# Homework 1

Pattern Mining and Social Network Analysis

BOUYSSOU Gatien , de POURTALES Caroline, LAMBA Ankit

03 novembre, 2020

## Contents

# Classification

## Overall

Classification algorithms have categorical responses. In classification, we create a function f(X) that takes a vector of input variables X and predicts its class Y belonging to the set of classes C.

## Possibilities of models

There are classifiers as logistic regression, Decision trees, Perceptron / Neural networks, K-nearest-neighbors, linear and quadratic logistic regression, Bayes classifiers …

## Some indicators to analyze the results

### Sensitivity and recall

The sensitivity (also named recall) is the percentage of true defaulters that are identified.
It is the probability that our model predicts a patient positive given the fact that he has the disease.

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

### Specificity

The specificity is the percentage of non-defaulters that are correctly identified.
It is the probability that our model predicts a patient negative given the fact that he doesn't have the disease.

1 - specificity is the Type 1 error, it is the probability that our model predicts a patient positive given the fact that he doesn't have the disease.

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

### Precision

The precision is the proportion of true positive tests (individuals tested positive and really positive) among the positive tests (which can be right or not).

$$precision = \frac{TruePositiveTests}{PositiveTests}$$

### 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 :

- a, the number of elements in $\pi_1$ and $\pi_2$
- b, the number of elements in $\pi_1$ and not in $\pi_2$

- c, the number of elements in $\pi_2$ and not in $\pi_1$
- d, the number of elements not in both $\pi_1$ and $\pi_2$

| | in $\pi_2$ | not in $\pi_2$ |
|---|---|---|
| in $\pi_1$ | a | b |
| not in $\pi_1$ | c | d |

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

### Mutual information

Mutual information is calculated between two variables. It measures the reduction of uncertainty for one variable given a known value of the other variable. 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 probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

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 compute a separate loss for each class label per observation and sum the result.

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

## Metrics

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

### MSE : Mean Squared Error

Let's define the mean squared error or MSE.

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

where :

$$1_{y_i - \hat{f}(x_i)} = \begin{cases} 1 & \text{if } y_i \neq \hat{f}(x_i) \\ 0 & \text{otherwise} \end{cases}$$

It is a too much simple indicator so let's look at other evalutaion criterions.

### AIC : Akaike information criterion

The Akaike information criterion (AIC) is an estimator of in-sample prediction error. In-sample fit estimates the likelihood of a model to predict the future values.
The AIC score rewards models that achieve a high goodness-of-fit score and penalizes them if they become overly complex.

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

$$AIC = -2 * \log L + 2p$$

where : L is the likelihood function of the parameters in the model and p the number of predictors used in the model.

$2p$ is a penalization term : adding predictor which don't improve well enough L gives a worse AIC.

To use AIC for model selection, we simply choose the model giving the smallest AIC over the set of models considered.

### BIC : Bayesian information criterion

In statistics, the Bayesian information criterion or Schwarz information criterion is a criterion for model selection among a finite set of models. It is based, in part, on the likelihood function and it is closely related to the Akaike information criterion.

BIC measures the trade-off between model fit and complexity of the model.

For the least squares model, the BIC is, up to irrelevant constants, given by :

$$BIC = -2 * \log L + p * log(n)$$

where : L is the likelihood function of the parameters in the model, p the number of predictors used in the model and n the number of data.

$p * log(n)$ is a penalization term : adding predictor which don't improve well enough L gives a worse BIC.

To use BIC for model selection, we simply choose the model giving the smallest BIC over the set of models considered.

## Logistic Regression

### How it works

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

$$p_i = \frac{e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + ...}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + ...}}$$

To come back to linear regression we define the logistic function as follow.

$$logit(p_i) = log(\frac{p_i}{1 - p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + ...$$

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 player 1 (win=1 or loose=0) based on : the number of aces won by each player, and, the number of unforced errors commited by both players. The data set is a subset of a data set from https://archive.ics.uci.edu/ml/datasets/Tennis+Major+Tournament+Match+Statistics.

