**Python code for web-scraping demographics data**

#Import the required libraries

from lxml import html

import requests

import random

import time

import pandas as pd

from fake\_useragent import UserAgent

#Global declaration of user agent required for get() request

ua = UserAgent()

#Import the unique datasets present in our complaints database

filename = 'uniquezip\_tx.csv'

data = pd.read\_csv(filename, converters={0:str})

#Convert the dataframe into list

mnop = data['ZIP'].values.tolist()

#Change the subset values, in order to scrape data, like 1-500 records per day

mn = mnop[1435:1847]

#Create a nested list for gathering the scraped data for each zip

final = []

final\_1 = []

for l in range(len(mn)):

final.append(final\_1)

#for loop for multiple requests

for k in range(len(mn)):

agent = {'User-Agent': ua.random}

url = 'http://www.city-data.com/zips/'+mn[k]+'.html'

response = requests.get(url,headers=agent)

#Decode the website content using ISO 8859

decode\_response = response.content.decode('ISO-8859-1')

tree = html.fromstring(decode\_response)

#Obtain the required data from the html element using xpath

head\_key = tree.xpath('//div[@id="body"][@class="row"]//following-sibling::b/following::text()'

#Create empty list to keep appending required values from the head\_key

ab\_list = []

for i in range(len(head\_key)):

#Houses and condos

if("Houses and condos:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Housing units with mortgage

if("Housing units in zip code" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Housing units without mortgage

if("Houses without a mortgage" in head\_key[i]):

1

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Estimated median house/condo value

if("Estimated median house/condo value in 2016" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#cost of living index

if("Mar. 2016 cost of living index in zip code" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Estimated median household income in 2016:

if("Estimated median household income in 2016: " in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+2])

#Median gross rent

if("Median gross rent in 2016:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#real estate property taxes paid for housing units with mortgages

if("Median real estate property taxes paid for housing units with mortgages" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#real estate property taxes paid for housing units with no mortgage

if("Median real estate property taxes paid for housing units with no mortgage" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Unemployment rate

if("Unemployed:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Males

if("Males:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Females

if("Females:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#resident age

if("Median resident age:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+2])

#commute

if("Mean travel time to work (commute):" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

2

#Population density

if("Population density:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

#Population density

if("% of renters here:" in head\_key[i]):

#ab\_list.append(head\_key[i])

ab\_list.append(head\_key[i+1])

x = ab\_list.copy()

#Cleansing the obtained values

for i in range(len(x)):

x[i] = x[i].replace('$','')

x[i] = x[i].replace('minutes','')

x[i] = x[i].replace('years','')

x[i] = x[i].replace('\r\n','')

x[i] = x[i].replace(' ','')

x[i] = x[i].replace('\xa0','')

x[i] = x[i].replace(',','')

x[i] = x[i].replace('%','')

#Further cleasning activity

y = x.copy()

for i in range(len(x)):

if("(" in x[i]):

y[i] = "(".join(x[i].split("(")[:-1])

final[k]=y.copy()

#Generate a random time delay between each web request

timeDelay = random.randrange(0, 25)

time.sleep(timeDelay)

#Create a new dataframe and export to csv file format

new\_df = pd.DataFrame(columns=['houses','percent\_rent', 'cost\_of\_living','pop\_density', 'm\_prop\_tax\_'male', 'female', 'unemp', 'commute', 'est\_m\_house\_val', 'm\_res\_age', 'est\_m\_house\_'houses\_w\_mtg', 'houses\_w\_out\_mtg', 'm\_gross\_rent'], data=final, index=mn)

new\_df.to\_csv("ZipTX/zip\_tx\_v2.csv")

**R programming code for model building**

#### Loading required libraries

library(dplyr)

library(cluster)

library(stringr)

library(factoextra)

library(caret)

library(rpart)

library(rpart.plot)

library(DMwR)

library(ROSE)

library(ebmc)

library(rusboost)

library(gridExtra)

library(e1071)

library(nnet)

library(corrplot)

