# Project Report
# Landform Classification Based on Cloudsourced Data
# ERG 2050

December 24, 2017


| | | |
|---|---|---|
| Partners: | Jimi He | 116010067 |
| | Xizhe Zhang | 116010299 |
| | Zhucheng Zhan | 116010282 |
| | Zhiwei Huang | 116010093 |
| Instructor: | Professor Chen | |


## 1 Introduction

We choose the third one as our project. In this project, more than 10,000 observations are given in the training dataset. Each of them have 28 predictors, among which 27 predictors are NVDI (normalized difference vegetation index) values extracted from satellite images acquired between January 2014 and July 2015, in reverse chronological order and the remaining one is the maximum NDVI value. Each of them belong to one of six classes(impervious, farm, forest, grass, orchard, water). The object is to make a prediction and classify the training datasets classes.

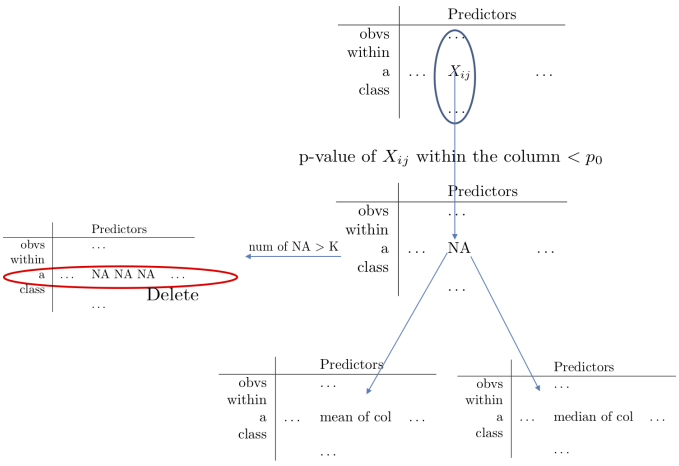
### 1.1 Methodology

Firstly, we need to process the data, to remove observations whose data are not complete and to denoise the data, because the data contain noise due to cloud cover. And there are three ways we use to denoise it: replacement by mean, replacement by median and the box plot method. We will explain it later. After that, we will have three different training datasets.
Secondly, we do the model selection and dataset selection. There are five models that we use: KNN, LDA, QDA, Logistic Regression and Pruned Tree. We fit each model to the three different datasets and use ten-fold cross validation method to calculate the error rate. And choose the lowest one as the model we choose.

# 2 Data Denoising

## 2.1 Chi-square Method

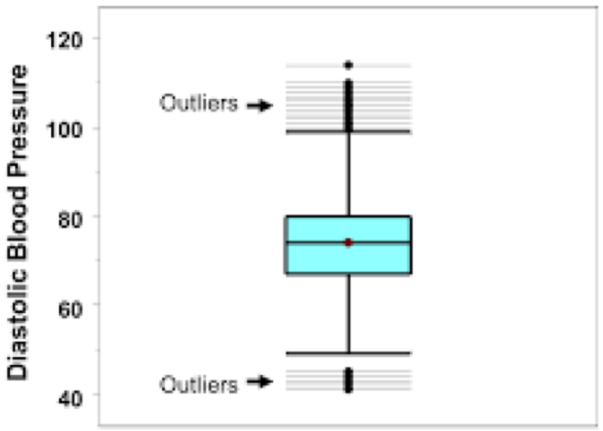
Since the main problem for this question is that there are noises in the dataset. So the first thing we did is to denoise the data. To quantify the criteria, we have developed two main approaches to solve this problem.
The first approach is based on the chi-square test. The main idea for this approach is that we first classify the data by its class. After classifying, we find the outliers by columns. With each loop, we find the highest and lowest value in one column, and then compute the p-value under chi-square distribution. If the p-value is too small, we will identify it as an outlier and change it to NA, and then keep doing the steps until the p-value of the highest and lowest value is greater than some certain value. So thats basically how we identify the outlier in each predictor.
Since there are also some noises in the response. To find the noise in the response, we set another parameter, K, that is, if in each observation, there are no less than K outliers are identified, then we will consider it as a noise in the response.
And the last step is to deal with those outliers. For the outliers in the predictors, considered that there is one special predictor, which is the MAX NDVI of each observation, so we first replace it by the maximum of the observation. For other outliers in the predictor, we have two solutions, the first is to replace the outliers by the sample mean of each column, the second is by the median. And for each outlier in the response, because to identify the outlier is too hard, we will simply delete the observation.