We want to predict if Player-1 is a winner.

```
##          Player1      Player2 Result ACE.1 UFE.1 ACE.2 UFE.2 ACEdiff UFEdiff
## 1      M.Koehler   V.Azarenka      0     2    18     3    14      -1       4
## 2     E.Baltacha   F.Pennetta      0     0    10     4    14      -4      -4
## 3      S-W.Hsieh      T.Maria      1     1    13     2    29      -1     -16
## 4       A.Cornet       V.King      1     4    30     0    45       4     -15
## 5  Y.Putintseva   K.Flipkens      0     2    28     6    19      -4       9
## 6 A.Tomljanovic B.Jovanovski      0     6    42    11    40      -5       2
```

Instead of 4 variables, we decrease to two variables :
ACEdiff = ACE.1 - ACE.2 and
UFEdiff = UFE.1 - UFE.2

ACEdiff is the amount of aces Player-1 in addition to Player-2.
UFEdiff is the amount of unforced errors Player-1 in addition to Player-2.

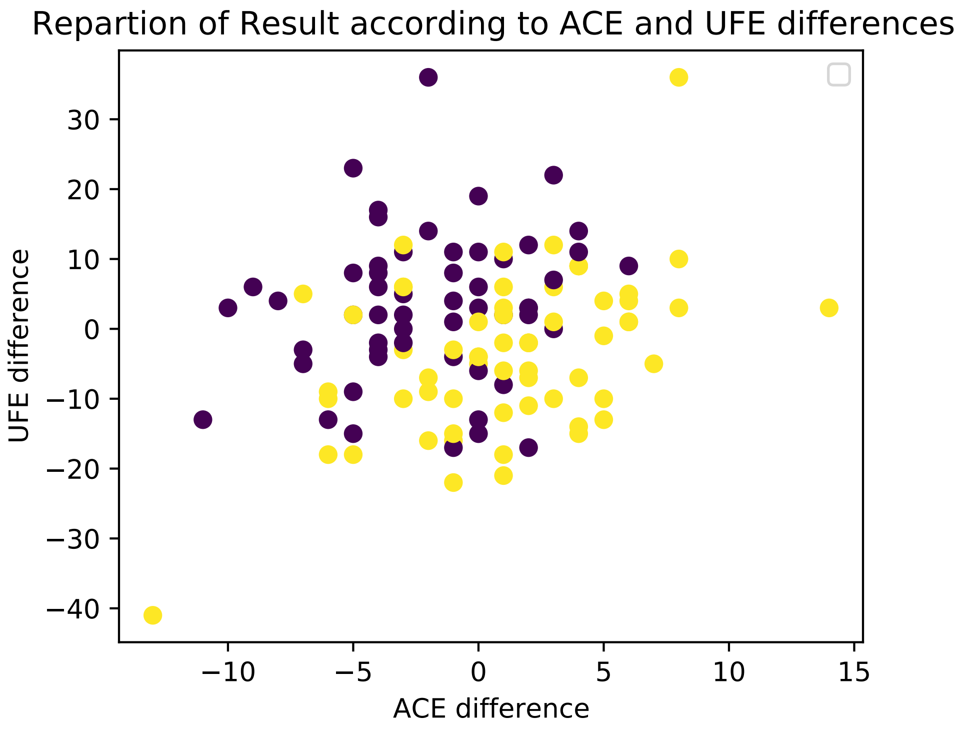
This is the plot of UFE difference according to ACE difference between Player 1 and 2 with different colors for different results.

```
## Looking if the dataset is balanced :
```

```
## 1    63
## 0    55
## Name: Result, dtype: int64
```

```
##             Result       ACE.1       UFE.1  ...       UFE.2     ACEdiff     UFEdiff
## count  118.000000  118.000000  118.000000  ...  118.000000  118.000000  118.000000
## mean     0.533898    2.974576   20.177966  ...   20.466102   -0.296610   -0.288136
## std      0.500977    2.835857   10.248728  ...   11.444912    4.356564   11.410822
## min      0.000000    0.000000    4.000000  ...    2.000000  -13.000000  -41.000000
## 25%      0.000000    1.000000   13.000000  ...   12.000000   -3.000000   -7.750000
## 50%      1.000000    2.000000   18.000000  ...   18.000000    0.000000    1.000000
## 75%      1.000000    4.000000   25.750000  ...   27.000000    2.000000    6.000000
## max      1.000000   14.000000   54.000000  ...   55.000000   14.000000   36.000000
```

