STA 208 Final Project

Wei-Chih Chen, Chi-Lu Chiu, Tianxiao Zhang

June 10, 2015

Introduction

Neighbor.dat is a labeled data. Therefore, we focus on supervised analysis and try some possible unsupervised analysis. Secondly, there are only 55 observations compared with 92 features in data. So we try to use CV for any possible model to train and test data within those 55 observations. Third, although there are 92 features, we do not know whether all of them are necessary for fitting model. For eliminating unnecessary features, we focus more on shrinkage methods. To make sure this is the best model for this data, we try almost every model we have taught in class to find minimum in-sample misclassification rate.

For exploratory data analysis, first we use **nearZeroVar** function in package **caret** to filter predictors that have a single unique value, which we refer to as zero variance predictors. Fortunately, there is no near-zero variance predictor in the data set. Then we deal with between-predictor correlations that may increase inaccuracy in training models later. We show a correlation matrix of the training set which don't include NA values. Each pairwise correlation is computed from the training data and colored according to its magnitude. In Figure 3, the predictor variables have been grouped using a clustering technique so that collinear groups of predictors are adjacent to one another. Looking along the diagonal, there are blocks of strong positive correlations that indicate "clusters" of collinearity. Therefore we filter out the predictors that have high correlations, and create the new data set in which all pairwise correlations are less than 0.75.

Unsupervised Analysis

This data set has label for classifying, so unsupervised analysis are not suitable for this project. However, in this section, we still try four different unsupervised analysis to find the structure of this data set and the possible model to fit data. **kMeans**, **four different types of hierarchy clusters**, **MDS** and **EM algorithm** are the four methods we used in this project. We selected the suitable model by choosing the model with lowest misclassification rate on in-sample data which are 55 observation with crime label value. We will introduce how to apply data on these four method and their results in the following sections.

kMeans

The data set neighbor.dat is a label data. Before fitting data to kmeans, we should select train data which not include data without crime value and label out the crime column. After preprocessing neighbor.dat, we fit data to kmeans. Since there are three level for crime, we choose **k=3** with **nstart = 10**. Besides, we need to fit cluster labels as same as original crime label. However, the best in-sample misclassification rate result for kMenas is **0.473**. This results is shown that no matter how hard do we try to cluster features by themselves, the result becomes meaningless without considering label. Table 1 is the table for predict and truth. Rest of the unsupervised analysis use the same table, so we will just show this one for instance.

Table 1: Truth and Predict

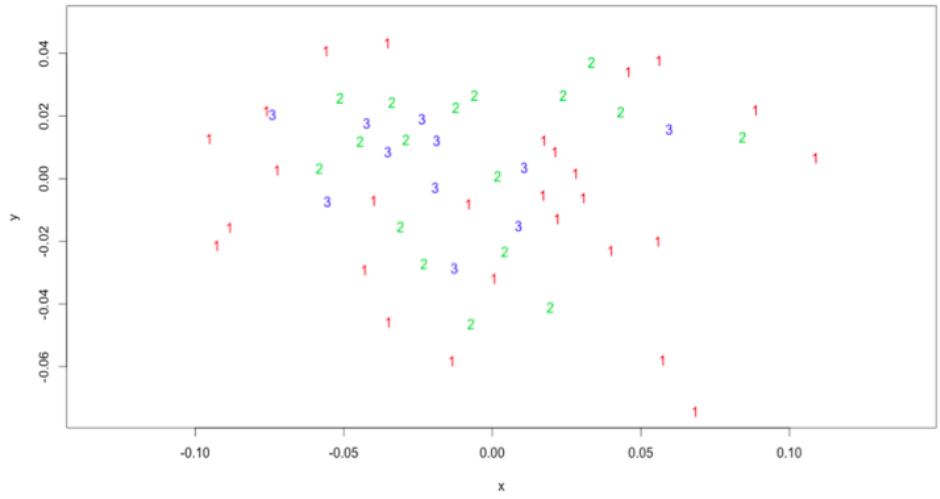
		truth		
		1	2	3
	1	16	4	4
predict	2	9	13	7
	3	2	0	0

Hierarchy clusters

We use **single linkage**, **complete linkage**, **average linkage** and **centroid linkage** for linkage to fit data. Since clustering need pairwise dissimilarities between data points, we use **pair.dist** data and find its distance between points to fit. After fitting for different linkages, we cut tree with **k=3**. And the in-sample misclassification rate of these four linkage are following **0.4583**, **0.5417**, **0.5417**, and **0.5**. The results are also not preferable because we use unsupervised analysis to fit supervised data.

