Coursework (CM3111)
Big Data Analytics

Alistair Quinn

December 12, 2017

---

1 Data Exploration

1.1 Dataset Choice

I have chosen a dataset that was used in the CoIL 2000 Challenge, which is called the Caravan Insurance Challenge dataset available here: https://www.kaggle.com/uciml/caravan-insurance-challenge

The data set is free to use for non-comercial use. Although sourced from the UCL Machine Learning organization's kaggle page, the dataset is owned by Sentient Machine Research.

1.2 Technology-Platform

The dataset is contained in a csv file that is 245KB in size. As the dataset is so small I not need to use Big Data technology such as Hadoop. Instead I will use RStudio on a windows PC. I chose the dataset based on my current ability, and I found the idea of the dataset interesting.

1.3 Problem Statement Data Exploration

Each row in the table corresponds to a post code.The task with this dataset is to identify potential purchasers of caravan insurance policies. The class label in the dataset is called CARAVAN and has two values, 0 or 1. CARAVARN is 1 when that row would potentially purchase a caravan insurance policy.

Below shows the number of rows and columns in the dataset

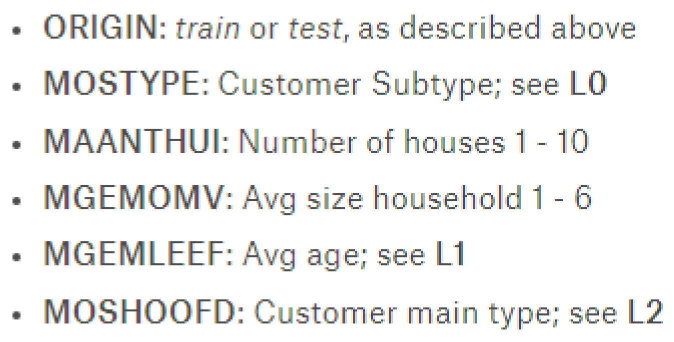
```r
#Set WD
setwd("D:/RGU/3rdYear/Semester1/Big Data Analytics/Coursework/wd")
#Load Data
df <- read.csv("Data/caravan-insurance-challenge.csv")
#Rows and Cols
```

```
nrow(df)

## [1] 9822

ncol(df)

## [1] 87
```

Below shows the names of the features

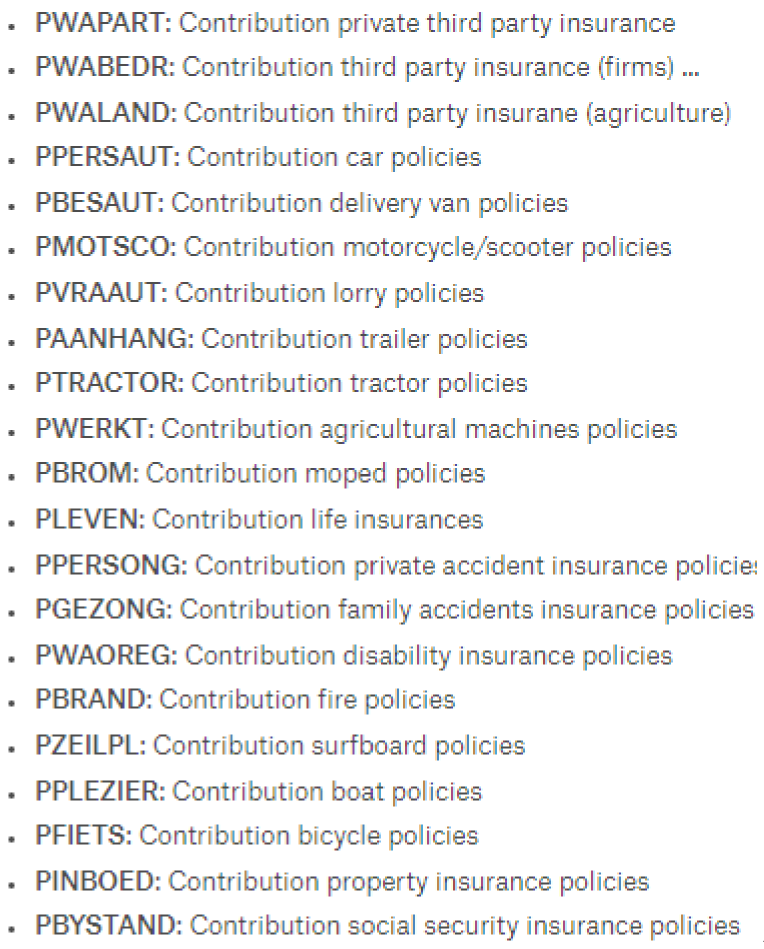
```
#Names of columns
names(df)

##  [1] "ORIGIN"   "MOSTYPE"  "MAANTHUI" "MGEMOMV"  "MGEMLEEF" "MOSHOOFD"
##  [7] "MGODRK"   "MGODPR"   "MGODOV"   "MGODGE"   "MRELGE"   "MRELSA"
## [13] "MRELOV"   "MFALLEEN" "MFGEKIND" "MFWEKIND" "MOPLHOOG" "MOPLMIDD"
## [19] "MOPLLAAG" "MBERHOOG" "MBERZELF" "MBERBOER" "MBERMIDD" "MBERARBG"
## [25] "MBERARBO" "MSKA"     "MSKB1"    "MSKB2"    "MSKC"     "MSKD"
## [31] "MHHUUR"   "MHKOOP"   "MAUT1"    "MAUT2"    "MAUT0"    "MZFONDS"
## [37] "MZPART"   "MINKM30"  "MINK3045" "MINK4575" "MINK7512" "MINK123M"
## [43] "MINKGEM"  "MKOOPKLA" "PWAPART"  "PWABEDR"  "PWALAND"  "PPERSAUT"
## [49] "PBESAUT"  "PMOTSCO"  "PVRAAUT"  "PAANHANG" "PTRACTOR" "PWERKT"
## [55] "PBROM"    "PLEVEN"   "PPERSONG" "PGEZONG"  "PWAOREG"  "PBRAND"
## [61] "PZEILPL"  "PPLEZIER" "PFIETS"   "PINBOED"  "PBYSTAND" "AWAPART"
## [67] "AWABEDR"  "AWALAND"  "APERSAUT" "ABESAUT"  "AMOTSCO"  "AVRAAUT"
## [73] "AAANHANG" "ATRACTOR" "AWERKT"   "ABROM"    "ALEVEN"   "APERSONG"
## [79] "AGEZONG"  "AWAOREG"  "ABRAND"   "AZEILPL"  "APLEZIER" "AFIETS"
## [85] "AINBOED"  "ABYSTAND" "CARAVAN"
```

Variables beginning with M are demographic statistics of the postal code.



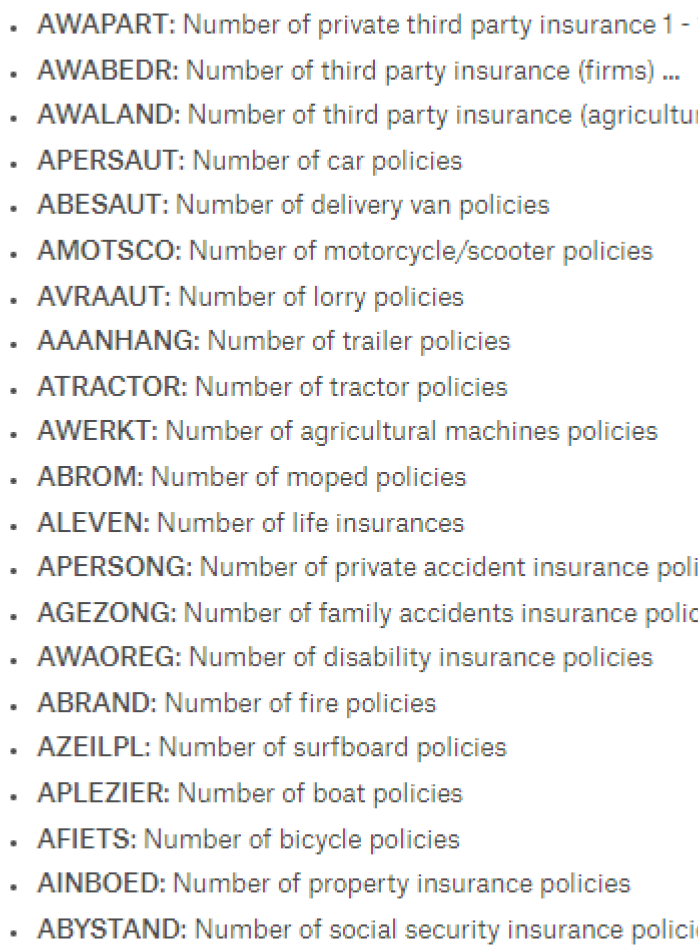
- **ORIGIN:** *train* or *test*, as described above
- **MOSTYPE:** Customer Subtype; see **L0**
- **MAANTHUI:** Number of houses 1 - 10
- **MGEMOMV:** Avg size household 1 - 6
- **MGEMLEEF:** Avg age; see **L1**
- **MOSHOOFD:** Customer main type; see **L2**

