subtitle: "Pattern Mining and Social Network Analysis"
title: "Homework 1"
author: "BOUYSSOU Gatien , de POURTALES Caroline, LAMBA Ankit"
date: "`r format(Sys.time(), '%d %B, %Y')`"
output:
pdf_document:
latex_engine: xelatex
toc: yes

toc_depth: 4

```{r setup, include=FALSE}
library(knitr)
knitr::opts_chunk$set(echo = TRUE,tidy=TRUE,message=FALSE,warning=FALSE,strip.white=TRUE,prompt=FALSE,
cache=TRUE, size="scriptsize",fig.width=6, fig.height=5)
library(reticulate)
#use_python("/Library/Frameworks/Python.framework/Versions/3.6/bin/python3", required = T)
knitr::knit_engines$set(python.reticulate = TRUE)
#py_install("matplotlib")
#py_install("scikit-learn")
```

{r rpackages,eval=TRUE,echo=FALSE}
#install.packages("rmarkdown")
library(magrittr)
library(knitr)
library(rmarkdown)
library(xlsx)
library(ggplot2)
library(ggfortify)
library(MASS)
library(dplyr)
library(ISLR)
library(readr)
library(randomForest)
library(tidyverse)
library(caret)
library(cluster)
library(factoextra)
library(fpc)

```
\clearpage

# Classification

## Overall

Classification algorithms have categorical responses. In classification we build a function f(X) that takes a

## Possibilities of models

There are classifiers as logistic regression, Decision trees, Perceptron / Neural networks, K-nearest-neighbo

## Some indicators to analyze the results
```

```
### Sensitivity and recall

The sensitivity (also named recall) is the percentage of true defaulters that are identified (True positive t
For example, probability of predicting disease given true state is disease.

$$sensitivity = recall = \frac{TruePositiveTests}{PositivePopulation}$$

### Specificity

The specificity  is the percentage of non-defaulters that are correctly identified (True negative tests).
1 - specificity is the Type 1 error, it is the false positive rate.
For example, probability of predicting non-disease given true state is non- disease.

$$specificity = \frac{TrueNegativeTests}{NegativePopulation}$$

### Precision

The precision is the proportion of true positive tests among the positive tests.
$$precision = \frac{TruePositiveTests}{PositiveTests}$$

### F-Mesure

The traditional F measure is calculated as follows:

$$F_Measure = \frac{(2 * Precision * Recall)}{ (Precision + Recall)}$$

### Rand index

The rand index is a mesure of similarity between two partitions from a single set.

Given two partitions $\pi_1$ and $\pi_2$ in E :
\begin{itemize}
\item a, the number of elements in $\pi_1$ and $\pi_2$
\item b, the number of elements in $\pi_1$ and not in $\pi_2$
\item c, the number of elements in $\pi_2$ and not in $\pi_1$
\item d, the number of elements not in both $\pi_1$ and $\pi_2$
\end{itemize}

\begin{center}
\begin{tabular} { | c | c | c |}
\hline
  & in $\pi_2$ & not in $\pi_2$ \\
\hline
in $\pi_1$ & a & b \\
not in $\pi_1$ & c & d \\
\hline
\end{tabular}
\end{center}

$$ RI(\pi_1, \pi_2) = \frac{a + d}{a + b + c + d}$$

### Mutual information

Mutual information is calculated between two variables and measures the reduction in uncertainty for one vari
The mutual information between two random variables X and Y can be stated formally as follows:

$$ MI= I(X ; Y) = H(X) – H(X | Y)$$

### Cross Entropy(log loss)

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probabi

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$ CE = − (y\log(p)+(1 − y)\log(1 − p))$$

If M>2 (i.e. multiclass classification), we calculate a separate loss for each class label per observation an

$$ CE = - \sum_{ c = 1 }^{b}y_{o, c}\log(p_{o, c})$$

## Accuracy of a model

How do we determine which model is best? Various statistics can be used to judge the quality of a model. \\
These include Mallow's $C_p$, Akaike information criterion (AIC), Bayesian information criterion (BIC), and a
```

```
### MSE : Mean Squarred Error

Let's define the mean squared error or MSE.
$$MSE = \frac{1}{n} \sum_i 1_{y_i-\hat{f}(x_i)}$$ where :

$$1_{y_i-\hat{f}(x_i)} = \left\{
    \begin{array}{ll}
        1 & \mbox{if } y_i != \hat{f}(x_i) \\
        0 & \mbox{otherwise}
    \end{array}
\right. $$

Recall : $$RSS = MSE * n$$

RSS and $R^2$ are not suitable for selecting the best model among a collection of models with different numbe

### Mallow’s Cp

Mallows’s Cp addresses the issue of overfitting, in which model selection statistics such as the residual sum

If there are d predictors :

$$ C_p = \frac{RSS + 2 d \hat{\sigma}^2}{n}$$

### AIC : Akaike information criterion

The Akaike information criterion (AIC) is an estimator of in-sample prediction error and thereby relative qua

The AIC criterion is defined for a large class of models fit by maximum likelihood.

$$ AIC = \frac{RSS + 2 d \hat{\sigma}^2}{n\hat{\sigma}^2}$$

To use AIC for model selection, we simply choose the model giving small- est AIC over the set of models consi

### BIC : Bayesian information criterion

In statistics, the Bayesian information criterion or Schwarz information criterion is a criterion for model s

BIC is derived from a Bayesian point of view, but ends up looking similar to Cp (and AIC) as well. For the le

$$ BIC = \frac{RSS + log(n) d \hat{\sigma}^2}{n}$$

### Adjusted R statistic

The adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors

$$ Adusted R^2 = 1 -  \frac{\frac{RSS}{n-d-1}}{\frac{TSS}{n-1}}$$

## Logistic Regression

### How it works

In logistic regression, for covariates (X_1 , . . . , X_p ), we want to estimate $p_i = P_r(Y_i = 1 | X_1,...

$$p_i = \frac{e^{\beta_0+ \beta_1x_{i1} + \beta_2x_{i2} + \beta_3x_{i3} + \beta_4x_{i4} + ...}}{1+ e^{\beta_0

To come back to linear regression we define the logistic function as follow.
$$ \begin{aligned}
logit(p_i) = log(\frac{p_i}{1-p_i}) &= \beta_0+ \beta_1x_{i1} + \beta_2x_{i2} + \beta_3x_{i3} + \beta_4x_{i4}
\end{aligned}$$

We can define the odds :

$$\frac{odds(Y_i=1 | X1 = x_{i1}+1)}{odds(Y_i=1 | X1 = x_{i1})} = e^{\beta_1}$$

### Which indicator to construct the model ?

We use Maximum Likehood :

$$ L(\beta) = \Pi_{i=1}^n{ p_i^{y_i} * (1 - p_i)^{y_i}} $$
The goal is to maximise it by adjusting $\beta$ vector.

### Example on the the Wimbledon tennis tournament
```

```
We use a dataset from the Wimbledon tennis tournament for Women in 2013. We will predict the result for playe
```