## 2.2 Boxplot Method

To solve this problem, we also developed another simpler approach. The approach is based on the boxplot function. In this data denoising method, the core idea is to remove those observations which its values are much larger than 3 quarters or much smaller than 1 quarters of its class box plot. So, there is only one simple parameter in this approach, which is to measure how far away the

extreme values is from the main box plot. The two advantages for this approach is that the method contain only one parameter which is simple, and the result can be seen straightforward from the plot, so it has greater interpretability. The second is that it does not depend on the normal assumption, so it is more general.




# 3 Model Selection

## 3.1 Model Candidates

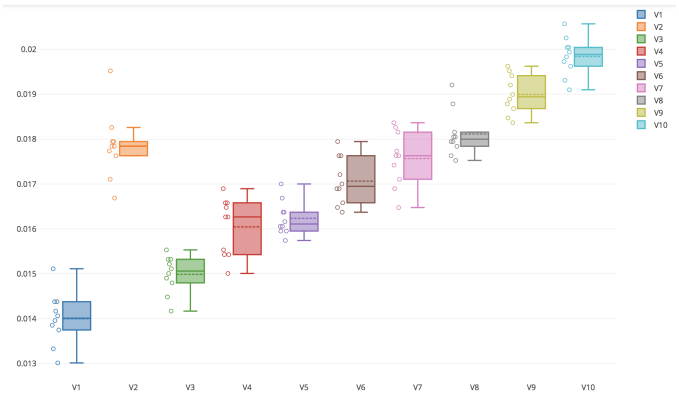
After denoising the data, we selected five candidates according to what we learnt: KNN, Logistic Regression, QDA, LDA and Tree. Aiming at getting a more accurate error rate, we decide to use k-folds cross validation to test each model so that we can finally find which one is the best.

## 3.2 K-folds cross validation

Based on the bias-variance tradeoff theory, we consider K=10 is a proper number so we choose to use 10-folds to test the model and find out which one is the best. To reduce randomness, for each model, we randomly ran the 10-fold cross validation for ten times, and then obtained the error set.

## 3.3 Select the best k in KNN

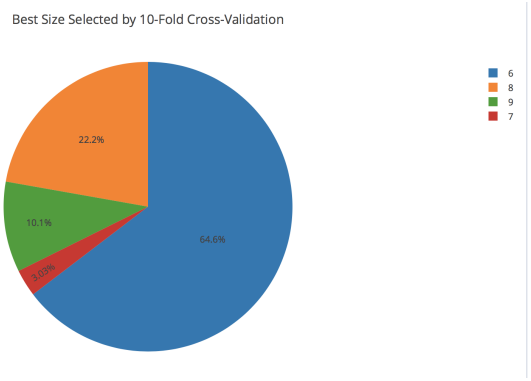
For KNN, we need to choose the best k to fit the data set. After using the 10-folds cross validation, we find out that KNN performs best when k=1 by seeing the distribution of the test error. So we decide to choose k=1 when using KNN to fit the dataset. Finally we find out KNN has the smallest error rate when k=1 and use the dataset denoised by boxplot.



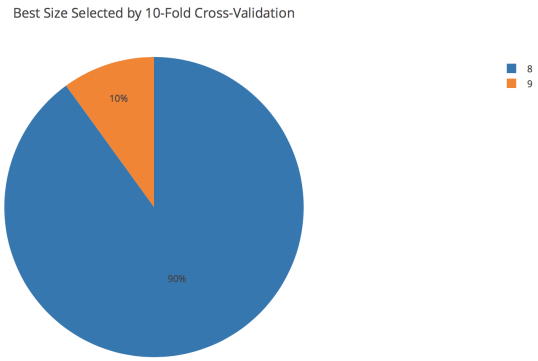

The x-axis represents the k in test, while the y-axis represents the 10-fold cross validation error rate.

