5th) Applied Data Science - Modeling and Prediction

by Kumaran Ponnambalam, Dedicated to Data Science Education Follow

This course focuses on **Modeling** and **Prediction**. Different algorithms for supervised and unsupervised learning are explored. Use cases are presented for the major types of algorithms.

20 Lessons (4h 40m)

URL: https://www.skillshare.com/classes/Applied-Data-Science-5-Modeling-and-Prediction/543485717?via=similar-classes

1. About Applied Data Science Series 8:12

Course goal

- Train students to be full-fledged data science **practitioners** who could execute **end-to-end** data science projects to achieve **business results**
- The course is oriented towards existing software professionals
  - Heavily focused on programming and solution building
  - Limited, as-required exposure to math and statistics
  - Overview of ML concepts, with focus on using existing tools to develop solutions

Achievements

- Understand the concepts and life cycle of Data Science
- Develop proficiency to use R for all stages of analytics
- Learn Data Engineering tools and techniques
- Acquire knowledge of different machine learning techniques and know when and how to use them.
- Become a full-fledged Data Science Practitioner who can immediately contribute to real-life Data Science projects

Course structure

- Concepts of Data Science
- Data Science Life Cycle
- Statistics for Data Science
- R Programming
  - Examples
- Data Engineering
- Modeling and Predictive Analytics
  - Use cases
- Advanced Topics
- Resource Bundle

Documents ▸ 2. Data Science (Scientist) ▸ Courses (DS) ▸ Resources ▸ 2_ADSR ▸

Nom	Modifié le	Type	Taille
AdvancedMethodsBreastCancer.pdf	16-02-15 17:45	Adobe Acrobat D...	5.575 Ko
AssociationRulesAccidents.pdf	12-02-15 00:09	Adobe Acrobat D...	215 Ko
DecisionTreesIris.pdf	10-02-15 19:19	Adobe Acrobat D...	312 Ko
KMeansClusteringAutoData.pdf	11-02-15 19:22	Adobe Acrobat D...	232 Ko
LinearRegressionAutoMpg.pdf	09-02-15 20:23	Adobe Acrobat D...	732 Ko
NaiveBayesSpamFiltering.pdf	11-02-15 00:35	Adobe Acrobat D...	274 Ko
RandomForestBankCustomer.pdf	17-02-15 07:22	Adobe Acrobat D...	7.079 Ko
accidents.csv	11-02-15 23:50	Fichier CSV	46 Ko
auto-data.csv	11-02-15 19:10	Fichier CSV	12 Ko
auto-miles-per-gallon.csv	09-02-15 18:39	Fichier CSV	17 Ko
bank.csv	14-02-12 23:38	Fichier CSV	451 Ko
breast_cancer.csv	01-09-13 00:35	Fichier CSV	123 Ko
capoi.csv	19-11-14 21:24	Fichier CSV	2.774 Ko
employee.csv	28-01-15 02:16	Fichier CSV	1 Ko
sms_spam_short.csv	31-10-14 06:25	Fichier CSV	44 Ko
timeseries.csv	20-11-14 03:14	Fichier CSV	8 Ko
Data Engineering Examples.R	05-02-15 23:03	Fichier R	4 Ko
R-Programming-Examples.R	15-02-15 09:34	Fichier R	12 Ko
Applied Data Science_ADSR-Resources.zip	06-10-20 15:31	zip Archive	14.714 Ko

Analytics and Predictions

2. Types of Analytics 12:08

Types of Analytics

Descriptive	Understand what happened
Exploratory	Find out why something is happening
Inferential	Understand a population from a sample
Predictive	Forecast what is going to happen
Causal	What happens to one variable when you change another
Deep	Use of advanced techniques to understand large and multi-source datasets

#6 types : descriptive, exploratory, inferential, predictive, causal, deep

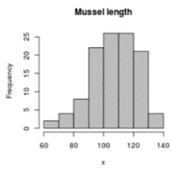
Reminder: a subset of a population is called a sample.

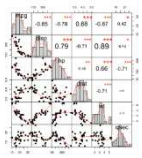
Exploratory Data Analysis (EDA)

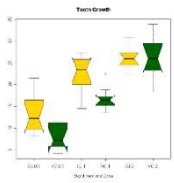
- Understand the predictors and targets in the data set
  - Spreads
  - Correlations
- Uncover the patterns and trends
- Find key variables and eliminate unwanted variables
- Detect outliers
- Validate previous data ingestion processes for possible mistakes
- Test assumptions and hypothesis

Tools used for EDA

- Correlation matrices
- Boxplots
- Scatterplots
- Principal component Analysis
- Histograms







Correlation (lien1,lien2),
Principal Component Analysis (PCA)

3. Types of Learning 17:16

Machine Learning

- Data contains attributes
- Attributes show relationships (correlation) between entities
- Learning – understanding relationships between entities
- Machine Learning – a computer analyzing the data and learning about relationships
- Machine Learning results in a model built using the data
- Models can be used for grouping and prediction

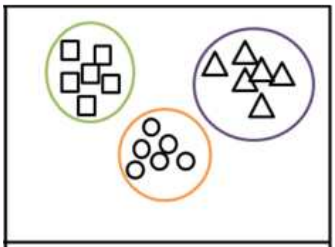
Data for Machine Learning

- Machines only understand numbers
- Text Data need to be converted to equivalent numerical representations for ML algorithms to work.
- Number representation
  - (Excellent, Good, Bad can be converted to 1,2,3)
- Boolean variables
  - 3 new Indicator variables called Rating-Excellent, Rating-Good, Rating-Bad with values 0/1
- Document Term matrix

DTM => https://www.tidytextmining.com/tidytext.html

Unsupervised Learning (UL)

- Finding hidden structure / similarity / grouping in data
- Observations grouped based on similarity exhibited by entities
- Similarity between entities could be by
  - Distance between values
  - Presence / Absence
- Types
  - Clustering
  - Association Rules Mining
  - Collaborative Filtering

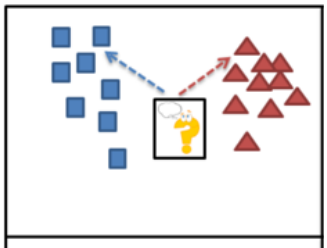


Apprentissage non-supervisé

UL : https://fr.wikipedia.org/wiki/Apprentissage_non_supervis%C3%A9

Supervised Learning (SL)

- Trying to predict unknown data attributes (outcomes) based on known attributes (predictors) for an entity
- Model used to predict future outcomes
- Types
  - Regression (continuous outcomes)
  - Classification (binary outcomes)

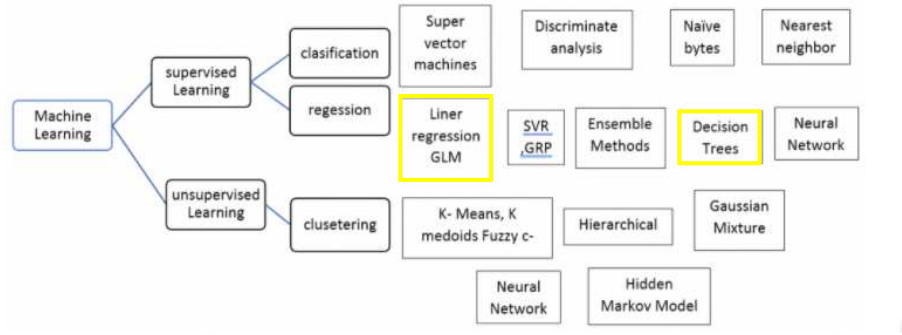


Apprentissage supervisé

SL : https://fr.wikipedia.org/wiki/Apprentissage_non_supervis%C3%A9

Machine Learning tree: SL vs. UL

Ref. https://www.educba.com/what-is-supervised-learning/

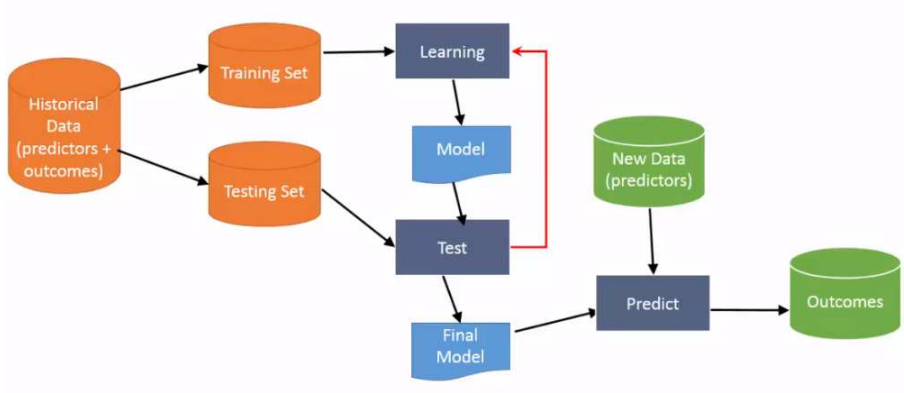


ML > SL > regression > LM, DT

ML > SL > classification

ML > UL > …

Supervised Learning Process



Training Data

- Historical Data contains both predictors and outcomes
- Split as training and testing data
- Training data is used to build the model
- Testing data is used to test the model
  - Apply model on training data
  - Predict the outcome
  - Compare the outcome with the actual value
  - Measure accuracy
- Training and Test fit best practices
  - 70-30 split
  - Random selection of records. Should maintain data spread in both datasets

+ Evaluate machine learning models using the train-test split (tutorial)

Reinforcement learning:

https://mc.ai/understanding-supervised-unsupervised-and-reinforcement-learning/

4. Analyzing Results and Errors 13:46

Comparing Results

Confusion Matrix

- Plots the predictions against the actuals for the test data
- Helps understand the accuracy of the predictions
- Predictions can be Boolean or classes

Confusion Matrix *is a tool to determine the performance of classifier. It contains information about actual and predicted classifications.*

Prediction Types

- The importance of prediction types vary by the domain
- True Positive (TP) and True Negative (TN) are the correct predictions
- False Negative (FN) can be critical in medical field
- False Positive (FP) can be critical in judicial field

		Actual	
		TRUE	FALSE
Predict ion	TRUE	True Positive	False Positive
	FALSE	False Negative	True Negative

False Negative *are not acceptable in* <u>*medical field*</u> *(incorrect detection of a disease: patient is positive (ill), but you detect as negative i.e. not ill!*
False Positive *are not acceptable in* <u>*judicial field*</u> *(incorrect detection of a criminal: client is negative (innocent), but you detect as positive i.e. a criminal!*

Confusion Matrix metrics

- Accuracy
  - Measures the accuracy of the prediction
  - Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Sensitivity
  - Hit rate or recall
  - Sensitivity = TP / (TP + FN)
- Specificity
  - True negative rate
  - Specificity = TN / (TN + FP)
- Precision
  - Precision = TP / (TP + FP)

#5 metrics: accuracy, sensitivity, specificity, precision, negative predictive value

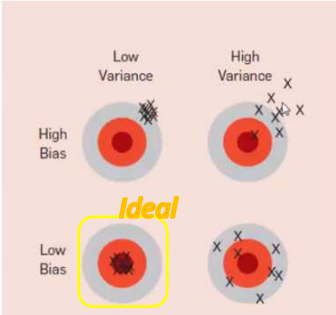
		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Ref. https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html

Prediction Errors

Bias and Variance

- Bias happens when the model "skews" itself to certain aspects of the predictors, while ignoring others. It is the error between prediction and actuals.
- Variance refers to the stability of a model – Keep predicting consistently for new data sets. It is the variance between predictions for different data sets.

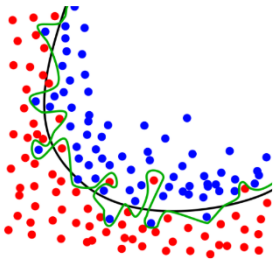


bias vs. variance (tutorial, trade-off)

1) *Low-bias ML algo: Decision Trees, k-Nearest Neighbours and Support Vector Machines.*
2) *High-bias ML algo: Linear Regression, Linear Discriminant Analysis, and Logistic Regression.*
3) *Low-variance ML algo: Linear Regression, Linear Discriminant Analysis, and Logistic Regression.*
4) *High-variance ML algo: Decision Trees, k-Nearest Neighbours and Support Vector Machines.*

Type of errors

- In-Sample error is the prediction error when the model is used to predict on the training data set it is built upon.
- Out-of-sample error is the prediction error when the model is used to predict on a new data set.
- Over fitting refers to the situation where the model has very low in-sample error, but very high out-of-sample error. The model has "over fit" itself to the training data.



Under vs. Over-fitting , In-sample vs. Out-of-sample

Sur-ajustement (overfitting) : https://fr.wikipedia.org/wiki/Surapprentissage

Sous-ajustement (underfitting) : https://www.statsoft.fr/concepts-statistiques/glossaire/s/surajustement.html

Ex. https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765

5. Linear Regression 19:00

Regression Analysis

- Method of investigating functional relationship between variables
- Estimate the value of dependent variables from the values of independent variables using a relationship equation
- Used when the dependent and independent variables are continuous and have some correlation.
- Goodness of Fit analysis is important.

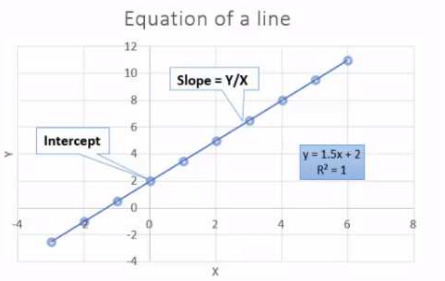
Linear Equation

- X is the independent variable
- Y is the dependent variable
- Compute Y from X using

$$Y = \alpha X + \beta$$

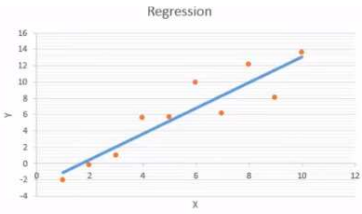
Coefficients:

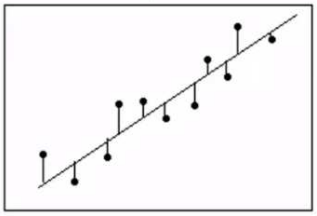
- α = Slope = Y/X
- β = Intercept = value of Y when X=0



Fitting a line

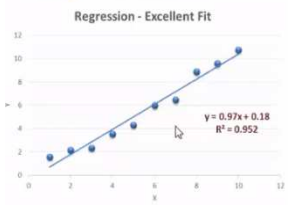
- Given a scatter plot of Y vs X, fit a straight line through the points so that the sum of square of vertical distances between the points and the line (called residuals) is minimized
- Best line = least residuals
- A line can always be fitted for any set of points

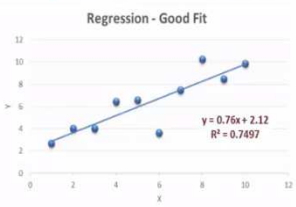


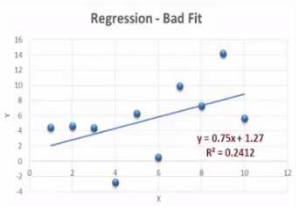


Goodness of Fit

- R-squared measures how close the data is to the fitted line
- R-squared varies from 0 to 1. The higher the value, the better the fit
- You can always fit a line. Use R-squared to see how good the fit is
- Higher correlation usually leads to better fit







Multiple Regression

- When there are more than one independent variable that is used to predict the dependent variable.
- The equation $Y = \beta + \alpha_1 * X_1 + \alpha_2 * X_2 + \ldots + \alpha_p * X_p$
- Same process used for prediction as a single independent variable
- Different predictors have different levels of impact on the dependent variable

Using Linear Regression for ML

- ML Technique to predict continuous data – supervised learning
- Predictors and outcomes provided as input
- Data analyzed (training) to come up with a linear equation
  - Coefficients
  - Intercept
  - R-squared
- Linear equation represents to model.
- Model used for prediction
- Typically fast for model building and prediction

Linear Regression - Summary

Advantages

- Fast
- Low cost
- Excellent for linear relationships
- Relatively accurate Continuous variables

Shortcomings

- Only numeric/ continuous variables
- Cannot model non-linear / fuzzy relationships
- Sensitive to outliers

Used in

- Oldest predictive model used in a wide variety of applications to predict continuous values

6. R Use Case : Linear Regression 18:01

Linear Regression exercise – predicting miles per gallon

Problem statement

The input data set contains data about details of various car models. Based on the information provided, the goal is to come up with a model to predict Miles-per-gallon of a given model.

