Sri Guru Gobind Singh College of Commerce
University of Delhi

Data Mining

Practical File

Submitted By
Om Gupta
Roll Number-214047
BSc. (Hons.) Computer Science

Submitted To
Dr. Megha Ummat

Index

Data Cleaning

```r
library("psych")

# Q1)
# a)
str(cars)

mean(cars$speed)
mean(cars$dist)

geometric.mean(cars$speed)
geometric.mean(cars$dist)

harmonic.mean(cars$speed)
harmonic.mean(cars$dist)

# b)
unique(cars$dist)

# c)
var(cars$dist)
var(cars$speed)

# d)
IQR(cars$speed)

# e)
quantile(cars$dist)
```

EditRules

Main.R

```
library("editrules")

# Q1)
people <- read.csv("./files/people.txt")
people

# Check violation of age
E_set <- editset(c("age >= 0", "age <= 150"))
violatedEdits(E_set, people)

# Check Violation using file with edit rules
E1 <- editfile("./files/edits.txt")
ve1 <- violatedEdits(E1, people)
summary(ve1)
plot(ve1)

# Q2)
employee <- read.csv("./files/employee.txt", sep=' ')
E2 <- editfile("./files/edits_employee.txt")
ve2 <- violatedEdits(E2, employee)
summary(ve2)
plot(ve2)
ve2
```

people.txt

```
age,agegroup,height,status,yearsmarried
21,adult,6.0,single,-1
2,child,3,married,0
18,adult,5.7,married,20
221,elderly,5,widowed,2
34,child,-7,married,3
```

employee.txt

```
Age Department Salary Increment Grade
28 Manufacturing 50000 60000 A
21 Sales 20000 5000 A
35 Management 65000 6500 A
22 Sales 14000 20000 B
70 Manufacturing 44000 20000 A
```

edits.txt

```
# numerical rules
age >= 0
height > 0
age <= 150
age > yearsmarried

# categorical rules
status %in% c("married", "single", "widowed")
agegroup %in% c("child", "adult", "elderly")
if (status == "married") agegroup %in% c("adult", "elderly")

# mixed rules
```

```
if (status %in% c("married", "widowed")) age - yearsmarried
> 17

if (age < 18) agegroup == "child"

if (age >= 18 && age < 65) agegroup == "adult"

if (age >= 65) agegroup == "elderly"
```

edits_employee.txt

```
# numerical rules

Age >= 15

Age <= 65

Salary > Increment


# categorical rules

Department %in% c("Manufacturing", "Sales")


# mixed rules

if (Salary < 25000) Grade == "B"
```

Practical Question 1

Main.R

```r
iris_dirty = read.csv('./files/iris_dirty.csv', colClasses = c("NULL", NA, NA, NA, NA, NA))
iris_dirty

# Q2)
# i)
complete_obs =  sum(complete.cases(iris_dirty))
percent_complete = complete_obs/nrow(iris_dirty) * 100

cat('Number of Observations that are complete: ', complete_obs)
cat('% of Observations that are complete', percent_complete)

# ii) Replace NA with mean of columns
for(i in colnames(iris_dirty)) {
  iris_dirty[,i][is.na(iris_dirty[,i])] <- mean(iris_dirty[,i], na.rm = TRUE)
}
iris_dirty

# iii)
library("editrules")

E <- editfile("./files/dirty_iris_rules.txt")
ve <- violatedEdits(E, iris_dirty)
ve

# iv)
summary(ve)
plot(ve)

# v)
boxplot(iris_dirty$Sepal.Length, main="Box plot", ylab="Sepal Length")
boxplot.stats(iris_dirty$Sepal.Length, coef = 1.5, do.conf = TRUE, do.out = TRUE)
```

dirty_iris_rules.txt

```
# numerical rules

Sepal.Length > 0

Sepal.Length < 30
```

```
Sepal.Width > 0
Petal.Length > 0
Petal.Width > 0

Petal.Length >= 2*Petal.Width

Sepal.Length > Petal.Length

# categorical rules
Species %in% c("setosa", "versicolor", "virginica")
```

Practical Question 3

```r
# Functions
standardize = function(x) {
  z <- (x - mean(x)) / sd(x)
  return(z)
}

show_df = function(df) {
  for (attr in colnames(df)) {
    cat("Attribute:", attr)
    cat("\n\tMean:", mean(df[, attr]))
    cat("\n\tStd. Dev.:", sd(df[, attr]))
    cat("\n\n")
  }
}

wine = read.csv("./data/wine.csv")
wine_data <- wine[-c(1, 1)]

show_df(wine_data)
wine_standardized <- apply(wine_data, 2, standardize)
show_df(wine_standardized)

library(datasets)
data(iris)

iris_data <- iris[-c(1, 5)]
iris_data

show_df(iris_data)
iris_standardized <- apply(iris_data, 2, standardize)
show_df(iris_standardized)
```