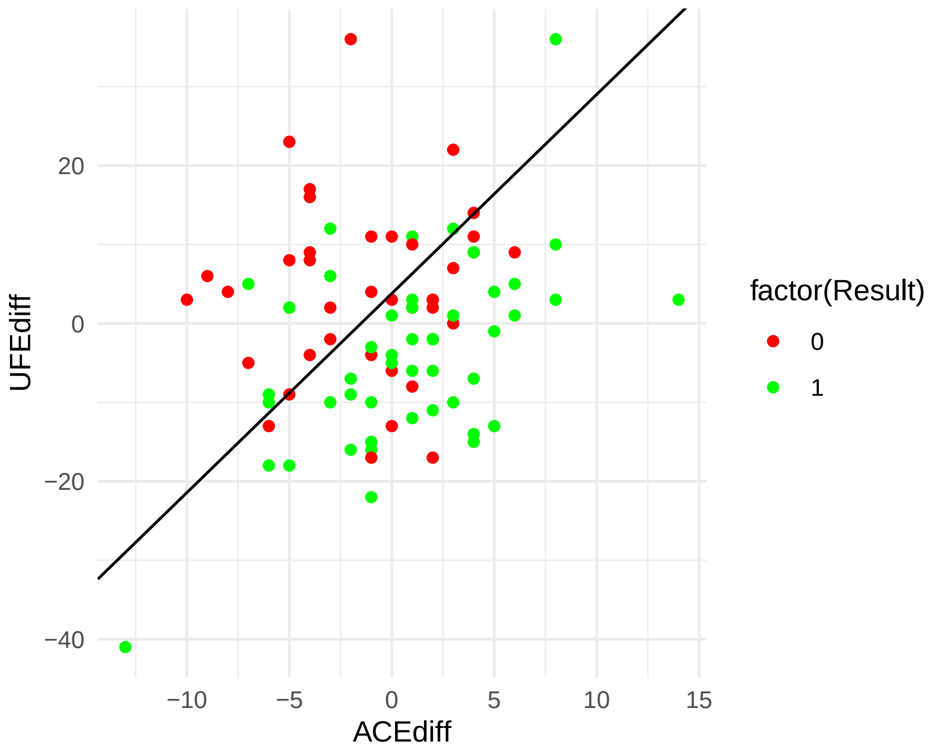
```
##
## [8 rows x 7 columns]
```



**On R**

```
##
## Call:
## glm(formula = Result ~ ACEdiff + UFEdiff, data = tennisTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9252  -0.4239   0.1520   0.3518   0.7938
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.56926    0.04756  11.970  < 2e-16 ***
## ACEdiff      0.04031    0.01096   3.678 0.000411 ***
## UFEdiff     -0.01620    0.00410  -3.950 0.000161 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1987929)
##
##     Null deviance: 21.591  on 87  degrees of freedom
## Residual deviance: 16.897  on 85  degrees of freedom
## AIC: 112.52
##
## Number of Fisher Scoring iterations: 2
```

With the model, we can draw the slope which indicates the sepration between classes.



We can write :

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

We can observe AIC = 105.1

The confusion matrix is :

```
##
## glm.Result_pred  0  1
##               0 15  5
##               1  2  8
```

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-winner :

```
## [1] 0.6153846
```

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

```
## [1] 0.8823529
```

The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-winner (even if not winner) :

```
## [1] 0.8
```

So the F_Mesure is :

```
## [1] 0.6956522
```

### On Python with Scikit-learn

```
## The mean accuracy is :

## 0.7538141799877036

## The confusion matrix  :

## [9.37, 3.08, 4.06, 12.76]

## The Sensitivity is : 0.6976917349218168

## The specificity is : 0.8055555555555556

## The precision is : 0.7526104417670683

## So, we can deduce that the F-mesure is : 0.7241112828438948
```

Using the Logistic Regression model in scikit, we obtain an accuracy of 0.74.
In average, this model has 9.48 True Positive, 3.23 False Positive, 4.4 False Negative, 12.63 True Positive.

## Decision trees and Random Forest

### Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model. It can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

### Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks. During their training, they are creating a multitude of decision trees to output the class that is the mode of the classes or mean/average prediction of the individual trees.

### 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 it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.

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 by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions and it is easy to implement, whereas information gain, favors smaller partitions with distinct values.

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

### Example on the the Wimbledon tennis tournament

**On R** The confusion matrix is :

```
##    0  1 class.error
## 0 21 17   0.4473684
## 1 10 40   0.2000000
```

The accuracy rate is :

```
## [1] 0.6931818
```

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

```
## [1] 0.7017544
```

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

```
## [1] 0.6774194
```

The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-winner (even if not winner) :

```
## [1] 0.8
```

So the F_Mesure is :

```
## [1] 0.7476636
```

### On Python with Scikit-learn

```
## The mean accuracy is :

## 0.6813511459128588

## The confusion matrix is :

## [8.69, 4.33, 5.31, 11.78]

## The Sensitivity is : 0.6207142857142857

## The specificity is : 0.7312228429546865

## The precision is : 0.6674347158218126

## So, we can deduce that the F-mesure is : 0.6432272390821614
```

## 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 training observations to x. Then, x is assigned to the class to which the most of these observations belong. We just need to define the notion of distance which can be euclidean.

