Jian Sun, Michael Chiang
December 15th, 2016

# R CODE

# Appendix

### 1 Logistic Regression

```
library(xlsx)
library(ElemStatLearn)
library(ROCR)
UCC=read.xlsx("/Users/LoveChina/Documents/6242/Final Project/UCCtrain.xlsx",
        sheetName = 'S', header = TRUE)
UTT=read.xlsx("/Users/LoveChina/Documents/6242/Final Project/UCCtest.xlsx",
        sheetName = 'S', header = TRUE)
Unpay.Train=UCC
Unpay.Test =UTT
Unpay.Train[,2]=as.factor(Unpay.Train[,2])
Unpay.Train[,4]=as.factor(Unpay.Train[,4])
Unpay.Train[,24]=as.factor(Unpay.Train[,24])

Unpay.Test[,2]=as.factor(Unpay.Test[,2])
Unpay.Test[,4]=as.factor(Unpay.Test[,4])
Unpay.Test[,24]=as.factor(Unpay.Test[,24])

LR.credi=glm(Unpay.Train$default.payment.next.month~., family=binomial("logit"), data=UCC)
drop1(LR.credi, test = "Chisq")
LR.credi=glm(Unpay.Train$default.payment.next.month~.-PAY_AMT4, family=binomial("logit"),
data=UCC)
drop1(LR.credi, test = "Chisq")
……
LR.credi=glm(Unpay.Train$default.payment.next.month~.-PAY_AMT4-BILL_AMT2-PAY_AMT6-PAY_
4-BILL_AMT3-SEX-PAY_AMT3-BILL_AMT5
        -BILL_AMT6-PAY_6-PAY_5-EDUCATION-LIMIT_BAL, family=binomial("logit"), data=UCC)
drop1(LR.credi, test = "Chisq")
LR.train1=glm(UCC$default.payment.next.month~MARRIAGE+AGE+PAY_0+PAY_2+PAY_3+BILL_AM
T1+
        BILL_AMT4+PAY_AMT1+PAY_AMT2+PAY_AMT5, family=binomial("logit"), data=UCC)
summary(LR.train1)
MX1=predict(LR.train1, UTT, type="response")
MSE1=sum((UTT$default.payment.next.month-MX1)^2)/nrow(UTT)
MSE1
pred.vals1 <- prediction(MX1, Unpay.Test$default.payment.next.month)

perf1 <- performance(pred.vals1, measure = "tpr", x.measure = "fpr")
```

## plot the ROC curve for the predicted response values by the fitted model

```
plot(perf1, colorize=TRUE)
performance(pred.vals1, measure = "auc")@y.values [[1]]
```

```
Call:
glm(formula = UCC$default.payment.next.month ~ MARRIAGE + AGE +
    PAY_0 + PAY_2 + PAY_3 + BILL_AMT1 + BILL_AMT4 + PAY_AMT1 +
    PAY_AMT2 + PAY_AMT5, family = binomial("logit"), data = UCC)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5895  -0.6789  -0.5481  -0.3035   3.1742

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.275e+00  2.318e-01  -5.501 3.77e-08 ***
MARRIAGE    -1.896e-01  7.728e-02  -2.453 0.014162 *
AGE          1.206e-02  4.257e-03   2.832 0.004629 **
PAY_0        5.649e-01  4.363e-02  12.948  < 2e-16 ***
PAY_2        1.111e-01  4.901e-02   2.267 0.023386 *
PAY_3        1.445e-01  4.445e-02   3.251 0.001152 **
BILL_AMT1   -4.858e-06  1.317e-06  -3.688 0.000226 ***
BILL_AMT4    3.113e-06  1.490e-06   2.089 0.036728 *
PAY_AMT1    -8.345e-06  4.368e-06  -1.910 0.056070 .
PAY_AMT2    -1.650e-05  5.616e-06  -2.939 0.003293 **
PAY_AMT5    -1.066e-05  4.908e-06  -2.171 0.029895 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5207.7  on 4999  degrees of freedom
Residual deviance: 4578.6  on 4989  degrees of freedom
AIC: 4600.6

Number of Fisher Scoring iterations: 6
```