MDS

Since we have distance data between cities, we try to see whether there are some relationship between distance. This distance image do not provide us some useful information about how label group together.



EM algorithm

This is another algorithm try to fit. For fitting EM algorithm, we use **Mclust** function and select parameter **G =3** which is the number of groups. EM algorithm gets a better results from previous unsupervised analysis. The in-sample misclassification rate result is **0.418**. But it is still too high for us to analysis.

Conclusion for Unsupervised Analysis

The results for unsupervised analysis are not good and we couldn't find out any specific structure. The best model for all of these unsupervised analysis is **EM algorithm**. But the in-sample misclassification rate is as high as **0.418**. Therefore, we would like to try using supervised analysis.

Supervised Analysis

The goal of this project is to make predictions for the missing crime scores rather than discovering the structure, so this is why we prefer supervised learning. We tried 10 different approaches, logistic regression, linear discriminant analysis, k-nearest neighborhood, classification tree, bagging, random forest, boosting, svm, and the shrinkage methods, lasso and ridge regression, and select the most suitable approach by checking the in-sample misclassification rate, and would use the approach we chose to do further discussion.

Logistic Regression

Since the score of neighborhood has three levels, we try to use penalized multinomial regression with more than two classes. Here we use **train** in package **caret** to train the model with 10-fold cross-validation, with method **multinom**. The final value used for the model is decay = 0.1 with in-sample misclassification rate **0.34**. However if we use the pre-processed data that is without the highly correlated predictors to train the model again, the in-sample misclassfication rate will be **0.485** which is worse. Our estimation is that high correlated variables may convey important information in the model.

Linear Discriminant Analysis (LDA)

By using LDA to predict the crime scores for the 55 neighborhoods which do not have missing crime scores, cross-validation was used. We tried both scaling and not scaling the data set and found out that not scaling the data set gave us lower misclassification rate. The misclassification rate is about 0.018, which means that only one crime score in the 55 neighborhood is incorrect. Although the misclassification rate is low, in LDA we used all of the 92 measurement, since we cannot do model selection in LDA, so there may be an overfitting issue. Therefore, we would like to try other methods and see whether there is any approach that can have similar misclassification rate without overfitting issue.

K-Nearest Neighborhood (K-NN)

By using K-NN to predict the crime scores for the 55 neighborhoods which do not have missing crime scores, cross validation was used. To decide which k should we use, we tried k = 1 to k = 54 and find out which one give us the smallest misclassification rate. We repeated this procedure for 100 times, since cross validation will gave us different smallest misclassification rate every time, and then we get the mean smallest misclassification rate which is about **0.435**.

Classification Tree

We firstly adopt the CART algorithm in package **tree** using cross-validation and **prune.misclass**. By CV, we get that the best size is 3 which yields the smallest deviation. Then we use size of 3 to prune the tree. The in-sample misclassification rate is 0.255 which may be overfitting since we use the whole dataset. Then we use **train** in package **caret** to train the tree model with 10-fold cross-validation, with method **rpart**. The final value used for the model was complexity parameter = 0.143, with an accuracy of 0.525. It indicates that the simple tree model is not satisfactory.

Bagging

We use **bagging.cv** in package **adabag** to do cross-validation for bagging. The estimated error is **0.364**. After doing importance plot, we could find the top 10 most important predictors shown as Figure 1.