### 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 of training observations well-balanced. Otherwise, if a class is over-represented, it would assign too much this label.

### Choosing k

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

## Discriminant Analysis

Discriminant analysis is a technique that helps classifying 2 or more groups into clusters. This technique uses - the Bayes' theorem - the covariance (the shape of one group of point) - and the center of this group.

The aim is to use those 3 components to create boundaries between the classes. The boundaries are composed by a set of points that have the same chance to belong to either classes.

Here, we are going to talk about two models. The first one is the Linear Discriminant Analysis (LDA). To use that model, each class need to be drawn from a multivariate Gaussian distribution and the covariance for each class needs to be equals (the groups of points need to have the same shape).

Quadratic Discriminant Analysis (QDA) is a bit similar to (LDA) because it assumes that each class are drawn from a Gaussian distribution. However, the covariance doesn't need to be the same for each class.

The advantages of such models is that they have a good accuracy and they are robust to outliers (because it is based on statistical distribution of all the observations).

## Support Vector machines

### Maximal margin classifier

A hyperplane in p dimensions can be written as follow :

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_p * x_p = \beta_0 + X^T\beta = 0$$

It is a a p-1 dimensional subspace of $R^p$

The hyperplane leads to a natural classifier, depending on the side of the hyperplane where the new observation lies.
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 labelled -1.

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

The distance of any point x to the hyperplane is given by :

$$d = \frac{1}{||\beta||}(\beta_0 + X^T\beta)$$

The best hyperplane is the maximal margin hyperplane which maximises the distance d from the training observations.
It is an optimization problem : we want to classify well all observations and maximize d.
Let's call M, the minimal distance from the observations to the hyperplan. We write the problem :

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

**Support vector classifier**

For some data sets a separating hyperplane does not exist, the data set is non-separable.
To obtain a support vector classifier, we relax the conditions that we had for the maximal margin hyperplane by allowing for a "budget" C of misclassifications.
We write the new problem :

$$\begin{aligned} &\max_{\beta_0,\beta} M && (2)\\ &\text{subject to } \Sigma_j\beta_j^2 = 1 && (3)\\ &\text{for all training observations } y_i * (\beta_0 + X^T\beta) > M * (1 - \epsilon_i) && (4)\\ &0 \le \epsilon_i \text{ and } \Sigma_i\epsilon_i \le C && (5) \end{aligned}$$

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

**Support Vector Machines**

For some datasets a non-linear decicion boundary between the classes is more suitable than a linear decision boundary.
To compute the boundary, we use interactions terms between predictors or functions on predictors.
An example can be :

$$\beta_0 + \beta_1 * x_1^2 + \beta_2 * x_2 * x_1 + ... + \beta_p * x_p^3 = 0$$

$$f(X_i) = \beta_0 + \Sigma_i\beta_i * K(x_i, x_j)$$

where K is a function between $x_i$ and $x_j$

We write the new problem :

$$\max_{\beta_0,\beta} M \tag{6}$$

$$\text{for all training observations } y_i * f(X_i) > M * (1 - \epsilon_i) \tag{7}$$

$$0 \leq \epsilon_i \text{ and } \Sigma_i \epsilon_i \leq C \tag{8}$$

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

## Neural networks

Neural networks can be decomposed as three parts the input the hidden layers and the output. Those parts are in fact nodes link together with edges that are given weights describing the strenght of the connection. With a given input the value of the first node of the hidden layer will correspond to the inputs time the weights on the edges going to the node. This product will give a result that needs to go through an activation function before giving the value of the current node. This process of computing the values of the nodes from the inputs to the output is called feedforward.

Once, we have computed the output, we compare our prediction to the answer. Then, we backpropagate the error. It means that depending on the error, the value of the nodes and the learning rate, we are going to updates the weights.

Once the weights are updated, we iterate until the amount of errors made by the model becomes acceptable.

# Regression

## Overall

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

## Metrics

### 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 used to fit a model while training.
The test MSE is used 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 with the model.

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 :

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

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

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

$(f(x_0) - E(\hat{f}(x_0))^2 = [Bias(\hat{f}(x_0))]^2$ is the squared Bias and refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. If the bias is low the model gives a prediction which is close to the true value.