{r, eval=TRUE, echo=FALSE}
id <- "1GNbIhjdhuwPOBr0Qz82JMkdjUVBuSoZd"
tennis <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",id), header = T)

test and train set

n = dim(tennis)[1]
n2 = n*(3/4)
set.seed(1234)
train = sample(c(1:n), replace = F)[1:n2]

reduction to two variables

tennis$ACEdiff = tennis$ACE.1 - tennis$ACE.2
tennis$UFEdiff = tennis$UFE.1 - tennis$UFE.2
head(tennis)

```
```{python, eval=TRUE, echo=FALSE, fig.height = 4, fig.width = 5, fig.align = "center"}
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#Collecting the dataset
tennis_dataset = r.tennis

# Overview of the data
#tennis_dataset.info()
print("Looking if the dataset is balanced : \n")
tennis_dataset["Result"].value_counts()
tennis_dataset.describe()

plt.scatter(tennis_dataset["ACEdiff"], tennis_dataset["UFEdiff"], c=tennis_dataset["Result"])
plt.show()
```

On R

```{r, eval=TRUE, echo=FALSE}
tennisTest = tennis[-train, ]
tennisTrain = tennis[train, ]
r.tennis2 = glm(Result ~ ACEdiff + UFEdiff, data = tennisTrain, family = "binomial")
summary(r.tennis2)
```

```
With the model, we can draw the slope which indicates the category of a point.
```

{r, eval=TRUE, echo=FALSE, fig.height = 4, fig.width = 5, fig.align = "center"}
#We calculate the slope
glm.b = -r.tennis2$coefficients[2]/r.tennis2$coefficients[3]
glm.a = -r.tennis2$coefficients[1]/r.tennis2$coefficients[3]

ggplot() + geom_point(aes(ACEdiff, UFEdiff, color = factor(Result)), data = tennisTrain,) + scale_color_manual(values = c("red", "green")) +
```

geom_abline(slope = glm.b, intercept = glm.a) +
theme_minimal()

```
We can write :

$$ \begin{aligned}
logit(p_i) = log(\frac{p_i}{1-p_i}) &= 0,31318 + 0,20856 * ACEDiff - 0,08272 * UFEDiff
\end{aligned}$$

We can observe AIC =  105.1


The confusion matrix is :
```