## 3.4 Select the best size for Tree

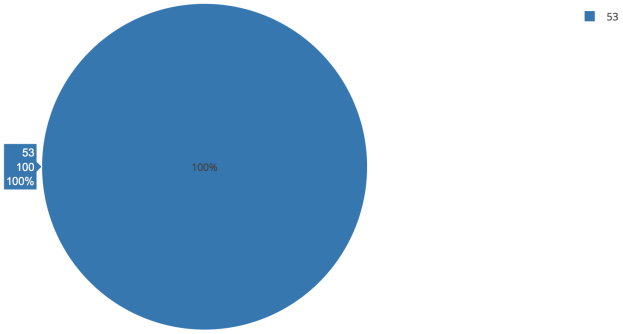
Based on 3 denoised datasets, we test each dataset 100 times to find out the best sizes of trees. After drawing the pie chart, we find:




For data set denoising by replacing outliers by median, we selects size = 6.



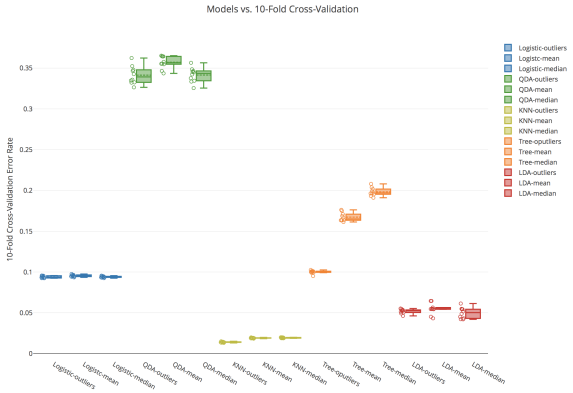

For data set denoising by replacing outliers by mean, we selects size = 8.




For data set denoising by boxplot, we selects size = 53.

### 3.5 Test five classification models by three denoised datasets

By running 10-folds 10 times on each model, we finally plot the error sets of all possible models in one picture. As the picture shows, it is obvious that KNN with k=1 based on the dataset obtained by boxplot outperforms other models.




# 4 Summary and Conclusions

Since we come to the conclusion that KNN with K=1 based on the dataset obtained by boxplot outperforms other models, we fit the KNN to the test dataset and find that the error rate is 61.33%, which is not so satisfying. After considering about the five classification models, we find the result meets our expectation:

a. KNN usually preforms well if the dataset is large it is hard to make some assumption for the predictors.

b. By intuition, the relationship between the predictors and the response is highly nonlinear and complicated, so K=1 which is the most flexible one is the best.

c. Since the chi-square test is based on the normal assumption, the boxplot method outperforming the chi-square method also indicates that the KNN method is better.

According to the test error rate, we think that although the boxplot method has greater interpretability, the parameter selection is still subjective and this makes the test error rate not so satisfying.