$Var(\hat{f}(x_0))$ is the Variance of the prediction at $\hat{f}(x_0)$ and refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. If the variance is high, there is a large uncertainty associated with the prediction.

### RMSE : Root Mean Squared Error

Root Mean Squared Error (RMSE), which measures the average prediction error made by the model in predicting the outcome for an observation. That is, the average difference between the observed known outcome values and the values predicted by the model. The lower the RMSE, the better the model.

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

**RSS : Residual Sum of Squares**

We define the residual sum of squares (RSS) as the sum of the squares of residuals (deviations predicted from actual empirical values of data) :

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 of freedom. Mean Square Error. The mean square error is the mean of the sum of squared residuals, i.e. it measures the average of the squares of the errors. Lower values (closer to zero) indicate better fit.

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

**R squared statistic**

In statistics, the coefficient of determination, denoted R$^2$ or r$^2$ and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable

$$R^2 = 1 - \frac{RSS}{TSS}$$
$$TSS = \Sigma(y_i - \bar{y}_i)^2$$

is the total sum of squares. TSS measures the total variance in the response Y.

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.

**Adjusted R statistic**

The adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance

$$AdustedR^2 = 1 - \frac{\frac{RSS}{n-p-1}}{\frac{TSS}{n-1}}$$

where : p is the number of predictors in the model and n the size of the data set.

**MAE : Mean Absolute Error**

Mean Absolute Error (MAE), an alternative to the RMSE that is less sensitive to outliers. It corresponds to the average absolute difference between observed and predicted outcomes. The lower the MAE, the better the model is.

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

### Mallow's Cp

Mallows's Cp addresses the issue of overfitting. Indeed, model selection statistics such as the residual sum of squares always get smaller as more variables are added to a model.

Recall : $RSS = MSE * n$

If there are p predictors :

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

where $\hat{\sigma}$ is the variance estimate of the error.

### AIC and BIC

AIC and BIC can both be used with the likelihood : $L = RSS/n$.

If p is the number a predictors used in the model and n the size of the data set :

$$AIC = -2 * \log\frac{RSS}{n} + 2p$$

$$BIC = -2 * \log\frac{RSS}{n} + p * log(n)$$

Both AIC and BIC penalize the model if adding one predictor doesn't improve well enough the model. They both need to be minimalized.

## Simple Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.

Simple linear regression lives up to its name: it is a very straightforward approach for predicting a quantitative response Y on the basis of a single predictor variable X. It assumes that there is approximately a linear relationship between X and Y. Mathematically, we can write this linear relationship as

