CLUSTERING: CHOOSE YOUR OWN ADVENTURE!

1 Overview

Today's adventure will involve a segmentation analysis of teenagers. The data is derived from status updates that the teens posted on a popular social networking platform. Our marketers honed in on 37 terms that they hoped would aid the segmentation of the sample into groups with different interests and different ideas of what was "cool." By getting to know the different types of teenage consumers, our brand hopes to find novel ways to connect with each group. Your job is to create and profile these groups in a way that lets us get to know the "prototype" of each cluster.

2 First things First: Explore your Data

No matter where you are going with it, the first step on your clustering adventure should be a deep exploratory dive into the data. This typically involves examining distributions - looking for signs that normalization or standardization might help space individuals out in a way that is amenable to clustering.

```
> load('teenSNS.RData')
> library(rpart)
> library(psych,ggplot2) #pairs.panels
> library(cluster) # Gap statistic of Tibshirani et al
> library(tidyverse)
> library(magrittr)
> library(flashClust)
> library(NbClust)
> library(clValid)
> library(ggfortify)
> library(clustree)
> library(dendextend)
> library(factoextra)
> library(FactoMineR)
> teens$cluster=NULL
> summary(teens)
> terms =  c("friends","basketball","football", "soccer" , "softball" ,"volleyball","swimming", "cheerleading","baseball" ,
+         "tennis" , "sports"   ,"cute"      ,"sex"     , "sexy"        ,"hot" ,
+         "kissed" , "dance"    ,"band"      ,"marching", "music"       ,"rock"  ,
+         "god"    , "church"   ,"jesus"     ,"bible"   , "hair"        ,"dress" ,
+         "blonde" , "mall"     ,"shopping"  ,"clothes" , "hollister"   ,"abercrombie" ,
+         "die"    , "death"    ,"drunk"     ,"drugs"  )
> demographics = c("gradyear","gender","age", "female","no_gender")
> # remove observations that have none of the terms:
> teens = teens[rowSums(teens[,terms])>0,]
> # observe basic summary stats for the terms
> summary(teens[,terms])
> #Downsample the points because 30K is a lot to draw
> samplePoints = sample(1:29074, 5000, replace=F)
> pairs.panels(teens[samplePoints,terms[1:5]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[6:10]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[11:15]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[16:20]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[21:25]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[26:30]], method = 'pearson', density=T,ellipses = F)
> pairs.panels(teens[samplePoints,terms[31:37]], method = 'pearson', density=T,ellipses = F)
>
>
```

If you choose SAS to conduct the analysis, a similar exploration would be done through proc univariate as follows:

```
cas;
caslib _all_ assign;

%let terms = friends basketball football soccer softball
        volleyball swimming cheerleading baseball tennis sports
        cute sex  sexy hot kissed dance band marching music rock
        god church jesus bible hair dress blonde  mall shopping
        clothes hollister abercrombie die death drunk drugs;
data teenterms;
set public.teensns;
total = sum(of &terms.);
if total>0; *eliminate observations with no terms;
keep &terms total;
run;

proc univariate data=teenterms;
var &terms.;
histogram &terms.;
run;
```

3 Processing your Data

Now that you've had a chance to explore your variables and their distributions, it's time to decide whether or not you want to use any form of standardization or normalization on your input features. Here is a list of some common approaches for data of this variety for you to choose from. There are other actions that you can take that are just as appropriate, so feel free to take liberty here if there is something else you'd like to try.

1. Do nothing. Let the data live on in its current form.

   ```
   > teens.norm = teens[,terms]
   ```

2. Employ statistical standardization (z-scores) by subtracting the mean of each column and dividing by the standard deviation of each column

   ```
   > teens.norm = scale(teens[,terms], center=T, scale=T)
   ```

   ```
   proc stdize data=teenterms out=teensnorm method=std;
     var &terms.;
   run;
   ```

3. Employ range standardization, putting each variable on a scale from 0 to 1, by transforming each column, $\mathbf{x}$, into $\mathbf{x}'$ as follows:

$$\mathbf{x}' = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$$

   ```
   > teens.norm = apply(teens[,terms], 2, function(x)(x-min(x))/(max(x)-min(x)))
   ```

```
proc stdize data=teenterms out=teensnorm method=range;
  var &terms.;
run;
```

