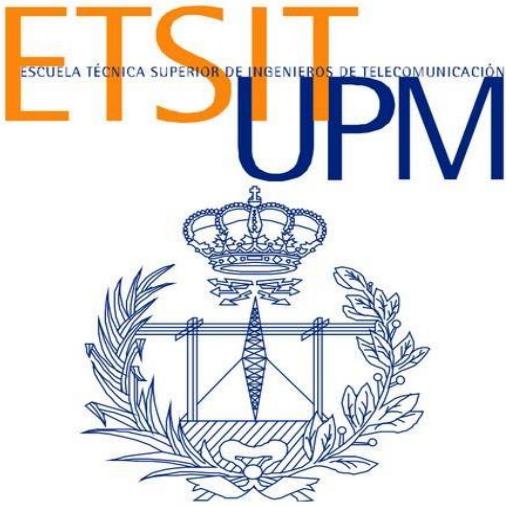


Course:

Machine Learning Lab

Mid-term Activities Report

Methodology and Tools for Developing Machine Learning models

Name: **Olle Elvland**

Date: 2022-11-23

CONTENTS

1. METHODOLOGY TO DEVELOP MACHINE LEARNING MODELS

OSA : Obstructive sleep Apnea Syndrome

With more than over 100 million individual cases of obstructive sleep worldwide (OSA), the sleeping disease is considered as one of the most common sleep-related breathing disorders in the world. OSA arises when the human's breathing cycle is disrupted due to the upper airway blocking or obstructed airway during their sleep. Which results in unwanted breathing patterns. Forcing the chest muscles to breath in a very undesired way, in order to pull air into your lungs. The condition causes the affected ones to breath too shallow or even i many cases stops breathing for a small amount of time. The disorder may occur multiple times during the same night.

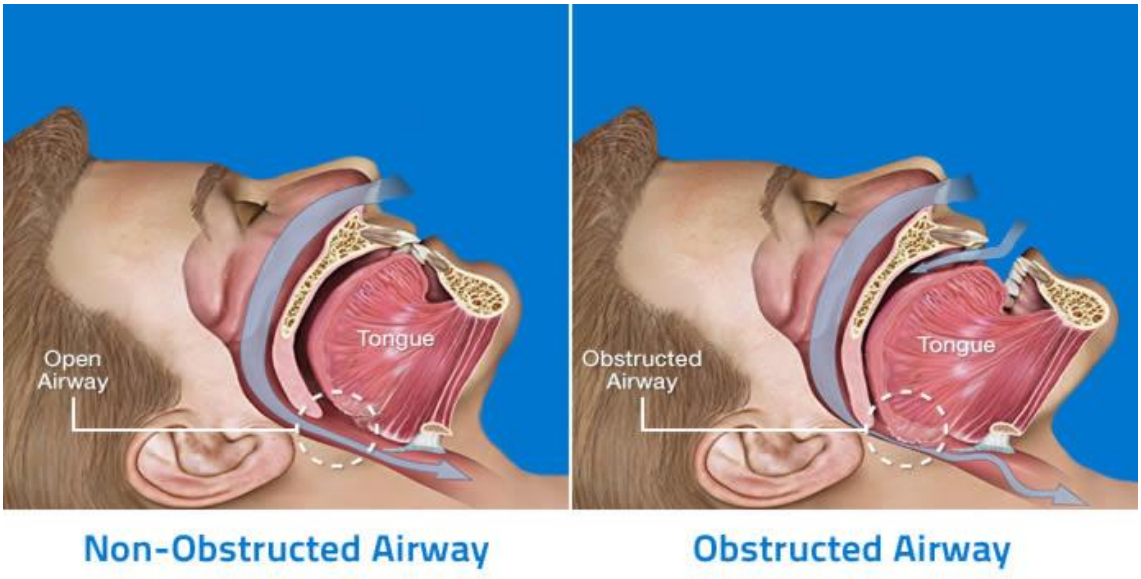


Figure 1 OSA illustration

However, approximately 80%-90 % (Kapur,1997) of the people who are suffering from the disease are unaware of their condition. This is a big issue since the disorder may heavily affect a person's impact negatively on the person's life. For example, some very typical OSA symptoms are:

- **Loudly snores**
- **Excessive sleepiness**
- **Morning headaches**
- **Not feeling rested enough after sleeping**
- **Dry mouth**

- **High blood pressure**
- **Daily mood changes**

Treatments

Unfortunately, in many cases, OAS is not easily detected because of the difficulty of its diagnosis. On other hand, there exist a few diagnosis procedures.

Generally, the most common way to diagnose a patient is through a Polysomnography (PSG), or a standard one-night sleep study, performed at a supervised laboratory or hospital sleep unit. The doctors are collecting data regarding the patient's sleep patterns. The data may be for e.g., checking brain activity, breathing patterns, movements and blood oxygen levels.