Variables beginning with P and A refer to product ownership and insurance statistics of the postal code. Variables beginning with P refer to contribution



- **PWAPART:** Contribution private third party insurance
- **PWABEDR:** Contribution third party insurance (firms) ...
- **PWALAND:** Contribution third party insurane (agriculture)
- **PPERSAUT:** Contribution car policies
- **PBESAUT:** Contribution delivery van policies
- **PMOTSCO:** Contribution motorcycle/scooter policies
- **PVRAAUT:** Contribution lorry policies
- **PAANHANG:** Contribution trailer policies
- **PTRACTOR:** Contribution tractor policies
- **PWERKT:** Contribution agricultural machines policies
- **PBROM:** Contribution moped policies
- **PLEVEN:** Contribution life insurances
- **PPERSONG:** Contribution private accident insurance policie
- **PGEZONG:** Contribution family accidents insurance policies
- **PWAOREG:** Contribution disability insurance policies
- **PBRAND:** Contribution fire policies
- **PZEILPL:** Contribution surfboard policies
- **PPLEZIER:** Contribution boat policies
- **PFIETS:** Contribution bicycle policies
- **PINBOED:** Contribution property insurance policies
- **PBYSTAND:** Contribution social security insurance policies

policies. vari-



- **AWAPART:** Number of private third party insurance 1 -
- **AWABEDR:** Number of third party insurance (firms) ...
- **AWALAND:** Number of third party insurance (agricultu
- **APERSAUT:** Number of car policies
- **ABESAUT:** Number of delivery van policies
- **AMOTSCO:** Number of motorcycle/scooter policies
- **AVRAAUT:** Number of lorry policies
- **AAANHANG:** Number of trailer policies
- **ATRACTOR:** Number of tractor policies
- **AWERKT:** Number of agricultural machines policies
- **ABROM:** Number of moped policies
- **ALEVEN:** Number of life insurances
- **APERSONG:** Number of private accident insurance poli
- **AGEZONG:** Number of family accidents insurance polic
- **AWAOREG:** Number of disability insurance policies
- **ABRAND:** Number of fire policies
- **AZEILPL:** Number of surfboard policies
- **APLEZIER:** Number of boat policies
- **AFIETS:** Number of bicycle policies
- **AINBOED:** Number of property insurance policies
- **ABYSTAND:** Number of social security insurance polici

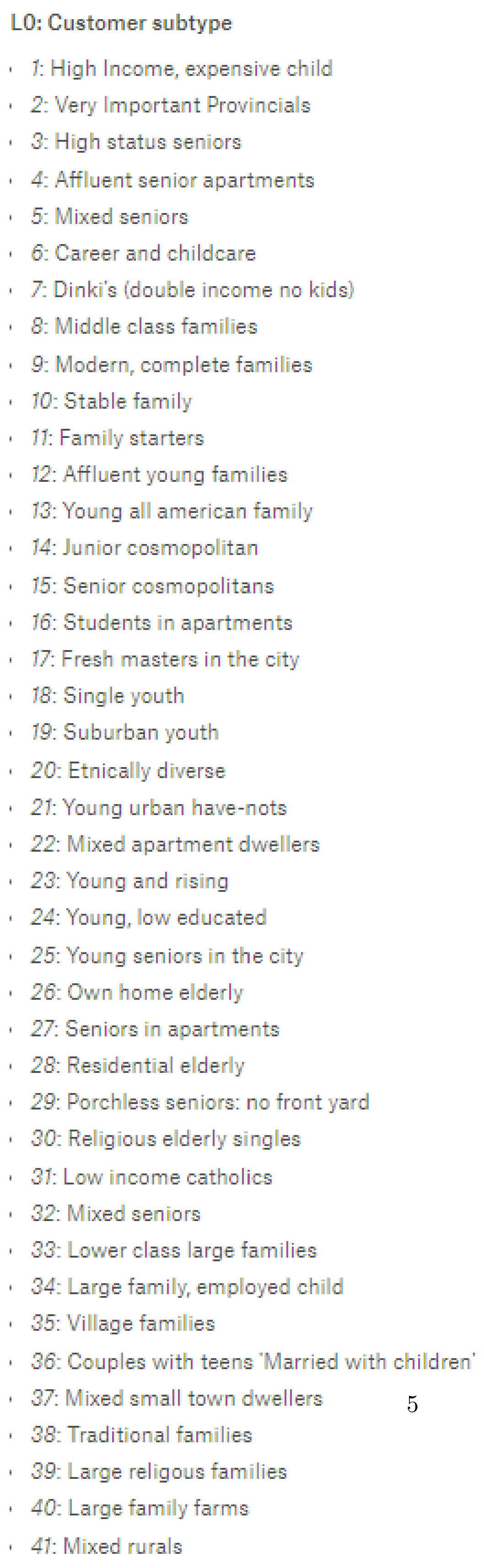
ables beggining with A refer to number of policies.

Going to check the factors of the dataset

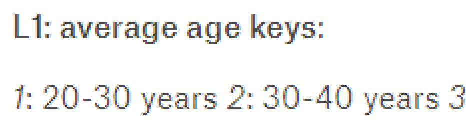
```r
#Names of columns
sapply(df,levels)
```

I have ommited the result of the above code, as it was far too large. Most of the columns in the current data are numeric values but are actually supposed to be factors. I will refactor these columns during pre-processing.

There are 4 keys that relate to this datset. A key for customer subtype:


L0: Customer subtype

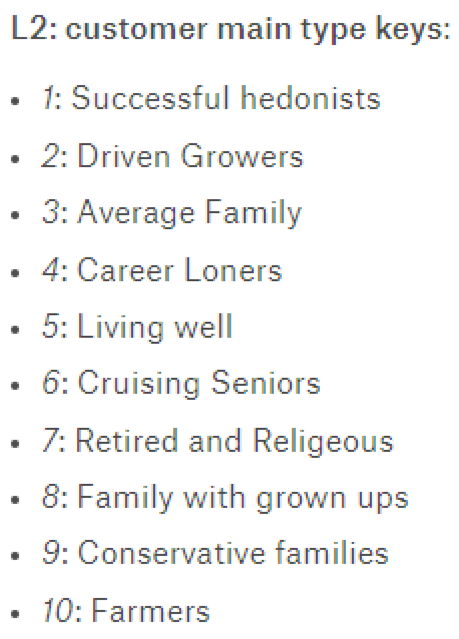
- *1*: High Income, expensive child
- *2*: Very Important Provincials
- *3*: High status seniors
- *4*: Affluent senior apartments
- *5*: Mixed seniors
- *6*: Career and childcare
- *7*: Dinki's (double income no kids)
- *8*: Middle class families
- *9*: Modern, complete families
- *10*: Stable family
- *11*: Family starters
- *12*: Affluent young families
- *13*: Young all american family
- *14*: Junior cosmopolitan
- *15*: Senior cosmopolitans
- *16*: Students in apartments
- *17*: Fresh masters in the city
- *18*: Single youth
- *19*: Suburban youth
- *20*: Etnically diverse
- *21*: Young urban have-nots
- *22*: Mixed apartment dwellers
- *23*: Young and rising
- *24*: Young, low educated
- *25*: Young seniors in the city
- *26*: Own home elderly
- *27*: Seniors in apartments
- *28*: Residential elderly
- *29*: Porchless seniors: no front yard
- *30*: Religious elderly singles
- *31*: Low income catholics
- *32*: Mixed seniors
- *33*: Lower class large families
- *34*: Large family, employed child
- *35*: Village families
- *36*: Couples with teens 'Married with children'
- *37*: Mixed small town dwellers
- *38*: Traditional families
- *39*: Large religous families
- *40*: Large family farms
- *41*: Mixed rurals


L1: average age keys:

1: 20-30 years *2*: 30-40 years *3*

A key for average age

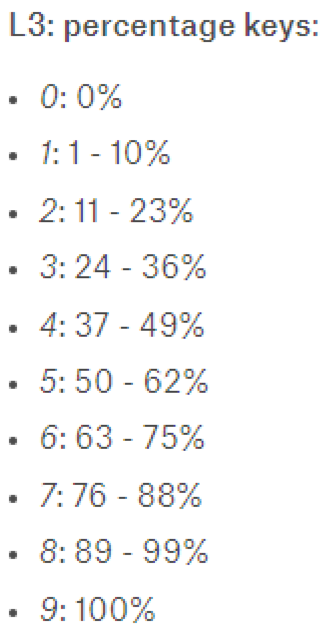
A key of customer main types



L2: customer main type keys:

- *1*: Successful hedonists
- *2*: Driven Growers
- *3*: Average Family
- *4*: Career Loners
- *5*: Living well
- *6*: Cruising Seniors
- *7*: Retired and Religeous
- *8*: Family with grown ups
- *9*: Conservative families
- *10*: Farmers

