Code ▾

Castellano_Lab06

Download the Allstate Claims Severity training data file from kaggle.

Hide

```
data = read.csv( "C:/Users/Castellano/Documents/Spring2020/CS636/All State Severity/train.csv")
```

Extracting numerical variables

Hide

```


#Applying the is.numeric function to the columns of "data", and then passing the output to the unlist function and selecting the resulting logical true columns from the original dataset, we get:
train_x0 <- data[unlist(lapply(data, is.numeric))]

str(train_x0)
```

```
'data.frame':	188318 obs. of  16 variables:
 $ id    : int  1 2 5 10 11 13 14 20 23 24 ...
 $ cont1 : num  0.726 0.331 0.262 0.322 0.273 ...
 $ cont2 : num  0.246 0.737 0.358 0.556 0.16 ...
 $ cont3 : num  0.188 0.593 0.484 0.528 0.528 ...
 $ cont4 : num  0.79 0.614 0.237 0.374 0.473 ...
 $ cont5 : num  0.31 0.886 0.397 0.422 0.704 ...
 $ cont6 : num  0.718 0.439 0.29 0.441 0.178 ...
 $ cont7 : num  0.335 0.437 0.316 0.391 0.247 ...
 $ cont8 : num  0.303 0.601 0.273 0.318 0.246 ...
 $ cont9 : num  0.671 0.351 0.261 0.321 0.221 ...
 $ cont10: num  0.835 0.439 0.324 0.445 0.212 ...
 $ cont11: num  0.57 0.338 0.381 0.328 0.205 ...
 $ cont12: num  0.595 0.366 0.373 0.322 0.202 ...
 $ cont13: num  0.822 0.611 0.196 0.605 0.246 ...
 $ cont14: num  0.715 0.304 0.774 0.603 0.433 ...
 $ loss  : num  2213 1284 3005 940 2764 ...
```

Data preparation

Hide

```
# Remove ID and loss features since they do not belong to the training data.
loss_1 <- vector()
loss_1 <- train_x0$loss # Save to use as target later.

# Remove features

train_x <- train_x0[,c(-1,-length(train_x0))] # Removes from all rows, first and last column (ID, loss)
```

1. Hierarchical Clustering

Do hierarchical clustering of the samples using the numeric features, and retrieve the two largest clusters. Then for each of these two clusters, find out how many samples with loss > the median loss; and how many with loss < the median loss.

Hide

```
set.seed(1) # Make runs reproducible
id = sample(1:dim(train_x)[1],10000) # Sample 10000 datapoints from all the rows of training data.
```

Hide

```
sample_x <- train_x[id,] # Creates a sampled dataset
str(sample_x)
```

```
'data.frame':	10000 obs. of  14 variables:
 $ cont1 : num  0.476 0.476 0.324 0.233 0.635 ...
 $ cont2 : num  0.299 0.621 0.489 0.682 0.489 ...
 $ cont3 : num  0.441 0.693 0.593 0.528 0.337 ...
 $ cont4 : num  0.534 0.565 0.284 0.237 0.895 ...
 $ cont5 : num  0.783 0.789 0.281 0.482 0.846 ...
 $ cont6 : num  0.326 0.373 0.38 0.455 0.456 ...
 $ cont7 : num  0.369 0.356 0.644 0.339 0.41 ...
 $ cont8 : num  0.361 0.361 0.308 0.308 0.361 ...
 $ cont9 : num  0.398 0.444 0.431 0.329 0.469 ...
 $ cont10: num  0.396 0.45 0.439 0.396 0.517 ...
 $ cont11: num  0.298 0.706 0.512 0.352 0.492 ...
 $ cont12: num  0.292 0.692 0.5 0.388 0.485 ...
 $ cont13: num  0.339 0.339 0.282 0.605 0.364 ...
 $ cont14: num  0.84 0.381 0.356 0.259 0.246 ...
```