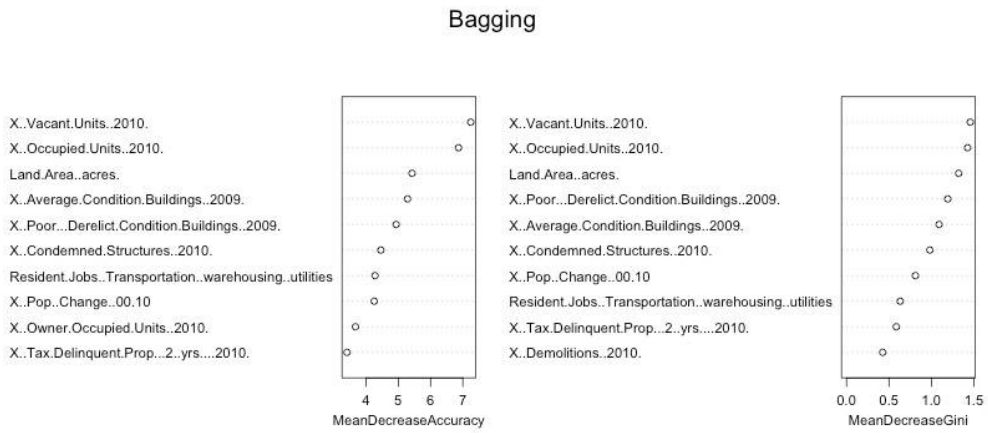


Figure 1: Bagging – 10 Most Important Predictors

Random Forest

We call function **randomForest** in package **randomForest**. By setting do.trace = 100, we examine the OOB error for different number of trees to grow from 100 to 500. The final OOB estimate of error rate is 0.291 which is quite satisfactory. Like bagging, we check the 10 most important predictors that have effect on RSS and Gini index, which is shown as Figure 2.

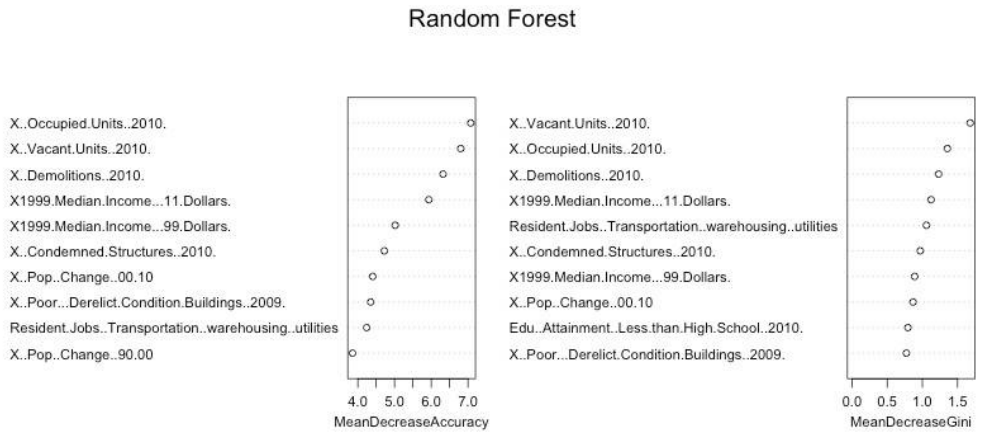


Figure 2: Random Forest – 10 Most Important Predictors

Boosting

During boosting procedure, we used 10-fold cross-validation and we tried seven different total number of trees to fit, 100, 200, 500, 1000, 1500, 2000 and 2500, to predict the crime scores for the 55 neighborhoods which do not have missing crime scores. We found that the number of variables used will increase from 31 to 79 while the number of trees to fit increase. The mean misclassification rate we get after fitting 100, 200, 500, 1000, 1500, 2000 and 2500 trees for 100 times is about 0.637, 0.637, 0.630, 0.606, 0.587, 0.580 and 0.580, respectively, which will decrease while the number of trees to fit increase. We also found that both number of variables used and misclassification rate will not change much when the number of trees to fit is larger than 2000. Therefore, we would conclude that when total number of trees to fit is 2000, we have the lowest misclassification rate 0.580.

SVM

For SVM, we build the model by tuning different parameters and using 10-fold cross-validation. Firstly we choose linear kernel and tune the cost among 0.001, 0.01, 0.1, 1, 5, 10, 100. The best parameter is that cost = 0.01, and the best in-sample classification rate is 0.313. Then we choose radial kernel and tune the cost among 0.1, 1, 10, 100, 1000, and tune the gamma among 0.5, 1, 2, 3, 4. The 10-fold CV gives us the best parameters of cost = 0.1, gamma = 0.5, and best performance = 0.513. The reason why the performance of radial kernel is poor is that the original data doesn't show centralized clustering structures, which is not suitable for radial kernel classifier.

Shrinkage Methods – Lasso and Ridge Regression

By using the shrinkage methods, Lasso and Ridge Regression, we use 10-fold cross-validation to pick out the smallest tuning parameter, λ, and use this λ to fit regression models for the 55 neighborhoods which do not have missing crime scores. Since there are 3 classes in crime score, we used multinomial family while fitting the models. We repeated this procedure for 100 times and calculate some summary statistics misclassification rate for each approach, which is shown as Table 2.