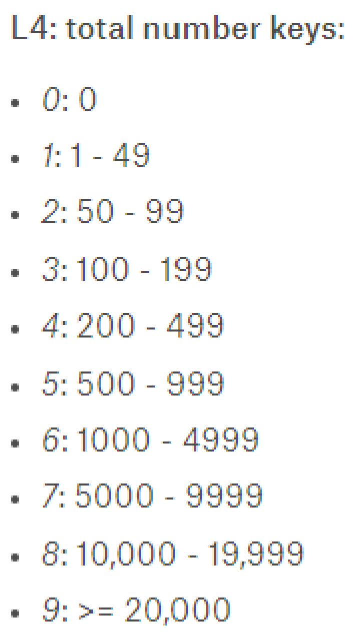
A key of percentage ranges



L3: percentage keys:

- *0*: 0%
- *1*: 1 - 10%
- *2*: 11 - 23%
- *3*: 24 - 36%
- *4*: 37 - 49%
- *5*: 50 - 62%
- *6*: 63 - 75%
- *7*: 76 - 88%
- *8*: 89 - 99%
- *9*: 100%

and a key of total number ranges



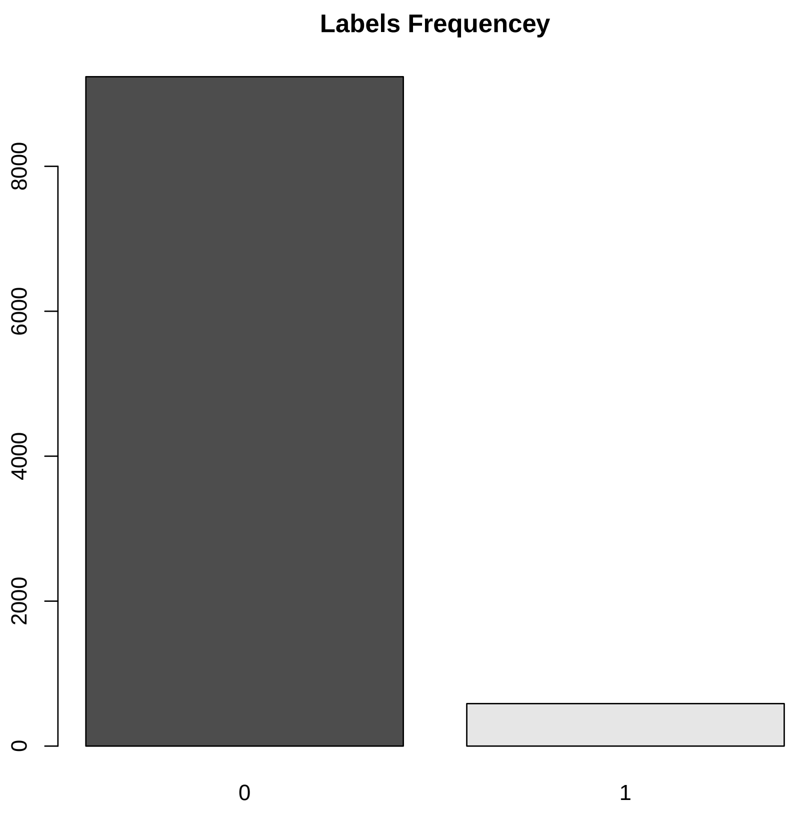
L4: total number keys:

- *0*: 0
- *1*: 1 - 49
- *2*: 50 - 99
- *3*: 100 - 199
- *4*: 200 - 499
- *5*: 500 - 999
- *6*: 1000 - 4999
- *7*: 5000 - 9999
- *8*: 10,000 - 19,999
- *9*: >= 20,000

I will use these keys to turn the appropriate columns into factors.

I will now take a look at the class label distribution

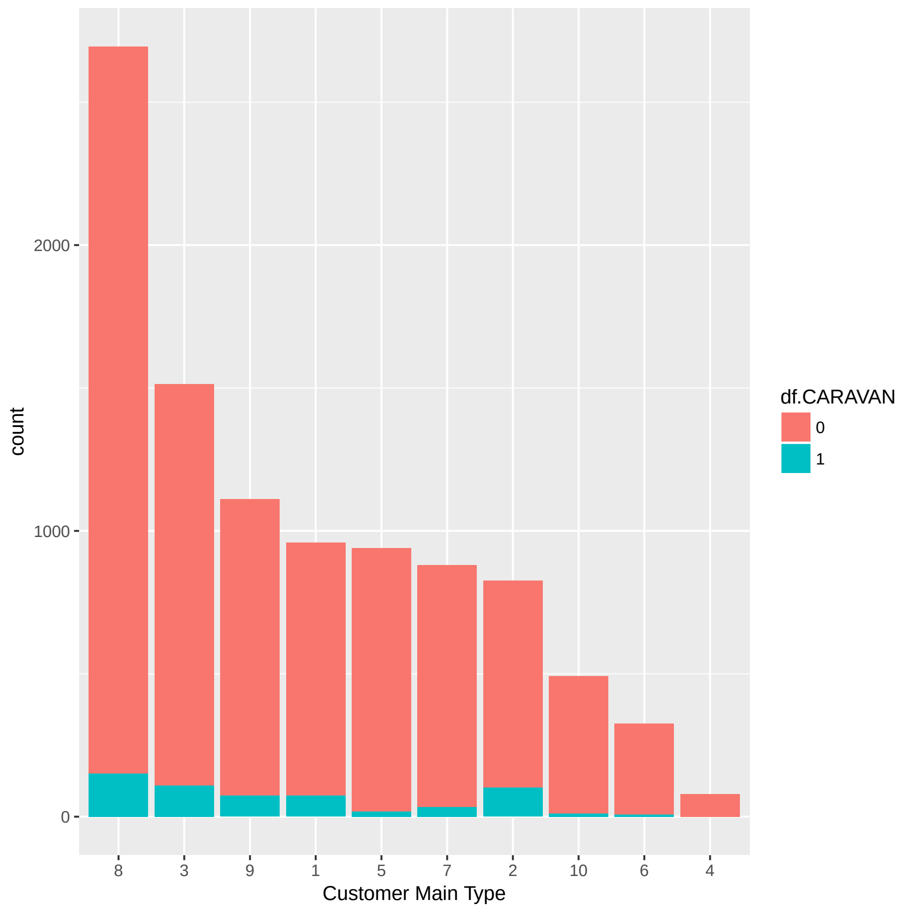
```r
#Class label Distribution Plot
classLabelFreq <- table(df$CARAVAN)
classLabelFreq

##
##    0    1
## 9236  586

barplot(classLabelFreq,col=gray.colors(2),main="Labels Frequencey")
```



There are 586 records that are likely to want caravan insurance. 9236 records that do not. Dataset has an imbalanced distribution in the class label. I will use a resampling technique during pre-processing to compensate for this.

Lets take a look at the disribution of Main Customer Type

```
library(ggplot2)

## Warning:  package 'ggplot2' was built under R version 3.4.2

#Temp refactor, will do properly later
custMainType <- data.frame(df$MOSHOOFD,df$CARAVAN)
custMainType$df.MOSHOOFD <- as.factor(custMainType$df.MOSHOOFD)
custMainType$df.CARAVAN <- as.factor(custMainType$df.CARAVAN)
#Plot of Customer Main Type
ggplot(custMainType,aes(x=reorder(df.MOSHOOFD,df.MOSHOOFD,function(x)-length(x)),fill=df.CA
```



Most frequent Main Customer Type is 8:Family with Grown Ups, 2nd Most frequent is 3:Average Family and the 3rd most frequent is 9:conservative families.

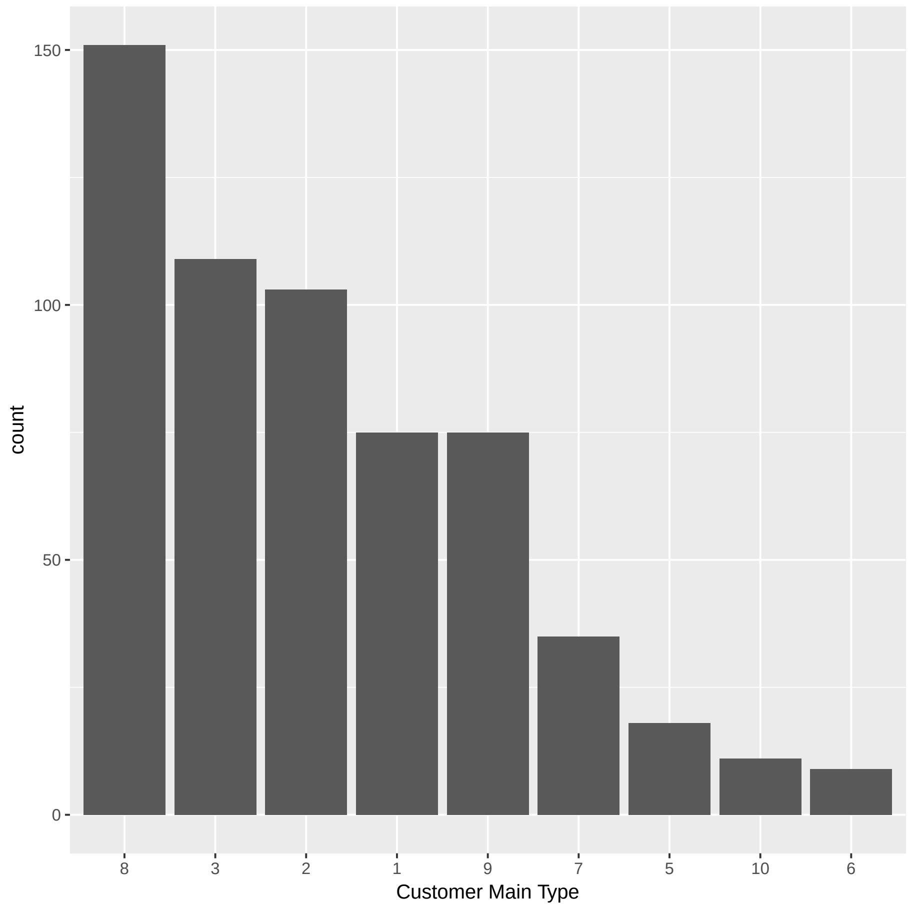
The least frequent type is 4:Career Loners ,2nd least frequent is 6:Cruising Seniors and the 3rd least frequent is 10:Farmers.