Techniques Used

1. Linear Regression (multi-variate)
2. Data Imputation
3. Variable Reduction

Data Engineering & Analysis

```
R program => C:/Users/User/Workspace/RprogProjects/AppliedDataScience_5
```

```
> auto_data <- read.csv("auto-miles-per-gallon.csv", stringsAsFactors = TRUE)
> str(auto_data)
'data.frame':   398 obs. of  8 variables:
 $ MPG         : num  18 15 18 16 17 15 14 14 14 15 ...
 $ CYLINDERS   : int  8 8 8 8 8 8 8 8 8 8 ...
 $ DISPLACEMENT: num  307 350 318 304 302 429 454 440 455 390 ...
 $ HORSEPOWER  : Factor w/ 94 levels "?","100","102",..: 17 35 29 29 24 42 47
 $ WEIGHT      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
 $ ACCELERATION: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
 $ MODELYEAR   : int  70 70 70 70 70 70 70 70 70 70 ...
 $ NAME        : Factor w/ 305 levels "amc ambassador brougham",..: 50 37 232
```

str(), summary(), head(), tail()

Data Cleansing

1. The ranges of values in each of the variables (columns) look ok without any kind of outliers
2. Horsepower is a number and R should have shown the quartiles like other numeric variables. It is being recognized as factor. Also, str() shows "?" as one of the values. So this means, the ? values should be imputed. We will replace the "?" with the mean value for Horsepower.

```
auto_data <- read.csv("auto-miles-per-gallon.csv", stringsAsFactors =
TRUE)
```

=>

```
$ HORSEPOWER  : Factor w/ 94 levels "?","100","102",..: 17 35 29 29 24 42 47
```

```
auto_data[auto_data$HORSEPOWER == '?',]
```

A	B	C	D	E	F	G	H
MPG	CYLINDERS	DISPLACEMENT	HORSEPOWER	WEIGHT	ACCELERATION	MODELYEAR	NAME
25	4	98	?	2046	19	71	ford pinto
21	6	200	?	2875	17	74	ford maverick
40.9	4	85	?	1835	17.3	80	renault lecar deluxe
23.6	4	140	?	2905	14.3	80	ford mustang cobra
34.5	4	100	?	2320	15.8	81	renault 18i
23	4	151	?	3035	20.5	82	amc concord dl

Cleansing

=> replacing ? (NA) by mean of HP

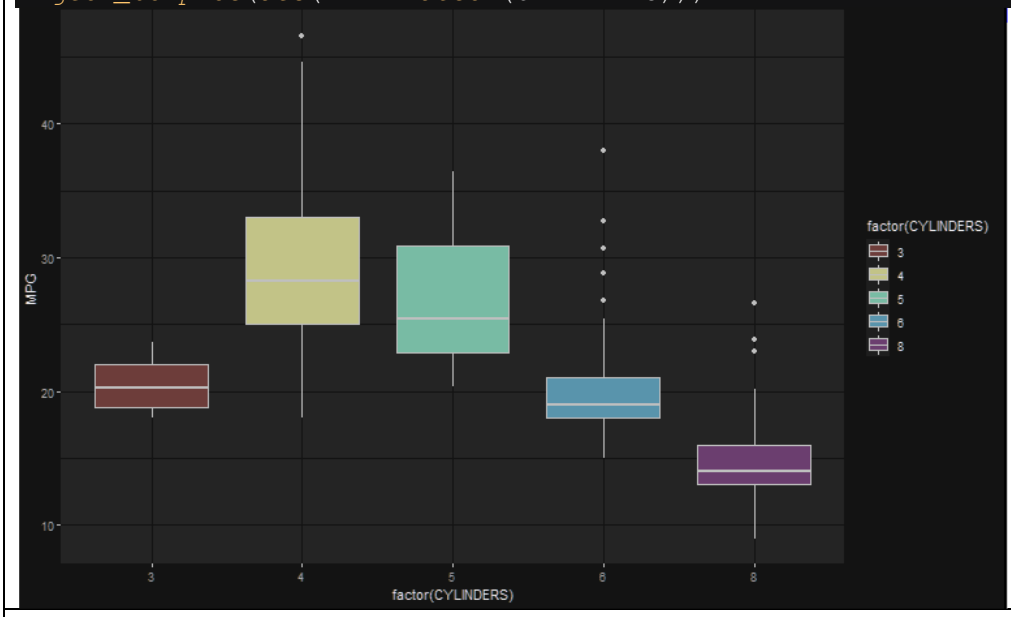
```
# except in column HORSEPOWER where we found some '?'
(33,127,331,337,355,375)
auto_data[auto_data$HORSEPOWER == '?',]
# convert ? to 1 via as.numeric !!!
auto_data$HORSEPOWER <- as.numeric(auto_data$HORSEPOWER)
auto_data[auto_data$HORSEPOWER == 82,]
print(auto_data$HORSEPOWER) # NA values are displayed
```

```
for (i in c(33,127,331,337,355,375)) {
  print(auto_data[i,]) # HORSEPOWER is NA
}
# now, we will replace NA value with the mean of all the HP !
hp_mean <- mean(auto_data$HORSEPOWER,na.rm="TRUE")
hp_mean  # i.e. 104.4694 ok - Mean:104.5 computed by summary
auto_data$HORSEPOWER[is.na(auto_data$HORSEPOWER)] <- hp_mean
for (i in c(33,127,331,337,355,375)) {
  print(auto_data[i,])   # HORSEPOWER is now 104.4694
}
summary(auto_data)
```

Exploratory Data Analysis

```
library(ggplot2)
ggplot(auto_data, aes(factor(CYLINDERS), MPG)) +
    geom_boxplot( aes(fill=factor(CYLINDERS)))
```

```
ggplot(auto_data, aes(factor(CYLINDERS),MPG)) +
  geom_boxplot(aes(fill=factor(CYLINDERS)))
```



So, when the nbr. of cyclinders increase, MPG decrease (pattern).

Dots in white are outliers ⇔ BoxPlot – Check for outliers

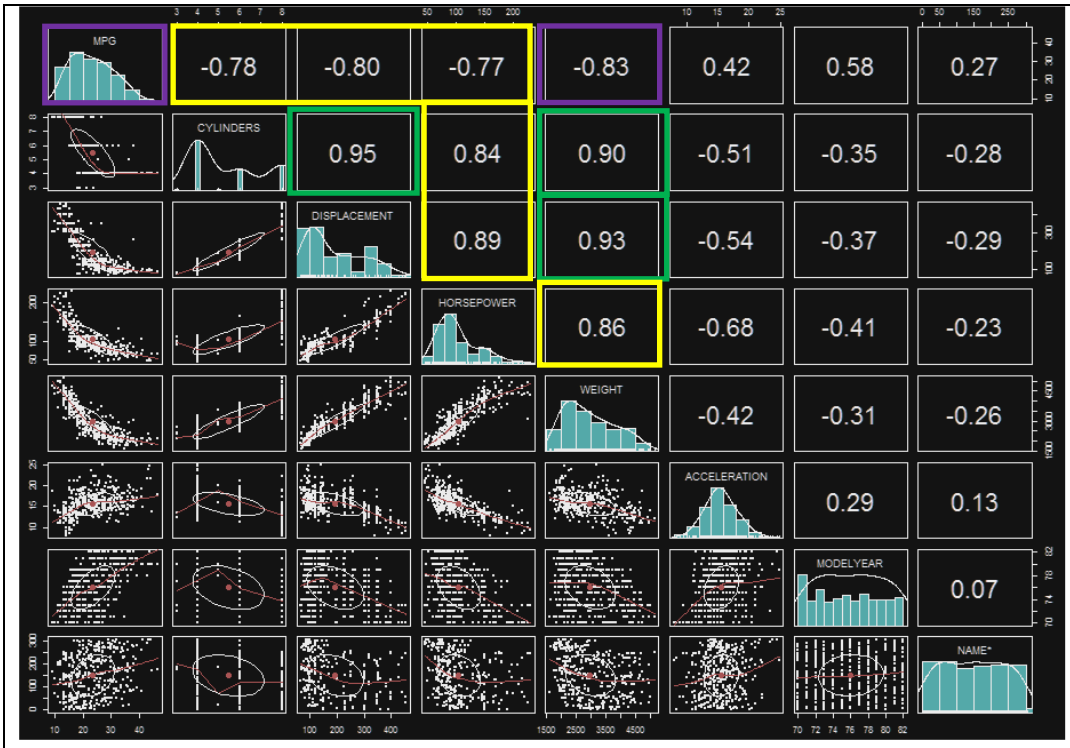
Correlation

```
library(psych)
pairs.panels(auto_data)
```

=>

Once you do correlations, it is important to find out domain (automobiles in this case) as to why they exist. In this example, we are trying to predict miles-per-gallon. The chart shows the Pearson correlation co-efficient (range -1 to + 1).

```
      MPG        CYLINDERS   DISPLACEMENT HORSEPOWER    WEIGHT    ACCELERATION  MODELYEAR       NAME
```



Ref. https://www.rdocumentation.org/packages/ggplot2/versions/3.3.2/topics/aes

High correlation between CYLINDERS and WEIGHT
High correlation between CYLINDERS and DISPLACEMENT
High correlation between WEIGHT and DISPLACEMENT
Good correlation between WEIGHT/CYLINDERS/DISPLACEMENT and HORSEPOWER

- Number of Cylinders has a high negative correlation to MPG (As Cylinders increase, MPG decreases). This is as expected.
- Same is the case with Displacement.
- The more the weight the less the acceleration. This also has a logical explanation
- Name has little correlation to MPG. True. This can be ignored.

Cylinders, Displacement and Weight have high correlation amongst themselves and hence one is a proxy of the other two. To limit the number of variables, we will eliminate displacement and cylinders and just keep Weight alone

```
auto_data$DISPLACEMENT <- NULL
auto_data$CYLINDERS <- NULL
```

Reduce from 3 variables to 1 => just keep one (Weight)

Modelling & Predicting

lm() => https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm

```
lm_model <- lm(MPG ~ . ,auto_data[1:5])
summary(lm_model, TRUE, TRUE, TRUE)
summary.lm(lm_model)
```

```
Call:
lm(formula = MPG ~ ., data = auto_data[1:5])

Residuals:
    Min      1Q  Median      3Q     Max 
-8.7152 -2.3362 -0.1052  2.0100 14.2742 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -15.599843   4.587024  -3.401 0.000741 ***
HORSEPOWER     0.005231   0.012873   0.406 0.684677    
WEIGHT        -0.006757   0.000454 -14.884  < 2e-16 ***
ACCELERATION   0.083131   0.096857   0.858 0.391257    
MODELYEAR      0.754431   0.051552  14.634  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.44 on 393 degrees of freedom
Multiple R-squared:  0.8083,	Adjusted R-squared:  0.8063 
F-statistic: 414.1 on 4 and 393 DF,  p-value: < 2.2e-16
```

The model gives the Intercept and co-efficients required for the linear regression equation. The R-Squared value is .8, which is a very good fit for the problem.

summary.lm

mathematical equation can be generalized as follows:

$$Y = \beta 1 + \beta 2X + \epsilon$$

where, β1 is the intercept and β2 is the slope (2 regression coefficients).

ϵ is the error term, the part of Y the regression model is unable to explain.

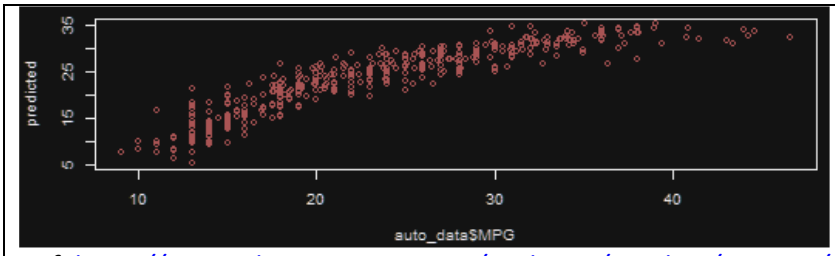
Ref. http://r-statistics.co/Linear-Regression.html

Testing

To test the accuracy of the equation, let us apply the equation on the same data set and predict the MPG for each record. Since we already know the actual value, let us compare the predicted value with the actual value and see the conformance/ error in the prediction.

```
predicted <- predict.lm(lm_model, auto_data)
summary(predicted)
```

```
> summary(auto_data[1])
      MPG
 Min.   : 9.00
 1st Qu.:17.50
 Median :23.00
 Mean   :23.51
 3rd Qu.:29.00
 Max.   :46.60
> summary(predicted)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  5.148  18.451  24.592  23.515  29.142  35.359 
```



Ref. https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/plot

```
# linear model
lm_model <- lm(MPG ~ . ,auto_data[1:5])
# prediction based on the model and actual data
predicted <- predict.lm(lm_model, auto_data)
summary(predicted)

plot(auto_data$MPG, predicted, col = "red")
# correlation
cor(auto_data$MPG, predicted)
```

[1] 0.8990293

The plot of prediction vs actual follows a diagnal straight line, which means this is very good prediction.
The correlation co-efficient is also very high, which again means very good prediction.

Conclusions

The model built can predict MPG with an accuracy of about 90% (based on the correlation co-efficient)

7. Decision Trees 10:42

Decision Trees (DT) - Overview

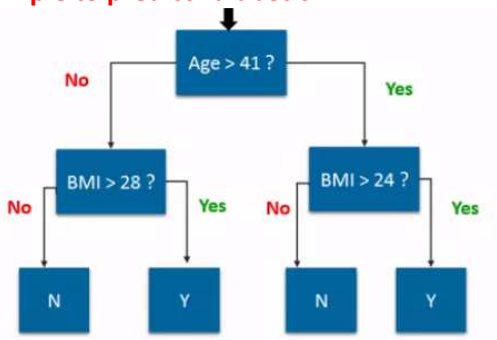
- The simplest, easy to understand and easy to explain ML technique.
- Predictor variables are used to build a tree that would progressively predict the target variable
  - Trees start with a root node that start the decision making process
  - Branch nodes refine the decision process
  - Leaf nodes provide the decisions
- Training data is used to build a decision tree to predict the target
- The tree becomes the model that is used to predict on new data

More about decision tree >> link
DT algorithms >> link1 link2

Ref. https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision

Decision Trees - Example to predict "diabetic"

Age	BMI	Is Diabetic
24	22	N
33	28	N
41	36	Y
48	24	N
58	31	Y
61	35	Y



Predicting "Is Diabetic ?"

Decision Trees - Choosing the right predictors

- The depth of trees are highly influenced by the sequence in which the predictors are chosen for decisions
- Using predictors with high selectivity gives faster results
- ML implementations automatically make decisions on the sequence /preference of predictors

Decision Trees - Summary

Advantages

- Easy to interpret and explain
- Works with missing data
- Sensitive to local variations
- Fast

Shortcomings

- Limited Accuracy
- Bias builds up pretty quickly
- Not good with large predictors

Used in

- Credit approvals
- Situations with legal needs to explain decisions
- Preliminary categorization

8. R Use Case : Decision Trees 19:36

Decision Trees exercise - Predicting Flower types

Problem Statement

The input data is the iris dataset. It contains recordings of information about flower samples. For each sample, the petal and sepal length and width are recorded along with the type of the flower. We need to use this dataset to build a decision tree model that can predict the type of flower based on the petal and sepal information.