Table 2: Summary Statistics of the Misclassification Rate for the Shrinkage Methods

	Lasso	Ridge
Minimum	0	0.2364
Median	0.0546	0.2545
Mean	0.0758	0.2464
Maximum	0.2909	0.2545
Standard Error	0.0773	0.0091

According to Table 2, Ridge has relatively high misclassification rate. Therefore, we decided to view Lasso as the best approach in the shrinkage methods. The mean misclassification rate for Lasso is about 0.0758.

Conclusion for Supervised Analysis

For all the supervised analysis approach, LDA has the smallest misclassification rate, and Lasso also has small misclassification rate. Since LDA has overfitting issue because it use all the variables while we use much lesser variables by using shrinkage methods, we prefer using Lasso to do our prediction for the missing values.

Discussion

After using several approaches in unsupervised analysis and supervised learning, we decided to use Lasso to predict the missing crime scores. Since we repeated the lasso procedure for 100 times, we also generated 100 predictions for the missing crime scores at the same time. We examine the predictions for the missing crime scores for each repetition, and found out that when the misclassification rate for the 55 neighborhoods which do not have missing crime scores is smaller than the mean misclassification we will have the same set

of predictions for the missing crime scores. Therefore, we randomly choose one repetition from the ones with in-sample misclassification rate equals to 0 to continue our discussion.

By the Lasso we used, the tuning parameter λ chosen by 10-fold cross-validation is about 0.036. Furthermore, in Lasso, number of variables used is lesser than LDA, boosting, and other approaches. Since we used multinomial family while fitting the models as we mentioned before, variables used for predicting each class of crime score is different. 16 variables are used to determine whether the neighborhood should be classified as crime score = 1, 9 variables are used to determine whether the neighborhood should be classified as crime score = 2 and 11 variables are used to determine whether the neighborhood should be classified as crime score = 3, which both the variables the their parameter estimate were shown as Table 4, Table 5 and Table 6, respectively. In the following paragraph, we will interpret some important variables that has effects on predicting different class of crime score.

According to Table 4, we can see that housing units in 2010 has significant effects on predicting a neighborhood as crime score equals to 1. For instance, Higher percent of house has people living in it makes the neighborhood has low level of crime, while Higher percent of house without people living does not. Besides, some type of resident jobs and jobs in the neighborhood also have effects on predicting a neighborhood as crime score equals to 1. According to Table 5, the neighborhood with resident having job in transportation, warehousing and utilities and has more resident ride bicycle to work are reasons that the neighborhood is predicted as crime score equals to 2. On the other hand, neighborhood with lots of job for professor, scientific, administration and waste management has less probability to be predicted as crime score equals to 2. According to Table 6, if a neighborhood has relatively high percent of population age $<$ 5, it is more likely to be predicted as crime score equals to 3. Besides, some type of resident jobs and jobs in the neighborhood also have effects on predicting a neighborhood as crime score equals to 3.

Based on all the condition we mentioned in this report, our final prediction for the missing crime scores are shown as Table 3:

Table 3: Prediction for the Missing Crime Scores

Index	1	8	13	17	18	20	23	30	31	42	46	47
Crime Score	1	3	2	2	1	1	1	1	1	2	1	1
Index	49	52	53	54	55	57	60	69	71	72	76	79
Crime Score	1	1	1	2	1	2	1	2	3	1	1	1