Comparing to where CARAVAN is TRUE, you can see that the two most frequent main types are the same as the whole dataset. Customer Main Type 2: Driven Growers seems to be a bit more prominent in the rows where CARAVAN is true. You can also see that there are no instances of group 4: Career Loners in the rows where CARAVAN is true.

Lets take a look at the rows where CARAVAN is TRUE:

```r
library(ggplot2)
#Wants caravan
wantsCaravan <- df[df$CARAVAN==1,]
wantsCaravan$MOSHOOFD <- as.factor(wantsCaravan$MOSHOOFD)
wantsCaravan$MOSTYPE <- as.factor(wantsCaravan$MOSTYPE)
```

```
#Plot of Customer Main Type where wants caravan
ggplot(wantsCaravan,aes(x=reorder(MOSHOOFD,MOSHOOFD,function(x)-length(x)))) + geom_bar() +
```



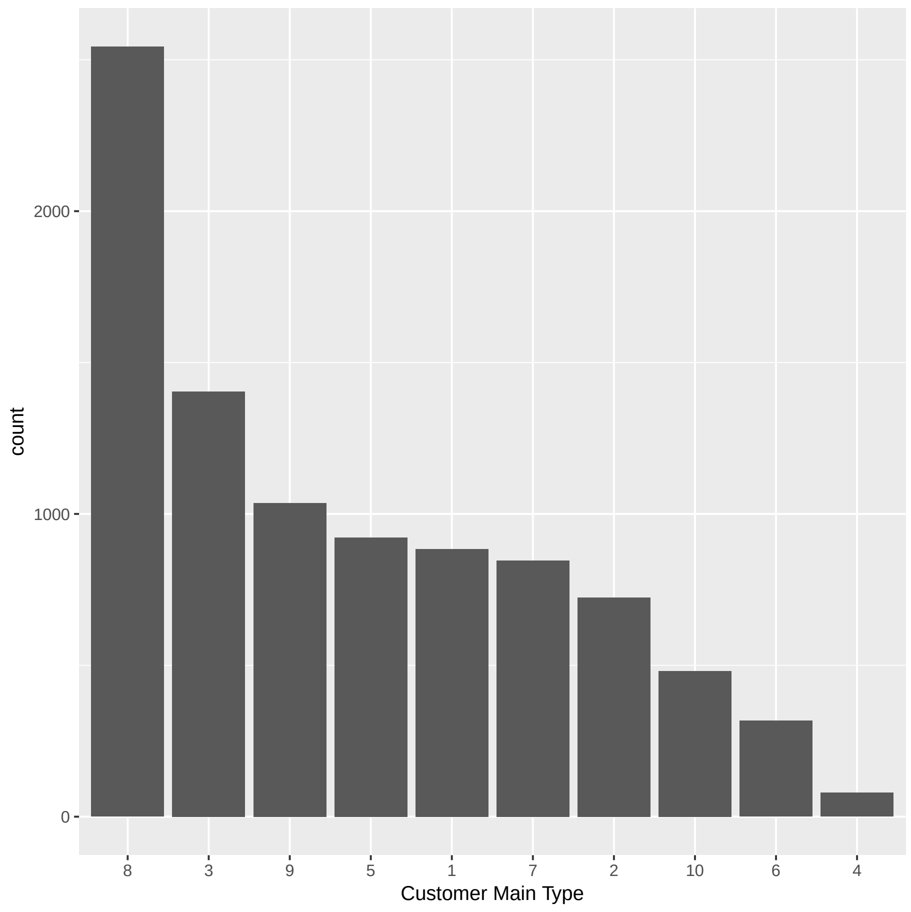
```
#Max and Min
mainCustType = table(wantsCaravan$MOSHOOFD)
names(which.max(mainCustType))

## [1] "8"

names(which.min(mainCustType))

## [1] "6"
```

The top 3 Main Customer Types are 8:Family with Grown Ups, 3: Average Family and 2:Driven Growers. As stated before the first 2 types are the same as the dataset as a whole. Type 2:Driven Growers has now overtaken 9:conservative families. 9:Conservative Families is joint 4th. Interesting that category 1: Successful Hedonists has the same number of occurances as 9:Conservaite Families. Hedonists are in the pursuit of pleasure. There is perhaps a connection there between the idea of traveling by caravan and a pursuit of happiness.

The 3 least frequent Main Custom Types 6:Cruising Seniors ,10:Farmers and 5:Living Well. As stated before there are no instances of 4:Career Loners when CARAVAN is TRUE. There is too small a number of instances of 4:Career Loners to really say there is correlation but it is possible.

Lets take a look at rows where CARAVAN is FALSE:

```r
library(ggplot2)
#Not want caravan
notWantCaravan <- df[df$CARAVAN==0,]
notWantCaravan$MOSHOOFD <- as.factor(notWantCaravan$MOSHOOFD)
notWantCaravan$MOSTYPE <- as.factor(notWantCaravan$MOSTYPE)

#Plot of Customer Main Type
ggplot(notWantCaravan,aes(x=reorder(MOSHOOFD,MOSHOOFD,function(x)-length(x)))) + geom_bar()
```

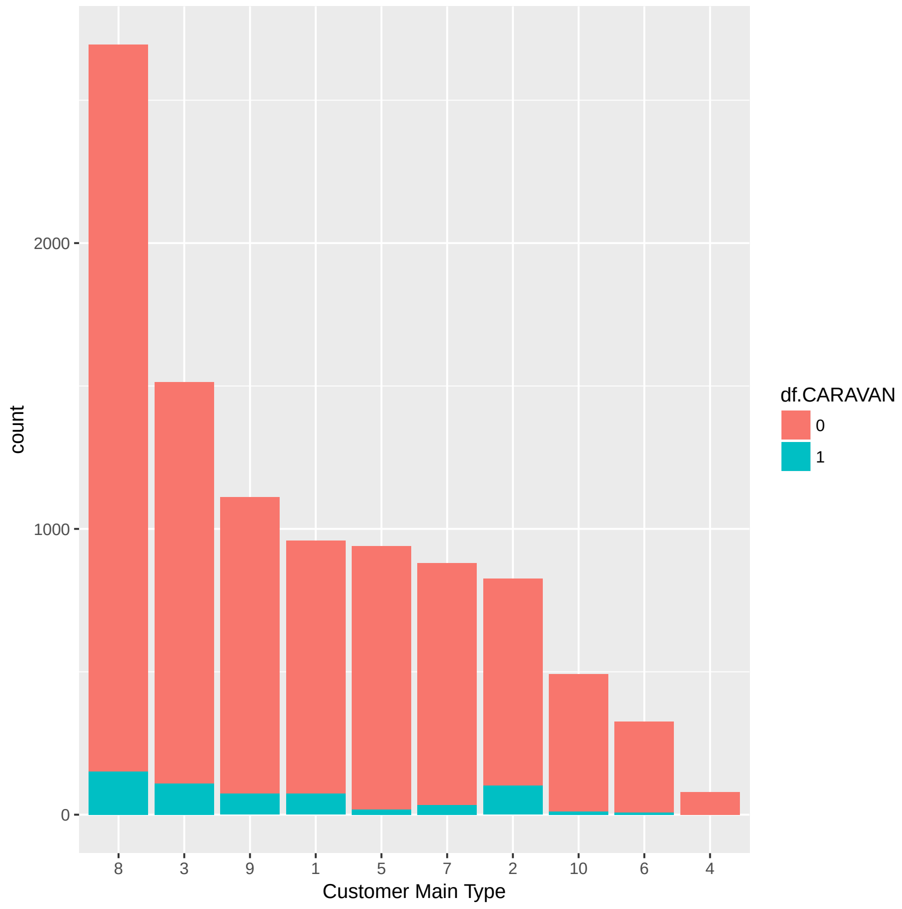


```
#Max and Min
mainCustType = table(notWantCaravan$MOSHOOFD)
names(which.max(mainCustType))

## [1] "8"

names(which.min(mainCustType))

## [1] "4"
```