Techniques Used

1. Decision Trees - C5.0
2. Training and Testing
3. Confusion Matrix

Data engineering & Analysis

```
library(datasets)
data(iris)
iris_data <- iris
str(iris)
summary(iris)
```

```
> str(iris_data)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
```

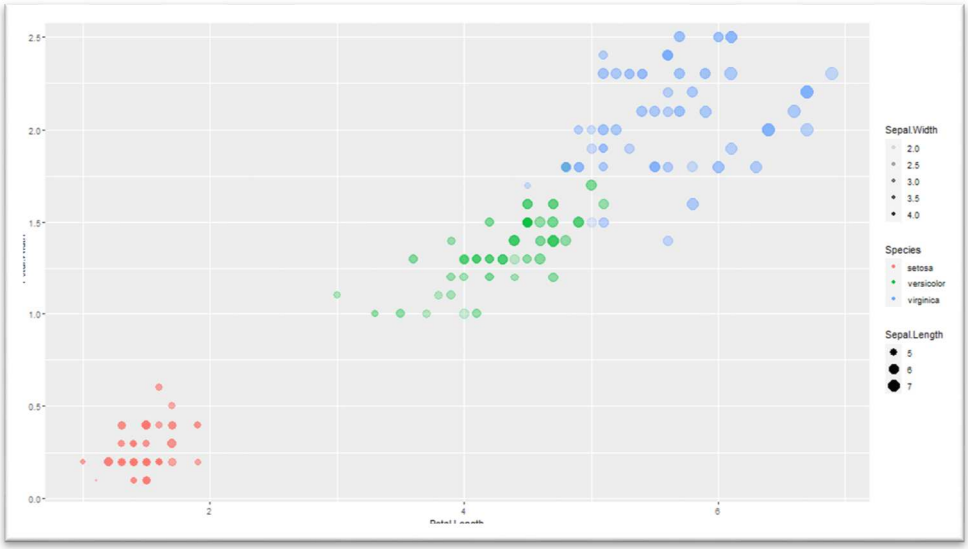
No cleansing required

Data Cleansing

1. The ranges of values in each of the variables (columns) look ok without any kind of outliers
2. There is equal distribution of the three classes - setosa, versicolor and virginia

Exploratory Data Analysis

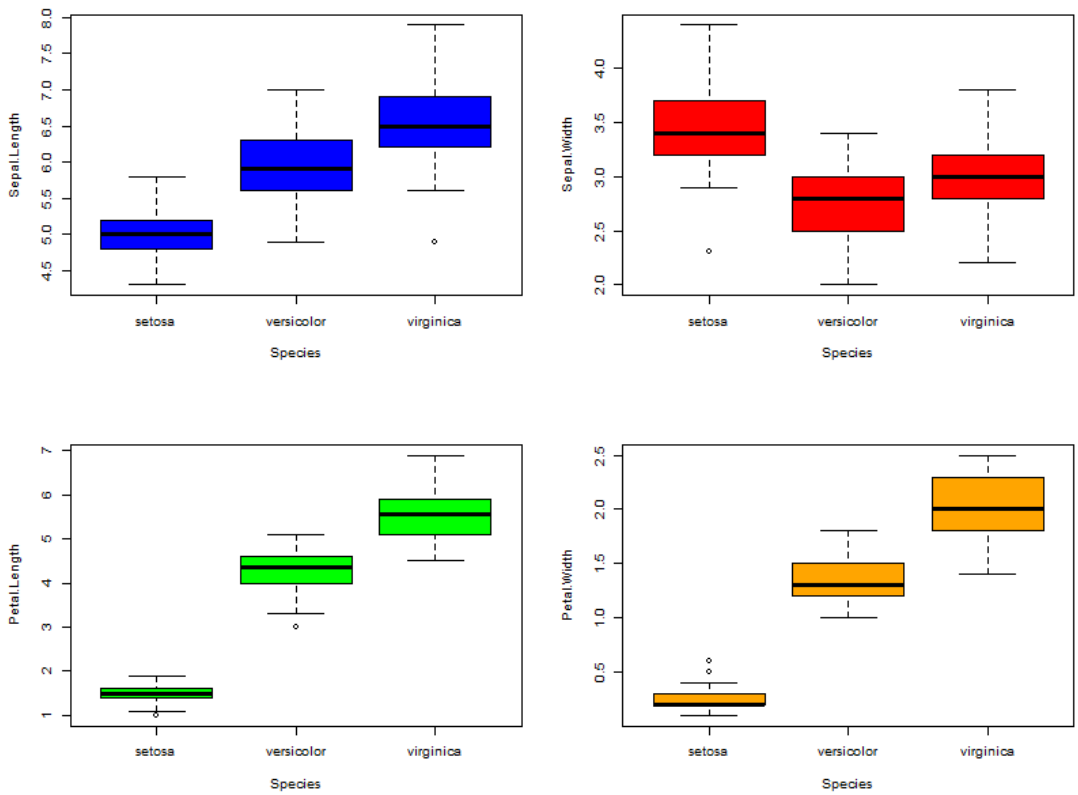
```
library(ggplot2)
ggplot(data = iris_data) +
 geom_point(aes(x = Petal.Length, y = Petal.Width, color = Species, size
= Sepal.Length, alpha = Sepal.Width))
```



So, Petal.Length and Petal.Width are good indicators to distinguish the species.

Ref. https://juba.github.io/tidyverse/08-ggplot2.html

```
#Box plots
par(mfrow=c(2, 2))  # divide graph area in 2 columns
boxplot(data = iris_data, Sepal.Length ~ Species, col = "blue")
boxplot(data = iris_data, Sepal.Width ~ Species, col = "red")
boxplot(data = iris_data, Petal.Length ~ Species, col = "green")
boxplot(data = iris_data, Petal.Width ~ Species, col = "orange")
```

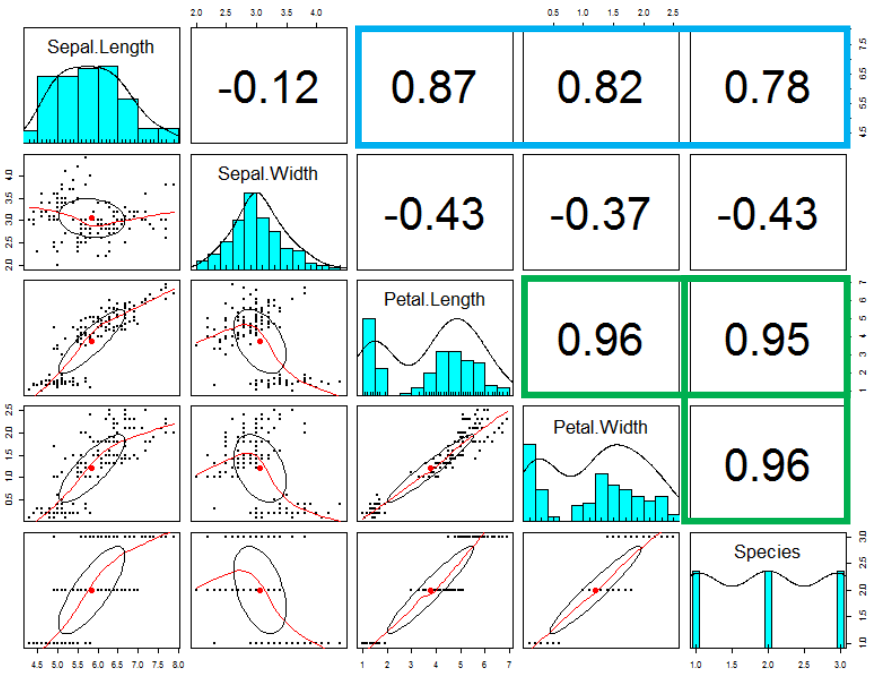


Again, Petal.Length and Petal.Width are very good indicators to distinguish the species. Septal.Length is ok too…

All 3 except Sepal Width seem to bring the significant differenciation between the 3 classes

Correlations

```
library(psych)
pairs.panels(iris_data)
```



All 3 except Sepal Width seem to bring the significant differenciation between the 3 classes

The correlation co-efficients confirm the findings of the Exploratory Data Analysis.

Modelling & Predictions

Split Training and Testing

Split training and testing datasets in the ratio of 70-30

```
library(caret)
# creation of a vector with the training data set (70%)
inTrain <- createDataPartition(y=iris_data$Species, p=0.7, list=FALSE)
class(inTrain); str(inTrain)
# training: 105 obs vs. testing : 45 obs
training <- iris_data[inTrain,]
class(training); str(training); dim(training)
table(training$Species)

testing <- iris_data[-inTrain,]
class(testing); str(testing); dim(testing)
table(testing$Species)
```

Model building

Build model based on the training data

```
#Ref. https://cran.r-project.org/web/packages/C50/index.html
library(C50)
# build a model based on training datasert (target is Species)
model <- C5.0(training[-5], training$Species)
summary(model)
```

The model clearly shows how the decision tree looks like. This is one of the advantages of decision trees.

```
C5.0 [Release 2.07 GPL Edition]  - Mon Oct 26 11:40:36 2020
  Class specified by attribute `outcome'
  Read 105 cases (5 attributes) from undefined.data
Decision tree:
  Petal.Width <= 0.4: setosa (35)
  Petal.Width > 0.4:
      :... Petal.Length <= 4.8: versicolor (33/1)
           Petal.Length > 4.8: virginica (37/3)
Evaluation on training data (105 cases):
          Decision Tree  - Size      Errors
                            3           4     ( 3.8%)   <<
               (a)   (b)   (c)    <-classified as
                35              (a): class setosa
                     32     3    (b): class versicolor
                      1    34    (c): class virginica
            Attribute usage:        100.00%    Petal.Width ,            66.67%     Petal.Length
```

Testing

Now let us predict the class for each sample in the test data. Then compare the prediction with the actual value of the class.

```
predicted <- predict(model, testing)
class(predicted)
str(predicted)
table(predicted)
confusionMatrix(predicted, testing$Species)
```

```
Confusion Matrix and Statistics – Reference

Prediction   setosa versicolor virginica
 setosa        13        0        0
 versicolor     2       14        2
 virginica      0        1       13
Overall Statistics
    Accuracy : 0.8889
      95% CI : (0.7595, 0.9629)
   No Information Rate : 0.3333
   P-Value [Acc > NIR] : 1.408e-14
    Kappa : 0.8333

Mcnemar's Test P-Value : NA
Statistics by Class:      setosa     versicolor     virginica
Sensitivity               0.8667         0.9333        0.8667
Specificity               1.0000         0.8667        0.9667
Pos Pred Value            1.0000         0.7778        0.9286
Neg Pred Value            0.9375         0.9630        0.9355
Prevalence                0.3333         0.3333        0.3333
Detection Rate            0.2889         0.3111        0.2889
Detection Prevalence      0.2889         0.4000        0.3111
Balanced Accuracy         0.9333         0.9000        0.9167
```

The model shows very high accuracy. The reason why the accuracy is so high is because, the data itself has very strong signals (seperation between the classes). Sepal.Length and Sepal.Width have very high correlations and they are used in the decision tree. In order to see how the tree will behave if it only had Sepal.Length and Sepal.Width, let us remove that data and see how accurate the tree is.

```
#get only Sepal Length, width and species
sub_data <- iris_data[, c(1,2,5)]
```

```
# Modelling with a sub-set of data (just Sepal & Species)
sub_data <- iris_data[c(1,2,5)]
sub_inTrain <- createDataPartition(y=sub_data$Species, p=0.7, list=FALSE)
sub_training <- sub_data[sub_inTrain,]
sub_testing <- sub_data[-sub_inTrain,]
model2 <- C5.0(sub_training[-3], sub_training$Species)
```

```
summary(model2)
sub_predicted <- predict(model2, sub_testing)
confusionMatrix(sub_predicted, sub_testing$Species)
```

Model2 is less accurate than first model

Decision Tree => Errors 4 25 (23.8%)

(a)	(b)	(c)	<- classified as
33	2		(a): class setosa
	15	20	(b): class versicolor
	3	32	(c): class virginica

Overall Statistics => Accuracy : 0.6889 … Kappa : 0.5333

You will notice that the decision tree itself has become more complex and the accuracy dropped significantly

Conclusions

. Irrespective of the algorithm used, we need high correlations between the predictor and target variables for good predictions

9. Naive Bayes Classifier 19:21

Naïves Bayes - theorem (overview)

- Probability of an event A = P(A) is between 0 and 1
- Baves' theorem gives the conditional probability of an event A given event B has already occurred.

  P(A/B) = P(A intersect B) * P (A) /P(B)
- Example
  - There are 100 patients
  - Probability of a patient having diabetes is P(A) = .2
  - Probability of patient having diabetes (A) given that the patient's age is > 50 (B) is P(A/B) = .4

Other Refs.

1) https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne

2) https://le-datascientist.fr/les-algorithmes-de-naives-bayes

Naïves Bayes – Classifications

- Application of Bayes' theorem to ML
- The target variable becomes event A
- The predictors become events B1 – Bn
- We try to find P(A / B1-Bn)

Age	BMI	Is Diabetic	
24	22	N	Probability of Is Diabetic = Y given that Age = 24 and BMI = 22
41	36	Y	Probability of Is Diabetic – Y given that Age = 41 and BMI = 36

Naïves Bayes - Model building and prediction

- The model generated stores the conditional probability of the target for every possible value of the predictor.

	Overall	Age						Gender	
Salary		1 to 20	20 to 30	30 to 40	40 to 50	50 to 60	60 to 100	Female	Male
< 50K	.75	0.1	0.3	0.25	0.17	0.1	0.08	0.39	0.61
> 50K	.25	0.03	0.08	0.3	0.32	0.2	0.07	0.15	0.85
Overall		.08	.24	.26	.21	.12	.08	.33	.67

- When a new prediction needs to be done, the conditional probabilities are applied using Bayes' formula to find the probability
  - To predict for Age = 25
  - P(Salary < 50K / Age=25) = 0.3 * 0.75 / 0.24 = ~ 0.92
  - P(Salary > 50K / Age=25) = 0.08 * 0.25 / 0.24 = ~ 0.08

Naïves Bayes – Summary

Advantages

- Simple and fast
- Works well with noisy and missing data
- Provides probabilities of the result
- Very good with categorical data

Shortcomings

- Limited Accuracy
- Expects predictors to be independent
- Not good with large numeric features

Used in

- Medical diagnosis
- Spam filtering
- Document classification
- Sports predictions

10. R Use Case : Naive Bayes 19:12

Naïve Bayes exercise – Spam Filtering

Problem Statement

The input data is a set of SMS messages that has been classified as either "ham" or "spam". The goal of the exercise is to build a model to identify messages as either ham or spam.

Ham email is an email that is wanted by the recipient and is not considered spam.