$$Y \approx \beta_0 + \beta_1 * X$$

## Multiple Linear Regression

### Definition

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable

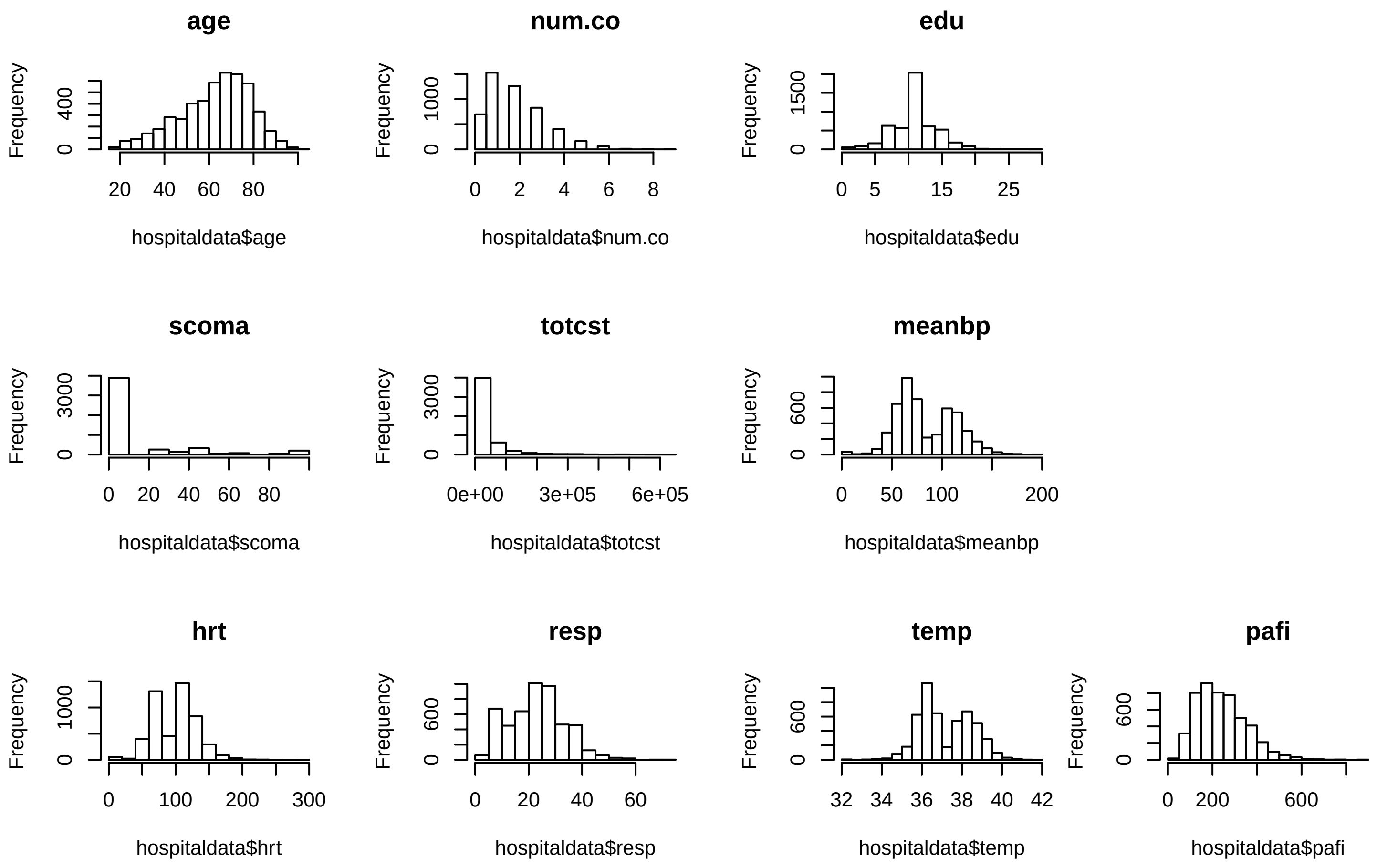
Formula and Calcualtion of Multiple Linear Regression

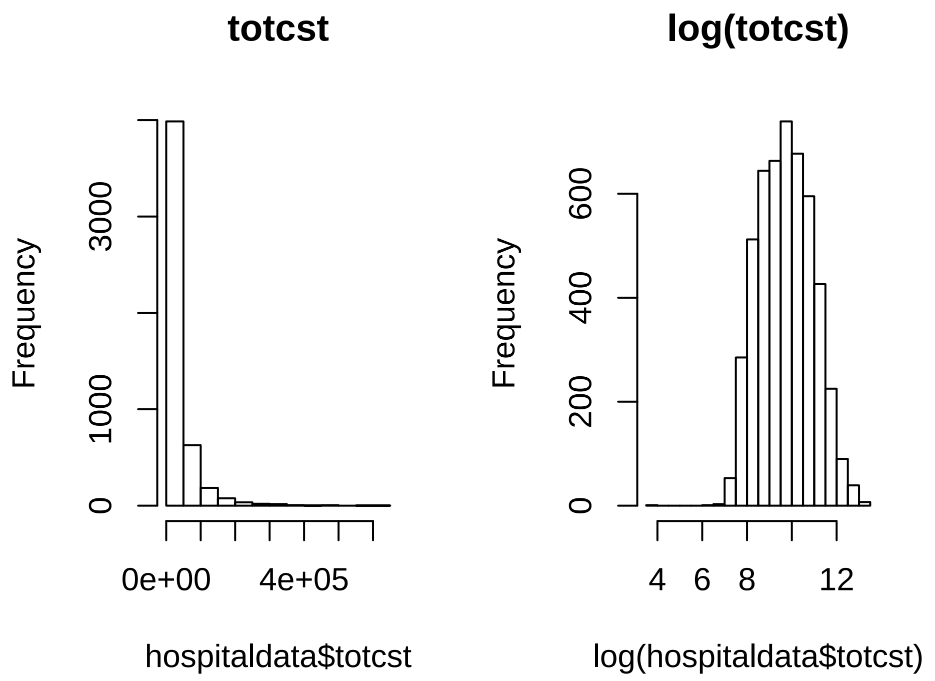
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