{r, eval=TRUE, echo=FALSE}
glm.Result_probs = predict(r.tennis2, newdata = tennisTest)
glm.Result_pred = ifelse(glm.Result_probs > 0.5, 1, 0)
glm.confusion_matrix = table(glm.Result_pred, tennisTest$Result)
glm.confusion_matrix

```
The accuracy rate is $\frac{15 + 8}{30} = 0.7667$.

The sensitivity is the percentage of true output giving Player1-winner among the population of true Player1-w
```

{r, eval=TRUE, echo=FALSE}
glm.sensitivity = glm.confusion_matrix[2,2]/(glm.confusion_matrix[1,2] + glm.confusion_matrix[2,2])
glm.sensitivity

```
The specificity is the percentage of true output giving Player2-winner (= Player1-looser) among the populatio
```

{r, eval=TRUE, echo=FALSE}
glm.specificity = glm.confusion_matrix[1,1]/(glm.confusion_matrix[1,1] + glm.confusion_matrix[2,1])
glm.specificity

```
The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-win
```

{r, eval=TRUE, echo=FALSE}
glm.precision = glm.confusion_matrix[2,2]/(glm.confusion_matrix[2,1] + glm.confusion_matrix[2,2])
glm.precision

```
So the F_Mesure is :
```

{r, eval=TRUE, echo=FALSE}
glm.fmesure = (2*glm.precision*glm.sensitivity)/(glm.sensitivity + glm.precision)
glm.fmesure

```
#### On Python with Scikit-learn
```

{python, eval=TRUE, echo=FALSE}
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix
import numpy as np

def defineTestTrainDatasetRandomly():
msk = np.random.rand(len(tennis_dataset)) < 0.75

```
train = tennis_dataset[msk]
test = tennis_dataset[~msk]

trainX = train[['ACEdiff', 'UFEdiff']]
trainY = train[['Result']]

testX = test[['ACEdiff', 'UFEdiff']]
testY = test[['Result']]
return (trainX, trainY, testX, testY)
```

meanAccuracy = 0
meanPerfModel = [0,0,0,0] # mean respectively of tn, fp, fn, tp
nbrOfIteration = 100
for i in range(0,nbrOfIteration):
trainX, trainY, testX, testY = defineTestTrainDatasetRandomly()
clf = LogisticRegression(C=1e5).fit(trainX, trainY.values.ravel())
labelsPredicted = clf.predict(testX)
meanPerfModel += confusion_matrix(labelsPredicted, testY.values.ravel()).ravel()
meanAccuracy += clf.score(testX, testY.values.ravel())
meanAccuracy /= nbrOfIteration
meanPerfModel = [i/nbrOfIteration for i in meanPerfModel]
print("Mean accuracy is : \n")
print(meanAccuracy)
print("Mean performance is : \n")
print(meanPerfModel)
sensivity = meanPerfModel[0]/(meanPerfModel[0]+meanPerfModel[2])
print("The Sensitivity is : " + str(sensivity))
specificity = meanPerfModel[3]/(meanPerfModel[3]+meanPerfModel[1])
print("The specificity is : " + str(specificity))
precision = meanPerfModel[0]/(meanPerfModel[0]+meanPerfModel[1])
print("The precision is : " + str(precision))
Fmesure = (2*precision*sensivity)/(precision+sensivity)
print("So, we can deduce that the F-mesure is : " + str(Fmesure))

```
Using the Logistic Regression model in scikit we obtain an accuracy of 0.74. In average, this model has 9.48 

## Decision trees and Random Forest


### Decision Tree
Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised lear

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of 

### Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and 


### Which indicator to construct the model ?

#### Entropy
```

```
Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder

Mathematically Entropy for 1 attribute is represented as:

$$ E(s) = \sum_{i = 1}^{c} -p_{i}\log_{2}(p_{i})$$

Mathematically Entropy for multiple attributes is represented as:

$$ E(T,X) = \sum_{c \in X}P(c)E(c)$$


#### Gini

You can understand the Gini index as a cost function used to evaluate splits in the dataset. It is calculated

$$ Gini = 1 - \sum_{i =1}^c(p_{i})^2$$


### Example on the the Wimbledon tennis tournament

#### On R
```