4. Standardize *each observation* so that the rows sum to 1. This would represent each individual's usage of each word as a proportion of the total number of words they used from this list.

```
> teens.norm = as.data.frame(t(apply(teens[,terms], 1, function(x)(x/sum(x)))))
```

```
data teensnorm;
        set teenterms;
        array vars [*] &terms;
        do i=1 to dim(vars);
                vars[i] = vars[i]/total;
        end;
run;
```

5. Try a log transformation, transforming each column, $\mathbf{x}$, into $\log(\mathbf{x}+1)$ where 1 is added to the argument to prevent the problem that would be associated with computing $\log(0)$ which is undefined.

```
> teens.norm = log(teens[,terms]+1)
```

```
data teensnorm;
        set teenterms;
        array vars [*] &terms;
        do i=1 to dim(vars);
                vars[i] = ln(vars[i]+1);
        end;
run;
```

4 Visualize the processed data

Once we've processed the data for input to a clustering method, we should explore once more to make sure that nothing strange happened. For this, we can repeat the steps taken in section 2 with our new data frame teens.norm. We can also consider a projection onto principal components. For all but option 2 above, we can still choose between two different versions of PCA - covariance or correlation (for option 2, these two versions would be identical). If you choose to observe the data projected onto principal components, you can first plot the first two PCs of each version:

```
> cov.pca = prcomp(teens.norm, scale=F)
> cor.pca = prcomp(teens.norm, scale=T)
> par(mfrow=c(2,1))
> plot(cov.pca$sdev^2, main = 'Covariance PCA',ylab='Eigenvalue')
> plot(cov.pca$x)
> plot(cor.pca$sdev^2, main = 'Correlation PCA',ylab='Eigenvalue')
> plot(cor.pca$x)
```

The corresponding SAS code would be:

```
proc princomp data = teensnorm out = corPCA;
var &terms.;
run;
proc princomp data = teensnorm cov out = covPCA;
var &terms.;
run;
proc sgplot data=corPCA;
```

```
scatter x=prin1 y=prin2 ;
run;
proc sgplot data=covPCA;
scatter x=prin1 y=prin2 ;
run;
```

One can go further and plot the 2D projections onto other components if desired. The purpose of such exploration is to better understand the structure of your data cloud in space and to inform any decisions regarding the number of clusters or outliers that could affect the progress of clustering algorithms.

```
> # Explore FURTHER principal components for visualization
> # Downsample the points because 30K is a lot to draw
> samplePoints = sample(1:29074, 8000, replace=F)
> par(mfrow=c(3,3),mar=c(4,4,2,1))
> plot(cov.pca$x[samplePoints,c(1,3)])
> plot(cov.pca$x[samplePoints,c(1,4)])
> plot(cov.pca$x[samplePoints,c(1,5)])
> plot(cov.pca$x[samplePoints,c(2,3)])
> plot(cov.pca$x[samplePoints,c(2,4)])
> plot(cov.pca$x[samplePoints,c(2,5)])
> plot(cov.pca$x[samplePoints,c(3,4)])
> plot(cov.pca$x[samplePoints,c(3,5)])
> plot(cov.pca$x[samplePoints,c(4,5)])
> mtext("COVARIANCE PCA", side = 3, line = -1.5, outer = TRUE)
> par(mfrow=c(3,3),mar=c(4,4,2,1))
> plot(cor.pca$x[samplePoints,c(1,3)])
> plot(cor.pca$x[samplePoints,c(1,4)])
> plot(cor.pca$x[samplePoints,c(1,5)])
> plot(cor.pca$x[samplePoints,c(2,3)])
> plot(cor.pca$x[samplePoints,c(2,4)])
> plot(cor.pca$x[samplePoints,c(2,5)])
> plot(cor.pca$x[samplePoints,c(3,4)])
> plot(cor.pca$x[samplePoints,c(3,5)])
> plot(cor.pca$x[samplePoints,c(4,5)])
> mtext("CORRELATION PCA", side = 3, line = -1.5, outer = TRUE)
```

In SAS, the same plots can be created with proc sgscatter:

```
proc sgscatter data=corPCA;
  matrix prin1 prin2 prin3 prin4 prin5;
run;
proc sgscatter data=covPCA;
  matrix prin1 prin2 prin3 prin4 prin5;
run;
```

We may choose to revisit these plots once we have some clusters and we'd like to visualize their layout. For now, we're ready to address the question of how many clusters exist in the data.

5 Determine your input data

While it is fine to use whatever normalized version of your original data as input to *k*-means, particularly here in the case of only 37 terms, it is very often a good idea to reduce the dimensionality of data prior to input to a clustering algorithm. Of course, now your choices again expand: you have 9 different options of matrix factorizations using PCA alone (from 2 different versions of PCA on any of the 5 varieties of standardized data outlined in section 3, minus 1 for the fact that covariance and correlation PCA are identical when the input data is standardized by z-score.)