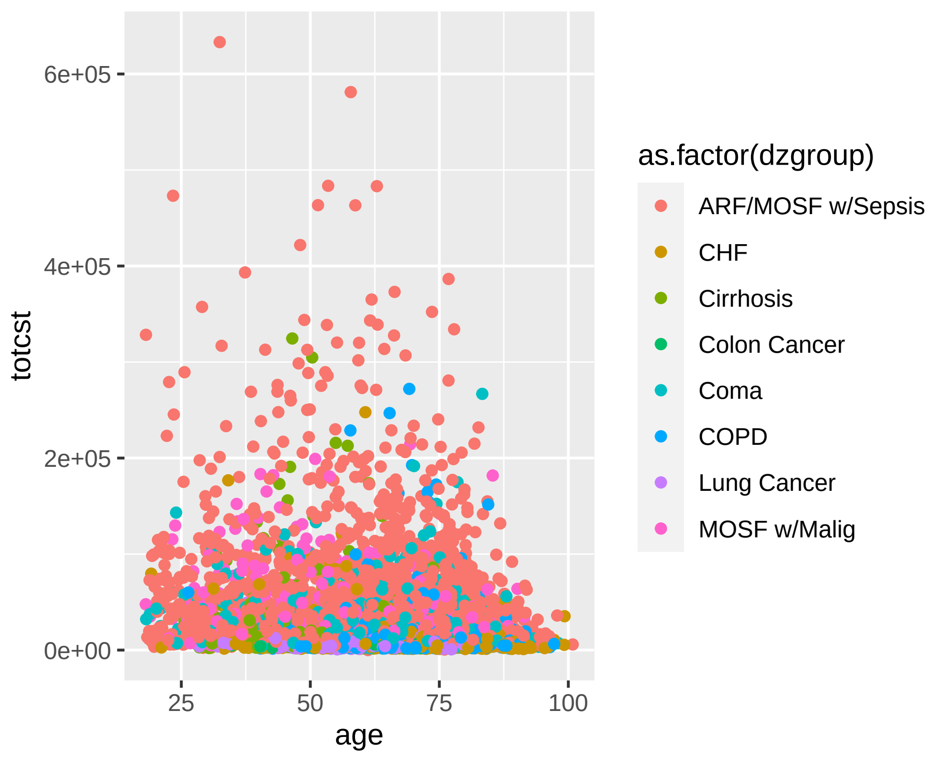
**Hospital Costs dataset**

The next dataset (source F. E. Harrell, Regression Modeling Strategies) contains the total hospital costs of 9105 patients with certain diseases in American hospitals between 1989 and 1991. The different variables are :

```
##     age           dzgroup num.co edu     income scoma totcst  race meanbp hrt
## 1 62.85       Lung Cancer      0  11   $11-$25k     0     NA other     97  69
## 2 60.34         Cirrhosis      2  12   $11-$25k    44     NA white     43 112
## 3 52.75         Cirrhosis      2  12 under $11k     0     NA white     70  88
## 4 42.38       Lung Cancer      2  11 under $11k     0     NA white     75  88
## 5 79.88 ARF/MOSF w/Sepsis      1  NA               26     NA white     59 112
## 6 93.02              Coma      1  14               55     NA white    110 101
##   resp  temp   pafi
## 1   22 36.00 388.00
## 2   34 34.59  98.00
## 3   28 37.40 231.66
## 4   32 35.00     NA
## 5   20 37.90 173.31
## 6   44 38.40 266.63
```







```
##           age           dzgroup  num.co  edu  ...    hrt  resp   temp    pafi
## 4955   70.42  ARF/MOSF w/Sepsis       4   12  ...   57.0    10  37.50  217.00
## 4956   68.62               COPD       2   12  ...  110.0    10  36.20  135.00
## 4957   66.07  ARF/MOSF w/Sepsis       1    8  ...  104.0    22  35.70  280.00
## 4958   55.15               Coma       1   11  ...    0.0     8  38.59  218.50
## 4959   81.54  ARF/MOSF w/Sepsis       1    8  ...   69.0    24  36.20  230.41
##
## [5 rows x 13 columns]
```

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

```
##                age       num.co  ...         temp         pafi
## count  4960.000000  4960.000000  ...  4960.000000  4960.000000
## mean     62.668768     1.912298  ...    37.218907   241.739718
## std      15.892083     1.397348  ...     1.301758   110.397054
## min      18.120000     0.000000  ...    32.000000    12.000000
## 25%      52.597500     1.000000  ...    36.200000   156.977500
## 50%      65.090000     2.000000  ...    36.900000   226.660000
## 75%      74.472500     3.000000  ...    38.300000   308.000000
## max     100.850000     9.000000  ...    41.700000   890.380000
##
## [8 rows x 10 columns]
```