{r, eval=TRUE, echo=FALSE}
accuracyrate <- rep(na,40) deg="1:40" for (d in deg) { model <- randomforest(result ~ ace.1 + ace.2 ufe.1 ufe.2, tennistrain,
mtry="2," ntree="500," nodesize="d,importance" = true) yrandomforest="predict(model," newdata="tennisTest)"
model.result_probs="predict(model," model.result_pred="ifelse(model.Result_probs"> 0.5, 1, 0)
model.confusion_matrix = table(model.Result_pred, tennisTest$Result)
model.accuracyrate = (model.confusion_matrix[2,2] + model.confusion_matrix[1,1]) /30
accuracyrate[d] = model.accuracyrate
}

#The model with the smallest MSE has 14 nodesizes
which.min(accuracyrate)

#The best model is
model <- randomForest(Result ~ ACE.1 + ACE.2 + UFE.1 + UFE.2, tennisTrain,
mtry = 2, ntree = 500, nodesize=which.min(accuracyrate),importance = TRUE)

predictions <- model %>% predict(tennisTest)
data.frame(MSE = mean((predictions - tennisTest$Result)^2),
R2 = R2(predictions, tennisTest$Result),
RMSE = RMSE(predictions, tennisTest$Result),
MAE = MAE(predictions, tennisTest$Result))
```

```
The confusion matrix is :
```

{r, eval=TRUE, echo=FALSE}
model.Result_probs = predict(model, newdata = tennisTest)
model.Result_pred = ifelse(model.Result_probs > 0.5, 1, 0)
model.confusion_matrix = table(model.Result_pred, tennisTest$Result)
model.confusion_matrix

```
The accuracy rate is :
```

{r, eval=TRUE, echo=FALSE}
model.accuracyrate = (model.confusion_matrix[2,2] + model.confusion_matrix[1,1]) /30
model.accuracyrate
```

```
The sensitivity is the percentage of true output giving Player1-winner among the population of true Player1-w
```

{r, eval=TRUE, echo=FALSE}
model.sensitivity = model.confusion_matrix[2,2]/(model.confusion_matrix[1,2] + model.confusion_matrix[2,2])
model.sensitivity

```
The specificity is the percentage of true output giving Player2-winner (= Player1-looser) among the populatio
```

{r, eval=TRUE, echo=FALSE}
model.specificity = model.confusion_matrix[1,1]/(model.confusion_matrix[1,1] + model.confusion_matrix[2,1])
model.specificity

```
The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-win
```

{r, eval=TRUE, echo=FALSE}
model.precision = model.confusion_matrix[2,2]/(model.confusion_matrix[2,1] + model.confusion_matrix[2,2])
model.precision

```
So the F_Mesure is :
```

{r, eval=TRUE, echo=FALSE}
model.fmesure = (2*model.precision*model.sensitivity)/(model.sensitivity + model.precision)
model.fmesure

```
#### On Python with Scikit-learn
```

{python, eval=TRUE, echo=FALSE}
from sklearn.ensemble import RandomForestClassifier

meanAccuracy = 0
meanPerfModel = [0,0,0,0] # mean respectively of tn, fp, fn, tp
nbrOfIteration = 100
for i in range(0,nbrOfIteration):
trainX, trainY, testX, testY = defineTestTrainDatasetRandomly()
clf = RandomForestClassifier(max_depth=6, random_state=0).fit(trainX, trainY.values.ravel())
labelsPredicted = clf.predict(testX)
meanPerfModel += confusion_matrix(labelsPredicted, testY.values.ravel()).ravel()
meanAccuracy += clf.score(testX, testY.values.ravel())
meanAccuracy /= nbrOfIteration
meanPerfModel = [i/nbrOfIteration for i in meanPerfModel]
print(meanAccuracy)
print(meanPerfModel)

sensivity = meanPerfModel[0]/(meanPerfModel[0]+meanPerfModel[2])
print("The Sensitivity is : " + str(sensivity))
specificity = meanPerfModel[3]/(meanPerfModel[3]+meanPerfModel[1])
print("The specificity is : " + str(specificity))
precision = meanPerfModel[0]/(meanPerfModel[0]+meanPerfModel[1])
print("The precision is : " + str(precision))