Techniques Used

1. Naive Bayes Classifier
2. Training and Testing
3. Confusion Matrix
4. Text Pre-Processing

Data Engineering & Analysis

Loading and understanding the dataset

```
sms_data <- read.csv("sms_spam_short.csv", stringsAsFactors = FALSE)
# data.frame - 500 obs. of  2 variables
class(sms_data); dim(sms_data); str(sms_data)
# type and text
head(sms_data[1]); head(sms_data[2])
# convert type column as factor
sms_type_factors <- as.factor(sms_data$type)
class(sms_type_factors)
```

```
str(sms_type_factors)
sms_data$type <- sms_type_factors
str(sms_data) # type are now seen as factor (2 levels)
summary(sms_data)  # 437 ham / 63 spam
```

Data Cleansing

The dataset contains raw text. The text need to be pre-processed and converted into a Document Term Matrix before it can be used for classification purposes. The steps required are documented as comments below

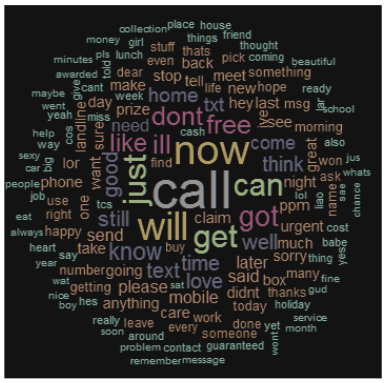
```
# data cleansing on email messages via Corpus
library(tm)
email_msg_corpus <- Corpus(VectorSource(sms_data$text))
class(email_msg_corpus)
inspect(email_msg_corpus) # [1:3])  # inspect first 3 on 500 documents
# convert to lowercase
refined_msg_corpus <- tm_map(email_msg_corpus,
content_transformer(tolower))
# removing punctuation, white space, numbers, stop and specific words
refined_msg_corpus <- tm_map(refined_msg_corpus, removePunctuation)
refined_msg_corpus <- tm_map(refined_msg_corpus, removeNumbers)
refined_msg_corpus <- tm_map(refined_msg_corpus, stripWhitespace)
refined_msg_corpus <- tm_map(refined_msg_corpus, removeWords,
stopwords())
refined_msg_corpus <- tm_map(refined_msg_corpus, removeWords,
c("else","the","are","for"))
refined_msg_corpus <- tm_map(refined_msg_corpus, removeWords,
c("as","they","has","a","his"))
refined_msg_corpus <- tm_map(refined_msg_corpus, removeWords,
c("on","when","is","in","already"))
# messages now look more cleaner ...
inspect(refined_msg_corpus)

# creation of a document-term sparse matrix
dtm <- DocumentTermMatrix(refined_msg_corpus)
class(dtm)  # "DocumentTermMatrix"    "simple_triplet_matrix"
str(dtm)    # Docs , Terms
# too many columns !!! ~2000
dim(dtm)    # 500    1967
# reduce to terms taht are occuring at least 10 times in this refined
corpus
filtered_dtm <- DocumentTermMatrix(refined_msg_corpus,
list(dictionary=findFreqTerms(dtm,10)))
filtered_dtm
dim(filtered_dtm)   # 500  59
t(inspect(filtered_dtm)[1:5,])
```

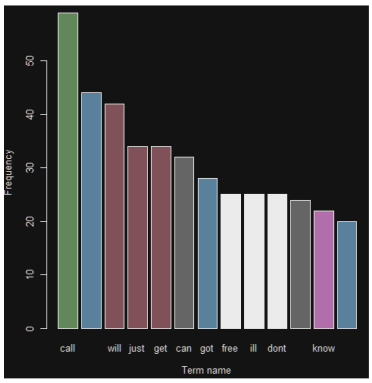
Exploratory Data Analysis

The following example shows a word cloud for both ham and spam message. The size of words shown in the word cloud is based on the frequency of occurance. It will clearly show that there is a difference in the most common occuring words between these types

```
# Build a matrix 'm' from 'filtered_dtm'
# rem. in a matrix all the elements are the same type of data
# ref. http://www.sthda.com/french/wiki/text-mining-et-nuage-de-mots-
avec-le-logiciel-r-5-etapes-simples-a-savoir
#      https://www.datamentor.io/r-programming/matrix/
m <- as.matrix(filtered_dtm)
class(m); attributes(m) # $dim 500 59, $dimnames$Docs , $dimnames$Terms
```

```r
head(colnames(m)) # Terms
head(rownames(m)) # Docs ids
print(head(m))

# Build a vector that is ordered descrease with sum of each term
occurence
v <- sort(colSums(m),decreasing=TRUE)
head(v); class(v[1]); str(v); names(v)
print(v[1]); print(names(v[1]));

# Convert 'v' to a data frame <term,ferq>
# rem. in a data frame the columns contain different types of data
d <- data.frame(term = names(v), freq = v)
head(d, 5) # top five term occurences

# Bar plot of 'd'
# Ref. https://www.statmethods.net/graphs/bar.html
barplot(d$freq, names.arg = d$term, col = d$freq,
        #main ="Most frequent terms",
        xlab ="Term name", ylab = "Frequency",
        horiz = FALSE)

# findFreqTerms(x, lowfreq = 0, highfreq = Inf)
# Ref. https://www.rdocumentation.org/packages/tm/versions/0.7-
7/topics/findFreqTerms
findFreqTerms(dtm, lowfreq = 40, highfreq = Inf)
findAssocs(dtm, terms = "call", corlimit = 0.3)

# world cloud
# Refs.
https://www.rdocumentation.org/packages/wordcloud/versions/2.6/topics/wor
dcloud
#
https://www.rdocumentation.org/packages/RColorBrewer/versions/1.1-
2/topics/RColorBrewer
library(wordcloud)
pal_colors <- brewer.pal(9, "Dark2")
display.brewer.pal(9, "Dark2")
wordcloud(refined_msg_corpus, min.freq = 5, colors = pal_colors,
random.order=FALSE)
wordcloud(refined_msg_corpus[sms_data$type == "ham"], min.freq = 5,
colors = pal_colors, random.order=FALSE)
wordcloud(refined_msg_corpus[sms_data$type == "spam"], min.freq = 5,
colors = pal_colors, random.order=FALSE, scale = 2)
```





Modeling & Prediction

Split Training and Testing

Split training and testing datasets in the ratio of 70-30

```
# Ref. http://topepo.github.io/caret/index.html (lattice, ggplot2)
# Classification And REgression Training
library(caret)
# ref. https://www.rdocumentation.org/packages/caret/versions/6.0-
86/topics/createDataPartition
inTrain <- createDataPartition(y=sms_data$type, p=0.7, list=FALSE)
#matrix
dim(inTrain); inTrain[1:8,1]
# splitting the raw data between training and testing
train_rawdata <- sms_data[inTrain,]
test_rawdata <- sms_data[-inTrain,]
dim(raw_data_train);dim(raw_data_test)
str(raw_data_train);str(raw_data_test)
raw_data_train[1:3,1:2]
# splitting the corpus between training and testing
train_corpus <- refined_msg_corpus[inTrain]
test_corpus <- refined_msg_corpus[-inTrain]
#splitting the refined/filtered dtm between training and testing
train_dtm <- filtered_dtm[inTrain,]
test_dtm <- filtered_dtm[-inTrain,]
#converting numeric data to factors (nbr. of occurences vs. Yes/No found
in a doc)
# via 'conv_counts' function
conv_counts <- function(x) {
  y <- ifelse(x > 0, 1, 0)
  x <- factor(y, levels = c(0,1), labels = c("No", "Yes") )
}
#applying this function on our 2 dtm (on all columns of a matrix)
# ref.
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/apply
train <- apply(train_dtm, MARGIN = 2, FUN = conv_counts)
test <- apply(test_dtm, MARGIN = 2, FUN = conv_counts)
# converting to data frames
df_train <- as.data.frame(train)
df_test <- as.data.frame(test)
df_train[1:5,1:6]
str(df_train)  # 351 obs of 59 var
str(df_test)   # 149 obs of 59 var
# add 'type' column
df_train$hamspam_type <- train_rawdata$type
df_test$hamspam_type <-  test_rawdata$type
str(df_train)  # 351 obs of 60 var
str(df_test)   # 149 obs of 60 var
```

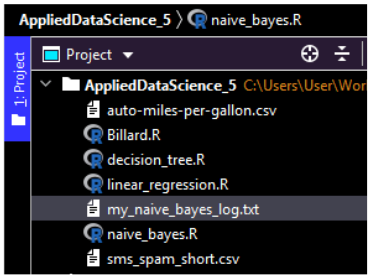
df_test

	dont	day	home	new	great	hamspam_type
470	No	No	No	No	No	ham
473	No	No	No	No	No	ham
477	No	No	No	No	No	ham
482	No	No	No	No	No	ham
489	No	No	No	No	No	ham
490	No	No	No	No	No	spam

Model Building

Build model based on the training data

```
library(e1071)
nb_model <- naiveBayes(df_train[,-60],df_train$hamspam_type)
filename <- "my_naive_bayes_log.txt"
# ref. http://www.cookbook-r.com/Data_input_and_output/Writing_text_and_output_from_analyses_to_a_file/
sink(filename) # prepare to write into output file
nb_model
sink() # return to R Console
```




```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = df_train[, -60], y = df_train$hamspam_type)

A-priori probabilities:
df_train$hamspam_type
      ham      spam
0.8717949 0.1282051

Conditional probabilities:
                      good
df_train$hamspam_type         No        Yes
                ham  0.95751634 0.04248366
                spam 0.97777778 0.02222222
```

...

```
                      call
df_train$hamspam_type         No        Yes
                ham  0.93137255 0.06862745
                spam 0.60000000 0.40000000
```

Testing

Now let us predict the class for each sample in the test data. Then compare the prediction with the actual value of the class.

```
predicted <- predict(nb_model, df_test)
confusionMatrix(predicted, df_test$hamspam_type)
sink(filename, append=TRUE) # prepare to write into output file
confusionMatrix(predicted, df_test$hamspam_type)
sink() # return to R Console
```

```
Confusion Matrix and Statistics

          Reference
Prediction ham spam
      ham  127    3
      spam   4   15

               Accuracy : 0.953
                 95% CI : (0.9056, 0.9809)
    No Information Rate : 0.8792
    P-Value [Acc > NIR] : 0.001811

                  Kappa : 0.784

 Mcnemar's Test P-Value : 1.000000
```

```r
# +
https://www.rdocumentation.org/packages/gmodels/versions/2.18.1/topics/CrossTable
library(gmodels)
CrossTable(predicted, df_test$hamspam_type,
           prop.chisq = FALSE, prop.t = FALSE,
           dnn = c('predicted', 'actual'))
```

```
Total Observations in Table:  149


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |       127 |         3 |       130 |
             |     0.977 |     0.023 |     0.872 |
             |     0.969 |     0.167 |           |
-------------|-----------|-----------|-----------|
        spam |         4 |        15 |        19 |
             |     0.211 |     0.789 |     0.128 |
             |     0.031 |     0.833 |           |
-------------|-----------|-----------|-----------|
Column Total |       131 |        18 |       149 |
             |     0.879 |     0.121 |           |
-------------|-----------|-----------|-----------|
```

How to test the model on real message???

11. Random Forests 10:31

Random Forest – Overview

- Random Forest is one of the most popular and accurate algorithms
- It is an Ensemble method based on decision trees
  - Builds multiple models – each model a decision tree
  - For prediction – each tree is used to predict an individual result
  - A vote is taken on all the results to find the best answer

Random Forest – How it works?

- Lets say the dataset contains m samples (rows) and n predictors (columns)
- x trees are built, each with a subset of data
- For each tree, a subset of m rows and n columns are chosen randomly.
- For example, if the data has 1000 rows and 5 columns, each tree is built using 700 rows and 3 columns
- The data subset is used to build a tree
- For prediction, new data is passed to each of the x trees and x possible results obtained
- For example, if we are predicting buy=Y/N and there are 500 trees, we might get 350 Y and 150 N results
- The most found result is the aggregate prediction.

Ref. https://fr.wikipedia.org/wiki/For%C3%AAt_d%27arbres_d%C3%A9cisionnels
https://cran.r-project.org/web/packages/randomForest/index.html

Random Forest – Summary

Advantages

- Highly accurate
- Efficient on large number of predictors
- Fully parallelizable
- Very good with missing data

Shortcomings

- Time and Resource consuming
- For categorical variables, bias might exist if levels are disproportionate

Used in

- Scientific Research
- Competitions
- Medical Diagnosis

12. R Use Case : Random Forests 18:47

Random Forest – Prospective customers of a bank

Problem Statement

The input data contains surveyed information about potential customers for a bank. The goal is to build a model that would predict if the prospect would become a customer of a bank, if contacted by a marketing exercise.

Techniques Used

1. Random Forests
2. Training and Testing
3. Confusion Matrix
4. Indicator Variables
5. Binning
6. Variable Reduction

Data Engineering & A

Loading and understandi

4. Indicator Variables
5. Binning
6. Variable Reduction

```
bank_data <- read.                    , header = TRUE, as.is = FALSE)  # i.e.
sep = ";" + convert to factors
str(bank_data)
summary(bank_data)
head(bank_data)
```

```
> head(bank_data)
  age         job marital education default balance housing loan  co
1  30  unemployed married   primary      no    1787      no   no cel
2  33    services married secondary      no    4789     yes  yes cel
3  35  management  single  tertiary      no    1350     yes   no cel
4  30  management married  tertiary      no    1476     yes  yes  un
```

No cleansing required

Correlations

Given the large number of predictors, we would like to start with a correlation analysis to see if some variables can be dropped

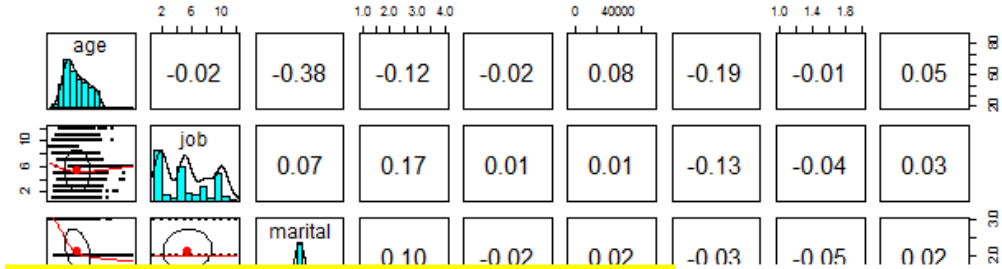
'data.frame': 4521 obs. of 17 variables:

1) $ age : int 30 33 35 30 59 35 36 39 41 43 ...
2) $ job : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
3) $ marital : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
4) $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
5) $ default : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
6) $ balance : int 1787 4789 1350 1476 0 747 307 147 221 -88 ...
7) $ housing : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
8) $ loan : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
9) $ contact : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
10) $ day : int 19 11 16 3 5 23 14 6 14 17 ...
11) $ month : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
12) $ duration : int 79 220 185 199 226 141 341 151 57 313 ...
13) $ campaign : int 1 1 1 4 1 2 1 2 2 1 ...
14) $ pdays : int -1 339 330 -1 -1 176 330 -1 -1 147 ...
15) $ previous : int 0 4 1 0 0 3 2 0 0 2 ...
16) $ poutcome : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
17) $ y : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

Correlation analysis in 2 parts (1->8 + 17 & 9 -> 16 + 17)

```
bank_data[,c(1:8,17)]
   age           job  marital education default balance housing loan   y
    30    unemployed  married   primary      no    1787      no   no  no
    33      services  married secondary      no    4789     yes  yes  no
    35    management   single  tertiary      no    1350     yes   no  no
```

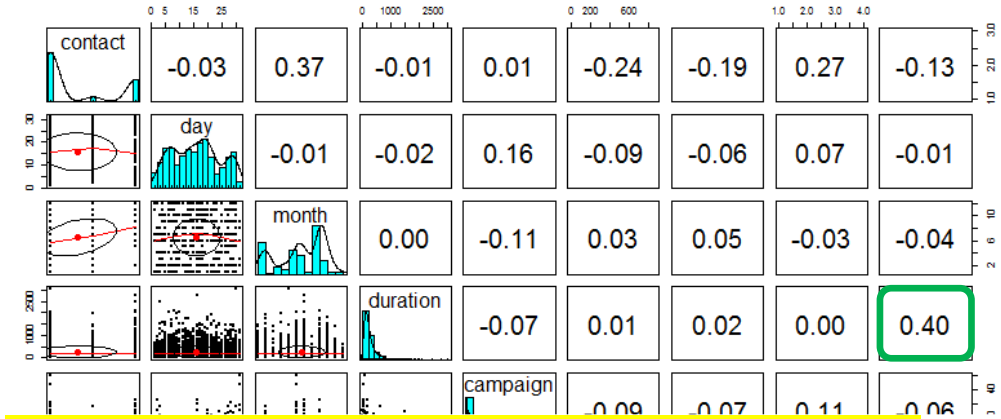
=>



No correlation between first 8 predictors (low, very low)

```
pairs.panels(bank_data[,c(9:16,17)])
```

=>



No correlation between first 8 predictors (low, very low), except for duration

We have weak predictors… but a combination of some of them can actually become strong !
SO, it's where Random Forest can be applied…

Based on the correlation co-efficients, let us eliminate default, balance, day, month, campaign, poutcome because of very low correlation. There are others too with very low correlation, but let us keep it for example sake.