Beyond the choice of the specific matrix factorization to use, you *also* must choose the rank (i.e. dimensionality/number of components) to which you will reduce the data. The number of components can be chosen based upon the screeplot or the cumulative % of variance explained.

Alter the code below to suit your desires and define your input data - **the lines of code below are just examples!**

```
> #input = teens.norm #raw normalized data
> #input = cov.pca$x[,1:4] #first 4 PCs from covariance PCA on normalized data
> input = cor.pca$x[,1:7] #first 7 PCs from correlation PCA on normalized data
```

```
data input;
  set teensnorm;
end;

data input;
  set corPCA;
  keep prin1--prin7;
run;
```

6 Choose the number of clusters, k

There are a multitude of options for exploring how many clusters might be appropriate for a given dataset. When it comes to implementation, it won't always be *easy* to use every method across every software platform. Thus, we will break this section into options for R and options for SAS.

6.1 Choose the number of clusters in R

1. Look for an elbow in the screeplot, much like we did when determining an appropriate dimensionality of the data. Using the options in this document, you have 9 different options for that screeplot (see section 5).

2. Examine the SSE value for clusterings with $k = 1, 2, 3, \ldots$ and look for an elbow in *that* graph. The code below actually creates *ten* clusterings for each value of k, and then displays a boxplot of the SSE for each value of k.

```
> obj = matrix(NA, nrow=10, ncol = 19)
> for(k in 2:20){
+   iter=1
+   while(iter<11){
+   obj[iter,(k-1)] = kmeans(input,k)$tot.withinss
+   iter=iter+1
+   }
+ }
> colnames(obj) = paste('k',2:20,sep='')
> rownames(obj) = paste('iter',1:10,sep='')
> # Use output to create data frame for boxplot visual
> obj = data.frame(obj)
> obj2 = gather(obj,key = 'K',value = 'SSE',  k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13,k14,k15,k16,k17,k18,k19,k20  )
> obj2$K = gsub("k",'',obj2$K)
> obj2$K = as.numeric(obj2$K)
> par(mfrow=c(1,1))
> boxplot(SSE~K,data=obj2, ylab = 'SSE Objective Function', xlab='Number of clusters, k', col='violet')
>  #,    main = 'Box plots of SSE for 10 runs of k-means with each k')
>
```

3. Trace(W_q) method. Let $\mathbf{W}_q$ be the within-group dispersion matrix for data clustered into q clusters:

$$.\mathbf{W}_q = \sum_{k=1}^{q} \sum i \in C_k (x_i - c_k)(x_i - c_k)^T \qquad (1)$$

In this method, we compute the $trace(W_q)$ for various numbers of clusters. This should decrease monotonically as the number of clusters grows, and we use the maximum of the second differences to determine the number of clusters (i.e. where the slope of the curve is increasing the fastest)

```
> res=NbClust(input, distance = "euclidean", min.nc=2, max.nc=8,
+             method = "kmeans", index = "tracew")
> (k=res$Best.nc)
> clusters = res$Best.partition
```

4. Marriot Index. Let $\mathbf{W}_q$ be the within-group dispersion matrix for data clustered into q clusters, as defined in equation 1. Marriot's index is:

$$q^2 \det(\mathbf{W_q})$$

Across multiple clusterings, we're looking for the maximum difference between successive levels to determine the optimal k.

```
> res=NbClust(input, distance = "euclidean", min.nc=2, max.nc=8,
+             method = "kmeans", index = "marriot")
> (k=res$Best.nc)
> clusters = res$Best.partition
```

5. Method of Friedman and Rubin (1967). Let $\mathbf{W}_q$ be the within-group dispersion matrix for data clustered into q clusters, as defined in equation 1. Friedman's index is computed as

$$trace(\mathbf{W}_q^{-1}\mathbf{B_q})$$

where $\mathbf{B}_q$ is the between-group dispersion matrix for data clustered into q clusters, defined as

$$\mathbf{B}_q = \sum_{k=1}^{q} n_k(\mathbf{c}_k - \bar{\mathbf{x}})(\mathbf{c}_k - \bar{\mathbf{x}})^T$$