Fmesure = (2*precision*sensivity)/(precision+sensivity)
print("So, we can deduce that the F-mesure is : " + str(Fmesure))

```
## KNN : K-nearest neighbors

### How does it work ?

To make predictions for an observation x, the KNN use the training observations to find the k closest trainin

### Comments

KNN is very good for non-linear classification. \\

However it doesn't give the coefficient for the predictors so we can't see their impacts. \\

It also needs a lot od training observations well-balanced. Otherwise if a class is over-represented, it woul

### Choosing k

To choose k, it can be useful to use cross-validation. We test multiple times the model with different k and

## Linear and quadratic Discriminant Analysis

### Bayes theorem

TO DO

### Linear Discriminant Analysis

TO DO

### Quadratic Discriminant Analysis

TO DO


## Support Vector machines

### Maximal margin classifier

A hyperplane in p dimensions can be written as follow : \newline

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_p * x_p = \beta_0 + X^T \beta = 0$$
It is a a p-1 dimensional subspace of $R^p$ \newline

The hyperplane leads to a natural classifier, depending on the side of the hyperplane where the new observati

If $\beta_0 + X^T \beta > 0 $ : it lies on one side of the hyperplane meaning it belongs to class labelled 1.
If $\beta_0 + X^T \beta < 0 $ : it lies on the opposite side of the hyperplane meaning it belongs to class la

Thus $y_i * (\beta_0 + X^T \beta) >0$ \newline

The distance of any point x to the hyperplane is given by :  \newline
$$d = \frac{1}{||\beta||}(\beta_0 + X^T \beta)$$

The best hyperplane is the maximal margin hyperplane which maximises the distance d from the training observa

It is an optimization problem : we want to classify well all observations and maximize d.  \newline

Let's call M, the minimal distance from the observations to the hyperplan. We write the problem :  \newline

\begin{align}
  & \max_{\beta_0, \beta} M \\
  & \text{subject to} \notag \Sigma_j \beta_j^2 = 1 \\
  & \text{for all training observations} \notag  y_i * (\beta_0 + X^T \beta) > M \\
\end{align}

### Support vector classifier

For some data sets a separating hyperplane does not exist, the data set is non-separable. \newline

To obtain a support vector classifier we relax the conditions that we had for the maximal margin hyperplane b
```

```
We write the new problem :  \newline

\begin{align}
& \max_{\beta_0, \beta} M \\
& \text{subject to} \Sigma_j \beta_j^2 = 1  \\
& \text{for all training observations} y_i * (\beta_0 + X^T \beta) > M*(1-\epsilon_i)\\
& 0 \leq \epsilon_i \text{and} \Sigma_i \epsilon_i \leq C \\
\end{align}

Once we have the hyperplane, we can deduce to which class the observation belong by projection. \newline

### Support Vector Machines

For some datasets a non-linear decicion boundary between the classes is more suitable than a linear decision

To compute the boundary, we use interactions terms between predictors or functions on predictors. \newline

An example can be : \newline
$$\beta_0 + \beta_1 * x_1^2 + \beta_2 * x_2 * x_1 + ... + \beta_p * x_p^3  = 0$$


$$f(X_i) = \beta_0 + \Sigma_i \beta_i * K(x_i,x_j) $$ where K is a function between $x_i$ and $x_j$ \newline

We write the new problem : \newline

\begin{align}
& \max_{\beta_0, \beta} M \\
& \text{for all training observations} y_i * f(X_i) > M*(1-\epsilon_i)\\
& 0 \leq \epsilon_i \text{and} \Sigma_i \epsilon_i \leq C \\
\end{align}

Once we have the hyperplane, we can deduce to which class the observation belong by projection. \newline

## Neural networks

TO DO

\clearpage

# Regression

## Overall

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determi

## Accuracy of a model

### MSE : Mean Squarred Error

The MSE mesures the mean accuracy of the predicted responses values for given observations.
There are two MSE : the train MSE and the test MSE. \\
The train MSE is use to fit a model while training. \\
The test MSE is use to choose between models already trained.
\\

Let's define the mean squared error or MSE.
$$MSE = \frac{1}{n} \sum_i(y_i-\hat{f}(x_i))^2$$ where $\hat{f}(x_i)$ is the prediction of $y_i$ obtained wit

Then the expected test MSE refers to the average test MSE that we would obtain if we repeatedly estimated
f using a large number of training sets, and tested each at $x_0$. So that the expected test MSE is :

$$E(y_0 - \hat{f}(x_0))^2 $$

$$ \begin{aligned}
E(y_0-\hat{f}(x_0))^2 &=
Var(\hat{f}(x_0))  + (f(x_0)-E(\hat{f}(x_0)))^2 + Var(\varepsilon)
\end{aligned}$$

$Var(\varepsilon)$ represents the irreductible error. This term can not be reduced regardless how well our st

$(f(x_0)-E(\hat{f}(x_0))^2 = [Bias(\hat{f}(x_0))]^2$ is the squared Bias and refers to the error that is intr

$Var(\hat{f}(x_0))$ is the Variance of the prediction at $\hat{f}(x_0)$ and refers to the amount by which $\h
```

```
### RMSE : Root Mean Squared Error

Root Mean Squared Error (RMSE), which measures the average prediction error made by the model in predicting t

$$RMSE = \sqrt {MSE} = \sqrt {\frac{1}{n} \sum_i (y_i-\hat{f}(x_i))^2} $$

### RSS : Residual Sum of Squares

We define the residual sum of squares (RSS) as the sum of the squares of residuals (deviations predicted from

Since $\hat{f}(x_i) = a + b * x_i$ and $y_i = a + b * x_i + \epsilon_i$ \\

$$RSS = \Sigma \epsilon_i^2 = \Sigma (y_i - \hat{f}(x_i))^2 = n * MSE$$

We want to minimize the RSS.

### RSE : Residual Standard Error

The residual standard error is the square root of the residual sum of squares divided by the residual degrees

$$RSE = \sqrt{\frac{1}{n-2}RSS}$$

### R squared statistic

In statistics, the coefficient of determination, denoted R² or r² and pronounced "R squared", is the proporti

$$R^2 = 1 - \frac{RSS}{TSS}$$
$$TSS = \Sigma (y_i - \bar{y}_i)^2$$ is the total sum of squares. TSS measures the total variance in the resp

TSS − RSS measures the amount of variability in the response that is explained.

$R^2$ measures the proportion of variability in Y that can be explained using X.

### MAE : Mean Absolute Error

Mean Absolute Error (MAE), an alternative to the RMSE that is less sensitive to outliers. It corresponds to t

$$MAE = \frac{1}{n} \sum_i |y_i-\hat{f}(x_i)|$$

## Simple Linear Regression

### Definition

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response

Simple linear regression lives up to its name: it is a very straightforward approach for predicting a quantit
$$Y ≈ \beta_0 + \beta_1 * X$$

## Multiple Linear Regression

### Definition

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that u

Formula and Calcualtion of Multiple Linear Regression
$$ y_{i}= \beta_{0} + \beta_{1}x_{i1} + \beta_{2}x{i2} + \beta_{p}x_{ip} + \epsilon$$

### Hospital Costs dataset

The next dataset (source F. E. Harrell, Regression Modeling Strategies) contains the total hospital costs of
```