1) $ age : int 30 33 35 30 59 35 36 39 41 43 ...
2) $ job : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
3) $ marital : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
4) $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
5) $ default : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
6) $ balance : int 1787 4789 1350 1476 0 747 307 147 221 -88 ...
7) $ housing : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
8) $ loan : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
9) $ contact : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
10) $ day : int 19 11 16 3 5 23 14 6 14 17 ...
11) $ month : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
12) $ duration : int 79 220 185 199 226 141 341 151 57 313 ...
13) $ campaign : int 1 1 1 4 1 2 1 2 2 1 ...
14) $ pdays : int -1 339 330 -1 -1 176 330 -1 -1 147 ...
15) $ previous : int 0 4 1 0 0 3 2 0 0 2 ...
16) $ poutcome : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
17) $ y : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

Eliminate variables with very low coeff < 0.1 (below 10%)

```
new data <-bank_data[, c(1:4,7:9,12,14,15,17)]
```

```
# Correlation between predictors?
library(psych)
pairs.panels(bank_data[,c(1:8,17)])
pairs.panels(bank_data[,c(9:16,17)])

# Keep predictors with coreff. above 0.1 (> 10%)
newbank_data <- bank_data[c(1:4,7:9,12,14,15,17)]
pairs.panels(newbank_data)
```

Do some transforamtions…

create bins on the age

```
# ref. cut -
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/cut
newbank_data$age <- cut(newbank_data$age, c(1,20,40,60,100))
```

Levels: (1,20] (20,40] (40,60] (60,100]

Do some transforamtions…

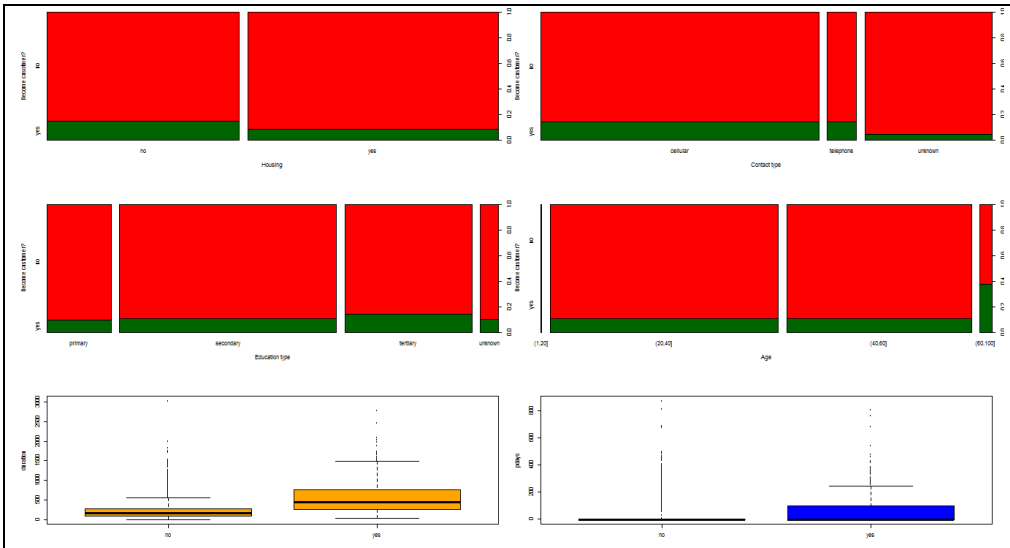
split mariage into three categories (is_divorced, is_single, is_married)

```
newbank_data$is_divorced <- ifelse(newbank_data$marital == "divorced" , 1, 0)
newbank_data$is_single   <- ifelse(newbank_data$marital == "single" , 1, 0)
newbank_data$is_married  <- ifelse(newbank_data$marital == "married", 1, 0)
newbank_data$marital      <- NULL  # to delete
```

```
str(newbank_data)
'data.frame':    4521 obs. of  13 variables:
$ age       : Factor w/ 4 levels "(1,20]","(20,40]",..: 2 2 2 2 3 2 2 2 3 3 ...
$ job       : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
$ education  : Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
$ housing    : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
$ loan       : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
$ contact    : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
$ duration   : int  79 220 185 199 226 141 341 151 57 313 ...
$ pdays      : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
$ previous   : int  0 4 1 0 0 3 2 0 0 2 ...
$ y         : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
$ is_divorced: num  0 0 0 0 0 0 0 0 0 0 ...
$ is_single  : num  0 0 1 0 0 1 0 0 0 0 ...
$ is_married : num  1 1 0 1 1 0 1 1 1 1 ...
```

Exploratory Data Analysis

```
par(mfrow=c(3,2), las=0)
plot(newbank_data$housing, newbank_data$y, xlab="Housing" , ylab="Become
cusotmer?" , col=c("darkgreen","red"))
plot(newbank_data$contact, newbank_data$y, xlab="Contact type" , ylab="Become
customer?" , col=c("darkgreen","red"))
plot(newbank_data$education, newbank_data$y, xlab="Contact type" , ylab="Become
customer?" , col=c("darkgreen","red"))
plot(newbank_data$age, newbank_data$y, xlab="Contact type" , ylab="Become
customer?" , col=c("darkgreen","red"))
boxplot(duration ~ y, data = newbank_data, col="orange" )
boxplot(pdays ~ y, data = newbank_data, col="blue" )
","red"))
```



We can see few insights:

- having no house gives a better chance to become a new customer
- telephone and mobile contact increase the chance to get a new customer
- being educate (tertiary) gives more chance to become a new customer
- boxplots shown many outliers

… no great confidence

Model building

```
library(caret)
inTrain <- createDataPartition(y=newbank_data$y, p=0.7, list=FALSE)
training_data <- newbank_data[inTrain,]
testing_data <- newbank_data[-inTrain,]
dim(training_data); dim(testing_data)
```

3165 13 1356 13

```
table(training_data$y);table(testing_data$y);
```

no	yes	no	yes
2800	365	1200	156

Radom Forest model

```
# Random forest
# ref. https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest
#  see example with 'iris' data set
library(randomForest)
model <- randomForest( y ~ . , data=training_data)
print(model)
```

randomForest(formula = y ~ . , data = training_data)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3
OOB estimate of error rate: 10.49%
Confusion matrix:

	no	yes	class.error
no	2719	81	0.02892857 (2%)
yes	251	114	0.68767123 (68%)

If the client is already a customer, then it's difficult to know if he will remains… (68% errors)

```
# ref. https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/importance
# Extract Variable Importance Measure - extractor function for variable importance measures as
# produced by randomForest.
importance(model)
```

round(importance(model),2)

	MeanDecreaseGini
age	25.94
job	61.40
education	25.79
housing	18.35
loan	9.02
contact	17.95
duration	220.49
pdays	49.97
previous	27.49
is_divorced	7.49
is_single	8.10
is_married	10.83

Testing

Now let us predict the class for each sample in the test data. Then compare the prediction with the actual value of the class.

Radom Forest test

```
predicted <- predict(model, testing_data)
str(predicted)
table(predicted)
cm <- confusionMatrix(predicted, testing_data$y)
print(cm)
cm$byClass
cm$overall
cm$positive
cm$table
```

```
predicted      no    yes
             1272     84
```

```
confusionMatrix(predicted, testing_data$y)
```

```
Reference - Prediction    no    yes
                    no  1162   110
                   yes    38    46
Accuracy : 0.8909
95% CI : (0.873, 0.907)
No Information Rate : 0.885
P-Value [Acc > NIR] : 0.2638
Kappa : 0.3293
Mcnemar's Test P-Value : 5.342e-09
Sensitivity : 0.9683
Specificity : 0.2949
Pos Pred Value : 0.9135
Neg Pred Value : 0.5476
Prevalence : 0.8850
Detection Rate : 0.8569
Detection Prevalence : 0.9381
Balanced Accuracy : 0.6316
'Positive' Class : no
```

It's still difficult to predict the YES because in our data set the proportion of the No is much higher than the Yes :

```
table(training_data$y);table(testing_data$y);
```

no	yes	no	yes
2800	365	1200	156

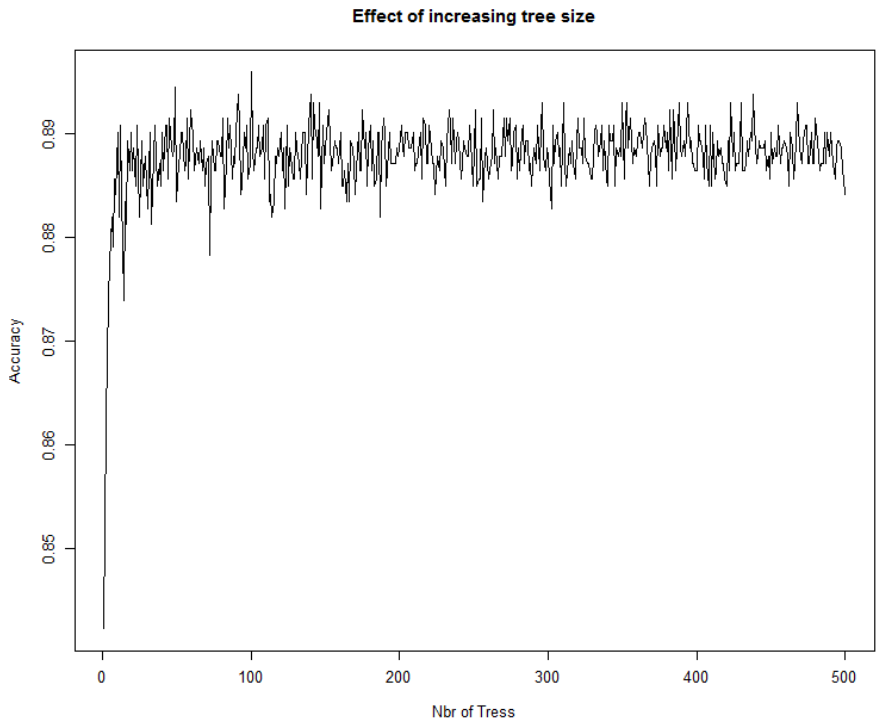
Inspite of the correlations being not so high, the accuracy is high because of the combined effect of predictors as well as the power of building multiple trees.

```
cm$overall
```

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
8.901180e-01	3.271390e-01	8.722508e-01	9.062734e-01	8.849558e-01	2.926765e-01	9.772923e-09

Effect of increasing tree count

Let us try to build different number of trees and see the effect of that on the accuracy of the prediction



Conclusions

Random forests provide better accuracy than plain decision trees because of the power of the number of trees built. The example shows that as we increase the number of trees, accuracy also increases. But that comes at a cost of increased time and resource usage.

13. K Means Clustering 11:53

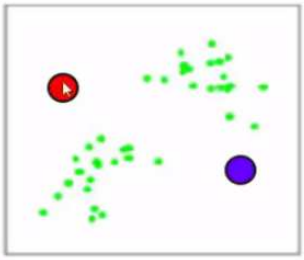
K Means Clustering – Overview

- Unsupervised Learning technique
- Popular method for grouping data into subsets based on the similarity
- Partitions n observations with m variables into k clusters where by each observation belongs to only one cluster
- How it works
  - An m dimensional space is created
  - Each observation is plotted based on this space based on the variable values
  - Clustering is done by measuring the distance between points and grouping them
- Multiple types of distance measures available like Euclidian distance and Manhattan distance

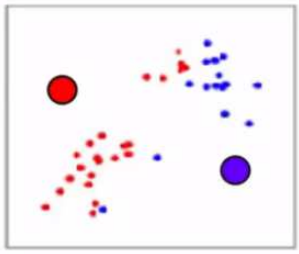
Ref. https://fr.wikipedia.org/wiki/K-moyennes , https://dataanalyticspost.com/Lexique/k-means-ou-k-moyennes/ … http://mathlasup5.fr/docs/manhattan.pdf

K Means Clustering – Stages

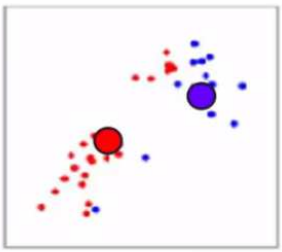
- Dataset contains only m=2 variables. We will create k=2 clusters
- Plot observations on a two dimensional plot



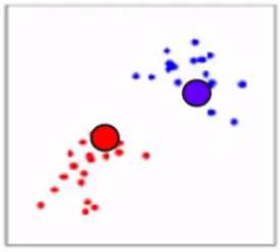
- Choose k=2 centroids at random
- Measure the distance between each observation to each centroid



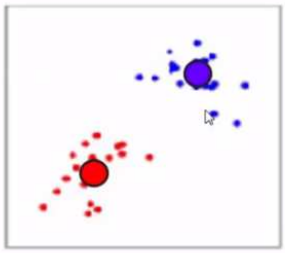
- Assign each observation to the nearest centroid
- This forms the clusters for round 1



- Find the centroid of each of the cluster
- Centroid is the point where the sum of distances between the centroid and each point is minimum



- Repeat the process of finding the distance between each observation to each centroid (the new one) and reassign each point to the nearest one



- Find the centroid for the new clusters
- Repeat the process until the centroids don't move

Iteration => convergence (stable)

K Means Clustering – Summary

Advantages

- Fast
- Efficient with large number of variables
- Explainable

Shortcomings

- K needs to be known
- The initial centroid position has influence on clusters formed

Used in

- Preliminary grouping of data before other classification
- General grouping
- Geographical clustering

14. R Use Case : K Means Clustering 16:24

K Means Clustering – Auto data

Problem Statement

The input data contains samples of cars and technical / price information about them. The goal of this problem is to group these cars into 4 clusters based on their attributes

Techniques Used

1. K-Means Clustering
2. Centering and Scaling

Data Engineering & Analysis

Loading and understanding the dataset

```
car_data <- read.csv("auto-data.csv", as.is=FALSE)
str(car_data)
head(car_data)
summary(car_data)
```

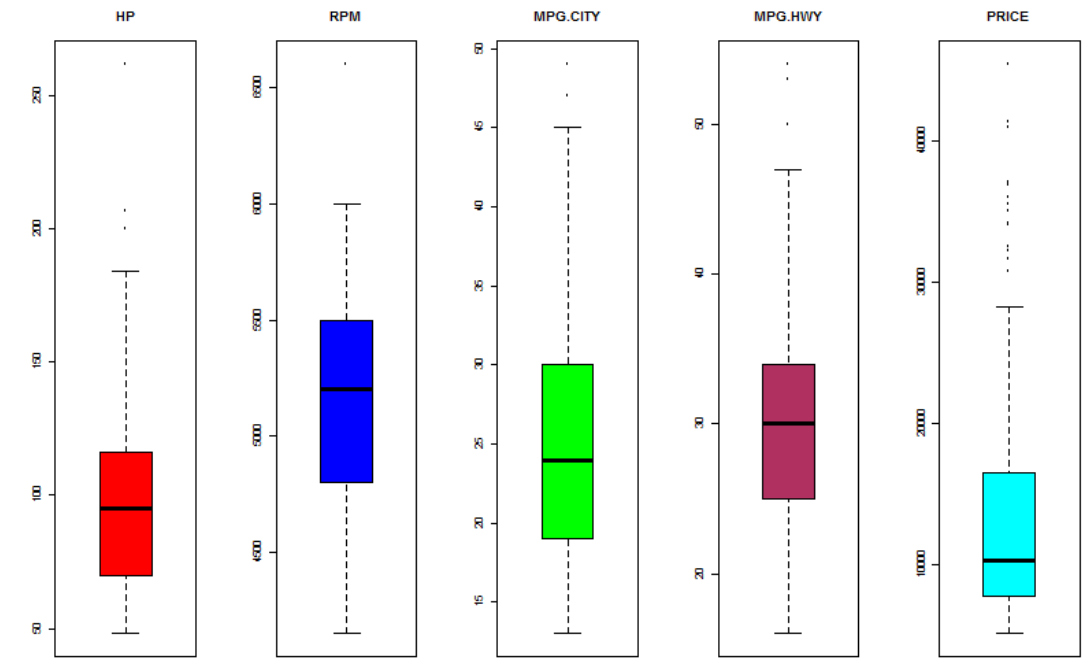
head(car_data)

	1	2	3	4	5	6	7	8	9	10	11	12
	MAKE	FUELTYPE	ASPIRE	DOORS	BODY	DRIVE	CYLINDERS	HP	RPM	MPG.CITY	MPG.HWY	PRICE
1	subaru	gas	std	two	hatchback	fwd	four	69	4900	31	36	5118
2	chevrolet	gas	std	two	hatchback	fwd	three	48	5100	47	53	5151