# Import CFPB Consumer dataset

data <- read.csv("ConsumerComplaintsAlltab.csv", sep = "\t")

str(data)

#remove complaint text, Tags, consumer\_consent, submitted\_via.

data\_nt <- data[-c(6,11,12,13)]

str(data\_nt)

#Demographics dataset

demo\_tx <- read.csv("WebScraping/ZipTX/zip\_tx\_v1.csv")

demo\_tx\_na <- na.omit(demo\_tx)

names(demo\_tx\_na)[1] <- "ZIP.code"

summary(demo\_tx\_na)

# Clustering on demographic dataset

demo\_tx\_na\_rn <- demo\_tx\_na[,-1]

demo\_tx\_scale <- demo\_tx\_na[,-1]

demo\_tx\_scale <- scale(demo\_tx\_scale)

rownames(demo\_tx\_scale) <- demo\_tx\_na[,1]

demo\_tx\_eucl <- dist(demo\_tx\_scale, method = "euclidean")

#Scree plot

fviz\_nbclust(demo\_tx\_scale, kmeans, method = "wss") + geom\_vline(xintercept = 3, linetype = 2)

#K means Clustering

set.seed(123)

km.res <- kmeans(demo\_tx\_scale, 3, nstart = 50)

demo\_tx\_clust <- cbind(demo\_tx\_na\_rn, cluster=km.res$cluster)

#Visualizing different cluster sizes

km.res.2 <- kmeans(demo\_tx\_scale, 2, nstart = 50)

km.res.3 <- kmeans(demo\_tx\_scale, 3, nstart = 50)

km.res.4 <- kmeans(demo\_tx\_scale, 4, nstart = 50)

km.res.5 <- kmeans(demo\_tx\_scale, 5, nstart = 50)

p1 <- fviz\_cluster(km.res.2, data = demo\_tx\_scale, geom = "point")

p2 <- fviz\_cluster(km.res.3, data = demo\_tx\_scale, geom = "point")

p3 <- fviz\_cluster(km.res.4, data = demo\_tx\_scale, geom = "point")

p4 <- fviz\_cluster(km.res.5, data = demo\_tx\_scale, geom = "point")

grid.arrange(p1, p2, p3, p4, nrow = 2)

#Cluster statistics

aggregate(demo\_tx\_na\_rn, by = list(cluster=km.res$cluster), mean)

demo\_tx\_clust <- cbind(demo\_tx\_na\_rn, cluster=km.res$cluster)

#cluster size

km.res$size

#cluster means

km.res$centers

#visualization

fviz\_cluster(km.res, data = demo\_tx\_scale, geom = "point")

#Merge with original TEXAS dataset

data\_zip\_merge\_tx <- data\_nt %>% filter(data\_nt$State=="TX")

demo\_tx\_zipclust\_prep <- cbind(ZIP.code=demo\_tx\_na$ZIP.code, demo\_tx\_na\_rn, cluster=km.res$cluster)

merged\_clust\_tx <- merge(data\_zip\_merge\_tx, demo\_tx\_zipclust\_prep, by="ZIP.code", all.x = T)

merged\_clust\_tx\_na <- merged\_clust\_tx %>% filter(!is.na(cluster))

str(merged\_clust\_tx\_na)

# Data preparation for predicting company response

#remove date, public response, state, date sent, consumer disputation, complaint id

tx\_clust <- merged\_clust\_tx\_na[-c(7, 9, 10, 12, 13, 31)]

tx\_clust[,"Month"] <- as.factor(format(as.Date(tx\_clust$Date.received, format="%m/%d/%Y"),"%m"))

#tx\_clust[,"Year"] <- as.factor(format(as.Date(tx\_clust$Date.received, format="%m/%d/%Y"),"%Y"))

#remove data received

tx\_clust <-tx\_clust[-2]

str(tx\_clust)

#replace sub issue

for(i in 2:5){

tx\_clust[,i] <- as.character(tx\_clust[,i])