Appendix 1 - Figures and Tables

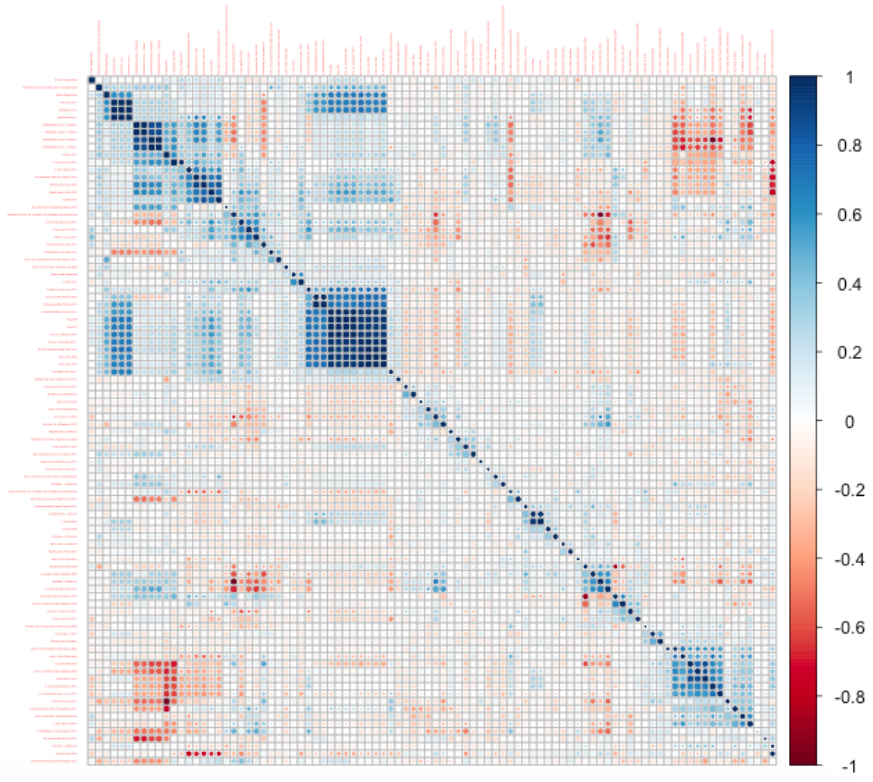


Figure 3: Pairwise Correlation Matrix

Table 4: Prediction for the Missing Crime Score = 1

Variable	Parameter Estimate
Intercept	-6.14
% Pop. Change, 90-00	1.92e-02
% Occupied Units (2010)	1.42e-01
% Vacant Units (2010)	-1.71e-15
% Owner Occupied Units (2010)	2.97e-03
% Renter Occupied Units (2010)	-3.50e-03
% Average Condition Buildings (2009)	8.35e-03
# Residential Bldg. Permits (2010)	-5.70e-03
# Residential Bldg. Permits (2010)	-1.69e-05
Resident Jobs: Retail Trade	3.90e-03
Resident Jobs: Finance, insurance, real estate	-1.27e-01
Resident Jobs: Public admin	2.81e-01
Jobs in 'Hood: Retail trade	-2.70e-02
Jobs in 'Hood: Information	-5.39e-03
Jobs in 'Hood: Finance, insurance, real estate	-5.10e-02
Edu. Attainment: Less than High School (2010)	-3.84e-02
1999 Median Income ('99 Dollars)	7.98e-06

Table 5: Prediction for the Missing Crime Score = 2

Variable	Parameter Estimate
Intercept	1.71
Cemetery (% of land area)	2.52e-02
# Sets of Steps	3.34e-02
Total # Units (2000)	1.77e-05
# Condemned Structures (2010)	3.65e-02
Resident Jobs: Transportation, warehousing, utilities	2.92e-01
Jobs in 'Hood: Prof, scientific, admin, and waste mgmt	-3.60e-02
Edu. Attainment: Less than High School (2010)	6.79e-02
Est. Pop. Under Poverty (2010)	8.49e-05
Commute to Work: Bicycle (2010)	3.34e-01

Table 6: Prediction for the Missing Crime Score = 3

Variable	Parameter Estimate
Intercept	4.426
% Pop. Change, 90-00	-0.006
%Pop. Age < 5 (2010)	8.248
Woodland (% of land area)	-0.016
Street Density (st. mi/area sq. mi)	0.009
# Sets of Steps	-0.006
Resident Jobs: Construction	-0.040
Resident Jobs: Retail Trade	-0.005
Resident Jobs: Prof, scientific, admin, and waste mgmt	0.003
Jobs in 'Hood: Information	0.002
Jobs in 'Hood: Educ, health, and social svc	-0.001
Commute to Work: Taxi (2010)	0.208