Exploratory Data Analysis

Typically, for Clustering problems, EDA is only required for finding out outliers and errors. If outliers are found, we would want to eliminate them since they might skew the clusters formed by moving the centeroids signficantly.

```
par(mfrow=c(1,5))
boxplot(x = car_data$HP, col = "red", main = "HP")
boxplot(x = car_data$RPM, col = "blue", main = "RPM")
boxplot(x = car_data$MPG.CITY, col = "green", main = "MPG.CITY")
boxplot(x = car_data$MPG.HWY, col = "maroon", main = "MPG.HWY")
boxplot(x = car_data$PRICE, col = "cyan", main = "PRICE")
```



We choose not to remove the outliers (dots in the charts) since they are many (hence may not be outliers

Modeling & prediction

Build Clusters for 2 variables

In order to demonstrate the clusters being formed on a 2-dimensional plot, we will only use 100 samples and 2 attributes - HP and PRICE to create 4 clusters.

```
library(class)          # classification - https://cran.r-project.org/web/packages/class/class.pdf
set.seed(11111)   # to generate the same random (here with ID=11111)
car_subset <- car_data[1:100, c(8,12)]   # 100 rows , 2 dim. { #HP ~ PRICE }
car_subset
```

```
> car_subset
     HP PRICE
1    69  5118
2    48  5151
3    68  5195
4    62  5348
```

```
car_clusters <- kmeans(car_subset, 4)   # 4 clusters
car_clusters
```

K-means clustering with 4 clusters of sizes 20, 15, 29, 36

Cluster means:

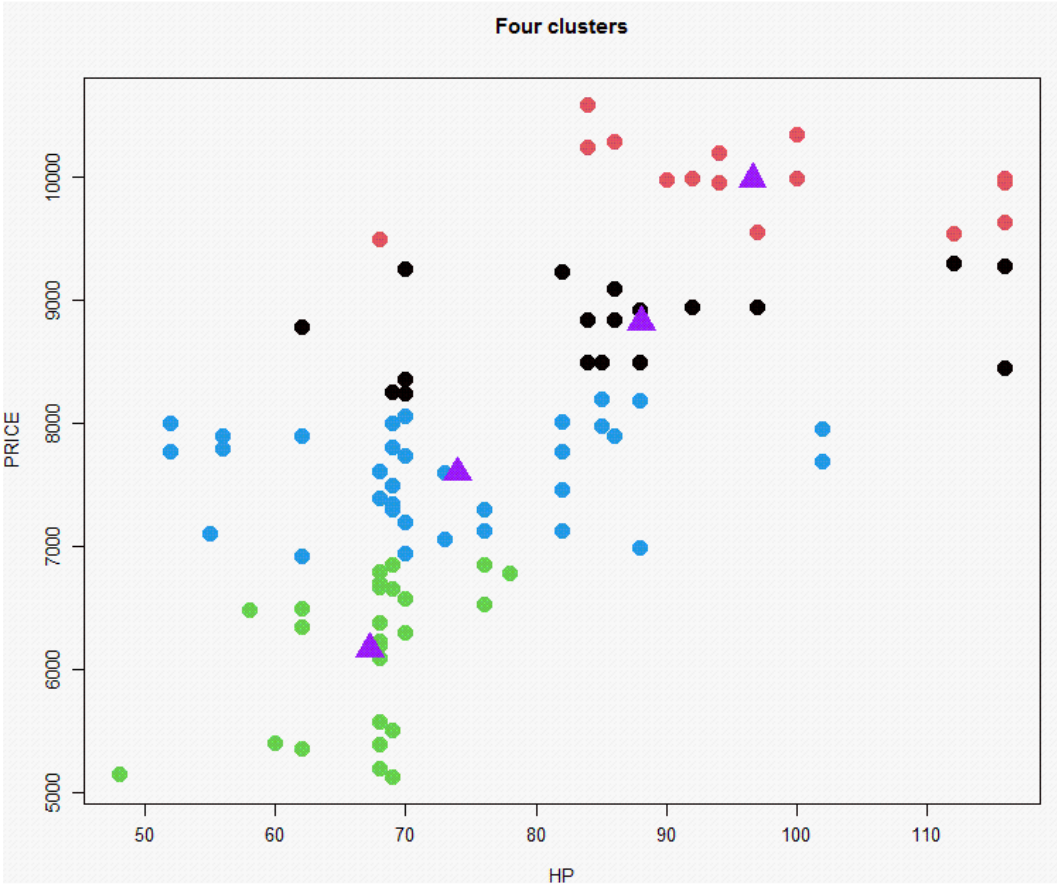
```
       HP       PRICE
1 88.05000 8821.600
2 96.60000 9984.667
3 67.24138 6163.690
4 74.00000 7596.083
```

Clustering vector:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  4  4
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
```

```
 4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
 4  4  4  4  4  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
 1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2

Within cluster sum of squares by cluster:
[1] 2479534 1464655 9728274 5071069      (between_SS / total_SS =  90.2 %)

Available components:
[1] "cluster"    "centers"    "totss"      "withinss"   "tot.withinss"
[6] "betweenss"  "size"       "iter"       "ifault"
```

```
library(class)        # classification - https://cran.r-project.org/web/packages/class/class.pdf
set.seed(11111)  # to generate the same random (here with ID=11111)
car_subset <- car_data[1:100, c(8,12)]  # 100 rows , 2 dim. { #HP ~ PRICE }
car_subset

car_clusters <- kmeans(car_subset, 4)  # 4 clusters
car_clusters; car_clusters$centers; car_clusters$cluster

# plot - https://www.datamentor.io/r-programming/plot-function/
par(mfrow=c(1,1), cex=1, mai=c(1,1,1,1))
plot(car_subset$HP, car_subset$PRICE, col=car_clusters$cluster,pch=20, cex=3, xlab="HP", ylab="PRICE", main="Four clusters")
 points(car_clusters$centers, col="purple", pch=17, cex=3)
 # legend("topleft", c(1:4), fill=c("green","blue","black","red"))
)
```



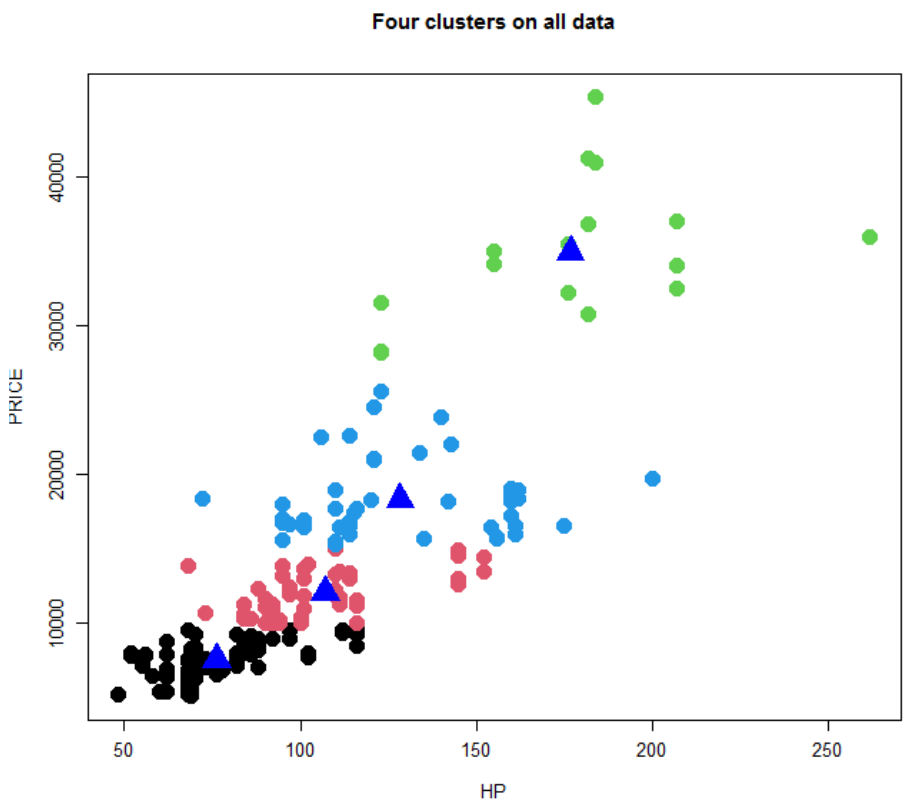
Clustering for all the data

Data transformation (from factors to numeric)

```
car_data2 <- car_data
for (columns in 1:8) {
  car_data2[,columns] <- as.numeric(car_data[,columns])
}
car_data[1:3,]; car_data2[1:3,]
```

```
> car_data[1:3,]; car_data2[1:3,]
       MAKE FUELTYPE ASPIRE DOORS      BODY DRIVE CYLINDERS HP  RPM MPG.CITY MPG.HWY PRICE
1    subaru      gas    std   two hatchback   fwd      four 69 4900       31      36  5118
2 chevrolet      gas    std   two hatchback   fwd     three 48 5100       47      53  5151
3     mazda      gas    std   two hatchback   fwd      four 68 5000       30      31  5195
  MAKE FUELTYPE ASPIRE DOORS BODY DRIVE CYLINDERS HP  RPM MPG.CITY MPG.HWY PRICE
1   18        2      1     2    3     2         3 69 4900       31      36  5118
2    4        2      1     2    3     2         5 48 5100       47      53  5151
3    9        2      1     2    3     2         3 68 5000       30      31  5195
```

```
dim(car_data2)  # 197  12

# to generate the same random (here with ID=11111)
set.seed(11111)
car_clusters2 <- kmeans(car_data2, 4)  # 4 clusters
car_clusters2$centers
car_clusters2$centers[,c(8,12)]

par(mfrow=c(1,1), cex=1, mai=c(1,1,1,1))
plot(car_data2$HP, car_data2$PRICE,
   col=car_clusters2$cluster, pch=20, cex=3,
   xlab="HP", ylab="PRICE", main="Four clusters on all data")
#points - https://www.math.ucla.edu/~anderson/rw1001/library/base/html/points.html
points(car_clusters2$centers[,c(8,12)] , col="blue" , pch=17 , cex=3 )
```



K-means clustering with 4 clusters of sizes 89, 47, 16, 45

Cluster means:

```
     MAKE FUELTYPE   ASPIRE    DOORS     BODY    DRIVE CYLINDERS       HP      RPM MPG.CITY
1 13.20225 1.932584 1.056180 1.471910 3.617978 2.022472  3.022472  76.04494 5157.865 30.17978
2 14.10638 1.914894 1.255319 1.425532 3.574468 2.276596  3.297872 107.17021 5094.681 22.91489
3  9.43750 1.812500 1.187500 1.500000 3.187500 3.000000  3.000000 176.75000 4996.875 16.68750
4 12.51111 1.866667 1.333333 1.333333 3.777778 2.800000  3.244444 128.37778 5106.667 20.55556
…
```

Clustering vector:

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [47] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
 [93] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4
[139] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3
[185] 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Within cluster sum of squares by cluster:

```
[1] 140048031 115551241 334439368 302547347
 (between_SS / total_SS =  92.9 %)
```

Available components:

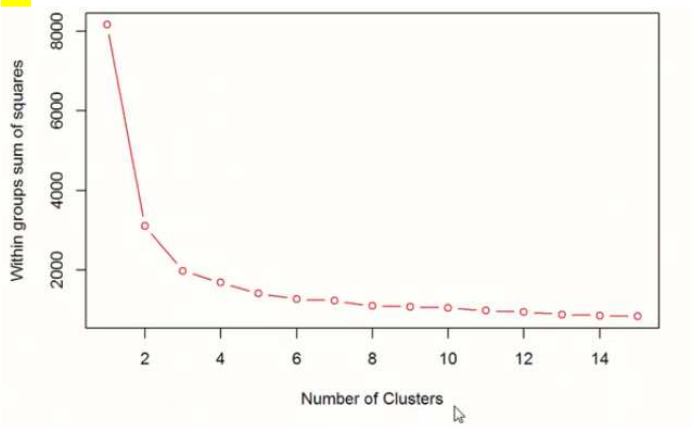
```
[1] "cluster"     "centers"     "totss"       "withinss"    "tot.withinss" "betweenss"   "size"        "iter"        "ifault"
```

Finding optimum number of clusters

???

```
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares", col="red")}

wssplot(auto_data)
```

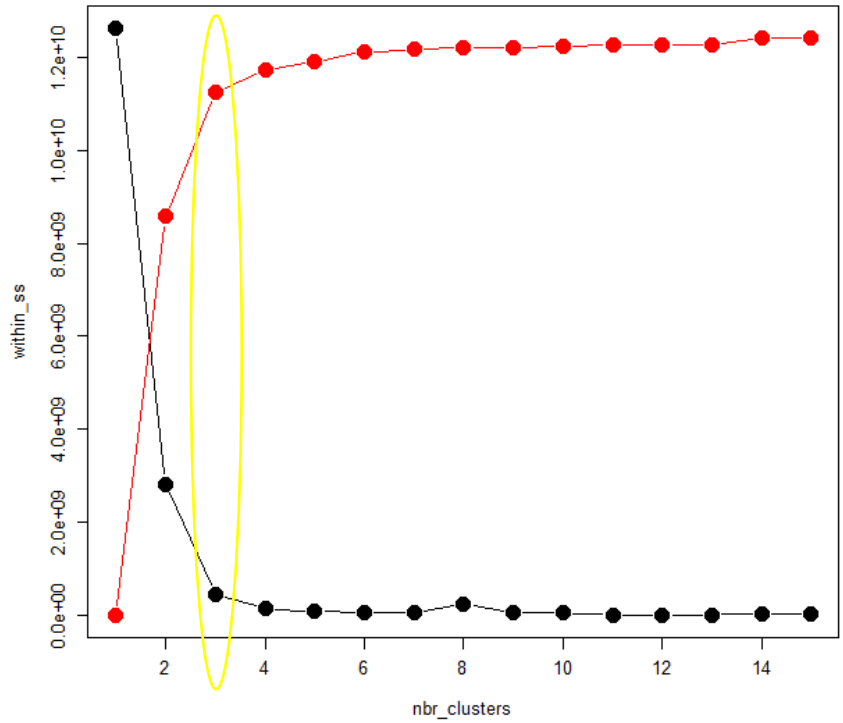
=>



3 seems to be the optimal number of clusters for this dataset

```
set.seed(11111)
within_ss <- c(); between_ss <- c();
nbr_clusters <- 1:15
for (i in nbr_cluster_range) {
  cluster_i <- kmeans(car_data2, i)
  within_ss[i] <- cluster_i$withinss
  between_ss[i] <- cluster_i$betweenss
}
```

```
within_ss; between_ss
plot(x=nbr_clusters,y=within_ss, col="black", type="b", pch=20, cex=3)
points(x=nbr_clusters,y=between_ss, col="red", pch=20, cex=3, type="b")
# => optimun is 3
```



3 seems to be the optimal number of clusters for this dataset

15. Association Rules Mining 11:30

Association Rules Mining – Overview of ARM

- ARM shows how frequently sets of items occur together
  - Find Items frequently brought together
  - Find fraudulent transactions.
  - Frequent Pattern Mining/ Exploratory Data Analysis
  - Finding the next word
- One of the clustering techniques
- Assumes all data are categorical, not applicable for numeric data
- Helps generate association rules that can be then used for business purposes like stocking aisles.

Association Rules Mining – Datasets

- Market basket transactions
  - Tran 1 { bread, cheese, milk}
  - Tran 2 { apple, eggs, yogurt}
  - Tran 3 {bread, eggs}
- Text document data set (bag of words)
  - Doc 1 { cricket, sachin, India }
  - Doc 2 { soccer, messi, Barcelona}
  - Doc 3 { sachin, messi, superstars}

Association Rules Mining – Measures

- Let N be the number of transactions
- Let X, Y and Z be individual items
- Support measures how frequently an combination of items occurs in the transactions
  - Support(X) = count(transactions with X)/ N
  - Support(X,Y)= count(transactions with X and Y)/N
- Confidence measures the expected probability that Y would occur when X occurs
  - Confidence(X -> Y) = support(X,Y) / support(X)
- Lift measures how many more times X and Y occurs together than expected
  - Lift(X -> Y) = confidence(X->Y) / support(Y)

Association Rules Mining – Rules and goals

- A rule specifies when one item occurs the other too occurs
  - When bread is brought, milk is brought 33% of the time.
  - When India occurs in the bag of words, sachin occurs 20% of the time.
- Goal is to find all rules that satisfy the user specified minimum support and minimum confidence
- A frequent itemset is an itemset whose support is > the minimum support level specified.
- Apriori algorithm is the most popular ARM algorithm

16. R Use Case : Association Rules Mi... 13:11

Association Rules Mining – Accident conditions

Problem Statement

The input dataset contains information about 1000 fatal accidents. It has different feature variables associated with the accident. The goal is to find patterns in the variables - which accident conditions frequently occur together.

Techniques Used

1. Association Rules Mining
2. Converting Feature data into Basket Data

Data Engineering & Analysis

Loading and understanding the dataset