# 5 Code Appendix

## 5.1 Chi-square Method

```
#Chi-square Method

library(outliers)
read.csv("(dataset␣path)",header=T)
data.all=read.csv("training.csv", head = T)

#classify the dataset by the classes

forest.index = which(data.all$class=='forest')
forest.data = data.all[forest.index,]

farm.index = which(data.all$class=='farm')
farm.data = data.all[farm.index,]

grass.index = which(data.all$class=='grass')
grass.data = data.all[grass.index,]

impervious.index = which(data.all$class=='impervious')
impervious.data = data.all[impervious.index,]

orchard.index = which(data.all$class=='orchard')
orchard.data = data.all[orchard.index,]

water.index = which(data.all$class=='water')
water.data = data.all[water.index,]

#Change the type of the data from character to numeric.

for (i in 1:28){
  for (k in 1:nrow(forest.data)){
    forest.data[k,i+1] = as.numeric(forest.data[k,i+1])
  }
}

for (i in 1:28){
  for (k in 1:nrow(farm.data)){
    farm.data[k,i+1] = as.numeric(farm.data[k,i+1])
  }
}

for (i in 1:28){
  for (k in 1:nrow(grass.data)){
    grass.data[k,i+1] = as.numeric(grass.data[k,i+1])
  }
}
```

```
for (i in 1:28){
  for (k in 1:nrow(impervious.data)){
    impervious.data[k,i+1] = as.numeric(impervious.data[k,i+1])
  }
}

for (i in 1:28){
  for (k in 1:nrow(orchard.data)){
    orchard.data[k,i+1] = as.numeric(orchard.data[k,i+1])
  }
}

for (i in 1:28){
  for (k in 1:nrow(water.data)){
    water.data[k,i+1] = as.numeric(water.data[k,i+1])
  }
}

data.classified=rbind(farm.data,forest.data,
grass.data,impervious.data,orchard.data,water.data)

#chi-suqare test to find the outliers:
#forest  with p.value=0.005 K=3
for (m in 1:28){
  for (n in 1:nrow(forest.data)){
    chisq.out.test(forest.data[,m+1])
    x=chisq.out.test(forest.data[,m+1])
    if (x$p.value<0.005)
      forest.data[which.min(forest.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(forest.data)){
    chisq.out.test(forest.data[,m+1],opposite = TRUE)
    x=chisq.out.test(forest.data[,m+1],opposite = TRUE)
    if (x$p.value<0.005)
      forest.data[which.max(forest.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.forest=sum(is.na(forest.data)) #the number of outliers in forest
#test if there is any outlier class
for (i in 1:nrow(forest.data)){
  if (sum(is.na(forest.data[i,]))>2)
    forest.data[i,1]=NA
  else next
}
number.outliers.class.forest=sum(is.na(forest.data[,1])) # the number of outliers o
#delete the wrong class
index.outlier.forest=which(is.na(forest.data$class))
forest.data.prune=forest.data[-index.outlier.forest,] #the pruned data
forest.prune.left=sum(is.na(forest.data.prune))
#convert the outlier to the normal values
#converted by mean
```

```
for (m in 1:27){
  for (n in 1:nrow(forest.data.prune)){
    if (is.na(forest.data.prune[n,m+2]))
      forest.data.prune[n,m+2]=mean(forest.data.prune[,m+2],na.rm= T)
    else next
  }
}

#converted by median
#for (m in 1:28){
  #for (n in 1:nrow(forest.data.prune)){
    #if (is.na(forest.data.prune[n,m+1]==TRUE)
        #forest.data.prune[n,m+1]==median(forest.data.prune[,m+1],na.rm = FALSE)
        #else next
  #}
#}

for (n in 1:nrow(forest.data.prune)){
  if (is.na(forest.data.prune[n,2]))
    forest.data.prune[n,2]=max(forest.data.prune[n,-1],na.rm= T)
  else next
}

#farm p.value=0.0005 K=4
for (m in 1:28){
  for (n in 1:nrow(farm.data)){
    chisq.out.test(farm.data[,m+1])
    x=chisq.out.test(farm.data[,m+1])
    if (x$p.value<0.0005)
      farm.data[which.min(farm.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(farm.data)){
    chisq.out.test(farm.data[,m+1],opposite = TRUE)
    x=chisq.out.test(farm.data[,m+1],opposite = TRUE)
    if (x$p.value<0.0005)
      farm.data[which.max(farm.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.farm=sum(is.na(farm.data)) #the number of outliers in farm
#test if there is any outlier class
for (i in 1:nrow(farm.data)){
  if (sum(is.na(farm.data[i,]))>3)
    farm.data[i,1]=NA
  else next
}
number.outliers.class.farm=sum(is.na(farm.data[,1])) # the number of outliers of cl
#delete the wrong class
index.outlier.farm=which(is.na(farm.data$class))
farm.data.prune=farm.data[-index.outlier.farm,] #the pruned data
farm.prune.left=sum(is.na(farm.data.prune))
#convert the outlier to the normal values
#converted by mean
```

```
for (m in 1:27){
  for (n in 1:nrow(farm.data.prune)){
    if (is.na(farm.data.prune[n,m+2]))
      farm.data.prune[n,m+2]=mean(farm.data.prune[,m+2],na.rm= T)
    else next
  }
}

#converted by median
#for (m in 1:27){
  #for (n in 1:nrow(farm.data.prune)){
    #if (is.na(farm.data.prune[n,m+1]==TRUE)
      #farm.data.prune[n,m+1]==median(farm.data.prune[,m+2],na.rm = FALSE)
    #else next
    #}
  #}

for (n in 1:nrow(farm.data.prune)){
  if (is.na(farm.data.prune[n,2]))
    farm.data.prune[n,2]=max(farm.data.prune[n,-1],na.rm= T)
  else next
}

#grass p.value=0.0005 K=4
for (m in 1:28){
  for (n in 1:nrow(grass.data)){
    chisq.out.test(grass.data[,m+1])
    x=chisq.out.test(grass.data[,m+1])
    if (x$p.value<0.0005)
      grass.data[which.min(grass.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(grass.data)){
    chisq.out.test(grass.data[,m+1],opposite = TRUE)
    x=chisq.out.test(grass.data[,m+1],opposite = TRUE)
    if (x$p.value<0.0005)
      grass.data[which.max(grass.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.grass=sum(is.na(grass.data)) #the number of outliers in grass
#test if there is any outlier class
for (i in 1:nrow(grass.data)){
  if (sum(is.na(grass.data[i,]))>3)
    grass.data[i,1]=NA
  else next
}
number.outliers.class.grass=sum(is.na(grass.data[,1])) # the number of outliers of
#delete the wrong class
index.outlier.grass=which(is.na(grass.data$class))
grass.data.prune=grass.data[-index.outlier.grass,] #the pruned data
grass.prune.left=sum(is.na(grass.data.prune))