Another way to check on your own is a through a home sleep apnea test. Like the PSG test, the doctors are collecting data through some handout machines in order to monitor certain data when it comes to normal sleep deviations. Containing certain machines which help the doctor measure airflow to your lungs, blood oxygen degrees, and snoring depth. Unfortunately, All the testing for potentially severe conditions may take long time, and the waiting list for a polysomnography investigation is in many cases long-lasting .

The discovery of sleep apnea disease was found back in 1965. Though, it wasn't until 1985 that any treatments were introduced for eliminating airway obstruction sleep apnea. The only effective treatment back in the days for OSA was tracheostomy, which bypasses the upper airway obstruction. Nowadays, there exist superior treatments depending on what the core problem of a person's sleeping disorder is. Some well-known treatments for OSA are e.g. Continuous positive airway pressure (CPAP), Bi-Level positive airway pressure (BiPAP) and Auto-titrating PAP (APAP).

The main objective for me as a data scientist in this assignment is to assist the medical team, by looking at certain relationships between different and relevant data which has been collected using a certified and reliable app. Then applying different machine learning technology and techniques to ease the detection of OSA on users. The goal is to research if there is a way to identify if a person is , or is more likely to suffer from severe Sleep apnea Syndrome by understanding the relationship between clinical data like gender and age with the probability of having an OSA diagnosis.

Apnea-hypopnea Index

The measurement which determines the severity of OSA is typically called Apnea-Hypopnea index (AHI). It represents the amount of apnea (times breathing stops) and hypopnea events per hour of sleep. The Apnea-Hypopnea scale is divided into 4 categories for adults:

Apnea-Hypopnea Index

- calculates sleep apnea severity based on the total number of complete cessations (apnea) and partial obstructions (hypopneas) of breathing per hour of sleep.

For adults:

- an AHI of less than 5 is considered *normal*
- an AHI of 5 to 15 is considered *mild*
- an AHI of 15 to 30 is considered *moderate*
- an AHI greater than 30 is considered *severe*

When combined with a mild to severe AHI score and self-reports of excessive daytime sleepiness, this positive diagnosis of sleep apnea will lead your doctor to talk to you about treatment options for sleep apnea.

Figure 2 AHI index table

If the AHI value is lower or equal to 10, it is considered as healthy. An AHI over 30 is considered severe OSA. This index variable will be the most vital variable in this project in order to determine whether a person should undergo treatment regarding OSA or not.

Figure 3. Sleep Apnea

1.1. Machine Learning approach

The methodology pipeline used is primarily based around the knowledge I've gained during the classes and recommended books on machine learning techniques on predictive and descriptive learning. Some main aspects and guidelines I followed during the project are:

- Data description. Data preparation: As data preprocessing and preparation is one of the most important and time-consuming processes within data science work. As the quality and representation of our data for our database, is crucial for understanding and making the database parameters as suitable as possible for all the machine-learning models. In terms of both accuracy and efficiency. The cleaning procedure and understanding of useful and non-useful data is a vital part for machine learning models, in order to fully inference and find patterns which may guide us to proper conclusions regarding our models' output evaluation.

- Exploratory data analysis: Once our database has been optimized, another essential task is to inference with all the different variables. In order to understand as much information about specific patterns or relationships depending on which values, in order to draw conclusions regarding the ML models output.

- Training, validation and testing datasets. Depending on whatever the goal is to find an optimal model for regression or a classification prediction task. The algorithms for the training procedure within our machine learning model were applied to a specific training dataset, which is a sample of data used to fit the model. Later on, a validation dataset will be applied for the purpose of providing an unbiased evaluation of the already pretrained model to find the best model. In this case, comparison of a few error metrics gives us a decent measurement regarding the models' predictive quality. Finally, the testing dataset was applied to our model and predictions and evaluation results were collected.

- **Machine Learning models**: The assignment is mainly split up into two different parts, one regression part and a classification section. The regression part is focused on applying various regression algorithms and comparing each machine learning model's performance

based on how well it predicts AHI with input data from various parameters. In the classification part, ML-models are used to try to find certain patterns in order to classify a patient's OSA severity. With intentions of hopefully finding patterns in medical records and data to prevent future patients from developing OSA.