```
accident_data <- read.csv("accidents.csv")
str(accident_data); dim(accident_data) # 1000 obs. of  16 variables
accident_data[1:5,]
summary(accident_data)
```

⇨ *1000 obs. of 16 variables*

```
> accident_data[1:5,]
  Accident_Index Police_Force Accident_Severity Number_of_Vehicles Number_of_Casualties
1  200501CW10664            1                 3                  3                    2
2  200501CW11407            1                 3                  1                    1
3  200501E040954            1                 3                  2                    1
```

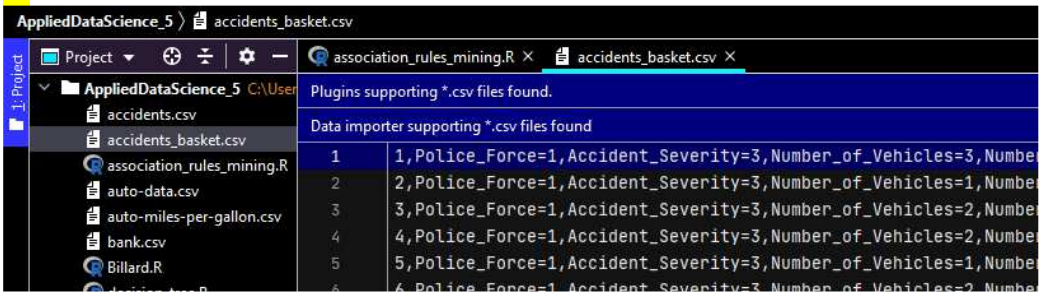
Data Transformation

The data frame needs to be converted into a Basket form to be loaded by the arules dataset. The following custom code does it.

```
colnames <- names(accident_data)
str(colnames); length(colnames)
# In a basket, every row represents a transaction (Id,<K=V>, ...)
# paste - ref. https://www.math.ucla.edu/~anderson/rw1001/library/base/html/paste.html
# char - ref. https://stat.ethz.ch/R-manual/R-devel/library/base/html/nchar.html
```

```
basket <- list()
for (n in 1:nrow(accident_data)) {
  kv <- paste0(n, SEP=",")
  for (c in 2:ncol(accident_data)) {  # without column n°2
    transaction <- paste0(colnames[c], sep = "=", accident_data[n,c], collapse = NULL)
    kv <- paste0(kv, transaction, SEP=",")
  }
  basket[n] <- substr(kv, 1, nchar(kv) - 1)
}
write(as.character(basket),"accidents_basket.csv")
```

=>



Exploratory Data Analysis

Typically, for Clustering problems, EDA is only required for finding out outliers and errors. If outliers are found, we would want to eliminate them since they might skew the clusters formed by moving the centeroids signficantly.

```
# arules - ref. https://www.rdocumentation.org/packages/arules/versions/1.6-6
#            https://www.rdocumentation.org/packages/arules/versions/1.6-6/topics/read.transactions
library(arules)  # load "Mining Association Rules and Frequent Itemsets"
accidents <- read.transactions("accidents_basket.csv", format = "basket", sep=",")
summary(accidents)
```

>

transactions as itemMatrix in sparse format with
***1000 rows** (elements/itemsets/transactions) and*
***1452 columns** (items) and a **density of 0.01**01928*

most frequent items (TOP 5) :

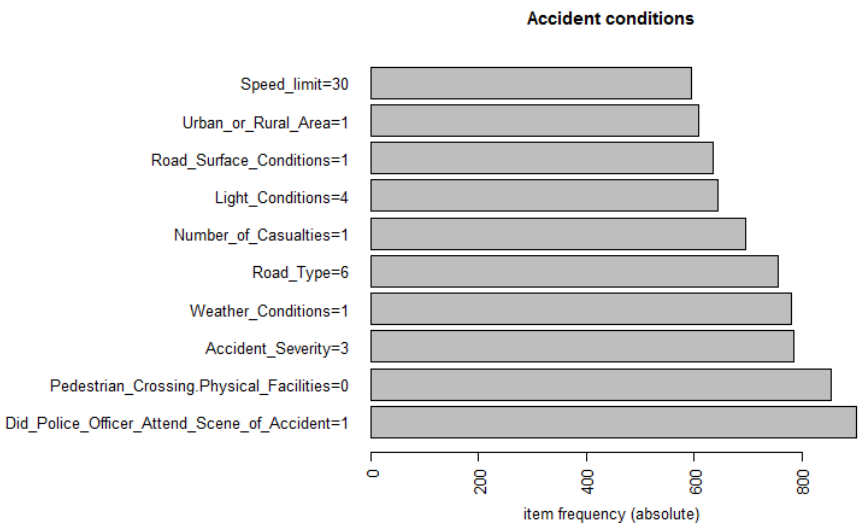
Did_Police_Officer_Attend_Scene_of_Accident=1	*902*
Pedestrian_Crossing.Physical_Facilities=0	*854*
Accident_Severity=3	*786*
Weather_Conditions=1	*781*
Road_Type=6	*755*
(Other)	*11922*

element (itemset/transaction) length distribution: *sizes 16 / 1000*

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
16	16	16	16	16	16

includes extended item information - examples:
labels 1 1 / 2 10 / 3 100

```
# Items Frequency Plot -
https://www.rdocumentation.org/packages/arules/versions/1.5-
5/topics/itemFrequencyPlot
itemFrequencyPlot(accidents, type = "absolute", topN = 10, popCol = "green",
popLwd = 1,
                  lift = FALSE, horiz = TRUE, names = TRUE,
cex.names=graphics::par("cex.axis"),
                  main = "Accident conditions")
```



Modeling & Prediction

We discover the frequently occuring patterns with arules.

```
# apriori - ref. https://www.rdocumentation.org/packages/arules/versions/1.6-6/topics/apriori
# The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.8,
#  maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).
rules <- apriori(accidents, parameter=list(supp=0.1, conf = 0.3))
```

=>

Apriori - Parameter specification:

confidence	minval	smax	arem	aval	originalSupport	maxtime	support	minlen	maxlen	target	ext
0.3	0.1	1	none	FALSE	TRUE	5	0.1	1	10	rules	TRUE

Algorithmic control:

Filter	tree	heap	memopt	load	sort	verbose
0.1	TRUE	TRUE	FALSE	TRUE	2	TRUE

Absolute minimum support count: 100

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[1452 item(s), 1000 transaction(s)] done [0.02s].
sorting and recoding items ... [29 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.00s].
writing ... [21772 rule(s)] done [0.02s].
creating S4 object ... done [0.01s].

```
# apriori - ref. https://www.rdocumentation.org/packages/arules/versions/1.6-6/topics/apriori
# The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.8,
#  maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).
rules <- apriori(accidents, parameter=list(supp=0.1, conf = 0.3))
class(rules)  # arules
length(rules) # set of 21772 rules
inspect(rules[1:40])
```

```
##    lhs                          rhs                          support confidence   lift
## 21 {Weather_Conditions=2}    => {Road_Surface_Conditions=2}   0.124      0.9841 2.9822
```

```r
# Example also here : http://r-statistics.co/Association-Mining-With-R.html
# Rules with confidence of 1 (see rules_conf) imply that, whenever the XXX item was purchased,
# the YYY item was also purchased 100% of the time.
rules_conf <- sort (rules, by="confidence", decreasing=TRUE) # 'high-confidence' rules.
inspect(head(rules_conf))

# A rule with a lift of 18 (see rules_lift) imply that, the items in XXX and YYY are 18 times more likely to be purchased together compared to the purchases when they are assumed to be unrelated.
rules_lift <- sort (rules, by="lift", decreasing=TRUE) # 'high-lift' rules.
inspect(head(rules_lift)) # show the support, lift and confidence for all rules
```

=>

```
> inspect(head(rules_conf))
    lhs                                                  rhs                                    support confidence coverage     lift coun
[1] {Junction_Detail=0,
     Number_of_Casualties=2}                         => {Pedestrian_Crossing.Physical_Facilities=0}   0.108          1    0.108 1.17096   108
[2] {Did_Police_Officer_Attend_Scene_of_Accident=1,
     Junction_Detail=0,
     Number_of_Casualties=2}                         => {Pedestrian_Crossing.Physical_Facilities=0}   0.100          1    0.100 1.17096   100
```

=>

```
> inspect(head(rules_lift)) # show the support, lift and confidence for all rules
    lhs                                                  rhs              support confidence coverage     lift count
[1] {Light_Conditions=6,
     Road_Type=6,
     Urban_or_Rural_Area=2,
     Weather_Conditions=1}                           => {Speed_limit=60}   0.118  0.8939394    0.132 3.990801   118
```

17. ANN and SVM 4:35

Artificial Neural Networks
Support Vector Machines

Artifical Neural Networks (ANN) – Overview

- Biologically inspired by how the human brain works.
- A Black box algorithm (a full explanation would require few hours and mathematical prerequisites)
- Used in artificial intelligence domain and of late for machine learning
- Helps discover complex correlations hidden in the data similar to the human brain
- Works well on noisy data and where the relationships between variables is vaguely understood.
- Fast prediction
- Very slow training and easy to over fit

Ref. ANN - https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
https://en.wikipedia.org/wiki/Artificial_neural_network

Support Vector Machines (SVM) – Overview

- A Black box method for machine learning – inner workings are complex, tricky and difficult to understand.
- One of the kernel methods.
- Algorithm based on vector geometry and statistical learning theory
- Can model highly complex relationships – very popular in pattern recognition (face recognition, text recognition etc.)
- Successful applications in many fields like bioinformatics, image recognition etc.
- Used for both classification and regression (discrete and continuous outcomes).

Ref. SVM - https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
https://en.wikipedia.org/wiki/Support_vector_machine

No use case… out of scope (too big) topic …

18. Bagging and Boosting 11:27

Bagging
Boosting

Bagging = Bootstrap Aggregating – Overview

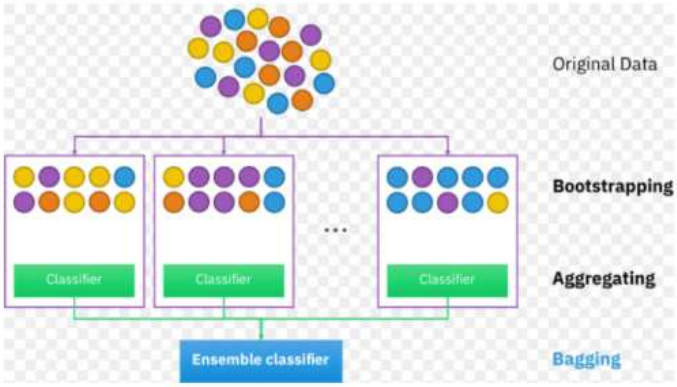
- Bootstrap Aggregating /Ensemble method
- Uses a base classifier Decision trees to train on multiple sample sets and to generate multiple models
- Prediction is done using each model and the most occurring result across all models is selected.
- For each training round a different bootstrap replicate dataset is constructed based on the original dataset.
  - If the original dataset has m examples, then n rounds of sampling is done to get m/n examples each
  - The n samples sets are then added up to for a dataset of m size
  - Examples could be duplicated in the sample sets or might not occur at all.

Bootstrap aggregating, also called **bagging** (from **b**ootstrap **agg**regat**ing**), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method.

Ref. https://fr.wikipedia.org/wiki/Bootstrap_aggregating ,
https://en.wikipedia.org/wiki/Bootstrap_aggregating

Bagging – How it works

- Suppose we want to run training 5 times on a dataset that has 8 records (Record ID 1:8).
- For each round, we do 2 sets of sampling with replacement to build the bootstrap aggregate.
- Training round 1:
  - sample 1 : 1,4,5,7
  - sample 2 : 2,4,6,7
  - bootstrap replicate : 1,2,4,4,5,6,7,7
- Training round 2:
  - sample 1 : 2,3,5,6
  - sample 2 : 1,2,6,8
  - bootstrap replicate : 1,2,2,3,5,6,6,8



Bagging – Remarks

- May produce improved results than the base classifier if the base classifier produces unstable results.
- High resource requirements and takes longer times to build models
- Various models available – difference is the base classifier used (some examples)
  - adaBag
  - Bagged CART
  - Bagged Flexible Discriminant Analysis
  - Bagged Logic Regression
  - Model Averaged Neural Network

Boosting – Overview

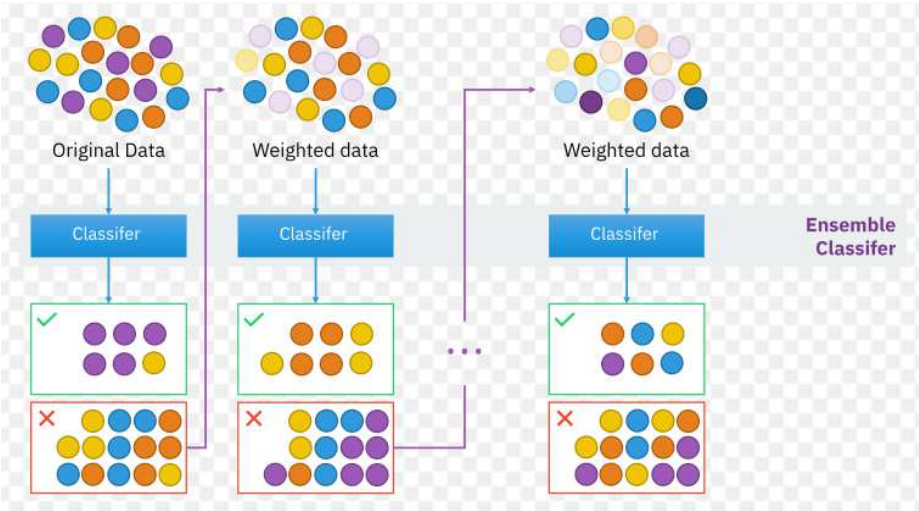
- Ensemble method like bagging
- Creates multiple models.
- Prediction done on multiple models and results aggregated to deliver the final prediction
- Different samples (records /observations) get different weights during learning for each of the training rounds
- Weight determined based on misclassification during previous round.

In machine learning, **boosting** is an ensemble meta-algorithm for primarily reducing bias, and also variance[1] in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.

Ref. https://en.wikipedia.org/wiki/Boosting_(machine_learning)

Boosting – How it works

- Multiple rounds of training. Weights of all records equal for the first round
- For each model building round
  - Build the model
  - Predict on the same training set using the model
  - Find misclassified records
  - Increase weights of misclassified records
  - Repeat model building
- Results in multiple models
- Predictions done using each model. Results aggregated.



Boosting – Remarks

- High resource requirements and takes longer times to build models
- Use a set of weak learners to create a strong learner
- Reduces bias
- Different algorithms available
  - Boosted Classification Trees
  - Boosted Generalized Additive Model
  - Boosted Generalized Linear Model

19. Dimensionality Reduction 7:28

Dimensionality Reduction – Overview

Reduce issues when too many predictors…

- Memory requirements
- CPU requirements / time taken for machine learning algorithms
- Correlation between predictors
- Over fitting
- Some ML algorithms don't work fine with too many predictors

Dimensionality Reduction – How to do

1) Manual selection

- Using domain knowledge
  - Purely based on hypothesis
  - Risky – there could be unknown correlations
- Using Correlation co-efficients
  - Variables with good correlation can only be picked up.
- Using Decision Trees
  - Decision trees are fast and choose variables based on correlation
  - Variables used in the decision trees can be picked for further processing

2) Principal Component Analysis (PCA)

- Used to reduce the number of predictors
- Based on Eigen Vectors and Eigen Values
- Given a set of M predictors, PCA transforms this to a set of N predictors such that N < M
- The new predictors are derived predictors called PC1, PC2, PC3
- The new predictors retain similar levels of correlation and predictability like the original predictors

Ref. https://fr.wikipedia.org/wiki/Analyse_en_composantes_principales
https://en.wikipedia.org/wiki/Principal_component_analysis

20. R Use Case : Advanced Methods 17:18

The Caret package - overview

Classification **A**nd **RE**gression **T**raining package is a set of functions that attempt to streamline the process for creating predictive models. It contains tools for:

1. data splitting
2. pre-processing
3. feature selection
4. model tuning using resampling
5. variable importance estimation

There are many different modelling functions in R. Some have different syntax for model training and/or prediction. The package started off as a way to provide a uniform interface the functions themselves, as well as a way to standardize common tasks (such parameter tuning and variable importance).

Ref. http://topepo.github.io/caret/index.html
https://github.com/topepo

Many algorithms…

topepo.github.io/caret/modelList.html

Most Visited | Getting Started | Data Science Central | Outlook Web App | Login - Cloudera Man... | TF hangout

Model	method Argument Value	Type	Packages	Tuning Param
Boosted Classification Trees	ada	Classification	ada, plyr	iter, maxdepth, nu
Bagged AdaBoost	AdaBag	Classification	adabag, plyr	mfinal, maxdepth
AdaBoost.M1	AdaBoost.M1	Classification	adabag, plyr	mfinal, maxdepth, c
Adaptive Mixture Discriminant Analysis	amdai	Classification	adaptDA	model
Adaptive-Network-Based Fuzzy	ANFIS	Regression	frbs	num.labels, max.ite

Breast cancer – predict disease

Problem Statement

The dataset contains diagnosis data about breast cancer patients and whether they are Benign (healthy) or Malignant (possible disease). We need to predict whether new patients are benign or malignant based on model built on this data.

Techniques Used

1. Principal Component Analysis
2. Training and Testing
3. Confusion Matrix
4. Neural Networks
5. Support Vector Machines
6. Bagging
7. Boosting

Use the caret package to build models for all the datasets provided in the use cases. Use the same algorithm used in the example use case, but with the caret package. Compare the results between the caret package and the output seen in the use cases.

Data Engineering & Analysis

Loading and understanding the dataset