#convert the outlier to the normal values
#converted by mean
```

```
for (m in 1:27){
  for (n in 1:nrow(grass.data.prune)){
    if (is.na(grass.data.prune[n,m+2]))
      grass.data.prune[n,m+2]=mean(grass.data.prune[,m+2],na.rm= T)
    else next
  }
}
#converted by median
#for (m in 1:28){
  #for (n in 1:nrow(grass.data.prune)){
    #if (is.na(grass.data.prune[n,m+1]==TRUE)
      #grass.data.prune[n,m+1]==median(grass.data.prune[,m+1],na.rm = FALSE)
    #else next
  #}
#}

for (n in 1:nrow(grass.data.prune)){
  if (is.na(grass.data.prune[n,2]))
    grass.data.prune[n,2]=max(grass.data.prune[n,-1],na.rm= T)
  else next
}

#impervious p.value=0.000001 K=6
for (m in 1:28){
  for (n in 1:nrow(impervious.data)){
    chisq.out.test(impervious.data[,m+1])
    x=chisq.out.test(impervious.data[,m+1])
    if (x$p.value<0.000001)
      impervious.data[which.min(impervious.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(impervious.data)){
    chisq.out.test(impervious.data[,m+1],opposite = TRUE)
    x=chisq.out.test(impervious.data[,m+1],opposite = TRUE)
    if (x$p.value<0.000001)
      impervious.data[which.max(impervious.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.impervious=sum(is.na(impervious.data)) #the number of outliers in i
#test if there is any outlier class
for (i in 1:nrow(impervious.data)){
  if (sum(is.na(impervious.data[i,]))>5)
    impervious.data[i,1]=NA
  else next
}
number.outliers.class.impervious=sum(is.na(impervious.data[,1])) # the number of ou
#delete the wrong class
index.outlier.impervious=which(is.na(impervious.data$class))
impervious.data.prune=impervious.data[-index.outlier.impervious,] #the pruned data

impervious.prune.left=sum(is.na(impervious.data.prune))
#convert the outlier to the normal values
#converted by mean
```

```
for (m in 1:27){
  for (n in 1:nrow(impervious.data.prune)){
    if (is.na(impervious.data.prune[n,m+2]))
      impervious.data.prune[n,m+2]=mean(impervious.data.prune[,m+2],na.rm= T)
    else next
  }
}
#converted by median
#for (m in 1:28){
  #for (n in 1:nrow(impervious.data.prune)){
    #if (is.na(impervious.data.prune[n,m+1]==TRUE)
      #impervious.data.prune[n,m+1]==median(impervious.data.prune[,m+1],na.rm = FALS
    #else next
  #}
#}

for (n in 1:nrow(impervious.data.prune)){
  if (is.na(impervious.data[n,2]))
    impervious.data.prune[n,2]=max(impervious.data.prune[n,-1],na.rm= T)
  else next
}

#orchard p.value=0.005 K=4
for (m in 1:28){
  for (n in 1:nrow(orchard.data)){
    chisq.out.test(orchard.data[,m+1])
    x=chisq.out.test(orchard.data[,m+1])
    if (x$p.value<0.005)
      orchard.data[which.min(orchard.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(orchard.data)){
    chisq.out.test(orchard.data[,m+1],opposite = TRUE)
    x=chisq.out.test(orchard.data[,m+1],opposite = TRUE)
    if (x$p.value<0.005)
      orchard.data[which.max(orchard.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.orchard=sum(is.na(orchard.data)) #the number of outliers in orchard
#test if there is any outlier class
for (i in 1:nrow(orchard.data)){
  if (sum(is.na(orchard.data[i,]))>3)
    orchard.data[i,1]=NA
  else next
}
number.outliers.class.orchard=sum(is.na(orchard.data[,1])) # the number of outliers
#delete the wrong class
index.outlier.orchard=which(is.na(orchard.data$class))
orchard.data.prune=orchard.data[-index.outlier.orchard,] #the pruned data
orchard.prune.left=sum(is.na(orchard.data.prune))
#convert the outlier to the normal values
#converted by mean
for (m in 1:27){
```

```
  for (n in 1:nrow(orchard.data.prune)){
    if (is.na(orchard.data.prune[n,m+2]))
      orchard.data.prune[n,m+2]=mean(orchard.data.prune[,m+2],na.rm= T)
    else next
  }
}
#converted by median

#for (m in 1:28){
  #for (n in 1:nrow(orchard.data.prune)){
    #if (is.na(orchard.data.prune[n,m+1]==TRUE)
      #orchard.data.prune[n,m+1]==median(orchard.data.prune[,m+1],na.rm = FALSE)
    #else next
  #}
#}

for (n in 1:nrow(orchard.data.prune)){
  if (is.na(orchard.data.prune[n,2]))
    orchard.data.prune[n,2]=max(orchard.data.prune[n,-1],na.rm= T)
  else next
}

#water p.value=0.00001 K=5
for (m in 1:28){
  for (n in 1:nrow(water.data)){
    chisq.out.test(water.data[,m+1])
    x=chisq.out.test(water.data[,m+1])
    if (x$p.value<0.00001)
      water.data[which.min(water.data[,m+1]),m+1]=NA
    else break
  }
  for (n in 1:nrow(water.data)){
    chisq.out.test(water.data[,m+1],opposite = TRUE)
    x=chisq.out.test(water.data[,m+1],opposite = TRUE)
    if (x$p.value<0.00001)
      water.data[which.max(water.data[,m+1]),m+1]=NA
    else break
  }
}
number.outliers.water=sum(is.na(water.data)) #the number of outliers in water
#test if there is any outlier class
for (i in 1:nrow(water.data)){
  if (sum(is.na(water.data[i,]))>4)
    water.data[i,1]=NA
  else next
}
number.outliers.class.water=sum(is.na(water.data[,1])) # the number of outliers of
#delete the wrong class
index.outlier.water=which(is.na(water.data$class))
water.data.prune=water.data[-index.outlier.water,] #the pruned data
water.prune.left=sum(is.na(water.data.prune))
#convert the outlier to the normal values
#converted by mean
for (m in 1:27){
```

```
  for (n in 1:nrow(water.data.prune)){
    if (is.na(water.data.prune[n,m+2]))
      water.data.prune[n,m+2]=mean(water.data.prune[,m+2],na.rm= T)
    else next
  }
}
#converted by median

#for (m in 1:28){
  #for (n in 1:nrow(water.data.prune)){
    #if (is.na(water.data.prune[n,m+1]==TRUE)
      #water.data.prune[n,m+1]==median(water.data.prune[,m+1],na.rm = FALSE)
    #else next
    #}
  #}

for (n in 1:nrow(water.data.prune)){
  if (is.na(water.data.prune[n,2]))
    water.data.prune[n,2]=max(water.data.prune[n,-1],na.rm= T)
  else next
}

data.denoising=rbind(farm.data.prune,
forest.data.prune,grass.data.prune,impervious.data.prune,
orchard.data.prune,water.data.prune)
```