Now going to take a look at Customer Subtype

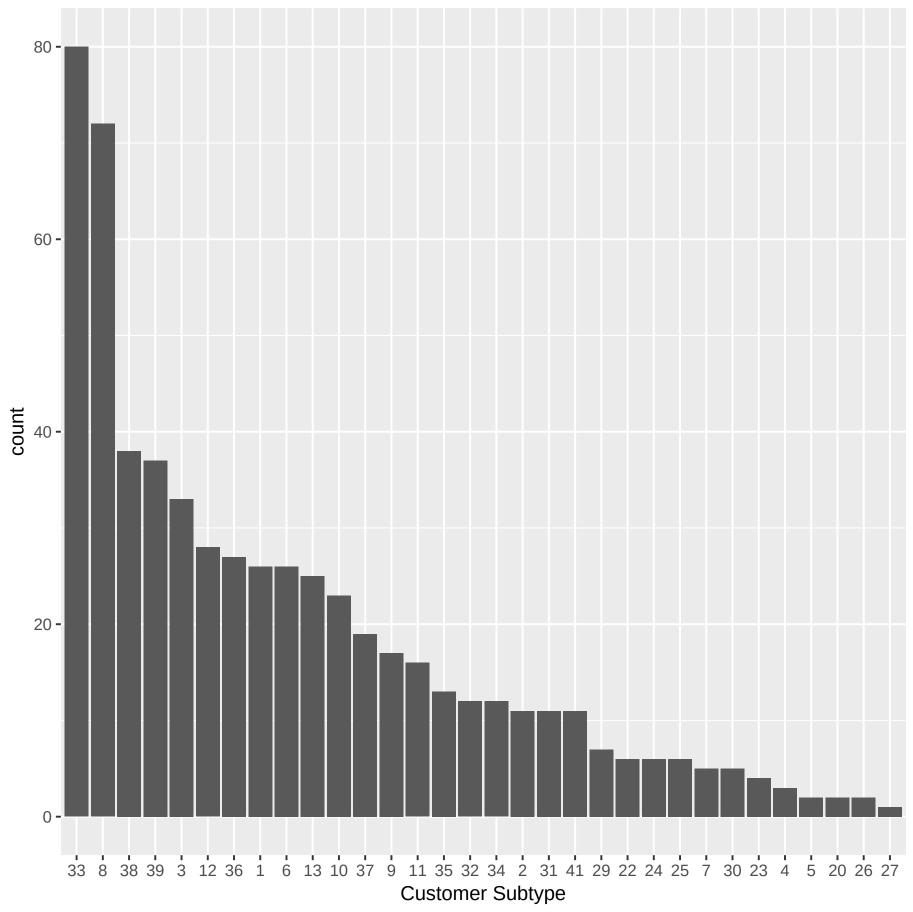
```
library(ggplot2)
#Temp refactor, will do properly later
custMainType <- data.frame(df$MOSHOOFD,df$CARAVAN)
custMainType$df.MOSHOOFD <- as.factor(custMainType$df.MOSHOOFD)
custMainType$df.CARAVAN <- as.factor(custMainType$df.CARAVAN)
```

```r
#Plot of Customer Main Type
ggplot(custMainType,aes(x=reorder(df.MOSHOOFD,df.MOSHOOFD,function(x)-length(x)),fill=df.CAR
```



Take a look at customersubtype when CARAVAN is TRUE

```r
#Plot of Customer Subtype where wants caravan
ggplot(wantsCaravan,aes(x=reorder(MOSTYPE,MOSTYPE,function(x)-length(x)))) + geom_bar() + la
```

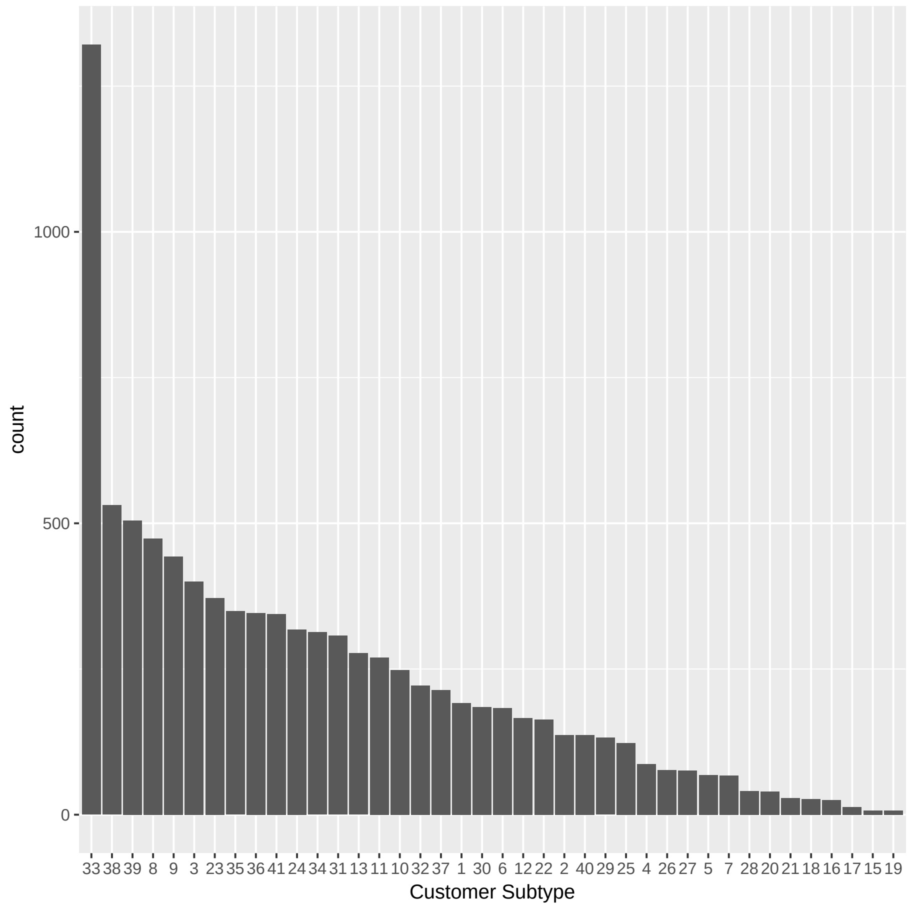


```
#Max and Min
subCustType = table(wantsCaravan$MOSTYPE)
names(which.max(subCustType))

## [1] "33"

names(which.min(subCustType))

## [1] "27"
```

Least frequent Main Custom Type is 6:Cruising Seniors.The Main Customer Subtype is 27:Seniors in apartments.

The 3 most frequent customer sub types are 33:Lower class large families, 8:Middle class families, 38:Traditional Families . This supports the theory that post codes containing most large families are most likey to purchase caravan insurance policies. The 3 least frequent customer sub types 27:Seniors in apartments, 26:Own home elderly and 20:Ethnically diverse. This supports the theory that seniors are the least likey group to purchase caravan insurance. Again I will need to compare to the rest of the data to confirm this.

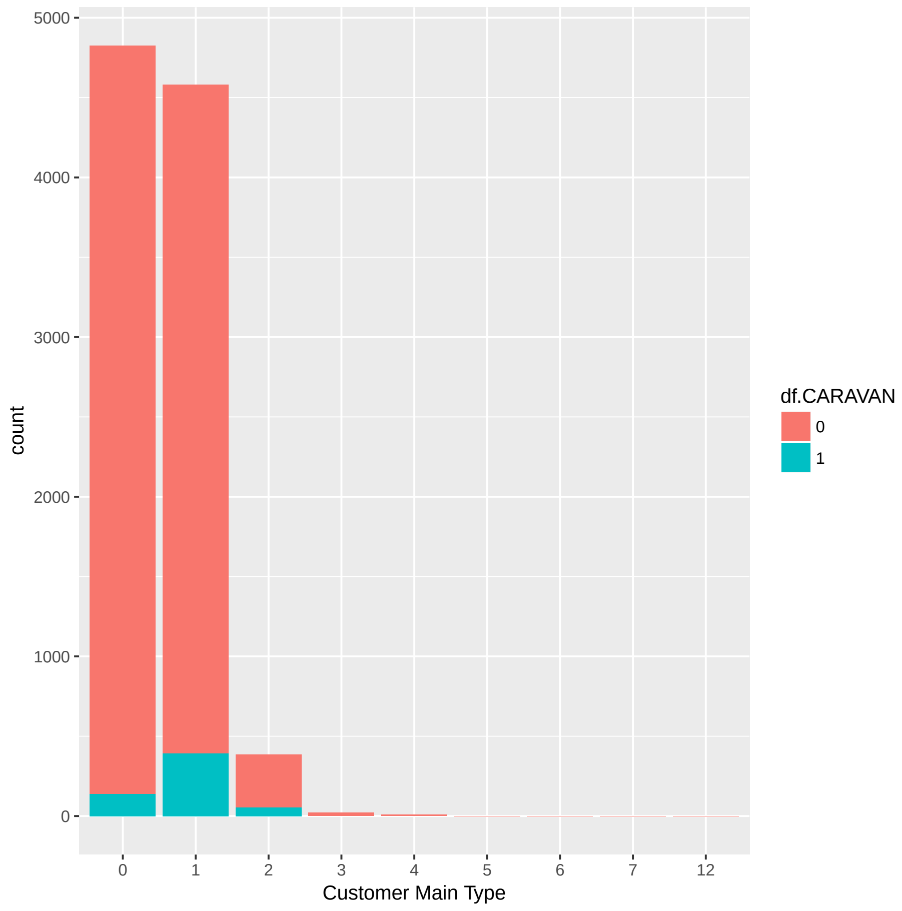
```
#Plot of Customer Subtype
ggplot(notWantCaravan,aes(x=reorder(MOSTYPE,MOSTYPE,function(x)-length(x)))) + geom_bar() +
```