```
cancer_data <- read.csv("breast_cancer.csv")
cancer_data[1:5,]    # breast cancer data frame
```

=>

```
> cancer_data[1:5,]   # breast cancer data frame
        id diagnosis radius_mean texture_mean perimeter_mean a
1 87139402         B       12.32        12.39          78.85
2  8910251         B       10.60        18.95          69.28
3   905520         B       11.04        16.83          70.92
```

```
dim(cancer_data)     # 569 rows, 32 columns
str(cancer_data)
```

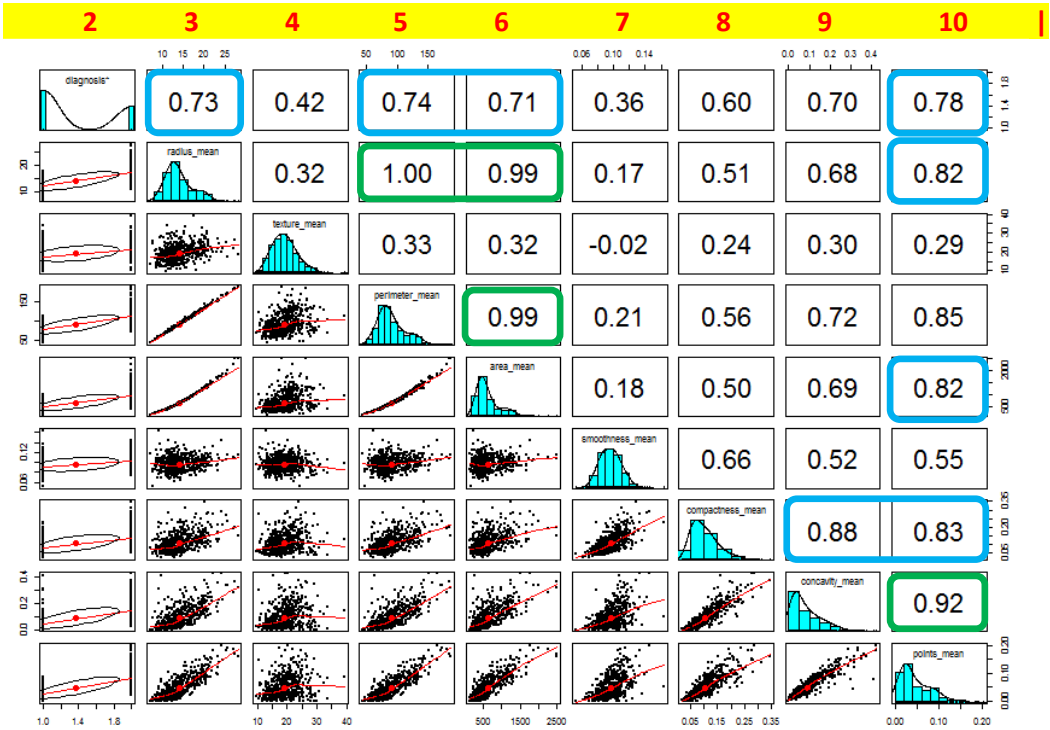
⇨ many variables (#31)

```
> str(cancer_data)
'data.frame':   569 obs. of  32 variables:
 $ id              : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 ...
 $ diagnosis       : chr  "B" "B" "B" "B" ...
 $ radius_mean     : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean    : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean  : num  78.8 69.3 70.9 73 97.7 ...
```

Exploratory Data Analysis

Correlation – split into 3 sets (2 ~ 3:10 / 11:21 / 22:32)

```
# ref.
https://www.rdocumentation.org/packages/psych/versions/2.0.9/topics/pairs.panels
library(psych)
pairs.panels(cancer_data[,c(2,3:10)])  # scatter plot of matrices (SPLOM)
pairs.panels(cancer_data[,c(2,10:20)])
pairs.panels(cancer_data[,c(2,21:32)])
```



...

Principal Components Analysis (PCA)

```
# ref. https://www.datacamp.com/community/tutorials/pca-analysis-r
#      http://www.sthda.com/french/articles/38-methodes-des-composantes-
principales-dans-r-guide-pratique/79-acp-dans-r-prcomp-vs-princomp/
#      https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp
scale_data <- scale(cancer_data[,3:32])
pca_data <- prcomp(scale_data)
str(pca_data); summary(pca_data)
```

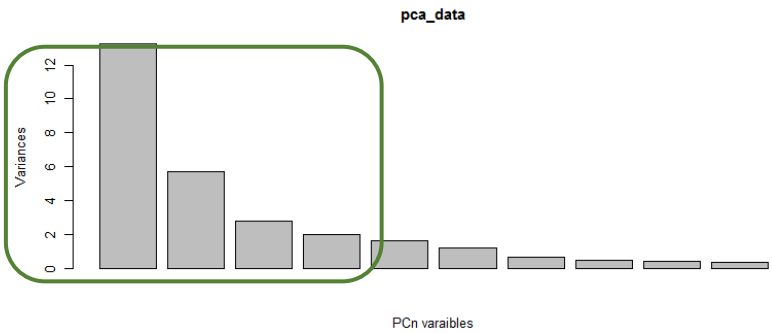
=>

```
> summary(pca_data)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172 0.69037 0.6457
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251 0.01589 0.0139
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010 0.92598 0.9399
                          PC10   PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
Standard deviation     0.59219 0.5421 0.51104 0.49128 0.39624 0.30681 0.28260 0.24372 0.22939
Proportion of Variance 0.01169 0.0098 0.00871 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175
Cumulative Proportion  0.95157 0.9614 0.97007 0.97812 0.98335 0.98649 0.98915 0.99113 0.99288
                          PC19    PC20   PC21    PC22    PC23   PC24    PC25    PC26    PC27
Standard deviation     0.22244 0.17652 0.1731 0.16565 0.15602 0.1344 0.12442 0.09043 0.08307
Proportion of Variance 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023
Cumulative Proportion  0.99453 0.99557 0.9966 0.99749 0.99830 0.9989 0.99942 0.99969 0.99992
                          PC28    PC29    PC30
Standard deviation     0.03987 0.02736 0.01153
Proportion of Variance 0.00005 0.00002 0.00000
```

With these four (4) first variables, we are covering 79% of the others

```r
# plotting this PCA
plot(pca_data, xlab="PCn variables")
```

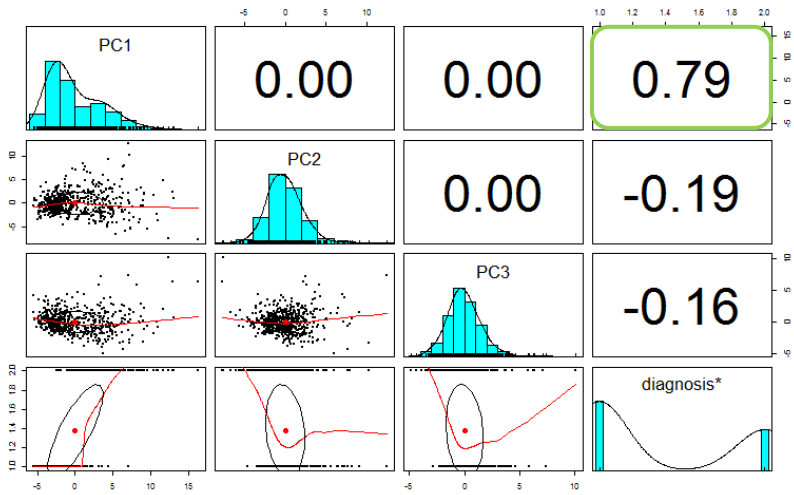
=>




```
> str(pca_data);
List of 5
 $ sdev    : num [1:30] 3.64 2.39 1.68 1.41 1.28 ...
 $ rotation: num [1:30, 1:30] 0.219 0.104 0.228 0.221 0.143 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:30] "radius_mean" "texture_mean" "perimeter_mean" "area_mean" ...
  .. ..$ : chr [1:30] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:30] -1.38e-16 6.15e-17 -1.19e-16 1.22e-16 1.62e-16 ...
  ..- attr(*, "names")= chr [1:30] "radius_mean" "texture_mean" "perimeter_mean" "area_mean" ...
 $ scale   : Named num [1:30] 3.524 4.301 24.299 351.9141 0.0141 ...
  ..- attr(*, "names")= chr [1:30] "radius_mean" "texture_mean" "perimeter_mean" "area_mean" ...
 $ x       : num [1:569, 1:30] -2.51 -1.46 -2.92 -1.99 -2.5 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:30] "PC1" "PC2" "PC3" "PC4" ...
 - attr(*, "class")= chr "prcomp"
```

```r
# use the first three (3) PC1, PC2 and PC3
set_pca_data <- pca_data$x[,1:3]
head(set_pca_data)
#convert pca to data frame (569 rows, 3 columns)
final_data <- data.frame(set_pca_data)
# add the 'diagnostics' column (569 rows)
final_data$diagnosis <- cancer_data$diagnosis
head(final_data)
# scatter plot of matrices (SPLOM)
pairs.panels(final_data)
```

=>



The first 3 principal components influences 75% of the target, so we only pick the top 3. A correlation analysis shows that these 3 have very good correlation to the target. Also the 3 PCs dont have any correlation amongst them.

Modelling and predicting

```
library(caret)
inTrain <- createDataPartition(y=final_data$diagnosis, p=0.7, list=FALSE)
training_data<- final_data[inTrain,]
test_data<- final_data[-inTrain,]
dim(training_data); dim(test_data)
table(training_data$diagnosis); table(test_data$diagnosis)
```

=> 399-4 / 170- 4

=> table(…_data$diagnosis) : B-M / 250-149 B-M / 107-63

Model Building and Testing

We will build different models based on 4 different algorithms. Then we predict on the test data and measure accuracy. Finally, we compare the algorithms for their accuracy and speed. The "caret" package in R provides a convenient unified interface for using any of the algorithms for modeling and prediction. It has an extensive library of algorithms http://topepo.github.io/caret/modelList.html . This can be used to compare performance of different algorithms for a given dataset.

```
predlist <- c("bagFDA",        #Bagging
              "LogitBoost",    #Boosting
              "nnet",          #Neural Networks
              "svmRadialCost") #Support vector machines
```

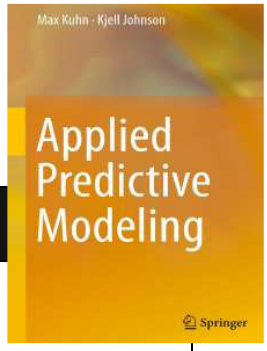
```
# with 4 algorithms : bagging, boosting, neural networks, support vector machines
algoList <- c("bagFDA","LogitBoost","nnet","svmRadialCost")
results <- data.frame(Algorithm=character(), Duration=numeric(),
Accuracy=numeric(), stringsAsFactors = FALSE )

for (i in 1:length(algoList)) {
  algo <- algoList[i]
  print(paste("Alogrithm:", algo))

  startTime <- as.integer(Sys.time())
  # Build model with 'train' function - ref.
https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train
  model <- train( diagnosis ~ . , data = training_data , method = algo)
  model

  # Predict against test data
  predicted <- predict(model, test_data)
  length(predicted); length(test_data$diagnosis)

  # Compare diagnosis prediction with test data
  comp <- confusionMatrix(predicted, as.factor(test_data$diagnosis))

  # Store the result
  stopTime <- as.integer(Sys.time())
  result <- c( as.character(algo) , stopTime - startTime ,
as.numeric(comp$overall[1]) )
  print(result)
  results[i,1] <- as.character(algo)
  results[i,2] <- ( stopTime - startTime )
  results[i,3] <- round(as.numeric(comp$overall[1]) * 100, 2)
}

#Print comparison results
results
```

```
> results
      Algorithm Duration Accuracy
1        bagFDA       49    95.88
2    LogitBoost       26    94.71
3          nnet        7    94.71
4 svmRadialCost      252    95.88
```

Conclusions

Given that there is one main principal component PC1, most algorithms will perform with excellent accuracy. This example shows how large predictors can be easily compressed using PCA and then used for prediction.

Annex



Max Kuhn and Kjell Johnson,
Applied Predictive Modeling (book, 2016)

Terminology

"Predictive modeling" is one of the many names that refers to the process of uncovering relationships within data for predicting some desired outcome.

Since many scientific domains have contributed to this field, there are synonyms for different entities:

- The terms ***sample***, ***data point***, ***observation***, or ***instance*** refer to a single, independent unit of data, such as a customer, patient, or compound.

- The term ***sample*** can also refer to a subset of data points, such as the training set sample. The text will clarify the appropriate context when this term is used.

- The ***training set*** consists of the data used to develop models while the ***test*** or ***validation*** sets are used solely for evaluating the performance of a final set of candidate models.

- The ***predictors***, ***independent variables***, ***attributes***, or ***descriptors*** are the data used as input for the prediction equation.

- ***Outcome***, ***dependent variable***, ***target***, ***class***, or ***response*** refer to the outcome event or quantity that is being predicted.

- ***Continuous*** data have natural, numeric scales. Blood pressure, the cost of an item, or the number of bathrooms are all continuous. In the last case, the counts cannot be a fractional number, but is still treated as continuous data.

- ***Categorical*** data, otherwise known as ***nominal***, ***attribute***, or ***discrete*** data, take on specific values that have no scale. Credit status ("good" or "bad") or color ("red," "blue," etc.) are examples of these data.

- ***Model building***, ***model training***, and ***parameter estimation*** all refer to the process of using data to determine values of model equations.

datacamp https://learn.datacamp.com/

SKILL SHARE https://www.skillshare.com/lists/saved-classes?via=header

___End___