**2 Linear Discriminant Analysis**

```
library(classifly)
Unpay.Train=UCC
Unpay.Test =UTT

Response1 = as.factor(Unpay.Train[,24])
Predictor1 = as.matrix(Unpay.Train[,1:23])

Response2 = as.factor(Unpay.Test[,24])
```

```
Predictor2 = as.matrix(Unpay.Test[,1:23])

TrainD=data.frame(Response1, Predictor1)
TestD =data.frame(Response2, Predictor2)

library(MASS)
Discri = lda(Predictor1, Response1,data=TrainD)
##test predictive
discri=predict(Discri, Predictor2, prior = Discri$prior, method = "predictive")

MSE2=sum((Unpay.Test[,24]-discri$x)^2)/length(Unpay.Test[,24])
MSE2
summary(discri)
ct <- table(discri$class,Response2)
RightRate2=sum(diag(prop.table(ct)))
RightRate2
###do some plot
library(ggplot2)
library(scales)
library(gridExtra)

proplda = Discri$svd^2/sum(Discri$svd^2)
plda <- predict(Discri, Predictor2, prior = Discri$prior, method = "predictive")

dataset = data.frame(Response2,lda = plda$posterior)

LD2=c(1:5000)
K1=dataset$lda.1

p1 <- ggplot(dataset) + geom_point(aes(LD2,K1, colour = Response2, shape = Response2), size = 2.5)
+
  labs(x = paste("LD1(", percent(proplda[1]), ")", sep=""),
       y = paste("LD2(", percent(proplda[1]), ")", sep=""))
p1
```

```
Coefficients of linear discriminants:
                    LD1
LIMIT_BAL -4.928587e-07
SEX       -6.585150e-02
EDUCATION -8.032433e-02
MARRIAGE  -2.488607e-01
AGE        1.581510e-02
PAY_0      6.701662e-01
PAY_2      1.601329e-01
PAY_3      1.733721e-01
PAY_4     -5.809809e-02
PAY_5      1.246299e-01
PAY_6     -8.132672e-02
BILL_AMT1 -3.876569e-06
BILL_AMT2 -6.001636e-07
BILL_AMT3 -3.147193e-06
BILL_AMT4  5.126440e-06
BILL_AMT5 -2.748228e-06
BILL_AMT6  2.587035e-06
PAY_AMT1  -4.613871e-06
PAY_AMT2  -3.912667e-06
PAY_AMT3  -5.001320e-06
PAY_AMT4   1.251129e-06
PAY_AMT5  -6.097643e-06
PAY_AMT6  -4.704967e-07
```

## 3 K-NN Classifier

```
set.seed (1)
KNum=c(1:100)
MSE3=rep(0,100)
RightRate3=rep(0,100)
for (i in 1:100) {
  knn.pred=knn(train.X,test.X,Response1, k=i)
  table(knn.pred,Response2)
  RightRate3[i]=mean(knn.pred==Response2)
  Fitted=as.numeric(knn.pred)
  Fitted=Fitted-1
  MSE3[i]=sum((Unpay.Test[,24]-Fitted)^2)/length(Unpay.Test[,24])
}

plot(RightRate3, type="p")
plot(MSE3, type='p')
MAXPoint=which.max(RightRate3)
MINPoint=which.min(MSE3)
knn.pred=knn(train.X,test.X,Response1, k=7)
summary(knn.pred)

table(knn.pred,Response2)
sum(diag(prop.table(ct)))
mean(knn.pred==Response2)
```

```
> table(knn.pred,Response2)
        Response2
knn.pred    0    1
       0 3636  930
       1  281  153
> ct=table(knn.pred,Response2)
> sum(diag(prop.table(ct)))
[1] 0.7578
```

### 4 Support Vector Machine (SVM) Classification

```
tune.out1=tune(svm,Response1~.,data=TrainD,kernel="linear",
         ranges=list(cost=c(0.001, 0.01,0.1,1,5,10,50,100)), decision.values=T)
summary(tune.out1)

tune.out1=tune(svm,Response1~.,data=TrainD,kernel="linear",
         ranges=list(cost=c(0.1,1,4,5,5.5,6)), decision.values=T)
summary(tune.out1)

##choose the best cost
bestmod=tune.out1$best.model
summary(bestmod)

##turn to do predict for testing dataset
ypred=predict(bestmod, TestD,decision.values=TRUE)
SVCT=table(predict=ypred, truth=TestD$Response2)
SVCT
sum(diag(prop.table(SVCT)))
```

```
> ypred=predict(bestmod, TestD)
> table(predict=ypred, truth=TestD$Response2)
       truth
predict    0    1
      0 3810  852
      1  107  231
> SVCT=table(predict=ypred, truth=TestD$Response2)
> sum(diag(prop.table(SVCT)))
[1] 0.8082
```