K-means Clustering

```
library (ggplot2)

data(iris)

iris2<-iris

iris2$Species<-NULL #removing dependent variable

str(iris2)

km<-kmeans(iris2,3) #forms '3' clusters
km$centers # centers of all clusters

km$cluster #cluster of every record


km$iter #no. of clustering iterations

km$withinss #sse of each cluster

iris2$cluster<-factor(km$cluster) # new column for cluster no.
iris2

I<-iris2[c(1,2,5)] # taking 1st & 2nd columns to plot and 5th for clustering
I

centers<-data.frame(cluster=factor(1:3),km$centers) #collecting clusters' centers into a
factor centers

#graph plotted b/w x-y
ggplot(data=I,aes(x=Sepal.Length,y=Sepal.Width,color=cluster,shape=cluster))+
  geom_point(alpha=0.2)+ #alpha is the width of points
  geom_point(data=centers,aes(x=Sepal.Length,y=Sepal.Width),size=3,stroke=2)

#size is the size of center points of each cluster
```

```r
# ========= NEW

wine<-read.csv('./data/wine.csv')

wine2 <- wine

wine2$Wine<-NULL #removing dependent variable

str(wine2)

km<-kmeans(wine2,3) #forms '3' clusters
km$centers # centers of all clusters

km$cluster #cluster of every record


km$iter #no. of clustering iterations

km$withinss #sse of each cluster

wine2$cluster<-factor(km$cluster) # new column for cluster no.
wine2

I<-wine2[c(1, 5, 14)] # taking 1st & 5nd columns to plot and 5th for clustering
I

centers<-data.frame(cluster=factor(1:3),km$centers) #collecting clusters' centers into a factor centers

#graph plotted b/w x-y
ggplot(data=I,aes(x=Alcohol,y=Mg,color=cluster,shape=cluster))+
  geom_point(alpha=0.7)+ #alpha is the width of points
  geom_point(data=centers,aes(x=Alcohol,y=Mg),size=3,stroke=2)

#size is the size of center points of each cluster
```

Practical Question 2

```r
data <- read.csv("./data/iris_dirty.csv")
data <- data[, -1]

# 1) Calculate the number and percentage of observations that are complete.
total_observations <- length(complete.cases(data))
complete_observations <- sum(complete.cases(data))

cat('The number of complete observations are: ', complete_observations)
cat('The % of complete observations are: ', complete_observations/total_observations * 100,
"%")

# ii) Replace NA with mean of columns
for(i in colnames(data)) {
  data[,i][is.na(data[,i])] <- mean(data[,i], na.rm = TRUE)
}
data

# iii) Define these rules in a separate text file and read them.
library("editrules")

E <- editfile("./files/dirty_iris_rules.txt")
ve <- violatedEdits(E, data)
ve

# iv)
summary(ve)
plot(ve)

# v)
boxplot(data$Sepal.Length, main="Box plot", ylab="Sepal Length")
boxplot.stats(data$Sepal.Length, coef = 1.5, do.conf = TRUE, do.out = TRUE)
```

Hierarchial Clustering

```r
# Iris Dataset
id1 <- sample(1:dim(iris)[1],30)
id1

irisSample <- iris[id1,]
irisSample$Species <- NULL
irisSample

hc <- hclust(dist(irisSample),method="ave")

plot(hc,hang=-1,labels=iris$Species[id1])

rect.hclust(hc,k=4)

groups <- cutree(hc,k=4)
groups


# Penguins Dataset
library(palmerpenguins)
dm <- penguins
data <- dm[complete.cases(dm), ]

id2 <- sample(1:dim(data)[1],30)
id2

dataSample <- data[id2,]
dataSample$species <- NULL
dataSample

hc <- hclust(dist(dataSample),method="ave")

plot(hc,hang=-1,labels=data$species[id1])

rect.hclust(hc,k=4)

groups <- cutree(hc,k=4)
groups
```

DBScan Clustering

```r
library(fpc)

# Iris Dataset
data(iris)

iris2 <- iris[-5]
iris2

ds <- dbscan(iris2,eps=0.45,MinPts=5)
ds

str(ds)

ds$cluster

table(ds$cluster,iris$Species)

plot(ds,iris2[c(1,4)])

plot(ds,iris2)

# Dm Dataset
library(caret)
library(palmerpenguins)
dm <- penguins
data <- dm[complete.cases(dm), ]

data$island

data2 <- data[c(3, 4, 5, 6)]
data2

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

data2 <- as.data.frame(lapply(data2[1:4], min_max_norm))

ds <- dbscan(data2,eps=0.2,MinPts=5)
ds

str(ds)

ds$cluster

data$species
```

```
table(ds$cluster, data$species)

plot(ds,data2[c(1,4)])

plot(ds,data2)
```