{r, eval=TRUE, echo=FALSE}
id <- "1heRtzi8vBoBGMaM2-ivBQI5Ki3HgJTmO" # google file ID
hospitaldata <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id), header = T)
head(hospitaldata)

{r, eval=TRUE, echo=FALSE }

We only look at complete cases

hospitaldata <- hospitaldata[complete.cases(hospitaldata),] hospitaldata <- hospitaldata[hospitaldata$totcst> 0,]

histograms

```
par(mfrow = c(3, 3))
hist(hospitaldata$age, main = 'age')
hist(hospitaldata$num.co, main = 'num.co')
hist(hospitaldata$edu, main = 'edu')
hist(hospitaldata$scoma, main = 'scoma')
hist(hospitaldata$totcst, main = 'totcst')
hist(hospitaldata$meanbp, main = 'meanbp')
hist(hospitaldata$hrt, main = 'hrt')
hist(hospitaldata$resp, main = 'resp')
hist(hospitaldata$temp, main = 'temp')
hist(hospitaldata$pafi, main = 'pafi')
```

{r, eval=TRUE, echo=FALSE, fig.height = 4, fig.width = 5, fig.align = "center" }

```
#transformation
par(mfrow = c(1, 2))
hist(hospitaldata$totcst, main = 'totcst')
hist(log(hospitaldata$totcst), main = 'log(totcst)')
```

{r, eval=TRUE, echo=FALSE, fig.height = 4, fig.width = 5, fig.align = "center"}

```
ggplot() + geom_point(aes(age, totcst, color = as.factor(dzgroup) ), data = hospitaldata)
```

{python, eval=TRUE, echo=FALSE}

```
from sklearn import linear_model
hopital_dataset = r.hospitaldata
hopital_dataset.tail()
```

```
We can see that there is a lot of NaN values. \\
Remove NaN and null data :
```

{python, eval=TRUE, echo=FALSE}

```
hopital_dataset.describe()
hopital_dataset = hopital_dataset.dropna()
#hopital_dataset.info()
```

