**Data Analytics (CMP330)**

# Practical 7 – Feature Selection

Please remember to save all .R files in your personal drive.

\*\*\*\* Typical example of errors you may come across in R \*\*\*\*\*\*

1. Remember to set your working directory at the start

Session -> set working directory -> choose file

Under Choose file you can select your P drive where you can save your code and .csv files

1. Handling “no package” errors. For example, suppose you are trying to load a library such as Bioconductor (just an example) with the code library(Bioconductor) and you get an error:

Error in library(bioconductor) :

there is no package called ‘bioconductor’

To handle this go to Tools -> install packages

In the box type in the package to install

**Part A – Principal Component Analysis**

In this analysis you will use the mtcars dataset (this is built into R). This dataset consists of data on 32 models of car, taken from an American motoring magazine (1974 Motor Trend magazine). For each car, you have 11 features, expressed in varying units (US units). Run the script to view a summary of the dataset:

**summary(mtcars)**

How many features are in this dataset?

**Compute the principle components**

1. Remove the two categorical variables (vs and am).
2. Pass mtcars matrix to the prcomp() function – more information on this function and parameters can be found here: <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/prcomp.html>
3. Set two arguments, center and scale, to be TRUE. Assign your output to mtcars.pca.

**mtcars.pca <- prcomp(mtcars[,c(1:7,10,11)], center = TRUE,scale. = TRUE)**

1. View your PCA object with **summary(mtcars.pca)**.

You should obtain 9 principal components PC1-9. Each of these explains a percentage of the total variation in the dataset. That is to say: PC1 explains 63% of the total variance, which means that nearly two-thirds of the information in the dataset (9 variables) can be encapsulated by just that one Principal Component. PC2 explains 23% of the variance. So, by knowing the position of a sample in relation to just PC1 and PC2, you can get a very accurate view on where it stands in relation to other samples, as just PC1 and PC2 can explain 86% of the variance.

1. Call **str(mtcars.pca)**

This provides details on the center point ($center), scaling ($scale), standard deviation(sdev) of each principal component. The relationship (correlation or anticorrelation, etc) between the initial variables and the principal components ($rotation). The values of each sample in terms of the principal components ($x).

**Plot the Principal Components**

In this part you will plot a biplot, which includes both the position of each sample in terms of PC1 and PC2 and also will show you how the initial variables map onto this. A biplot is a type of plot that will allow you to visualize how the samples relate to one another in our PCA (which samples are similar and which are different) and will simultaneously reveal how each variable contributes to each principal component.

1. install packages:

Tools -> install packages: factoextra

**library(factoextra)**

1. Call fviz\_eig on your PCA data

**fviz\_eig(mtcars.pca, addlabels=TRUE)**

fviz\_eig() visualize eigenvalues using a scree plot. The x-axis represents 9 PC dimensions while y-axis represents the percentage of explained variance. You can find that the first two PC explains 85.9% variances.

1. Create variable plot and Control variable colours using their contributions

**fviz\_pca\_var(mtcars.pca, col.var="contrib",**

**repel = TRUE # Avoid text overlapping**

**)**

Look at the plot, positive correlated variables point to the same side of the plot. Negative correlated variables point to opposite sides of the graph. Longer arrows indicate better explanation by this variable. Variable contributions are indicated as colour shows.

**Interpreting the results**

Look at the origin of each of the cars putting them into one of three categories: US, Japanese and European cars. You make a list for this info, then pass it the col.ind argument of fviz\_pca\_biplot. You'll also set ellipse.type to "convex" , which will draw an convex around each group.

## interpret the results

**mtcars.country <- c(rep("Japan", 3), rep("US",4), rep("Europe", 7),rep("US",3), "Europe", rep("Japan", 3), rep("US",4), rep("Europe", 3), "US", rep("Europe", 3))**

**fviz\_pca\_biplot(mtcars.pca,**

**col.ind = mtcars.country, palette = "jco",**

**addEllipses = TRUE, label = "all", ellipse.type = "convex",**

**col.var = "black", repel = TRUE)**

Having performed a PCA using the mtcars dataset, we can see a separation between American and Japanese cars along a principal component that is closely correlated to cyl, disp, wt, and mpg. This provides us with some clues for future analyses; if we were to try to build a classification model to identify the origin of a car, these variables might be useful.

**Part B Feature Selection – Correlation, importance, Random Forest**

**Finding highly correlated features in the dataset**

Data can contain attributes that are highly correlated with each other. Many methods perform better if highly correlated attributes are removed. The Caret R package provides the findCorrelation which will analyze a correlation matrix of your data’s attributes report on attributes that can be removed.

1. Open R studio go to tools -> install the following packages:

skimr

mlbench

caret

randomForest

e1071

Load the installed libraries and find the highly correlated features in the PimaIndiansDiabetes dataset.

1. Review a summary of the data

**skim(PimaIndiansDiabetes)**

1. Calculate correlation matrix

**correlationMatrix <- cor(PimaIndiansDiabetes[,1:8])**

1. summarize the correlation matrix

**print(correlationMatrix)**

1. find attributes that are highly corrected (e.g. >0.5)

**highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.5)**

1. print indexes of highly correlated attributes

**print(highlyCorrelated)**

1. Try different correlation cut-offs – how do the number of highly correlated attributes differ?

**Finding important features in the dataset**

The importance of features can be estimated from data by building a model. Some methods like decision trees have a built in mechanism to report on variable importance. In this section you will use the Pima Indians Diabetes dataset and construct an [Learning Vector Quantization](http://en.wikipedia.org/wiki/Learning_vector_quantization) (LVQ) model to identify important features in the dataset.

1. load the library

**library(mlbench)**

**library(caret)**

1. load the dataset

**data(PimaIndiansDiabetes)**

1. prepare training scheme

**control <- trainControl(method="repeatedcv", number=10, repeats=3)**

4. train the model using lvq (diabetes is your class attribute pos or neg)

**model <- train(diabetes~., data=PimaIndiansDiabetes, method="lvq", preProcess="scale", trControl=control)**

For more information on the train function parameters view here: https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train

1. estimate variable importance

**importance <- varImp(model, scale=FALSE)**

1. summarize importance

**print(importance)**

1. plot importance

**plot(importance)**

The varImp is used to estimate the variable importance, which is printed and plotted. It shows that the glucose, mass and age attributes are the top 3 most important attributes in the dataset and the insulin attribute is the least important.

Now try using a different machine learning approach to train the model such as bayesglm. Does this change the importance results?

**Recursive Feature Elimination**

Automatic feature selection methods can be used to build many models with different subsets of a dataset and identify those attributes that are and are not required to build an accurate model. A popular automatic method for feature selection provided by the caret R package is called **Recursive Feature Elimination (**RFE).

View the file **feature\_engineering\_RFE.r** on Blackboard that provides an example of the RFE method on the Pima Indians Diabetes dataset.

A Random Forest algorithm is used on each iteration to evaluate the model. The algorithm is configured to explore all possible subsets of the attributes. All 8 attributes are selected in this example, although in the plot showing the accuracy of the different attribute subset sizes, we can see that using just 4 attributes gives comparable results.