Hide

```
loss_x = train_x0[id,15] # Selects the corresponding targets of the sampled data set f
or training
str(loss_x)
```

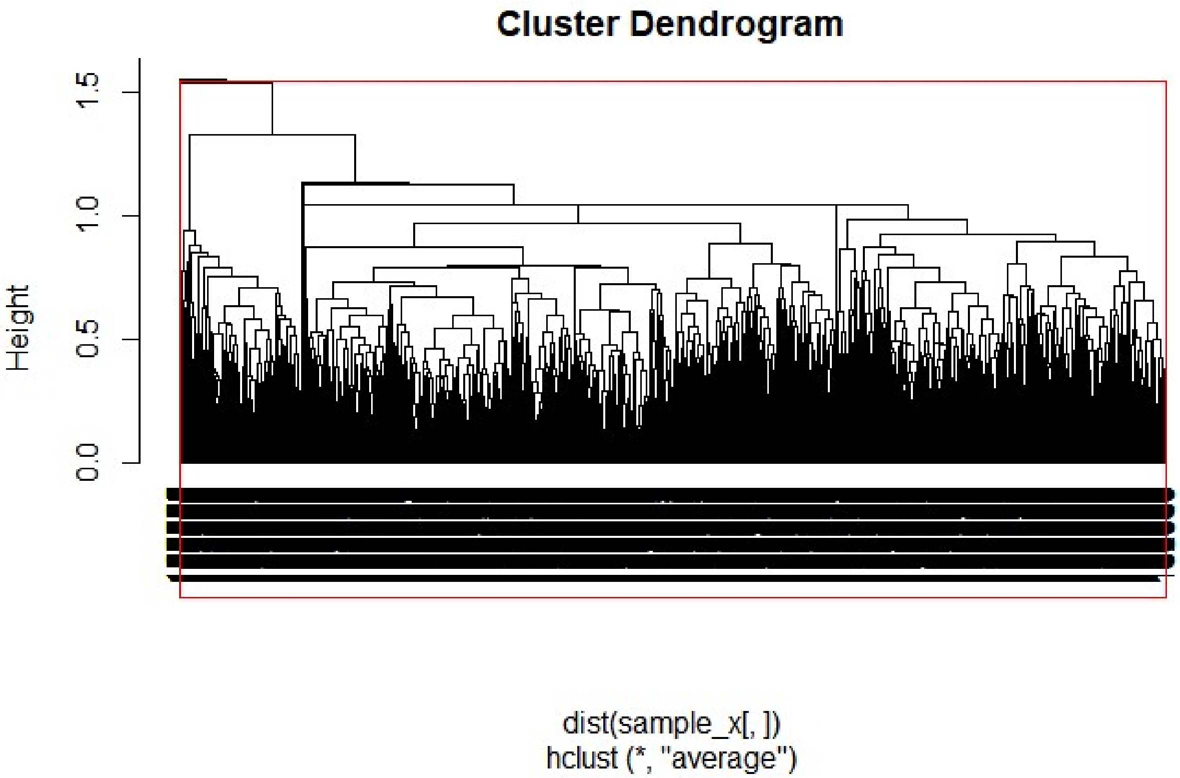
```
 num [1:10000] 0.84 0.381 0.356 0.259 0.246 ...
```

Training (Clustering)

Hide

```


clusters <- hclust(dist(sample_x[,]), method = 'ave') # Trains Clustering Algo
plot(clusters, hang = -1)
rect.hclust(clusters,k=2) # Selects two largest clusters
```



Hide

```
groups = cutree(clusters,k=2) # Selects two largest clusters

# Cluster Sizes
table(groups)
```

```
groups
   1    2 
9999    1 
```