### 5 Classification Tree

```
sex = as.factor(UCI_train[,2])
marriage = as.factor(UCI_train[,4])
def = ifelse(UCI_train$default.payment.next.month==1, "Yes", "No")

sex2 = as.factor(UCI_test[,2])
marriage2 = as.factor(UCI_test[,4])
def2 = ifelse(UCI_test$default.payment.next.month==1, "Yes", "No")
```

```
UCI_tr = data.frame(UCI_train[,-2], sex, marriage, def)
UCI_tr <- UCI_tr[c(-3)]
UCI_tr <- UCI_tr[c(-22)]

UCI_te = data.frame(UCI_test[,-2], sex2, marriage2, def2)
UCI_te <- UCI_te[c(-3)]
UCI_te <- UCI_te[c(-22)]

tree = tree(def ~ ., data = UCI_tr)
summary(tree)

# Fitting and Pruning
plot(tree, type = "uniform")
text(tree, all=T)
opt = cv.tree(tree, FUN = prune.misclass)
opt
optimal = opt$size[opt$dev == min(opt$dev)]
par(mfrow =c(1,2))
plot(opt$size, opt$dev, pch=16 , cex=0.9, col="red", type="b")
plot(opt$k, opt$dev, pch=16 , cex=0.9, col="red", type="b")

prune = prune.misclass(tree, best = optimal)
summary(prune)
pred = predict(prune, UCI_te, type = "class")
summary(pred)

plot(prune , type = "uniform")
text(prune, all=T)
# Classification Table
table = table(pred, UCI_te$def2)
table
# Accuracy Rate
ar = sum(diag(prop.table(table)))
ar
# MSE
predict = predict(prune, UCI_test)
mse_class <- mean((predict - UCI_test$default.payment.next.month)^2)
mse_class # 0.35886678
# Plot pruned tree
par(mfrow =c(1,1))
plot(prune)
text(prune, pretty = 0)
```

## 6 Ridge Regression

```
library(glmnet)
train_matrix <- model.matrix(default.payment.next.month ~ ., data = UCI_train)
test_matrix <- model.matrix(default.payment.next.month ~ ., data = UCI_test)
```

```
grid <- 10 ^ seq(4, -2, length = 100)

ridge <- glmnet(train_matrix, UCI_train$default.payment.next.month, alpha = 0, lambda = grid, thresh = 1e-12)
cv_ridge <- cv.glmnet(train_matrix, UCI_train$default.payment.next.month, alpha = 0, lambda = grid, thresh = 1e-12)
cv_ridge

best_lambda <- cv_ridge$lambda.min
best_lambda

# Fit model using best lambda
predict_ridge <- predict(ridge, s = best_lambda, newx = test_matrix)
mse_ridge <- mean((predict_ridge - UCI_test$default.payment.next.month)^2)
mse_ridge # 0.1512249
```

### 7 Principal Component Regression

```
library(pls)
pcr <- pcr(default.payment.next.month ~ ., data = UCI_train, scale = TRUE, validation = "CV")
validationplot(pcr, val.type = "MSEP")

predict_pcr <- predict(pcr, UCI_test, ncomp = 15)
mse_pcr <- mean((predict_pcr - UCI_test$default.payment.next.month)^2)
mse_pcr # 0.151333
```

### 8 Partial Least Squares

```
pls <- plsr(default.payment.next.month ~ ., data = UCI_train, scale = TRUE, validation = "CV")
validationplot(pls, val.type = "MSEP")

predict_pls <- predict(pls, UCI_test, ncomp = 4)
mse_pls <- mean((predict_pls - UCI_test$default.payment.next.month)^2)
mse_pls # 0.1512132
```

### 9 Correlation Heat Map

```
dc<-cor(UCI_train[,1:ncol(UCI_train)])
dc<-round(as.matrix(dc),2)
dc
melted_cormat <- melt(dc)
head(melted_cormat)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}
# Get upper triangle of the correlation matrix
```

```
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}
upper_tri <- get_upper_tri(dc)

melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Heatmap

ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                      midpoint = 0, limit = c(-1,1), space = "Lab",
                      name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
##Reorder the correlation matrix
reorder_cormat <- function(cormat){
  # Use correlation between variables as distance
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}
cormat <- reorder_cormat(dc)
upper_tri <- get_upper_tri(dc)
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                      midpoint = 0, limit = c(-1,1), space = "Lab",
                      name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
# Print the heatmap
print(ggheatmap)

ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
```

```
  panel.border = element_blank(),
  panel.background = element_blank(),
  axis.ticks = element_blank(),
  legend.justification = c(1, 0),
  legend.position = c(0.6, 0.7),
  legend.direction = "horizontal")+
guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                             title.position = "top", title.hjust = 0.5))
```