}

x1 <- tx\_clust$Sub.product==""

x2 <- tx\_clust$Sub.issue==""

tx\_clust$Sub.product[x1] <- tx\_clust$Product[x1]

tx\_clust$Sub.issue[x2] <- tx\_clust$Issue[x2]

for(i in 2:5){

tx\_clust[,i] <- as.factor(tx\_clust[,i])

}

#combine outcome categories

tx\_clust$Company.response.to.consumer <-as.character(tx\_clust$Company.response.to.consumer)

x1 <- tx\_clust$Company.response.to.consumer == "Closed"

tx\_clust$Company.response.to.consumer[x1] = "Consumer\_fault"

x1 <- tx\_clust$Company.response.to.consumer=="Closed with explanation"

tx\_clust$Company.response.to.consumer[x1] = "Consumer\_fault"

x1 <- tx\_clust$Company.response.to.consumer=="Closed without relief"

tx\_clust$Company.response.to.consumer[x1] = "Consumer\_fault"

x1 <- tx\_clust$Company.response.to.consumer == "Closed with monetary relief"

tx\_clust$Company.response.to.consumer[x1] = "Company\_fault"

x1 <- tx\_clust$Company.response.to.consumer=="Closed with non-monetary relief"

tx\_clust$Company.response.to.consumer[x1] = "Company\_fault"

x1 <- tx\_clust$Company.response.to.consumer=="Closed with relief"

tx\_clust$Company.response.to.consumer[x1] = "Company\_fault"

x1 <- tx\_clust$Company.response.to.consumer == "Company\_fault" | tx\_clust$Company.response.to.consumer == "Consumer\_fault"

tx\_clust$Company.response.to.consumer[!x1] = "Late\_Response"

final\_tx\_resp\_na <- tx\_clust %>% filter(tx\_clust$Company.response.to.consumer!="Late\_Response")

final\_tx\_resp\_na$Company.response.to.consumer <-as.factor(final\_tx\_resp\_na$Company.response.to.consumer)

#-------------------------------------------------------------------------------------Mortgage data

tx\_clust\_rep\_mortgage <- final\_tx\_resp\_na %>% filter(Product=="Mortgage")

tx\_clust\_rep\_mortgage\_issues <- tx\_clust\_rep\_mortgage %>% filter(Issue=="Loan modification,collection,foreclosure" | Issue=="Loan servicing, payments, escrow account")

final\_tx\_resp\_na\_mortgage <- tx\_clust\_rep\_mortgage\_issues[-c(2)]

for(i in c(1,2,3,4,5,6,24)){

final\_tx\_resp\_na\_mortgage[,i]<-as.character(final\_tx\_resp\_na\_mortgage[,i])

final\_tx\_resp\_na\_mortgage[,i]<-as.factor(final\_tx\_resp\_na\_mortgage[,i])

}

## Modeling

#Remove complaint ID

final\_tx <- final\_tx\_resp\_na\_mortgage[,-7] #final\_tx\_prep[-c(1,2, 7, 10, 14, 15)]

str(final\_tx)

for(i in c(1,2,3,4,5,6,23)){

final\_tx[,i]<-as.character(final\_tx[,i])

final\_tx[,i]<-as.factor(final\_tx[,i])

}

for(i in 7:22){

final\_tx[,i]<-as.numeric(final\_tx[,i])

}

final\_tx[,7:22] <- scale(final\_tx[,7:22])

#-----------------------------------------------------------------------------------------data exports

texas\_mortgage <- final\_tx\_resp\_na\_mortgage[,-7]

#write.csv(texas\_mortgage, "texas\_mortgage.csv")

ref\_final\_tx <- final\_tx

#correlation

x <- final\_tx

for(i in 1:ncol(x)){

x[,i] <- as.numeric(x[,i])

}

x <- scale(x)

#write.csv(cor(x), "Correlation.csv")

#Excluding multi collinear variables

final\_tx <- final\_tx[,-c(1, 4, 7, 11, 12, 14, 15, 19)]