Appendix 2 - R Code

```
#### Unsupervised Learning
trainneighbor = neighbor.dat
ind.na = which(is.na(neighbor.dat))
trainneighbor = trainneighbor[ind.na,]
train = data.frame(x=trainneighbor[,-1],y=as.factor(trainneighbor[,1]))
testneighbor = neighbor.dat[-ind.na,]
test = data.frame(x=testneighbor[,-1])

###kMeans### MR = 0.473
kms = kmeans(train, 3, nstart = 10)
table(predict=kms$cluster, truth=trainneighbor[,1])
Mis.kMeans <- 1-mean(kms$cluster==trainneighbor[,1]) ##misclassification rate of kMeans

library(clue)
vector = cl_predict(kms, newdata = test) #kMeans results

##misclassification rate of kMeans = 0.5

###cluster### single clustering best MR = 0.4583
###(a)###
hc.single=hclust(dist(pair.dist),method="single")
cut=cutree(hc.single, 3)
#plot(hc.single,main="Single Linkage",xlab="",sub="",cex=.9)
plot(x, col=cut, pch = (cut + 48))

##misclassification rate of single clustering = 0.45833

###(b)###
hc.complete=hclust(dist(pair.dist),method="complete")
#plot(hc.complete,main="Complete Linkage",xlab="",sub="",cex=.9)
cutb=cutree(hc.complete, 3)
plot(x, col=cutb)

##misclassification rate of complete clustering = 0.54167

###(e)###
hc.average=hclust(dist(pair.dist),method="average")
plot(hc.average,main="Average␣Linkage",xlab="",sub="",cex=.9)
cut.e1=cutree(hc.average, 3)
plot(x, col=cut.e1)

##misclassification rate of average clustering = 0.54167

hc.average2=hclust(dist(pair.dist)^2,method="average")
plot(hc.average2,main="Average␣Linkage␣with␣square␣distance",xlab="",sub="",cex=.9)
cut.e2=cutree(hc.average2, 3)
plot(x, col=cut.e2)

##misclassification rate of average clustering with square distance = 0.54167
## reusult is the same as e1
```

```
###(f)###
hc.centroid=hclust(dist(pair.dist),method="centroid")
plot(hc.centroid,main="Centroid␣Linkage",xlab="",sub="",cex=.9)
cut.f1=cutree(hc.centroid, 3)
plot(x, col=cut.f1)

##misclassification rate of centroid clustering = 0.5

hc.centroid2=hclust(dist(pair.dist)^2,method="centroid")
plot(hc.centroid2,main="Centroid␣Linkage␣with␣square␣distance",xlab="",sub="",cex=.9)
cut.f2=cutree(hc.centroid2, 3)
plot(x, col=cut.f2)

##misclassification rate of centroid clustering with square distance = 0.54167

###EM algorithm ###  MR = 0.418
library(mclust)
mc <- Mclust(train, 3)
table(predict=mc$classification, truth=train$y)
Mis.em <- 1-mean(mc$classification==train$y)
##misclassification rate of EM algorithm = 0.5

###MDS###
loc <- cmdscale(pair.dist)
x <- loc[, 1]
y <- -loc[, 2]
plot(x, y, type = "n", asp=1)
text(x, y, neighbor.dat[,1], cex = 1, col = as.integer(neighbor.dat[,1]+1) )

library(class)
knnpred=knn(pair.dist,x.014.te,train$y,k=1)
table(knnpred,y.014.te)
1-mean(knnpred==y.014.te)


#### Supervised Learning
neighbor.dat = data.frame(neighbor.dat)
na <- which(is.na(neighbor.dat[, 1]))
trainData = neighbor.dat[-na, ]
testData = neighbor.dat[na, ]

trainData.factor = cbind(as.factor(trainData[, 1]), trainData[, -1])
testData.factor = cbind(as.factor(testData[, 1]), testData[, -1])
colnames(trainData.factor)[1] <- "Crime"
colnames(testData.factor)[1] <- "Crime"

### Multinomial Logistic regression
library(nnet)
library(caret)
set.seed(25)
multinom.fit = train(Crime ~., data = trainData.factor, method = "multinom", trControl =
    trainControl(method = "cv", number = 10))
multinom.fit
multinom.filter = train(Crime ~., data = filteredtrainData, method = "multinom", trControl =
```

```
     trainControl(method = "cv", number = 10))
multinom.filter

### Tree
library(tree)
