <- theta1 + alpha * gradient
  #Initiliaze History
```

```
  error_trace <- NULL
  test_error_trace <- NULL
  theta_trace <- theta2
  for (i in 1:num.iterations){
    for (j in 1:417){
      xn = X[j,]
      yn = Y[j]
      theta1 <- theta2
      gradient <- gradFun(xn,yn, theta1)
      theta2 <- theta1 + alpha * gradient
      cost <-  rmse(as.vector(X * theta2) , Y)
      # Traces needed for the graph
      theta_trace <- c(theta_trace,theta2)
      error_trace <- c(error_trace,cost)
      test_error_trace <- c(test_error_trace, cost(xtest, ytest, theta2))

      if(abs(theta2 - theta1) > threshold)
        break
    }
  }
  return (list(error= error_trace, theta= theta_trace, testError = test_error_trace))
}
# plot data and converging fit
vary.alpha <- lapply(alphas, function(alpha) gradient.descent(x,
                                                              y, alpha=alpha, num.
    iterations=num.iterations))

par(mfrow = c(2, 3))
for (j in 1:3) {
  plot((vary.alpha[[j]])$error, ylab="RMSE", xlab=paste("iteration alpha",alphas[j],
    sep="-"), type="l")
}
#print graphs for the iterations
dev.copy(png,'Q1plot2.png')
dev.off()

lm_model<- lm(area ~ FFMC + DMC + DC + ISI + temp + RH + wind + rain + 1, data=train)
summary(lm_model)

png("Q1plot_lm_all.png")
layout(matrix(1:4, ncol = 2))
plot(lm_model)
layout(1)
dev.off()
pred_lm <- predict(lm_model, newdata = test)
lm.mse <- regr.eval(test$area, pred_lm,
          stats = c( "mse"),
          train.y = NULL)
lm.mse

mse_alpha1<- (vary.alpha[[1]])$testError[4170]
mse_alpha1
mse_alpha2<- (vary.alpha[[2]])$testError[4134]
mse_alpha2
mse_alpha3<- (vary.alpha[[3]])$testError[4133]
mse_alpha3
```

Problem 3

```
#install.packages("randomForest")
rm(list = ls(all.names = TRUE))
setwd("/Users/bansal/Google Drive/Spring2014/DataMining/HW3/")
library(MASS)
library(hydroGOF)
library(ggplot2)
library(rpart)
library(caTools)
library(grid)
library(lattice)
library(DMwR)
library(DAAG)
library(caret)
library(randomForest)

train_data = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/Machine1.csv"
    , header = TRUE)
test_data = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/Machinetest.
    csv", header = TRUE)

#regression
lm_model<- lm(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data=train_data)
summary(lm_model)

png("Q3plot_lm_all.png")
layout(matrix(1:4, ncol = 2))
plot(lm_model)
layout(1)
dev.off()

#Some additional values which can com en handy
#lm.rmse<- sqrt(mse(lm.model$fitted.values,train_data$PRP))
#lm.coef<-coefficients(lm.model) # model coefficients
#lm.ci<-confint(lm.model, level=0.95) # CIs for model parameters
#lm.resd<-residuals(lm.model) # residuals
#lm.anova<-anova(lm.model) # anova table
#lm.cov<-vcov(lm.model) # covariance matrix for model parameters
#lm.influ<-influence(lm.model) # regression diagnostics

pred_lm<-fitted(lm_model) # predicted values
pred_lm_test<-predict(lm_model, newdata = test_data )
regr.eval(train_data$PRP, pred_lm,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_lm, train_data$PRP)**2

cv.lm(df=train_data, lm_model, m=5)
dev.copy(png,'Q3plot_lm_cross.png')
dev.off()

#Regression Tree RT
rt_model <- rpart(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data = train_data
```

```
    ,
                    control = rpart.control(xval = 5, minbucket = 2, cp = 0))
#use of cp and max depth to control over and under fitting of the model
plot(rt_model)
text(rt_model,use.n=TRUE, all=TRUE, cex=.5)
dev.copy(png,'Q3plot_rt.png')
dev.off()