{python, eval=TRUE, echo=FALSE}
hopital_dataset = hopital_dataset[hopital_dataset["totcst"]>0]

cost to log(cost)

hopital_dataset["totcst"] = np.log(hopital_dataset["totcst"])

change the text labels to numbers because it's easier to process

hopital_dataset["dzgroup"] = pd.factorize(hopital_dataset["dzgroup"])[0]

hopital_dataset = hopital_dataset.drop("scoma", axis=1)
hopital_dataset = hopital_dataset.drop("race", axis=1)
hopital_dataset = hopital_dataset.drop("meanbp", axis=1)
hopital_dataset = hopital_dataset.drop("income", axis=1)
hopital_dataset = hopital_dataset.drop("hrt", axis=1)
hopital_dataset = hopital_dataset.drop("pafi", axis=1)

```
#### On R

We would like to build models that help us to understand which predictors are mostly driving the total cost.

Looking at the distribution of the cost we see we should apply a log transformation for a better distribution

We can calculate the MSE on the test set to evaluate the simple linear regression model.
```

{r, eval=TRUE, echo=FALSE}
set.seed(12345)
train.proportion = 0.7
train.ind = sample(1:nrow(hospitaldata), train.proportion* nrow(hospitaldata))
hospitaldata.train = hospitaldata[train.ind,]
hospitaldata.test = hospitaldata[-train.ind,]

fit = lm(log(totcst)~ age + temp + edu + resp + num.co + as.factor(dzgroup),
data = hospitaldata.train)

fit$coefficients

predictions <- fit %>% predict(hospitaldata.test)
data.frame(MSE = mean((predictions - log(hospitaldata.test$totcst))^2),
R2 = R2(predictions, log(hospitaldata.test$totcst)),
RMSE = RMSE(predictions, log(hospitaldata.test$totcst)),
MAE = MAE(predictions, log(hospitaldata.test$totcst)))

```
#### On Python with Scikit-learn
```

{python, eval=TRUE, echo=FALSE}

def defineTestTrainDatasetRandomly():
msk = np.random.rand(len(hopital_dataset)) < 0.75

```
train = hopital_dataset[msk]
test = hopital_dataset[~msk]

trainX = train.drop("totcst", axis=1)
trainY = train[['totcst']]

testX = test.drop("totcst", axis=1)
testY = test[['totcst']]
return (trainX, trainY, testX, testY)
```

trainX, trainY, testX, testY = defineTestTrainDatasetRandomly()
lm_reg = linear_model.Ridge(alpha=.5)
lm_reg.fit(trainX, trainY.values.ravel())
print(lm_reg.coef_)

lm_reg.intercept_

```
## Linear regression with interaction terms

### Definition

It is a regression which introduces an operation between two predictors like multiplication, division ...

### Hospital Costs dataset

####  On R

We use the same example than for simple linear regression.
```

{r, eval=TRUE, echo=FALSE}
fit_multiple = lm(log(totcst)~age*as.factor(dzgroup) + temp + edu + resp + num.co, data = hospitaldata.train)
fit_multiple$coefficients

```
We can calculate the MSE on the test set to evaluate the multiple linear regression model.
```

{r, eval=TRUE, echo=FALSE}
predictions <- fit_multiple %>% predict(hospitaldata.test)
data.frame(MSE = mean((predictions - log(hospitaldata.test$totcst))^2),
R2 = R2(predictions, log(hospitaldata.test$totcst)),
RMSE = RMSE(predictions, log(hospitaldata.test$totcst)),
MAE = MAE(predictions, log(hospitaldata.test$totcst)))

```
The MSE-test for multiple linear regression is worst than for simple linear regression.

Simple linear regression is the best model so far for this problem.

## K-nearest neighbor regression

It works the same way as the KNN for classification. \\

Given a value for k and a prediction point $x_i$, KNN regression identifies the k closest training observatio

Then it estimates $f(x_i)$ using the average of all the training responses in $N_i$.  \\

$$\hat{y_i} = f(x_i) = \frac{1}{k} \Sigma_{x_j \in N_i} y_j$$
\clearpage

# Validation techniques

## Sampling
```

```
This consists in dividing the dataset into a training set and a test set.

## Cross validation

R2, RMSE and MAE are used to measure the regression model performance during cross-validation.

### Validation set approach

The Validation Set Approach is a type of method that estimates a model error rate by holding out a subset of

#### Example on R
```

{r, eval=TRUE, echo=FALSE}

Split the data into training and test set

set.seed(123)
training.samples <- log(hospitaldata$totcst) %>% createDataPartition(p = 0.75, list = FALSE)
hospitaldata.train2 <- hospitaldata[training.samples,]
hospitaldata.test2 <- hospitaldata[-training.samples,]