tree.fit = tree(Crime ~., data = trainData.factor)
set.seed(10)
cv.tree.fit = cv.tree(tree.fit, method = "misclass")
plot(cv.tree.fit$size, cv.tree.fit$dev, type = "b") # Best size is 3
prune.fit = prune.misclass(tree.fit, best = 3)
prune.pred = predict(prune.fit, type = "class")
mean(trainData.factor$Crime != prune.pred)

tree.fit = train(Crime ~., data = trainData.factor, method = "rpart", trControl =
    trainControl(method = "cv", number = 10))
tree.fit

### Bagging
library(adabag)
library(randomForest)
set.seed(25)
bag.fit = bagging.cv(Crime ~., data = trainData.factor)
bag.fit
bag.fit2 = randomForest(Crime ~., data = trainData.factor, mtry = 92, importance = TRUE)
varImpPlot(bag.fit2, n.var = 10, cex = .7, main = "Bagging")

### Random Forest
set.seed(5)
rf.fit = randomForest(Crime ~., data = trainData.factor, mtry = sqrt(92), do.trace=100, na.
    action=na.omit, imp = T)
rf.fit
varImpPlot(rf.fit, n.var = 10, cex = .7, main = "Random␣Forest")

### SVM
library(e1071)
set.seed(25)
tune.out = tune(svm, Crime ~., data = trainData.factor, kernel = "linear", ranges = list(
    cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)), tunecontrol = tune.control(sampling = "cross
    ", cross = 10, best.model = TRUE))
tune.out
bestmod = tune.out$best.model
bestmod

tune.out2 = tune(svm, Crime ~., data = trainData.factor, kernel = "radial", ranges = list(
    cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)), tunecontrol = tune.control
    (sampling = "cross", cross = 10, best.model = TRUE))
tune.out2
bestmod2 = tune.out2$best.model
bestmod2

####
data.clean = neighbor.dat
View(data.clean)
ind.na = which(is.na(neighbor.dat))
```

```
data.na = neighbor.dat[ind.na,]
data.full = neighbor.dat[-ind.na,]

### LDA
lda.fit.scale = lda(scale(data.full[, -1]), data.full[, 1], cv = TRUE)
class.pred.scale = predict(lda.fit.scale, data.full[, -1])$class
class.pred.scale
mis.rate.scale = table(class.pred.scale == data.full[, 1])[1]/length(data.full[, 1])
mis.rate.scale
lda.fit = lda(data.full[, -1], data.full[, 1], cv = TRUE)
class.pred = predict(lda.fit, data.full[, -1])$class
class.pred
mis.rate = table(class.pred == data.full[, 1])[1]/length(data.full[, 1])
mis.rate
class.pred.na = predict(lda.fit, data.na[, -1])$class
class.pred.na

### K-NN
k.nn = function(data, class){
  knn.pred = matrix(, nrow = 54, ncol = 55)
  knn.mis = matrix(, nrow = 54, ncol = 1)
  for(i in 1:54){
    knn.pred[i, ] = knn.cv(data, class, i)
    knn.mis[i] = table(knn.pred[i, ] == class)[1]/length(class)
  }
  min.mis = which(knn.mis == min(knn.mis))
  knn.mis[min.mis[1], ]
}

knn.miss = rep(NA, nrow = 100)
for(i in 1:100){
  knn.miss[i] = k.nn(data.full[, -1], data.full[, 1])
}
mean(knn.miss)

### Boost
boosting = function(tree){
  boost = gbm(factor(Crime)~., distribution = "multinomial", data = data.frame(data.full),
              verbose = F, n.trees = tree)
  # View(summary(boost))
  pred.boost.full = predict(boost, newdata = data.frame(data.full), n.trees = 2500)
  crime.boost.full = rep(NA, nrow(data.full))
  for (i in 1:nrow(data.full)) {
    crime.boost.full[i] = which.max(pred.boost.full[(3*i-2):(3*i)])
  }
  crime.boost.full
  length(which(crime.boost.full != data.full[,1]))/length(crime.boost.full)
}
t = c(100, 200, 500, 1000, 1500, 2000, 2500)

boost.mis = matrix(, nrow = 100, ncol = 7)
for(i in 1:100){
  boost.mis[i, ] = sapply(t, function(t)boosting(t))
}
```

```
boost.miss = rep(NA, 7)
for(i in 1:7){
  boost.miss[i] = mean(boost.mis[, i])
}