Classification

```
library(caret)
```

```r
library(RWeka) # for methos="J48"
library(klaR) # for method="knn"
library(e1071) # for method="nb"

data <- read.csv("./data/BreastCancer.csv")

min_max_normalisation <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Species -> Dependent Variable
summary(data)

data <- data[c(-1)]
data$diagnosis <- replace(data$diagnosis, data$diagnosis %in% == c("B", "M"), c(0, 1))

# ================== NB

# ================== a) Hold-out Method 0.75
train_index <- createDataPartition(data$Species, p=0.75, list=FALSE) # 70-30 split
train_index

data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species

summary(data_test)

set.seed(42)

# ML Model
model <- train(Species~., data_train, method="nb") # method="J48" for DT (not
working)
model

predictions <- predict(model, data_test)
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================== a) Hold-out Method 0.66
train_index <- createDataPartition(data$Species, p=0.66, list=FALSE) # 70-30 split
train_index
```

```r
data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species

summary(data_test)

set.seed(42)

# ML Model
model <- train(Species~., data_train, method="nb") # method="J48" for DT (not working)
model

predictions <- predict(model, data_test)
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================= b) K-Fold Cross Vaidation
set.seed(42)

# repeated cross fold, number is k, repeats is folds
fitControl <- trainControl(method="repeatedcv", number=10, repeats=10)

# ML Model
model <- train(Species~., data, method="nb", trainControl=fitControl)
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))

# ================= b) Hold-out One Vaidation
set.seed(42)

# Leave-One Out Cross Validation
fitControl <- trainControl(method="loocv", number=10)

# ML Model
model <- train(Species~., data, method="nb", trainControl=fitControl)
model
```

```r
predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))

# ================== b) K-Fold Vaidation
set.seed(42)

# K-Fold Validation
fitControl <- trainControl(method="cv", number=10)

# ML Model
model <- train(Species~., data, method="nb", trainControl=fitControl)
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))




# ================== KNN
data[, c(1, 2, 3, 4)] <- lapply(data[, c(1, 2, 3, 4)], min_max_normalisation)
data

# ================== a) Hold-out Method 0.75
train_index <- createDataPartition(data$Species, p=0.75, list=FALSE) # 70-30 split
train_index

data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species
summary(data_test)

set.seed(42)

# ML Model
model <- train(Species~., data_train, method="knn", preProcess=c("center", "scale")) #
method="J48" for DT (not working)
model

predictions <- predict(model, data_test)
```

```r
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================== a) Hold-out Method 0.66
train_index <- createDataPartition(data$Species, p=0.66, list=FALSE) # 70-30 split
train_index

data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species
summary(data_test)

set.seed(42)

# ML Model
model <- train(Species~., data_train, method="knn", preProcess=c("center", "scale")) #
method="J48" for DT (not working)
model

predictions <- predict(model, data_test)
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================== b) K-Fold Cross Vaidation
set.seed(42)

# repeated cross fold, number is k, repeats is folds
fitControl <- trainControl(method="repeatedcv", number=10, repeats=10)

# ML Model
model <- train(Species~., data, method="knn", trainControl=fitControl, preProcess=c("center",
"scale"))
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))

# ================== b) Hold-out One Vaidation
set.seed(42)
```

```r
# Leave-One Out Cross Validation
fitControl <- trainControl(method="loocv", number=10)

# ML Model
model <- train(Species~., data, method="knn", trainControl=fitControl, preProcess=c("center",
"scale"))
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))

# ================== b) K-Fold Vaidation
set.seed(42)

# K-Fold Validation
fitControl <- trainControl(method="cv", number=10)

# ML Model
model <- train(Species~., data, method="knn", trainControl=fitControl, preProcess=c("center",
"scale"))
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))


# ================ Decision Tree
data(data)

# ================== a) Hold-out Method 0.75
train_index <- createDataPartition(data$Species, p=0.75, list=FALSE) # 70-30 split
train_index

data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species
summary(data_test)

set.seed(42)
```

```r
# ML Model
model <- train(Species~., data_train, method="J48") # method="J48" for DT (not working)
model

predictions <- predict(model, data_test)
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================= a) Hold-out Method 0.66
train_index <- createDataPartition(data$Species, p=0.66, list=FALSE) # 70-30 split
train_index

data_train <- data[train_index, ] # Training Dataset
data_test <- data[-train_index, ] # Testing Dataset

summary(data_train) # Check number of values for each species
summary(data_test)
set.seed(42)

# ML Model
model <- train(Species~., data_train, method="J48") # method="J48" for DT (not working)
model

predictions <- predict(model, data_test)
predictions

confusionMatrix(predictions, as.factor(data_test$Species))

# ================= b) K-Fold Cross Vaidation
set.seed(42)

# repeated cross fold, number is k, repeats is folds
fitControl <- trainControl(method="repeatedcv", number=10, repeats=10)

# ML Model
model <- train(Species~., data, method="J48", trainControl=fitControl)
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))
```

```r
# ================== b) Hold-out One Vaidation
set.seed(42)

# Leave-One Out Cross Validation
fitControl <- trainControl(method="loocv", number=10)

# ML Model
model <- train(Species~., data, method="J48", trainControl=fitControl)
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))

# ================== b) K-Fold Vaidation
set.seed(42)

# K-Fold Validation
fitControl <- trainControl(method="cv", number=10)

# ML Model
model <- train(Species~., data, method="J48", trainControl=fitControl)
model

predictions <- predict(model, data)
predictions

confusionMatrix(predictions, as.factor(data$Species))
```