Here, we look for the maximum difference in values of this criterion to determine the optimal k.

```
> res=NbClust(input, distance = "euclidean", min.nc=2, max.nc=10,
+             method = "kmeans", index = "friedman")
> (k=res$Best.nc)
> clusters = res$Best.partition
```

6. Use the Gap Statistic, a measure proposed by Tibshirani et al in 2001. **(too slow for 30K+ observations)** It's not really a great idea to downsample to employ these methods, but if you'd like to see this type of analysis in action, the cluster package has a nice implementation:

```
> samplePoints = sample(1:29074, 2000, replace=F)
> out.gap =clusGap(input[samplePoints,], kmeans, 10, B = 50, d.power = 2,
+         spaceH0 = c("scaledPCA"),
+         verbose = interactive())
> fviz_gap_stat(out.gap)
>   + theme_minimal()
>     + ggtitle("fviz_gap_stat: Gap Statistic")
```

7. Use the average Silhouette value. **(Too slow for 30K+ observations.)** For each observation, the **silhouette** value of that observation is

$$\frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the sum of its distances to points in the closest neighboring cluster and $b(i)$ is the sum of its distances to points in the same cluster. Thus, silhouette values take values between -1 and 1. Values of -1 would signify that the given point is, on average, *closer* to points in a neighboring cluster than it is to points in its own cluster and values of +1 would indicate that the given point is *much* closer to points in its own cluster than it is to points in the neighboring cluster. To determine a number of clusters, the average silhouette value is calculated across the dataset and the clustering with the maximum value is determined to have the optimal number of clusters.

It's not really a great idea to downsample to employ these methods, but if you'd like to see this type of analysis in action, the factoextra package has a nice implementation:

```
> fviz_nbclust(input[samplePoints,], kmeans, method = "silhouette", k.max = 10)
>   + theme_minimal()
>     + ggtitle("The Silhouette Plot")
```

8. Explore 24 metrics that have been proposed in the literature via the Nbclust package. **(Most methods too slow for 30K+ observations.)**

   Again, it's not really a great idea to downsample to employ these methods, but if you'd like to see this ensemble approach in action:

   ```
   > samplePoints = sample(1:29074, 2000, replace=F)
   > res=NbClust(input[samplePoints,], distance = "euclidean", min.nc=2, max.nc=10,
   +             method = "kmeans", index = "all")
   ```

Hopefully after exploring all these methods *and* considering the use-case to the brands' marketers you have some idea about how many clusters would be ideal. You can always try solutions with different number of clusters and leave the choice of cluster solution to the final profiling stage.

6.2 Choose the number of clusters in SAS

In SAS, each clustering procedure contains different approaches for determining the number of clusters. We'll enumerate some of those techniques here:

1. **Aligned Box Criteria**. SAS's computationally efficient response to the Gap Statistic. Video introduction for more details: https://mlconf.com/sessions/estimating-the-number-of-clusters-in-big-data-with/

   ```
   proc hpclus data=input
           noc=abc
           maxclusters=20
           maxiter = 15;
   input _all_;
   run;
   ```

2. **Cubic Clustering Criteria (CCC)**. This is likely to have run-time that is impractical for datasets as large as 30K observations. To use the CCC, we're generally looking for a local maximum in the plot of the ccc. The authors of this metric posit that a partition into *k* clusters is good when we see a dip in CCC for k-1 clusters and a peak for k clusters. After k clusters, the CCC should either gradually decrease or gradually rise (the latter event happens when more isolated groups or points are present).

3. **Pseudo-F.** You'll get both the CCC and the pseudo-F values after hierarchical clustering in SAS. To use the Pseudo-F statistic, we're looking for a peak in the chart. Higher values of pseudo-F indicate more "valid" clusters.

   ```
   data input;
     set input;
     id=_n_;
   run;

   proc cluster data=input
     method=centroid
     ccc pseudo
     outtree=cluster_tree;
   var LIST ALL EXCEPT ID;
   id id;
   run;
   ```

7 Determine a Final Clustering

Now that we've finished our exploration into the data and the potential number of clusters it could contain, it's time to *try*. To determine a final clustering for this adventure, choose from the 2 algorithms we covered in class: *k*-means and hierarchical clustering.