Hide

```
# Adding Loss and Class columns to the dataset
sample_x$loss = loss_x
sample_x$class = groups
str(sample_x)
```

```
'data.frame':	10000 obs. of  16 variables:
 $ cont1 : num  0.476 0.476 0.324 0.233 0.635 ...
 $ cont2 : num  0.299 0.621 0.489 0.682 0.489 ...
 $ cont3 : num  0.441 0.693 0.593 0.528 0.337 ...
 $ cont4 : num  0.534 0.565 0.284 0.237 0.895 ...
 $ cont5 : num  0.783 0.789 0.281 0.482 0.846 ...
 $ cont6 : num  0.326 0.373 0.38 0.455 0.456 ...
 $ cont7 : num  0.369 0.356 0.644 0.339 0.41 ...
 $ cont8 : num  0.361 0.361 0.308 0.308 0.361 ...
 $ cont9 : num  0.398 0.444 0.431 0.329 0.469 ...
 $ cont10: num  0.396 0.45 0.439 0.396 0.517 ...
 $ cont11: num  0.298 0.706 0.512 0.352 0.492 ...
 $ cont12: num  0.292 0.692 0.5 0.388 0.485 ...
 $ cont13: num  0.339 0.339 0.282 0.605 0.364 ...
 $ cont14: num  0.84 0.381 0.356 0.259 0.246 ...
 $ loss  : num  0.84 0.381 0.356 0.259 0.246 ...
 $ class : int  1 1 1 1 1 1 1 1 1 1 ...
```

Hide

```
# Calculating the Median of Loss column for the 2 Cluster classes
median_1_hr = median(sample_x[sample_x$class==1,"loss"])
median_2_hr = median(sample_x[sample_x$class==2,"loss"])

# For each Cluster, finding # of samples with loss > median and loss < median
length(sample_x[sample_x$loss > median_1_hr & sample_x$class == 1,"loss"])
```

```
[1] 4999
```

Hide

```
length(sample_x[sample_x$loss <= median_1_hr & sample_x$class == 1,"loss"])
```

```
[1] 5000
```

Hide

```
length(sample_x[sample_x$loss <= median_2_hr & sample_x$class == 2,"loss"])
```

```
[1] 1
```

Hide

```
length(sample_x[sample_x$loss > median_2_hr & sample_x$class == 2,"loss"])
```

```
[1] 0
```

For a Cluster of 10k observations, it makes sense than 5k of the obvservations fall above the median, and the other 5k below.

2. K - means Clustering

Do kmeans clustering of the samples using the numeric features, by setting k=2. Then for each of these two clusters, find out how many samples with loss > the median loss; and how many with loss < the median loss.

Hide

```
# Applying KMeans cwith two clusters
kmeans.result = kmeans(train_x, 2)
```

Hide

```
# Checking the sizes of the 2 Clusters
kmeans.result$size
```

```
[1] 115851  72467
```

Hide

```
#Adding the Cluster values(1 or 2) as a column to the dataset
x = vector()
x = kmeans.result$cluster
train_x$loss = loss_1
train_x$class = x
```

Hide

```
# Calculating the Median of Loss column for the 2 Cluster classes
median_1 = median(train_x[train_x$class==1,"loss"])
median_2 = median(train_x[train_x$class==2,"loss"])


length(train_x[train_x$loss > median_1 & train_x$class == 1,"loss"])
```

```
[1] 57925
```

Hide

```
length(train_x[train_x$loss <= median_1 & train_x$class == 1,"loss"])
```

```
[1] 57926
```

Hide

```
length(train_x[train_x$loss <= median_2 & train_x$class == 2,"loss"])
```

```
[1] 36234
```

Hide

```
length(train_x[train_x$loss > median_2 & train_x$class == 2,"loss"])
```

```
[1] 36233
```

3. PCA

Do PCA of the data using the numeric features and use the first two PCs to represent a sample and re-do Question 1 and Question 2. Also generate barplot of the variance of the first five PCs. Hint: use prcomp function in R