## 5.2 Boxplot Method

```
#Boxplot Method
forest.index = which(data.all$class=='forest')
forest.data = data.all[forest.index,]

farm.index = which(data.all$class=='farm')
farm.data = data.all[farm.index,]

grass.index = which(data.all$class=='grass')
grass.data = data.all[grass.index,]

impervious.index = which(data.all$class=='impervious')
impervious.data = data.all[impervious.index,]

orchard.index = which(data.all$class=='orchard')
orchard.data = data.all[orchard.index,]

water.index = which(data.all$class=='water')
water.data = data.all[water.index,]

outs=NULL
for (i in 1:27){
    target=forest.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
```

```
forest.data=forest.data[-outs,]
outs=NULL
for (i in 1:27){
    target=farm.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
farm.data=farm.data[-outs,]
outs=NULL
for (i in 1:27){
    target=grass.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
grass.data=grass.data[-outs,]
outs=NULL
for (i in 1:27){
    target=impervious.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
impervious.data=impervious.data[-outs,]
outs=NULL
for (i in 1:27){
    target=orchard.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
orchard.data=orchard.data[-outs,]
outs=NULL
for (i in 1:27){
    target=water.data[,i+1]
    outs=c(which(target %in% boxplot.stats(target,coef=2.5)$out),outs)
}
water.data=water.data[-outs,]

data_clean=rbind(water.data,grass.data,forest.data,
farm.data,impervious.data,orchard.data)
```