1. ***k*-means**

```
> set.seed(11117)
> final.clusters = kmeans(input,k)
> teens$cluster = final.clusters$cluster
```

```
proc fastclus data=input
        maxclusters=20
        out = final_clusters;
var _all_;
run;
```

2. **Hierarchical clustering**. Not recommended for users of R, as it will require in-memory storage of a similarity matrix that is roughly 10gb.

```
> tree = flashClust(dist(input),method='centroid')
> teens$cluster =cutree(tree,k=3)
```

The SAS implementation is a little more clunky, as one must first *create* the dendrogram (perhaps you've already done this in an earlier step to compute CCC or Pseudo-F - if so, feel free to use that outtree dataset and start in the 3rd procedure below), then *cut* the dendrogram, then *merge* the cluster assignments back into their original dataset for profiling.

```
data input;
  set input;
  id=_n_;
run;

proc cluster data=input
  method=centroid
  outtree=cluster_tree;
var LIST ALL EXCEPT ID;
id id;
run;

proc tree data = cluster_tree nclusters=2 out=final_clusters;
run;

proc sort data = final_clusters ;
by  _NAME_;
run;

data final_clusters;
merge input final_clusters;
run;
```

That was the easy part! Remember that k-means is, by default, randomly initialized in R. If you're displeased with your solution after the profiling step (next section) then you can always go back and re-create a final solution with a different seed.

8 Profile the Clusters

This step is where the actionable insights are generated - describe each cluster. Create a profile for each cluster that describes key attributes which differentiate it from the other clusters. There are many ways to approach this goal, here you are presented with two options:

1. **Visualize variable distributions, cluster vs. all**. By overlaying histograms for each variable on the whole dataset and each cluster, we can quickly see where cluster distributions differ greatly from population distributions. The following R function takes as input: a dataframe, a character vector listing the variables of interest for profiling, and a string that identifies the cluster variable in the dataframe.

```
> clusterProfile = function(df, clusterVar, varsToProfile){
+   k = max(df[,clusterVar])
+   for(j in varsToProfile){
+     if(is.numeric(df[,j])){
+     for(i in 1:k){
+       hist(as.numeric(df[df[,clusterVar]==i ,j ]), breaks=50, freq=F, col=rgb(1,0,0,0.5),
+        xlab=paste(j), ylab="Density", main=paste("Cluster",i, 'vs all data, variable:',j))
+       hist(as.numeric(df[,j ]), breaks=50,freq=F, col=rgb(0,0,1,0.5), xlab="", ylab="Density", add=T)
+
+       legend("topright", bty='n',legend=c(paste("cluster",i),'all observations'),
+              col=c(rgb(1,0,0,0.5),rgb(0,0,1,0.5)), pt.cex=2, pch=15 )}
+       }
+     if(is.factor(df[,j])&length(levels(df[,j]))<5){
+       (counts = table( df[,j],df[,clusterVar]))
+       (counts = counts%*%diag(colSums(counts)^(-1)))
+       barplot(counts, main=paste(j, 'by cluster'),
+               xlab=paste(j),legend = rownames(counts), beside=TRUE)
+     }
+   }
+ }
> clusterProfile(teens, 'cluster', terms)
```

   In SAS, to create such descriptive graphs, your best bet is the drag-and-drop interface of SAS Visual Analytics. To promote your data for use in the SAS VA module, you'll have to first save it to your casuser library and then promote it with proc casutil so that you can access it in SAS VA:

```
data teens;
set public.teensns;
total = sum(of &terms.);
if total>0; *eliminate observations with no terms;
drop total;
run;

data casuser.final;
merge teens fc(keep=cluster);
run;

proc casutil;
        promote casdata='final'
        incaslib ='casuser'
        outcaslib='casuser'
        casout ='final_clustering';
run;
```

   Once you can access the dataset final_clustering in SAS VA, you should be able to drag any variable to the chart along with cluster and get a visual that compares the clusters for that variable.

2. **Decision Trees to Predict Cluster.** Another profiling method is to create a decision tree with your cluster variable as your target. This will give you a set of rules (in the form of a tree) to place new observations in each cluster,

which in turn ought to help you explore the profiles of each cluster. This can be done in a one-vs-all fashion (make k trees, each tree predicting membership in a single cluster) or in a multi-class prediction fashion (simply predict the multi-level target "cluster") - the choice of tactic here probably depends on how many clusters you have. If you have more than 3-4 clusters, the multi-class prediction tree might be harder to draw conclusions from.

```
> teens$cluster=as.factor(teens$cluster)
> dtree = rpart(cluster~.,data=teens)
> plot(dtree)
> text(dtree)
```

