Machine learning notes :

1st chapter : data preprocessing

* Missing data: accomplishing
* Catagorical variabial: Label Encoder : Associate every catagorie to a number : i.e. france = 1 , spain = 2 ….

Problem : france < spain which is not true : =🡺 we create instead dummy variable :

Column for each catagorie, i.e spain = 1 0 , france = 0 1

* We need to split our data into two data, taining + test
* Note : transform and fit used to center the data, using the standard score

X\* = (X-E)/V where E = mean , E = mean (moyenne), and V = standard deviance

(Ecart type)

* For Feature scalling , we need to fit (calculate E and V) and transform(replace the data) of the training set, and then , we should only transform the test, based on E and V calculated previously

2nd Chapter : Regression:

* Simple Regression: y = aX+b, Simplest machine learning model, the model algorithm calculate a and b that give the least SUM (y-y’)², y is the actuall data, y’, the calculated data based on a and b

In python we use sklear.linear\_model.LinearRegressor, class, we contrcust, and then we fit in to our training set ,

* Dummy variables: Categorical data, as mentioenned in data sets, in our multiple linear regression model we use n-1 for each variables for each categorie
* Notes: SL = Significance Level, the propability that the null hypothesis(variable independant) is true, the lower the value, the more significant it is , we consier a bare minimum of 5% (Variables having SL score more than 5% won’t be considered )
* Multiple Linear regression Is same as simple linear regression
* Polynominal linear regression : y = b0 + b1\*x1 + b2\*x22 ….., basicly turning one independent variable into a matrix of feature and applying same princicple as multiple linear regression
* Decision Tree regression : divied dataset by value of indepandant variables, then get the mean of each area for value(DecisionTreeRegressor)

Notes: all these regression works the same way in python, loading class, calling constructor, fitting regressor into the dataset

Due the discontinuity of Decision Tree Model we need to use higher resolution to plot results properly

* Random forest: multiple Decision tree based on subsets from initial dataset, predicition made by taking the mean of all the predicted value from the Decision trees (still discontinued, needing higher resolution plot

Notes: in R, dataset[1] returns the dataframe, matrix(n,1) unlike dataset$Salary which return a vector

3rd Chapter : Classification

* Logisitic regression: basicly linear regression for categorical data, return propabitlies of such feature occurring
* KNN(K-Nearest Neighbours) : choose k nearest observations based on some metric(Euclidience Distance, Manhatin Distance) , then associated the current observation the dominant categorie, exp: k = 5, 3 observations A and 2 observations were B, then the current observation will be associated to A
* SVM: Support Vector Machine: picks two observation , any observation is basicly a vector, and these two points are kinda the ‘extreme’ case in each category, for example ML for identifying apples from oranges, the two observations chosen would be the most “appely” orange(orange looking more likely like an apple) and vice versa for the second point, these two observations are called “Support Vectors” which will determine the new linear separation by maximizing the sum of distances between the separator line and the two support vector
* For linearly inseparable vectors, we can map these vectors to a higher dimension (1D -> 2D by squaring function) then we might be able to separate these vectors, if not , we keep increasing mapping to higher dimension.

Note : Even though its great technique to use, its very resource consuming task (computer intensive) to map to higher dimension, separate, and then go back to the normal dimension , so we would mainly use a technique called the “Kernel Trick”.

Keywords :

* + RBF : Radius Basis Function, real-valued function whose value depends only on the distance between the origin/the two points :

Notes : is generally the Euclidian distance but it can be any norm function

* Map/mapping : refers to either, morphism or function, generalizes the idea of a function
* Kernel function intuition : , where is mapping for n to m dimensional space, usually m is higher than n, , will require us to to compute first the and and then do the dot product, which can be really expensive and m can be really large, after all the trouble of going to higher dimensional spaces, the result is just a scalar, we can avoid the trouble of going to higher dimensional space if we manage to find a clever kernel function

The Gaussian RBF Kernel : , where is the landmark ( the center of the function) , and determines the radius of the function, then we can divide our observation into two part where  > 0 and = 0

For more complicated cases we can use more than one Kernel function to divide our observations :

Other Kernel functions :

* + - Sigmoid Kernel :
    - Polynomial Kernel :
* Naïve Bayes : Classification based on Bayes Theorem, example : let’s take two categories A and B , and some oberservation having some features X and we calculate and by the following formula for A (resp. B ) , where :
  + is the prior propabilty
  + is the Mariginal Likelihood , which is calculated based on some pre-chosen radius
  + is the Likelihood

Notes: it’s called “Naïve” because it supposes that all the variables are initially independent

* CART : Classification and Regression Trees, Classification for predicting certain categorical variables i.e “YES” or “NO” , and regression trees for predicting variables (based on the mean) , it’s the one of the easiest models yet also one of the oldest, that’s why it’s being boosted/used by new methods/models such as Random forest tree
* Decision Tree (Note): Feature Scalling only needed for plotting, since it’s not based on the Euclidian distance
* Random Forest : multiple decision trees, built from randomly chosen subset from the dataset, then each new observation with the feature that has was the voted the most by the decision trees