### 5.3 Select the best k in KNN

```
library(ISLR)
library(class)
library(e1071)

#KNN vs. 10-fold CV
setwd("~/Desktop/ERG␣2050/project")

train = read.csv("training.csv", head = T)
#In this example train is raw data given by instruction
#we simply asign the denoised data to the variable train to select model
attach(train)
error.set = matrix(nrow = 10, ncol =10)
for(i in 1:10){
  print(i)
  set.seed(i)
```

```
  kfolds = tune.knn(train[,-c(1,2)], class, k = c(1:10))
  for(j in 1:10){
    error.set[i,j] = kfolds$performances$error[j]
  }
}
```

## 5.4 Select the Best Size of Tree

```
#Tree classification

library(tree)
library(ISLR)
attach(train)

setwd("~/Desktop/ERG␣2050/project")
train = read.csv("training.csv", head = T)
#In this example train is raw data given by instruction
#we simply asign the denoised data to the variable train to select model

best.size=rep(1,100)
tree.train.com = tree(class~., train, control = tree.control(minsize = 2, mincut =
for(i in 1:100){
  cv.train.com = cv.tree(tree.train.com, FUN=prune.misclass)
  best.size = cv.train.com$sizew[wwhich.min(cv.train.com$dev)]
}
```

## 5.5 Other Models

```
#-------Logistic Regression----------


library(boot)

setwd("~/Desktop/ERG␣2050/project")
train = read.csv("training.csv", head = T)
#In this example train is raw data given by instruction
#we simply asign the denoised data to the variable train to select model

attach(train)
error.set = rep(0,10)
lgst = glm(class~., data = train, family = binomial)
for(i in 1:10){
  print(i)
  set.seed(i)
  cv.lgst = cv.glm(train, lgst, K = 10)
  error.set[i] = cv.lgst$delta[1]
}

#----------LDA------------

library(MASS)
```

```
library(caret)

setwd("~/Desktop/ERG␣2050/project")
train = read.csv("training.csv", head = T)
#In this example train is raw data given by instruction
#we simply asign the denoised data to the variable train to select model

attach(train)
error.set = rep(0,10)
cv.error = rep(0,10)

for(i in 1:10){
  print(i)
  set.seed(i)
  folds = createFolds(1:nrow(train))
  for(j in 1:10){
    test.index = unlist(folds[j])
    lda.train = lda(class~., data = train[test.index,])
    lda.pred = predict(lda.train,train[-test.index,-1])
    cv.error[j] = 1 - mean(lda.pred$class == train[-test.index,1])
  }
  error.set[i] = mean(cv.error)
}

#------------QDA------------

library(MASS)
library(caret)

setwd("~/Desktop/ERG␣2050/project")
train = read.csv("training.csv", head = T)
#In this example train is raw data given by instruction
#we simply asign the denoised data to the variable train to select model

attach(train)
error.set = rep(0,10)
cv.error = rep(0,10)

for(i in 1:10){
  print(i)
  set.seed(i)
  folds = createFolds(1:nrow(train))
  orchard.index = which(train$class=='orchard')
  orchard.data = train[orchard.index,]
  for(j in 1:10){
    test.index = unlist(folds[j])
    test.data=rbind(orchard.data,train[test.index,])
    qda.train=qda(class~.,data=train[-test.index,])
    qda.pred = predict(qda.train,train[test.index,-1])
    cv.error[j] = 1 - mean(qda.pred$class == train[test.index,1])
  }
  error.set[i] = mean(cv.error)
}
```