#Train the model using the Cross validation (upto 5)
indx <- createFolds(train_data$PRP, returnTrain = TRUE)
ctrl <- trainControl(method = "cv", index = indx, number = 5)
set.seed(187)

cartTune <- train(x = train_data, y = train_data$PRP,
                    method = "rpart",
                    tuneLength = 25,
                    preProc = c("center", "scale"),
                    trControl = ctrl)
cartTune
### Plot the tuning results
plot(cartTune, scales = list(x = list(log = 10)))
dev.copy(png,'Q3plot_rt_crossValidation.png')
dev.off()

#visualise the Corss validation of the model
plotcp(rt_model)
dev.copy(png,'Q3plot_rt_complexity.png')
dev.off()

#prune Chose the tree with ideal value of CP based on the Error values collected.
cptable <- as.data.frame(rt_model$cptable)
alpha <- cptable$CP[which.min(cptable$xerror)]
rt_pruned <- prune(rt_model,cp=alpha)
plot(rt_pruned, branch = 0.5 )
text(rt_pruned,use.n=TRUE, all=TRUE, cex=.5)
dev.copy(png,'Q3plot_rt_prunedTree.png')
dev.off()

#Capture the Error Values
#Rsquare
rsq.rpart(rt_pruned)
pred_rt <- predict(rt_pruned)
cor(pred_rt, train_data$PRP)**2
pred_rt_test<-predict(rt_pruned, newdata = test_data )
cor(pred_rt_test, test_data$PRP)**2
#sqrt(mse(pred, train_data$PRP))
regr.eval(train_data$PRP, pred_rt,
           stats = c("mae", "mse", "rmse"),
           train.y = NULL)


#part c
library(kernlab)

svr <- ksvm(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data=train_data,
             type="eps-svr", kernel="vanilladot", cross=5)
```

```r
summary(svr)

pred_svr <- predict(svr, train_data )
pred_svr_test<-predict(svr, newdata = test_data )

regr.eval(train_data$PRP, pred_svr,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_svr, train_data$PRP)**2

#Part d
pred <- pred_lm
pred <- append(pred,pred_rt)
pred <- append(pred,pred_svr)
target <- train_data$PRP
target <- append(target,train_data$PRP)
target <- append(target,train_data$PRP)
model <- rep("MLR",dim(train_data)[1])
model <- append(model,rep("RT",dim(train_data)[1]))
model <- append(model,rep("SVR",dim(train_data)[1]))

dataframe <- data.frame(Pred = pred, Target = target, Model = model)
ggplot(data.frame(dataframe), aes(x = dataframe$Pred, y = dataframe$Target, color=dataframe$Model, shape=dataframe$Model)) + theme_bw() +
  geom_point() + geom_abline(intercept=0, slope=1, color="grey") +
  scale_colour_brewer(palette="Set1") + xlab("Predicted PRP") +
  ylab("Target PRP") + theme(legend.position = "top")
dev.copy(png,'Q3plot_part_d.png')
dev.off()


#part e
regr.eval(test_data$PRP, pred_lm_test,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_lm_test, test_data$PRP)**2
regr.eval(test_data$PRP, pred_rt_test,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_rt_test, test_data$PRP)**2
regr.eval(test_data$PRP, pred_svr_test,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_svr_test, test_data$PRP)**2

train_data2 = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/Machine2.csv", header = TRUE)

lm_model<- lm(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data=train_data2)
cv.lm(df=train_data, lm_model, m=5)
pred_lm_test<-predict(lm_model, newdata = test_data )
regr.eval(test_data$PRP, pred_lm,
          stats = c("mae", "mse", "rmse"),
          train.y = NULL)
cor(pred_lm_test, test_data$PRP)**2
```

```
#Regression Tree RT
rt_model <- rpart(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data = train_
    data2,
                    control = rpart.control(xval = 5, minbucket = 2, cp = 0))
#prune Chose the tree with ideal value of CP based on the Error values collected.
cptable <- as.data.frame(rt_model$cptable)
alpha <- cptable$CP[which.min(cptable$xerror)]
rt_pruned <- prune(rt_model,cp=alpha)

#Rsquare
rsq.rpart(rt_pruned)
pred_rt_test<-predict(rt_pruned, newdata = test_data )
regr.eval(test_data$PRP, pred_rt,
           stats = c("mae", "mse", "rmse"),
           train.y = NULL)
cor(pred_rt_test, test_data$PRP)**2