Hide

```

res.pca = prcomp(train_x[,-c(15,16)])
```

Calculating Variances

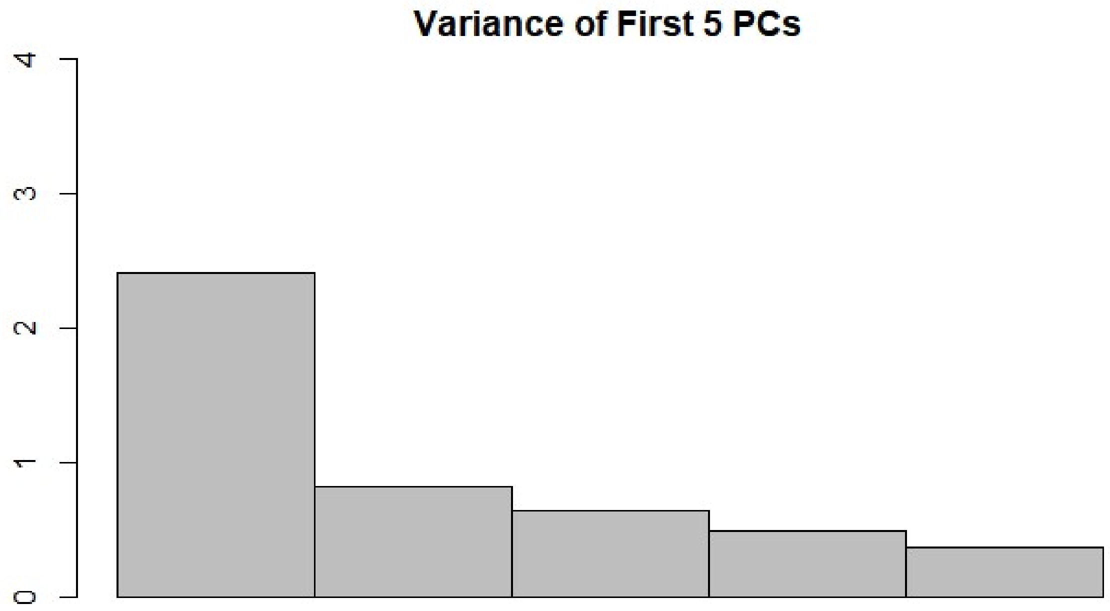
Hide

```
sd = res.pca$sdev
var = sd^2
```

Barplot of the Variance of the First 5 PCAs

Hide

```
barplot(var[1:5]*10,space = 0,xlim = c(0,5),ylim = c(0,4),main = 'Variance of First 5
PCs')
```



Hide

```
new_train = -res.pca$x[,1:2]
```

Converting to a DataFrame

Hide

```

new_train = as.data.frame(new_train)
```

Kmeans Clustering after PCA

Hide

```
kmeans.result_pca = kmeans(new_train,2)
```

Cluster Sizes

Hide

```
kmeans.result_pca$size
```

```
[1] 113785  74533
```

Adding Loss and Cluster(as Class) to the Dataset

Hide

```
new_train_kmeans = new_train
new_train_kmeans$loss = loss_1
new_train_kmeans$class = kmeans.result_pca$cluster
```

Calculating the Median of Loss column for the 2 Cluster classes

Hide

```
median_1_kmean = median(new_train_kmeans[new_train_kmeans$class==1,"loss"])
median_2_kmean = median(new_train_kmeans[new_train_kmeans$class==2,"loss"])


length(new_train_kmeans[new_train_kmeans$loss > median_1_kmean & new_train_kmeans$clas
s == 1,"loss"])
```

```
[1] 56892
```

Hide

```
length(new_train_kmeans[new_train_kmeans$loss <= median_1_kmean & new_train_kmeans$cla
ss == 1,"loss"])
```

```
[1] 56893
```

Hide

```
length(new_train_kmeans[new_train_kmeans$loss <= median_2_kmean & new_train_kmeans$cla
ss == 2,"loss"])
```

```
[1] 37267
```

Hide

```
length(new_train_kmeans[new_train_kmeans$loss > median_2_kmean & new_train_kmeans$clas
s == 2,"loss"])
```

```
[1] 37266
```