- **Evaluation:** Evaluation of a final model is a very crucial step for getting a solid understanding of how well the trained hypothesis function will generalize to new and future target data. One important thing to remember is that the model should preferably be as unbiased as possible from the trained data. Unfortunately, we only have a limited amount of data with various samples provided to work with. A solid method to overcome the bias problem is to split up the samples into a training subset and a test subset. Where the test data is used for the evaluation part. Another approach is to apply cross-validation techniques. By applying the "unseen" test data to evaluate the model, a good way to analyze the result is by using different accuracy metrics such as MSE,MAE and R-Squared to calculate the error term for regression models. In classification problems, Accuracy, confusion matrixes and ROC graphs are used to evaluate a models classify performance.

- **Conclusions:** In the final part, a discussion will take place. Where an analytic aspect will take place, to discuss the most interesting results provided throughout the work.

2. MLOps

MLOps or machine learning operations is a core strategy between developers, operations and data scientist. This concept comes in handy when the workflow for data scientist project needs to be improved and increase in efficiency. In many cases the better quality of data, the better the final quality of your model will provide. Therefore, MLOps goal is to make the experimentation of the data and model development better than it is today. The concept is also to make the deployment of newer and updated models into production with quality assurance. This is and will be a very important topic in future development och machine learning and AI. The reasoning for this is that the development and popularity of AI is growing immense. This growth won't stop since we are constantly developing new methods,

algorithms and models. Therefore, the collaboration between developers need to easier needs to be dealt with. I personally think MLOps will be one of the hot topics in upcoming years since the collaboration needs to find better strategies for optimizing the organizations abilities to work with future AI techniques.

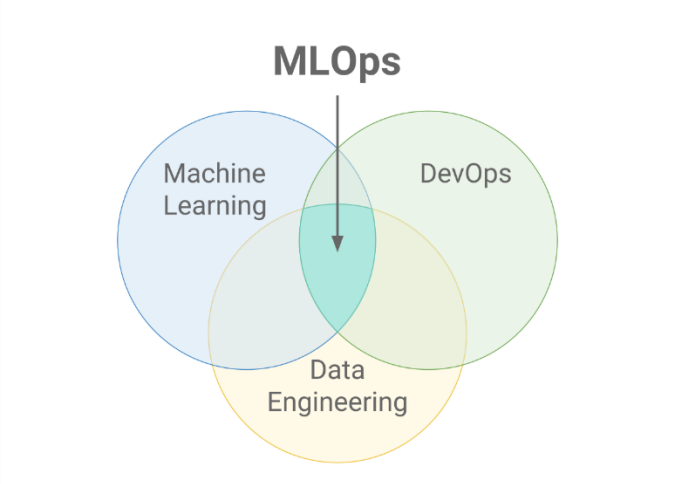


Figure 4 MLOps

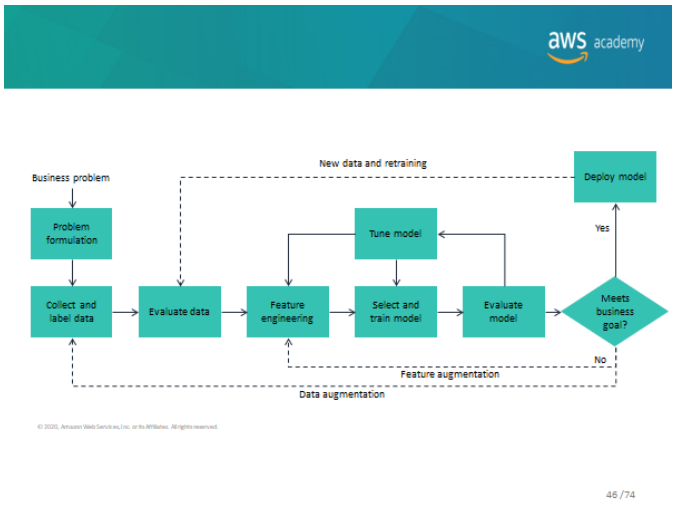


Figure 5 Pipeline for machine learning problem

2.1. Tools to develop Machine Learning models

Two of the most widely used data science programming language for visualizing and conducting data analysis is being used for this machine learning project: *Python* and *R.*

R is a well-known programming mainly focused on statistical analysis. With its strength of computing different kind of statistical and many veterans considered R is the state of art in terms of graphical visualization. The language was developed in 1993, and just like many other programming languages R is an open-source free software and is used by statiscaians, data miners and analysts. R provides some of what many data scientist believe to be the best library tools for helping with various challenges a standard data science project contains. The list below is a few famous libraries and packages which was used in the OSA project:

- **Readxl:** A package to ease the reading of data from by importing Excel sheets into R file.
- **Writexl : :** A package to ease the writing of data from by importing to an Excel sheets from an R script.
- **Dplyr:** Data manipulation library.
- **Visdat:** Visualization library for representing data graphically.
- **Naniar:** Package for manipulate missing data values.
- **Corrplot:** Package for visual investigative tool on correlation matrix.
- **ggplot2 :** Solid R package dedicated for data visualization.
- **lattice :** add-on package is a powerful and elegant high-level data visualization system with an emphasis on multivariate data.