boost.miss
boost = gbm(factor(Crime)~., distribution = "multinomial", data = data.frame(data.full),
            verbose = F, n.trees = 2000, cv.folds = 10)
summary(boost)

pred.boost.na = predict(boost, newdata = data.frame(data.na), n.trees = 1000)
crime.boost.na = rep(NA, nrow(data.na))
for (i in 1:nrow(data.na)) {
  crime.boost.na[i] = which.max(pred.boost.na[(3*i-2):(3*i)])
}
crime.boost.na


### Shinkage Method
## Ridge
ridge.mis = rep(NA, 100)
for(j in 1:100){
  cv.glm.dat.r = cv.glmnet(data.full[,-1], data.full[,1], alpha = 0)
  lambda.min.r = cv.glm.dat.r$lambda.min
  glm.dat.r = glmnet(data.full[,-1], data.full[,1], family = "multinomial",
                     lambda = lambda.min.r, alpha = 0)
  pred.glm.full.r = predict(glm.dat.r, newx = data.full[,-1], s = lambda.min.r,
                            type = "response")
  crime.glm.full.r = rep(NA, nrow(data.full))
  for ( i in 1:nrow(data.full)){
    crime.glm.full.r[i] = which.max(pred.glm.full.r[i,,1])
  }
  ridge.mis[j] = length(which(crime.glm.full.r != data.full[,1]))/length(data.full[, 1])
}
summary(ridge.mis)
sd(ridge.mis)
# coef
coef.r = predict(glm.dat.r, newx = data.full[,-1],
                 s = lambda.min.r, type = "nonzero")
coef.r

## Lasso
ela.mis = rep(NA, 100)
for(j in 1:100){
  cv.glm.dat.l = cv.glmnet(data.full[,-1], data.full[,1])
  lambda.min.l = cv.glm.dat.l$lambda.min
  glm.dat.l = glmnet(data.full[,-1], data.full[,1], family = "multinomial",
                     lambda = lambda.min.l)
  pred.glm.full.l = predict(glm.dat.l, newx = data.full[,-1], s = lambda.min.l,
                            type = "response")
  crime.glm.full.l = rep(NA, nrow(data.full))
  for ( i in 1:nrow(data.full)){
    crime.glm.full.l[i] = which.max(pred.glm.full.l[i,,1])
  }
  ela.mis[j] = length(which(crime.glm.full.l != data.full[,1]))/length(data.full[, 1])
```

```
}
summary(ela.mis)
sd(ela.mis)

ela.mis = rep(NA, 100)
crime.full.pred = matrix(, nrow = 100, ncol = nrow(data.full))
crime.na.pred = matrix(, nrow = 100, ncol = nrow(data.na))
lambda = rep(NA, 100)
coef.all = vector("list", 100)
for(j in 1:100){
  cv.glm.dat.l = cv.glmnet(data.full[,-1], data.full[,1])
  lambda.min.l = cv.glm.dat.l$lambda.min
  lambda[j] = lambda.min.l
  glm.dat.l = glmnet(data.full[,-1], data.full[,1], family = "multinomial",
                     lambda = lambda.min.l)
  coef.all[[j]] = coef(glm.dat.l, s = lambda.min.l)
  pred.glm.full.l = predict(glm.dat.l, newx = data.full[,-1], s = lambda.min.l,
                            type = "response")
  crime.glm.full.l = rep(NA, nrow(data.full))
  for ( i in 1:nrow(data.full)){
    crime.glm.full.l[i] = which.max(pred.glm.full.l[i,,1])
  }
  pred.glm.na.l = predict(glm.dat.l, newx = data.na[,-1],
                          s = lambda.min.l, type = "response")
  crime.full.pred[j, ] = crime.glm.full.l
  ela.mis[j] = length(which(crime.glm.full.l != data.full[,1]))/length(data.full[, 1])
  crime.glm.na.l = rep(NA, nrow(data.na))
  for (i in 1:nrow(data.na)) {
    crime.glm.na.l[i] = which.max(pred.glm.na.l[i,,1])
  }
  crime.na.pred[j, ] = crime.glm.na.l
}
View(crime.na.pred)
crime.full.pred[no.mis, ]
table(ela.mis)
no.mis = which(ela.mis == 0)
sam = sample(no.mis, 1)
table(lambda[no.mis])
lambda[sam]
crime.na.pred[sam, ]

coef.all[[sam]][1]
coef.all[[sam]][2]
coef.all[[99]][3]
coef.all[[no.mis[1]]][1]

# coef
coef.l = predict(glm.dat.l, newx = data.full[,-1],
                 s = lambda.min.l, type = "nonzero")
unlist(coef.l[1])
coef(glm.dat.l)
```