#----------------------------------------------------------------------------------------Train and test

set.seed(123)

partition<-createDataPartition(final\_tx$Company.response.to.consumer,p=0.80,list = FALSE)

train\_tx<-final\_tx[partition,]

test\_tx<-final\_tx[-partition,]

#-------------------------------------------------------------------decision tree model

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_tx, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

#rpart.plot(tx\_model\_resp,digits = 2,fallen.leaves = TRUE,type = 3,extra = 1)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault", mode = "everything")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#----------------------------------------------------------------------OVer sampling

train\_over <- ovun.sample(Company.response.to.consumer ~ ., data=train\_tx, method = "over", N=9000, seed=123)$data

table(train\_over$Company.response.to.consumer)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_over, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault", mode = "everything")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test\_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#----------------------------------------------------------------------Undersampling

train\_under <- ovun.sample(Company.response.to.consumer ~ ., data=train\_tx, method = "under", N=972, seed=123)$data

table(train\_under$Company.response.to.consumer)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_under, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test\_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test\_tx$Company.response.to.consumer, wp1[,2], plotit = F)

#----------------------------------------------------------------------Both

train\_both <- ovun.sample(Company.response.to.consumer ~ ., data=train\_tx, method = "both", p=0.5, seed = 123)$data

table(train\_both$Company.response.to.consumer)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_both, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault", mode="everything")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test\_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test\_tx$Company.response.to.consumer, wp1[,2], plotit = F)

#----------------------------------------------------------------------ROSE

train\_rose <- ROSE(Company.response.to.consumer ~ ., data=train\_tx, seed = 123)$data

table(train\_rose$Company.response.to.consumer)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_rose, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test\_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test\_tx$Company.response.to.consumer, wp1[,2], plotit = F)

#------------------------------------------------------------------------------DMwR SMOTE

set.seed(123)

train\_SMOTE <- SMOTE(Company.response.to.consumer ~ ., data = train\_tx, perc.over = 100, perc.under = 200)

table(train\_SMOTE$Company.response.to.consumer)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_SMOTE, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test\_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test\_tx$Company.response.to.consumer, wp1[,2], plotit = F)

# SVM

tx\_model\_resp <- svm(Company.response.to.consumer ~ ., data=train\_tx)

#varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)

#nnet

num\_tx\_mortgage <- final\_tx

num\_tx\_mortgage\_y <- num\_tx\_mortgage[4]

num\_tx\_mortgage\_x <- num\_tx\_mortgage[,-4]

for(i in (1:ncol(num\_tx\_mortgage\_x))){

num\_tx\_mortgage\_x[,i] <- as.numeric(num\_tx\_mortgage\_x[,i])

}

num\_tx\_mortgage\_x <- scale(num\_tx\_mortgage\_x)

num\_tx <- cbind(num\_tx\_mortgage\_x, num\_tx\_mortgage\_y)

set.seed(123)

partition<-createDataPartition(num\_tx$Company.response.to.consumer,p=0.70,list = FALSE)

train\_tx<-num\_tx[partition,]

test\_tx<-num\_tx[-partition,]

train\_SMOTE <- SMOTE(Company.response.to.consumer ~ ., data = train\_tx, perc.over = 100, perc.under = 200)

table(train\_SMOTE$Company.response.to.consumer)

train\_over <- ovun.sample(Company.response.to.consumer ~ ., data=train\_tx, method = "over", N=9000, seed=123)$data

table(train\_over$Company.response.to.consumer)

tx\_model\_resp <- nnet(Company.response.to.consumer ~ ., data=train\_SMOTE, size = 10)

#varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(as.factor(wp1), test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, as.factor(wp1), plotit = F)

tx\_model\_resp <- rpart(Company.response.to.consumer ~ ., data=train\_over, method = "class")

#rpart.plot(tx\_model\_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx\_model\_resp)

wp1 <- predict(tx\_model\_resp,test\_tx, type = "class")

confusionMatrix(wp1, test\_tx$Company.response.to.consumer, positive = "Company\_fault")

roc.curve(test\_tx$Company.response.to.consumer, wp1, plotit = F)