Build the model

model <- lm(log(totcst) ~ age + as.factor(dzgroup) + temp + edu + resp + num.co, data = hospitaldata.train2)

Make predictions and compute the R2, RMSE and MAE

predictions <- model %>% predict(hospitaldata.test2)
data.frame(MSE = mean((predictions - log(hospitaldata.test2$totcst))^2),
R2 = R2(predictions, log(hospitaldata.test2$totcst)),
RMSE = RMSE(predictions, log(hospitaldata.test2$totcst)),
MAE = MAE(predictions, log(hospitaldata.test2$totcst)))

```
#### Example on Python
```

{python, eval=TRUE, echo=FALSE}
from sklearn import linear_model
from sklearn.model_selection import cross_validate
import numpy as np
import pandas as pd

hopital_dataset = r.hospitaldata
hopital_dataset = hopital_dataset.dropna()
hopital_dataset = hopital_dataset[hopital_dataset["totcst"]>0]
hopital_dataset["totcst"] = np.log(hopital_dataset["totcst"])
hopital_dataset["dzgroup"] = pd.factorize(hopital_dataset["dzgroup"])[0]
hopital_dataset = hopital_dataset.drop("scoma", axis=1)
hopital_dataset = hopital_dataset.drop("race", axis=1)
hopital_dataset = hopital_dataset.drop("meanbp", axis=1)
hopital_dataset = hopital_dataset.drop("income", axis=1)
```
```

hopital_dataset = hopital_dataset.drop("hrt", axis=1)
hopital_dataset = hopital_dataset.drop("pafi", axis=1)

def defineTestTrainDatasetRandomly():
msk = np.random.rand(len(hopital_dataset)) < 0.75

```
train = hopital_dataset[msk]
test = hopital_dataset[~msk]

trainX = train.drop("totcst", axis=1)
trainY = train[['totcst']]

testX = test.drop("totcst", axis=1)
testY = test[['totcst']]
return (trainX, trainY, testX, testY)
```

trainX, trainY, testX, testY = defineTestTrainDatasetRandomly()
lm_reg = linear_model.Ridge(alpha=.5)
lm_reg.fit(trainX, trainY.values.ravel())
cv_results = cross_validate(lm_reg, trainX, trainY, cv=5,
scoring={'r2':'r2', 'MSE': 'neg_mean_squared_error',
'MAE':"neg_median_absolute_error",
'RMSE': "neg_root_mean_squared_error"})

cv_results

```
### Leave One out cross-validation

Leave-one-out cross-validation is a special case of cross-validation where the number of folds equals the num

This method works as follow:

Leave out one data point and build the model on the rest of the data set
Test the model against the data point that is left out at step 1 and record the test error associated with th
Repeat the process for all data points
Compute the overall prediction error by taking the average of all these test error estimates recorded at step

#### Example on R
```

{r, eval=TRUE, echo=FALSE}

Define training control

train.control <- trainControl(method = "LOOCV")

Train the model

model <- train(log(totcst) ~ age + as.factor(dzgroup) + temp + edu + resp + num.co, data = hospitaldata, method = "lm",
trControl = train.control)

Summarize the results

print(model)

```
### k-Fold Cross-Validation

K-Fold Cross-Validation is where a given data set is split into a K number of sections/folds where each fold

We divide the set of data in k equals part and we use k-1 parts to train the model and 1 to test. We do do tha

Here are the steps :

1.Split the dataset into k equal partitions (or "folds") \\

2.For each fold \\

  One fold is used as the testing set and the union of the other folds as the training set \\

  Calculate testing accuracy for this fold : \\

  $$\hat{f_i}=\frac{1}{K}\sum_{j \in N_0}(y_j)$$
  $$MSE = \frac{k}{n}\sum_i{I(y_i≠\hat{y_i})}$$

3.Use the average testing accuracy as the estimate of out-of-sample accuracy : \\

We would use the cross-validation error : \\
$$CV_{k}=\frac{1}{k}\sum_i{MSE_i}$$
with $I(y_i≠\hat{y_i}) = 1$ if $y_i≠\hat{y_i}$, 0 else. So that we calculate the average of wrong predicted v

#### Example on R
```

{r, eval=TRUE, echo=FALSE}

Define training control

train.control <- trainControl(method = "cv", number = 10)

Train the model

model <- train(log(totcst) ~ age + as.factor(dzgroup)+ temp + edu + resp + num.co, data = hospitaldata, method = "lm",
trControl = train.control)

Summarize the results

print(model)
```
```