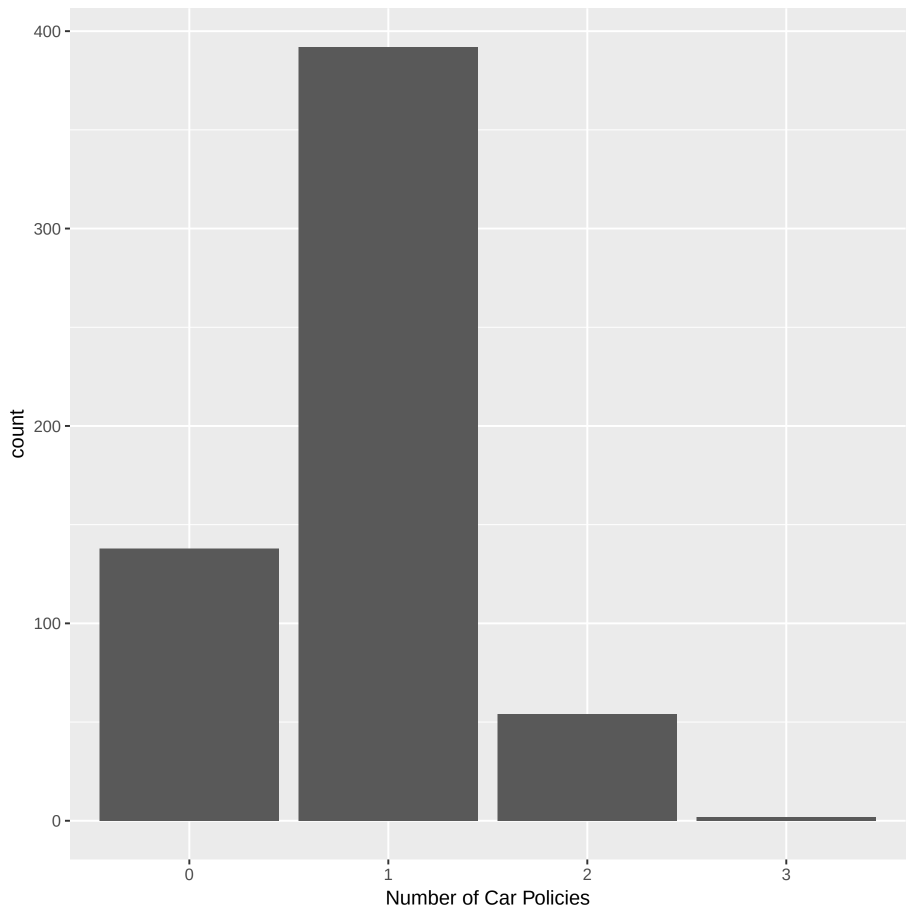
```
#Max and Min
subCustType = table(notWantCaravan$MOSTYPE)
names(which.max(subCustType))

## [1] "33"

names(which.min(subCustType))

## [1] "15"
```

I am not going to take a look at the APERSAUT column, which is the Number of Car Policies column. This is a factor, where each level equates to a range of values. The ranges are defined in the total number key.

```r
#Temp refactor
numberOfCarPolicies <- data.frame(df$APERSAUT,df$CARAVAN)
numberOfCarPolicies$df.APERSAUT <- as.factor(numberOfCarPolicies$df.APERSAUT)
numberOfCarPolicies$df.CARAVAN <- as.factor(numberOfCarPolicies$df.CARAVAN)
#Plot of APERSAUT
ggplot(numberOfCarPolicies,aes(x=reorder(df.APERSAUT,df.APERSAUT,function(x)-length(x)),fill
```



```r
library(ggplot2)
#Comparing levels of APERSAUT(Number of car policies)
wantsCaravan$APERSAUT <- as.factor(wantsCaravan$APERSAUT)
ggplot(wantsCaravan,aes(x=APERSAUT)) + geom_bar() + labs(x="Number of Car Policies")
```

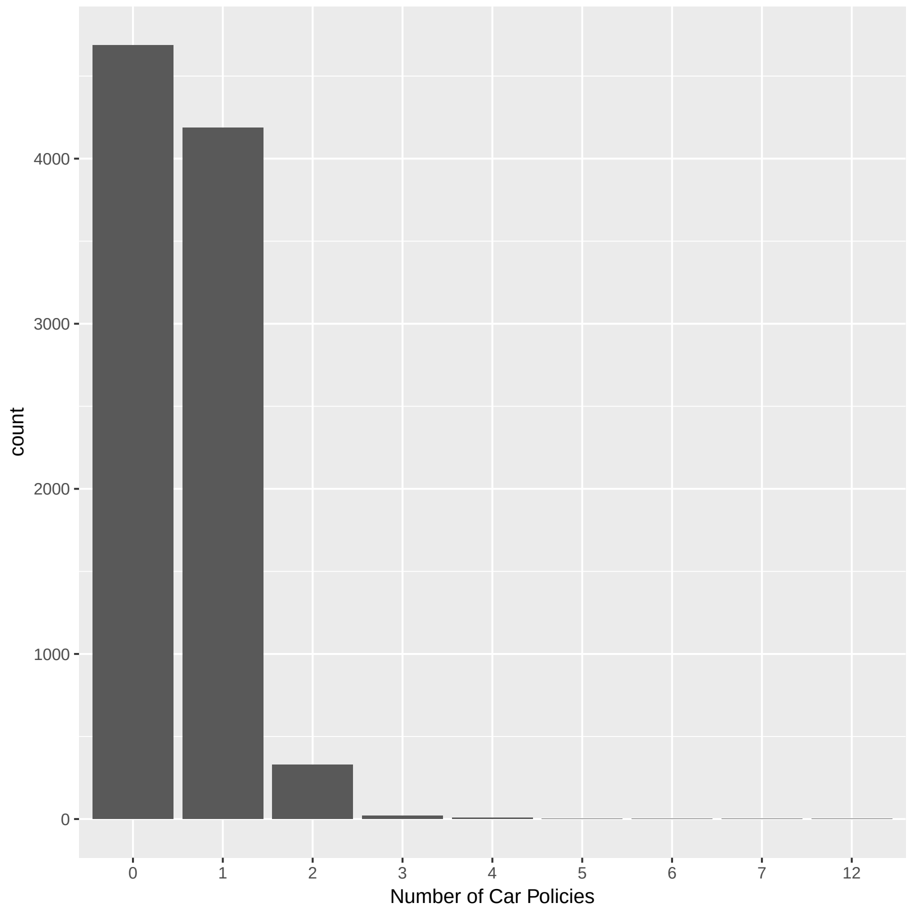


This factor is supposed to have 9 levels in total, but this column factored only contains the lowest 4 possible factors, 0,1,2 and 3. 1 is the most frequent. In this case 1 represents the range of values 1-49. 0 represents 0, 2 represents the range of values 50-99 and 3 represents the range of values 100-199. This would suggest that post codes that contain a number of cars in the range of 1-49 are most likey to purchase caravan insurance. There are only 2 rows in the dataset that contain 3 for this column so this could be considered an outlier. Based on the graph the range of values for number of cars could be 0-99, or 0-199 if you include the two rows that have the value 3. There isn't enough data here to be sure.

I was originally confused by this result. I had assumed that I would find the opposite, that areas with large numbers of cars would likey need caravan insurance as I assumed you would need a car to tow a caravan. After further study of the information about the dataset, in this case CARAVAN actually refers to mobile homes. This would suggest the data is based on american post codes (or area codes) and that 'caravan' refers to a self driving vehicle that you can sleep in rather than a traditional british caravan.