**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 linear regression model.

```
##                    (Intercept)                           age
##                    7.848450651                  -0.006610500
##                           temp                           edu
##                    0.075106886                   0.026642892
##                           resp                        num.co
##                   -0.004025088                  -0.043124831
##          as.factor(dzgroup)CHF    as.factor(dzgroup)Cirrhosis
##                   -1.405057960                  -0.923605089
## as.factor(dzgroup)Colon Cancer        as.factor(dzgroup)Coma
##                   -1.458633036                  -0.452564283
##         as.factor(dzgroup)COPD  as.factor(dzgroup)Lung Cancer
##                   -1.242045040                  -1.688831933
## as.factor(dzgroup)MOSF w/Malig
##                   -0.256302673
```

```
##         MSE        R2      RMSE       MAE
## 1 0.8903145 0.3742044 0.9435648 0.7440975
```

### On Python with Scikit-learn

```
## Ridge(alpha=0.5)
```

```
## The MSE is :1.1343804263062733
```

```
## The R squared statistic is :0.2210964033805175
```

## Linear regression with interaction terms

### Definition

It is a regression which introduces operations between multiple predictors like multiplication, division …

**Hospital Costs dataset**

**On R** We use the same example than for linear regression with interaction terms.

```
##                          (Intercept)                                   age
##                         7.6621089858                         -0.0042188616
##                 as.factor(dzgroup)CHF          as.factor(dzgroup)Cirrhosis
##                        -1.1077199267                         -0.5713961342
##     as.factor(dzgroup)Colon Cancer                 as.factor(dzgroup)Coma
##                        -1.3449083302                          0.3636117761
##                as.factor(dzgroup)COPD        as.factor(dzgroup)Lung Cancer
##                        -1.2782357031                         -2.1009737479
##     as.factor(dzgroup)MOSF w/Malig                                   temp
##                         0.3648729712                          0.0762561613
##                                  edu                                  resp
##                         0.0263008321                         -0.0039559477
##                               num.co            age:as.factor(dzgroup)CHF
##                        -0.0431664138                         -0.0046449011
##    age:as.factor(dzgroup)Cirrhosis age:as.factor(dzgroup)Colon Cancer
##                        -0.0062251497                         -0.0018857174
##         age:as.factor(dzgroup)Coma           age:as.factor(dzgroup)COPD
##                        -0.0129436877                          0.0002001618
##   age:as.factor(dzgroup)Lung Cancer age:as.factor(dzgroup)MOSF w/Malig
##                         0.0066143639                         -0.0103087258
```

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

```
##           MSE        R2      RMSE       MAE
## 1 0.8872275 0.3764346 0.9419275 0.7408729
```

The MSE-test for linear regression with interaction terms is better than for mutliple linear regression.

## 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 observations to $x_i$ represented by $N_i$.

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

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

## Validation techniques

These are techniques of Cross validation :

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

### Validation set approach

The Validation Set Approach, also called Sampling, is a type of cross-validation that estimates a model error rate by holding out a subset of the data from the fitting process (creating a testing dataset). The model is then built using the other set of observations (the training dataset).

### Example on R

```
##         MSE        R2      RMSE       MAE
## 1 0.8842554 0.3711666 0.9403485 0.7451318
```

### Example on Python

```
## Ridge(alpha=0.5)

## The MSE is : 1.1124098575803072

## The R squared statistic is : 0.2012559523176728
```

### 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 is used as a testing set at some point.

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 that k times in order to use each part as a test part.

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 :

$$\hat{f}_i = \frac{1}{K} \sum_{j \in N_0} (y_j)$$

$$MSE = \frac{k}{n} \sum_i I(y_i \neq \hat{y_i})$$

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 $I(y_i \neq \hat{y_i}) = 1$ if $y_i \neq \hat{y_i}$, 0 else. So that we calculate the average of wrong predicted values.

### Example on R

```
## Linear Regression
##
## 4960 samples
##    6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4464, 4464, 4464, 4464, 4464, 4464, ...
## Resampling results:
##
##   RMSE       Rsquared  MAE
##   0.9348169  0.379955  0.7467515
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

### Leave One out cross-validation

Leave-one-out cross-validation is a special case of cross-validation where the number of folds equals the number of instances in the data set.

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 the prediction
- 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 2.

### Example on R

```
## Linear Regression
##
## 4960 samples
##    6 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 4959, 4959, 4959, 4959, 4959, 4959, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9348588  0.3785367  0.7467919
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```