Association Rule Mining

```r
library(arules)

# a) Use minimum support as 50% and minimum confidence as 75%

# Dataset 1000
dataset1000=read.transactions("./data/1000-out1.csv",sep=",",rm.duplicate=TRUE)
dataset1000

summary(dataset1000)

itemFrequencyPlot(dataset1000, topN=10)
```

```r
rules1000=apriori(data=dataset1000,parameter=list(support=0.5,confidence=0.75)) # support=0.005,confidence=0.6
rules1000

inspect(rules1000)
inspect(sort(rules1000,by="confidence"))

# Dataset 5000
dataset5000=read.transactions("./data/5000-out1.csv",sep=",",rm.duplicate=TRUE)
dataset5000

summary(dataset5000)

itemFrequencyPlot(dataset5000, topN=10)

rules5000=apriori(data=dataset5000,parameter=list(support=0.5,confidence=0.75))
rules5000

inspect(rules5000)
inspect(sort(rules5000, by="confidence"))

# Market Basket
datasetmb=read.transactions("./data/Market_Basket.csv",sep=",",rm.duplicate=TRUE)
datasetmb

summary(datasetmb)

itemFrequencyPlot(datasetmb, topN=10)

rulesmb=apriori(data=datasetmb,parameter=list(support=0.5,confidence=0.75))
rulesmb

inspect(rulesmb)
inspect(sort(rulesmb, by="confidence"))


# b) Use minimum support as 60% and minimum confidence as 60%

# Dataset 1000
dataset1000

summary(dataset1000)

itemFrequencyPlot(dataset1000, topN=10)
```

```r
rules1000=apriori(data=dataset1000,parameter=list(support=0.6,confidence=0.6)) # support=0.005,confidence=0.6
rules1000

inspect(rules1000)
inspect(sort(rules1000,by="confidence"))

# Dataset 5000
dataset5000

summary(dataset5000)

itemFrequencyPlot(dataset5000, topN=10)

rules5000=apriori(data=dataset5000,parameter=list(support=0.6,confidence=0.6))
rules5000

inspect(rules5000)
inspect(sort(rules5000, by="confidence"))

# Market Basket
datasetmb

summary(datasetmb)

itemFrequencyPlot(datasetmb, topN=10)

rulesmb=apriori(data=datasetmb,parameter=list(support=0.6,confidence=0.6))
rulesmb

inspect(rulesmb)
inspect(sort(rulesmb, by="confidence"))


# c) Use different support and confidence thresholds.

# Dataset 1000
dataset1000

summary(dataset1000)

itemFrequencyPlot(dataset1000, topN=10)

rules1000=apriori(data=dataset1000,parameter=list(support=0.005,confidence=0.8)) # support=0.005,confidence=0.6
```

```
rules1000

inspect(rules1000)
inspect(sort(rules1000,by="confidence"))

# Dataset 5000
dataset5000

summary(dataset5000)

itemFrequencyPlot(dataset5000, topN=10)

rules5000=apriori(data=dataset5000,parameter=list(support=0.035,confidence=0.8))
rules5000

inspect(rules5000)
inspect(sort(rules5000, by="confidence"))

# Market Basket
datasetmb

summary(datasetmb)

itemFrequencyPlot(datasetmb, topN=10)

rulesmb=apriori(data=datasetmb,parameter=list(support=0.015,confidence=0.3))
rulesmb

inspect(rulesmb)
inspect(sort(rulesmb, by="confidence"))
```