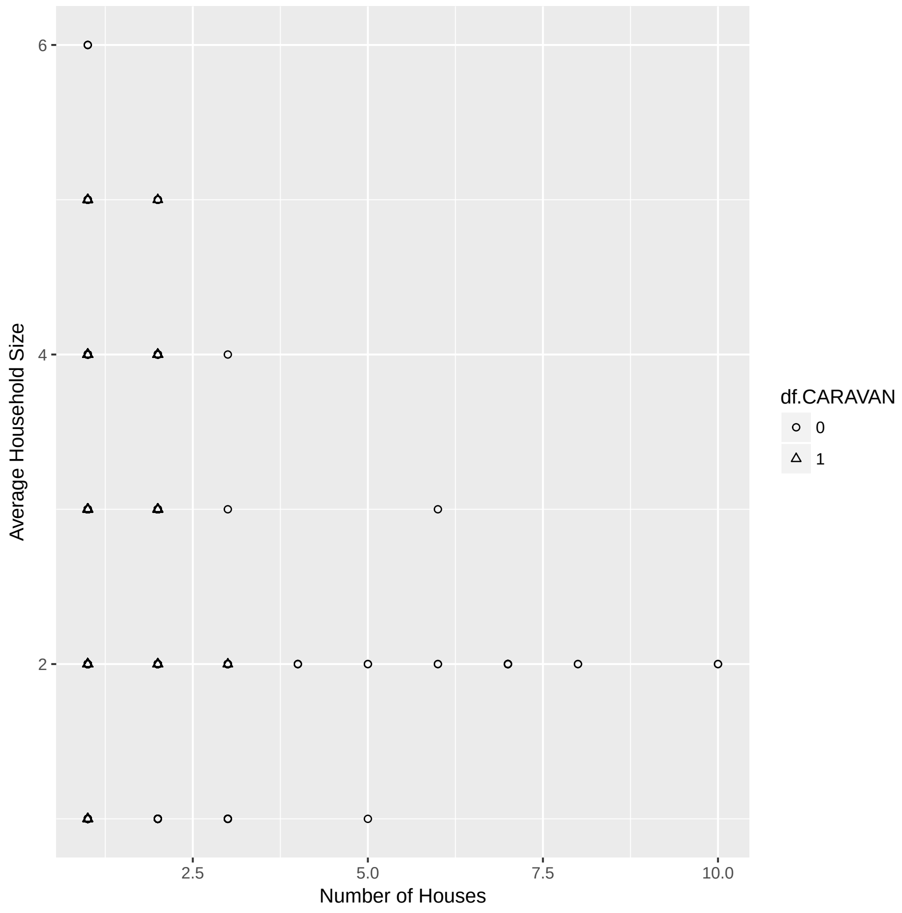
Going to take a look at APERSAUT when CARAVAN is FALSE

```
#Not Want Caravan
notWantCaravan$APERSAUT <- as.factor(notWantCaravan$APERSAUT)
ggplot(notWantCaravan,aes(x=APERSAUT)) + geom_bar() + labs(x="Number of Car Policies")
```



I am not going to take a look at two columns. MAANTHUI(Number of houses) and MGEMOMV(Avg size of household). MAATHUI is in the range of 1-10 and MGEMOMV is in the range of 1-6. They are the only two numeric values in the dataset the rest are factors. I will look at them together to see if there is any corelation. I will use ggplot2 again to make a scatter plot.

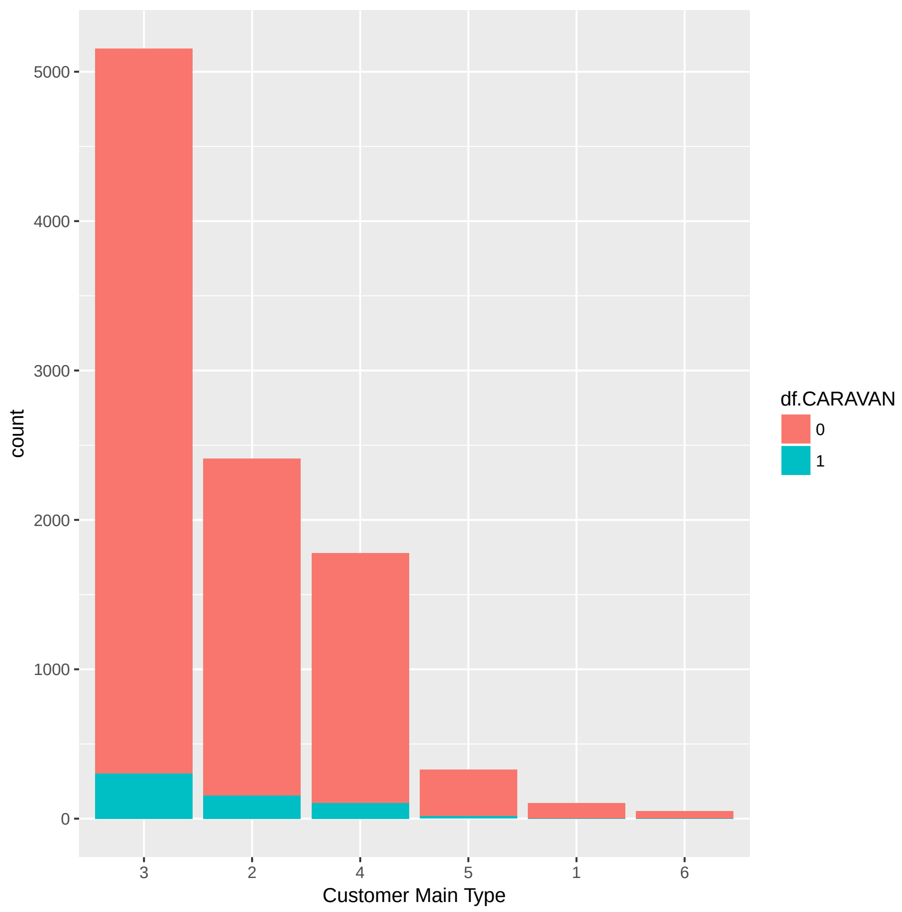
```
#Number of Houses and Avg size of household
houseData<-data.frame(df$MAANTHUI,df$MGEMOMV,df$CARAVAN)
houseData$df.CARAVAN<-as.factor(houseData$df.CARAVAN)
#ScatterPlot of both
ggplot(houseData,aes(x=df.MAANTHUI,y=df.MGEMOMV)) + geom_point(aes(shape=df.CARAVAN)) + scal
```



Taking a look at the results, average size of household decreases as the number of houses increases which makes sense. Looking at the points where CARAVAN is TRUE, There are no points when the number of houses is greater than 3. There are also no points when the average house size is greater than 5. This shows a potential connection between number of houses and CARAVAN. If I had to remove one of the two variables I would remove Average house size as I think number of houses has a greater corelation to CARAVAN equaling TRUE.

I will now take a look at the average age variable, MGEMLEEF.

```
#Average Age
library(ggplot2)
averageAge <- data.frame(df$MGEMLEEF,df$CARAVAN)
averageAge$df.MGEMLEEF <- as.factor(averageAge$df.MGEMLEEF)
averageAge$df.CARAVAN <- as.factor(averageAge$df.CARAVAN)
#Plot of Average Age
ggplot(averageAge,aes(x=reorder(df.MGEMLEEF,df.MGEMLEEF,function(x)-length(x)),fill=df.CARAV
```



Average age is a factor, where each level is a range of ages. Lowest age range is 1:20-30 years, highest is 6:70-80 years. Looking at the graph, levels 3:40-50, 2:30-40 and 4:50-60 are the top 3 most frequent values. The extremes of 1 and 6 are the two lowest and do not have very many occurances. Looking at instances where CARAVAN is TRUE, They are in the same order as the whole dataset. Except there are no instances where CARAVAN is TRUE that contain 1 and 6 for average age.There might be a trend that areas were the average age is 1:20-30 or 6:70-80 would not buy caravan insurance. There isn't enough data to be sure and as the data where CARAVAN is true follows a similar trend to the data as a whole its unlikely there is a corelation between average age and CARAVAN.

The first column in the data set is ORIGIN. It has two values

```
#Class label Distribution Plot
levels(df$ORIGIN)
```

```
## [1] "test"  "train"
```

This is the original source of the row from the challenge. The rows are already split into a train set and test set. I will remove this column later during pre-processing, as I plan to resample the data and split the data into train and test sets myself.