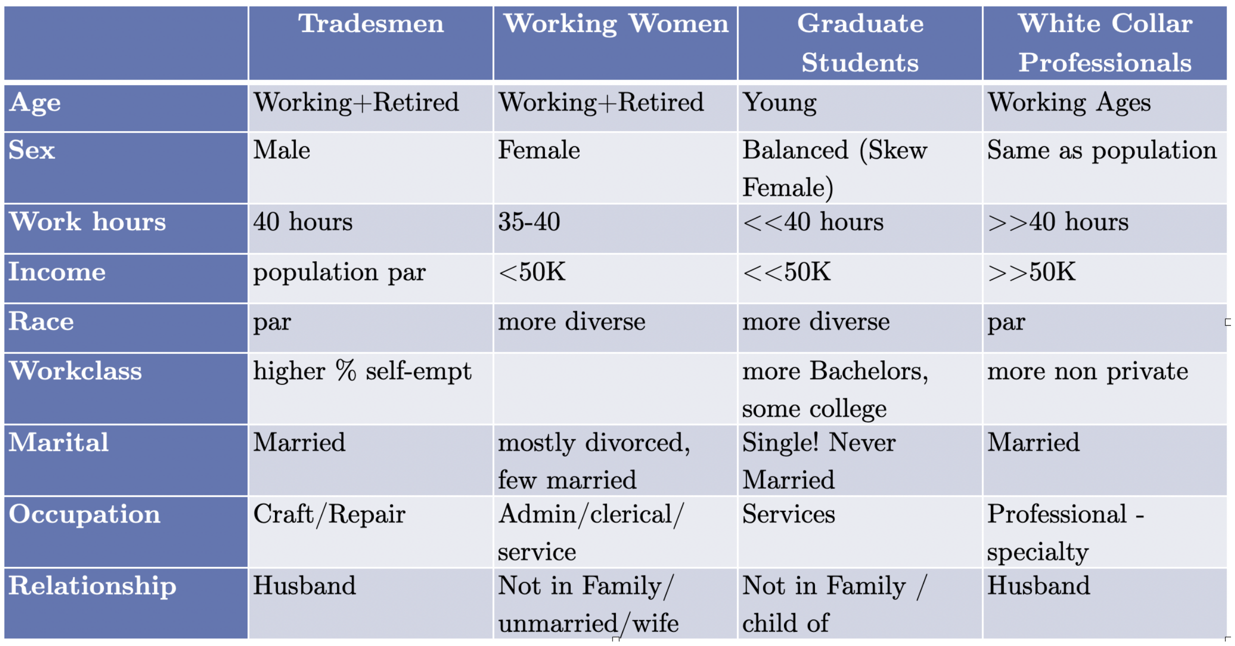
In SAS, we can either use the Visual Analytics point-and-click decision tree (probably faster to get the answer this way!) or the Decision Tree task in the Develop SAS Code Studio:

```
proc treesplit data=CASUSER.FINAL_CLUSTERING maxdepth=10;
        input gradyear age friends basketball football soccer softball volleyball
                swimming cheerleading baseball tennis sports cute sex sexy hot kissed dance
                band marching music rock god church jesus bible hair dress blonde mall
                shopping clothes hollister abercrombie die death drunk drugs female no_gender
                / level=interval;
        target CLUSTER / level=nominal;
        grow igr;
        prune costcomplexity(kfold=10);
run;
```

9 Create the Deliverable

At the end of the day, you want to sell your segmentation solution to the marketing team by describing your cluster profiles in a way that is honest and memorable. This means you'll have to give each cluster a name that would help any lay person soak up the profiled information, and probably create a table that highlights how each cluster stands out (or doesn't stand out) across each variable. A good way to get to the final deliverable is to create a table that is $p \times k$ where p=number of variables and k=number of clusters. Highlight an entry in the table if something stands out for that cluster and that variable. Make a quick note in the cell about *how* they stand out (e.g. "older than population"). Hopefully such a table will give way to final descriptions and names of clusters.

For an example, see the figure below.



	Tradesmen	**Working Women**	**Graduate Students**	**White Collar Professionals**
Age	Working+Retired	Working+Retired	Young	Working Ages
Sex	Male	Female	Balanced (Skew Female)	Same as population
Work hours	40 hours	35-40	<<40 hours	>>40 hours
Income	population par	<50K	<<50K	>>50K
Race	par	more diverse	more diverse	par
Workclass	higher % self-empt		more Bachelors, some college	more non private
Marital	Married	mostly divorced, few married	Single! Never Married	Married
Occupation	Craft/Repair	Admin/clerical/ service	Services	Professional - specialty
Relationship	Husband	Not in Family/ unmarried/wife	Not in Family / child of	Husband