#SVR
svr <- ksvm(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX , data=train_data2,
             type="eps-svr", kernel="vanilladot", cross=5)
pred_svr_test<-predict(svr, newdata = test_data )
regr.eval(test_data$PRP, pred_svr,
           stats = c("mae", "mse", "rmse"),
           train.y = NULL)
cor(pred_svr_test, test_data$PRP)**2

#part f
randf <- randomForest(PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX, data=train_
    data, ntree=10 )
pred_randf <- predict(randf, newdata = train_data)
cor(pred_randf, train_data$PRP)**2

regr.eval(train_data$PRP, pred_randf,
           stats = c("mae", "mse", "rmse"),
           train.y = NULL)
```

Problem 6

```
#install.packages("nlme")
rm(list = ls())
setwd("/Users/bansal/Google Drive/Spring2014/DataMining/HW3/")
library(MASS)
library(hydroGOF)
library(ggplot2)
library(nlme)
oxboys = read.csv("~/Google Drive/Spring2014/DataMining/HW3/HW3Data/oxboys.csv")

global <- lm(formula = height ~ year , data = oxboys)
summary(global)
png("Q6_Part_a.png")
layout(matrix(1:4, ncol = 2))
plot(global)
layout(1)
dev.off()

ggplot = ggplot(oxboys, aes(x=year, y=height, group=id)) + stat_smooth(method="lm")
```

```
ggsave('Q6_Part_b.png')

oxboys.train <- subset(oxboys, year < 3)
oxboys.test <- subset(oxboys, year >= 3)

global <- lm(formula = height ~ year , data = oxboys.train)
testGolbal <- predict.lm(global , newdata = oxboys.test)
mse(testGolbal,oxboys.test$height )
summary(global)
png("Q6_Part_c_a.png")
layout(matrix(1:4, ncol = 2))
plot(global)
layout(1)
dev.off()
list.mse  <- NULL

for( key in unique(oxboys.train$id)){
  oxboys.train.ind <- subset(oxboys.train, id==key)
  oxboys.test.ind <- subset(oxboys.test, id==key)
  # build a linear model with oxboys.train.ind
  mlr <- lm(height ~ year , data = oxboys.train.ind)
  summary(mlr)
  #plot(mlr)

  # predict on oxboys.test.ind, and measure MSE
  testModel<-predict.lm(mlr, newdata = oxboys.test.ind)
  mlrMSE<-mse(testModel,oxboys.test.ind$height)
  list.mse <- c(list.mse,mlrMSE)
}
sum(list.mse)

#lme(height ~ year + id , data = oxboys , random = ~id/year , method = "ML", na.action = na.omit)
mlm.obj <- lme(height~year, data=oxboys.train, random=list(id=pdDiag(~year)))
testmlm <- predict(mlm.obj, newdata=oxboys.test, level=1)
# calculate MSE
mse(testmlm, oxboys.test$height)


#PART D
oxboys.train <- subset(oxboys, year < 7)
oxboys.test <- subset(oxboys, year >= 7)

global <- lm(formula = height ~ year , data = oxboys.train)
testGolbal <- predict.lm(global , newdata = oxboys.test)
mse(testGolbal,oxboys.test$height )
summary(global)
png("Q6_Part_d_a.png")
layout(matrix(1:4, ncol = 2))
plot(global)
layout(1)
dev.off()
list.mse  <- NULL
for( key in unique(oxboys.train$id)){
  oxboys.train.ind <- subset(oxboys.train, id==key)
  oxboys.test.ind <- subset(oxboys.test, id==key)
```

```r
  # build a linear model with oxboys.train.ind
  mlr <- lm(height ~ year , data = oxboys.train.ind)
  #summary(mlr)
  # predict on oxboys.test.ind, and measure MSE
  testModel<-predict.lm(mlr, newdata = oxboys.test.ind)
  mlrMSE<-mse(testModel,oxboys.test.ind$height)
  list.mse <- c(list.mse,mlrMSE)
}
sum(list.mse)

mlm2.obj <- lme(height~year, data=oxboys.train, random=list(id=pdDiag(~year)))
test2mlm <- predict(mlm2.obj, newdata=oxboys.test, level=1)
# calculate MSE
mse(test2mlm, oxboys.test$height)
```