1.4 Pre-processing

First I will refactor all the appropriate columns

```
#Refactor
#Customer Subtype
df$MOSTYPE = factor(df$MOSTYPE,levels=c(1:41),labels=c("High Income, expensive child","Very

#Average Age
df$MGEMLEEF = factor(df$MGEMLEEF,levels=c(1:6),labels=c("20-30 years","30-40 years","40-50 y

#Custom Main Type
df$MOSHOOFD = factor(df$MOSHOOFD,levels=(1:10),labels=c("Successful hedonists","Driven Growe

#Percentages
for (i in which(colnames(df)=="MGODRK"):which(colnames(df)=="MKOOPKLA")){
  df[,i] = factor(df[,i],levels=c(0:9),labels=c("0%","1-10%","11-23%","24-36%","37-49%","50-
}

#Number of
for (i in which(colnames(df)=="PWAPART"):which(colnames(df)=="ABYSTAND")){
  df[,i] = factor(df[,i],levels=c(0:9),labels=c("0","1-49","50-99","100-199","200-499","500-
}

#Set class label as factor
df$CARAVAN <- factor(df$CARAVAN,levels=c(0:1))
```

I will now remove the column ORIGIN. The column origin is a factor with two values, TRAIN and TEST. It is the original set that the data came from in the challenge that this dataset was creaeted for. TRAIN data was given to contestants, and TEST was the data used to test the submitted models. As I am going to be resampling the data and partitioning my own train and test sets this column is useless so I will remove it.

```
#Get rid of ORIGIN
df$ORIGIN <- NULL
```

I will now remove any rows with missing values

```
#Remove NA's
df<-df[complete.cases(df),]
```

I will now resample the dataset to balance the distribution of the class label. I will do this using a function called ovun.sample from the ROSE library.

```
#Resample Train (Oversampling)
library(ROSE)

## Warning:  package 'ROSE' was built under R version 3.4.3
## Loaded ROSE 0.0-3

df<-ovun.sample(CARAVAN~.,data=df,method="over")$data
```

2 Modelling/Classification

I will now create my random forest model. I will repeat this 10 times with different train and test sets generated by createDataPartician. I will then display the accuracies of each model and the average.

```
memory.limit()

## [1] 8046

memory.limit(size=100000)

## [1] 1e+05

library(caret)

## Warning:  package 'caret' was built under R version 3.4.2
## Loading required package:  lattice

library(randomForest)

## Warning:  package 'randomForest' was built under R version 3.4.2
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package:  'randomForest'
## The following object is masked from 'package:ggplot2':
##
##    margin
```

```
calculateAvgAccuracy <- function(data,ntrees=100,nodeSize=1){
  results<-data.frame(Accuracy=as.numeric())
  for (i in 1:10){
    part<-createDataPartition(y=data$CARAVAN,p=0.7,list=FALSE)
    trainData<-data[part,]
    testData<-data[-part,]
    model<-randomForest(trainData[,-ncol(trainData)],trainData[,ncol(trainData)],xtest=testD
    preds<-levels(trainData[,ncol(trainData)])[model$test$predicted]
    auc<-(sum(preds==testData[,ncol(testData)])/nrow(testData))*100
    results<-rbind(results,data.frame(Accuracy=auc))
  }
  return(results)
}
#Get Acuracies
result<-calculateAvgAccuracy(df)
#Show accuracies
result

##    Accuracy
## 1  57.34417
## 2  64.29991
## 3  57.92231
## 4  55.57362
## 5  57.09124
## 6  56.53117
## 7  55.50136
## 8  56.69377
## 9  56.83830
## 10 57.48871

#Get average
avg<-sum(result)/nrow(result)
paste("Average Accuracy: ",avg,"%",sep="")

## [1] "Average Accuracy: 57.5284552845528%"
```

During initial testing, accuracies where around 55-57 range. This isn't great and could be improved. Mode data where CARAVAN=TRUE is really needed. Perhaps different resampling methods like bootstraping might generate better results.

3 Improving Performance

I will start by tring to fine tune the ntrees attribute of my model by testing my model with values between 100 and 1000 for ntree.

```
#Create train and test sets
part<-createDataPartition(y=df$CARAVAN,p=0.7,list=FALSE)
train<-df[part,]
test<-df[-part,]
#Tweak number of trees
testNTrees <- function(trainData,testData){
  ntrees<-100
  results<-data.frame(NTrees=as.numeric(),Accuracy=as.numeric())
  for (i in 1:10){
    model<-randomForest(trainData[,-ncol(trainData)],trainData[,ncol(trainData)],xtest=testD
    preds<-levels(trainData[,ncol(trainData)])[model$test$predicted]
    auc<-(sum(preds==testData[,ncol(testData)])/nrow(testData))*100
    results<-rbind(results,data.frame(NTrees=ntrees,Accuracy=auc))
    ntrees <-ntrees + 100
  }
  return(results)
}
#Getting accuracies for ntrees between 100 and 1000 (inclusive)
ntreeResults<-testNTrees(train,test)
#Get row with max accuracy
maxRow<-which.max(ntreeResults$Accuracy)
ntrees<-ntreeResults$NTrees[maxRow]
#Num trees
ntrees
```

```
## [1] 200
```

```
#Get Acuracies
result<-calculateAvgAccuracy(df,ntrees=ntrees)
#Show accuracies
result
```

```
##    Accuracy
## 1  56.65763
## 2  55.35682
## 3  56.80217
## 4  54.61608
## 5  56.82023
## 6  56.35050
## 7  55.15808
## 8  56.51310
## 9  55.82656
## 10 56.00723
```

```
#Get average
avg<-sum(result)/nrow(result)
paste("Average Accuracy: ",avg,"%",sep="")
```

```
## [1] "Average Accuracy: 56.0108401084011%"
```

I will not try and fine tune the randomForest function variable nodesize.

```
#Tweek Nodesize
testNodeSize <- function(trainData,testData){
  nsize<-0
  results<-data.frame(Nodesize=as.numeric(),Accuracy=as.numeric())
  for (i in 1:(nrow(trainData)/100)){
    model<-randomForest(trainData[,-ncol(trainData)],trainData[,ncol(trainData)],xtest=test
    preds<-levels(trainData[,ncol(trainData)])[model$test$predicted]
    auc<-(sum(preds==testData[,ncol(testData)])/nrow(testData))*100
    results<-rbind(results,data.frame(Nodesize=nsize,Accuracy=auc))
    nsize<-nsize+1
  }
  return(results)
}
#Getting accuracies for ntrees between 100 and 1000 (inclusive)
nodeSizeResults<-testNodeSize(train,test)
#Get row with max accuracy
maxRow<-which.max(nodeSizeResults$Accuracy)
nodeSize<-nodeSizeResults$Nodesize[maxRow]
#Get Acuracies
result<-calculateAvgAccuracy(df,ntrees=ntrees,nodeSize=nodeSize)
#Show accuracies
result

##    Accuracy
## 1  56.51310
## 2  60.92141
## 3  58.35592
## 4  55.84463
## 5  56.27823
## 6  53.47787
## 7  60.95754
## 8  61.24661
## 9  57.00090
## 10 55.50136

#Get average
avg<-sum(result)/nrow(result)
paste("Average Accuracy: ",avg,"%",sep="")

## [1] "Average Accuracy: 57.609756097561%"
```

To improve the performance of my model I will take a look at the variable importance in the random forrest model.

I will use the importance function, from the randomForest library, to create plots of mean decrease in accuracy and mean decrease in Gini.

```
#Mean Decrease in accuracy
meanDecreaseAccuracy<-importance(model,type=1)

## Error in importance(model, type = 1):  object 'model' not found

#Order highest to lowest
meanDecreaseAccuracy<-meanDecreaseAccuracy[order(meanDecreaseAccuracy),,drop=FALSE]

## Error in eval(expr, envir, enclos):  object 'meanDecreaseAccuracy'
not found

meanDecreaseAccuracy

## Error in eval(expr, envir, enclos):  object 'meanDecreaseAccuracy'
not found

#Plot
varImpPlot(model,type=1)

## Error in inherits(x, "randomForest"):  object 'model' not found

#Mean decrease in node impurity
meanDecreaseGini<-importance(model,type=2)

## Error in importance(model, type = 2):  object 'model' not found

#Order highest to lowest
meanDecreaseGini<-meanDecreaseGini[order(-meanDecreaseGini),,drop=FALSE]

## Error in eval(expr, envir, enclos):  object 'meanDecreaseGini' not
found

meanDecreaseGini

## Error in eval(expr, envir, enclos):  object 'meanDecreaseGini' not
found

#Plot
varImpPlot(model,type=2)

## Error in varImpPlot(model, type = 2):  object 'model' not found
```

I will remove any columns that have a negative mean decrease in accuracy value(if any).

```
#Get negative or 0 MDA
cols<-rownames(meanDecreaseAccuracy[meanDecreaseAccuracy<=0,,drop=FALSE])
```

```
## Error in rownames(meanDecreaseAccuracy[meanDecreaseAccuracy <= 0,
, drop = FALSE]): object 'meanDecreaseAccuracy' not found

cols

## Error in eval(expr, envir, enclos):  object 'cols' not found

#Remove cols
if (!is.null(cols)){
  df<-df[-cols]
}

## Error in eval(expr, envir, enclos):  object 'cols' not found
```

I will now re-test the accuracy of my model

```
#Calculate accuracies, with new dataset minus cols and new ntrees from testNTrees function
result<-calculateAvgAccuracy(df,ntrees=ntrees,nodeSize=nodeSize)
#Show accuracies
result

##    Accuracy
## 1  58.77145
## 2  63.34237
## 3  61.73442
## 4  58.50045
## 5  58.01265
## 6  58.24752
## 7  58.22945
## 8  59.22313
## 9  57.72358
## 10 60.39747

#Get average
avg<-sum(result)/nrow(result)
paste("Average Accuracy: ",avg,"%",sep="")

## [1] "Average Accuracy: 59.4182475158085%"
```