Big data is frequently occurring in machine learning projects. R provides a lot of great features and tools for managing large amount out data and is therefore a great language for certain data science problems.

As mentioned, *Python* is considered as the base for many data scientist. It is believed to be one of the most popular programming languages in the word. The language was first released in 1990 and has been open source ever since. It has become very popular over last year's due to its extremely high uses for all types of programming tasks. Thanks to its simplicity the high-level language is very easy to understand and learn, since most of the syntaxes try to mimic the human language as much as possible. Also, du to its open source availability, has led to all sorts of people have helped the language grow incredibly fast. Pythons possibility are limitless, and the learning curve is said to be fairly low. Thanks to its popularity, hundreds of different useful libraries and frameworks is being ready to be. The growth of interest in recent years within AI, big data, machine learning and data

sciene has also been one of the main factors why Python has become one state of art language. Some of the most helpful libraries (which was also used in the assignment) in python that is tailor suited for a data scientist :

- **Pandas:** Great framework for data analysis and data handling. Great tool for provide data manipulation control like data frames.
- **Numpy:** Library used high-level math functions and numerical computing with data operations.
- **Matplotlib:** data visualization, cross-platform and graphical plotting library for python
- **Scikit-Learn:** Great library for working with complex data

Anaconda is an open-source distribution of the Python and R languages that aims to simplify package managements and development. The python distribution may be interfered as an environment , package and data collection management.

Python is one of the fast growing programming language and is also considered as one of the superior data science, because of its rich tools in terms of performing great mathematical and statistical tools.

The IDE used for the python part of the machine learning laboratory work is the web IDE called **Google Colaboratory.** The environment is 100 percent web based only suited for python. Colab provide excellent tools for data scientist since its main focus is to implement Deep learning and Machine learning projects assisted by cloud storage. Google Colab is a very popular web IDE because of the system's configurations you are working on is not relevant. The computational resources won't matter. This is a very essential for AI projects as it is a very computational heavy field.

3. Data Preparation

In the beginning of the project the initial assignment was to perform a data cleansing of the provided dataset. The cleaning part was singly performed by various R tools.The tools used and why be described more in details below. **Readxl** and **Writexl** was used for importing and exporting all the data from and to an excel sheet which is where all the data are stored just like a database. **Dplyr** is another library used for manipulating the provided data. **Tidyr** was applied for "tidying" open certain data values. **Visdat** is used for plotting to the ease understanding of the data. The **Naniar** package was used for easily having a way to deal with some missing values.

To differentiate syntax code and normal text, I've emphasized the corresponding syntax representing by highlighting the syntax code for R text in blue.

Initially as described above we read the data and then converting it into a dataframe with the using the Readxl library package:

```
df_tmp <- read_excel(paste(Data_Directory, Input_file, sep = ""))
```

The data needs to be converted into a data frame to be able to format the data as table:

```
class(df_tmp)
```

```
df_tmp = as.data.frame(df_tmp)
```

```
class(df_tmp)
```

```
head(df_tmp,2)
```
Shows two instances of rows of the data frame

```
  Patient
    P0001
    P0002
                                                                                                    Comentarios
  es el Patient0002 (fotos) 3 (sentado) y 4 (tumbado) original,\r\n el uno era una prueba y \r\nse ha quitado
                                                                      Es el Patient0006 (sentado) y 7 (tumbado)
  Audios tumbado Fotos Audio fs KHz Gender EPWORTH  IAH IAH Supino IAH Lateral Peso Talla IMC Edad PerCervical
              si    si           16 hombre      NA 71.0          -1          -1   82   168  -1   39          -1
              si    si           16 hombre      NA 29.6          -1          -1  119   174  -1   56          48
  Fumador Roncador         Enfermedades Sala/Ruidos Imagen Dialecto DIST EXT OJOS DIST BARB-LOB Cansancio
       ns       ns Septo_Nasal_Desviado          nd     nd       nd            NA          <NA>      <NA>
       si       ns                   nd          nd     nd       nd            NA          <NA>      <NA>
  Concentrarse PerdRespNoche HiperT EstHOSP
          <NA>          <NA>     NA      NA
          <NA>          <NA>     NA      NA
```

summary(df_tmp)

```
   Patient           Comentarios        Audios tumbado         Fotos           Audio fs KHz         Gender
 Length:873         Length:873         Length:873         Length:873         Length:873         Length:873
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character



    EPWORTH           IAH           IAH Supino         IAH Lateral           Peso              Talla
 Min.   : 8.00   Min.   :  0.00   Length:873         Length:873         Length:873         Min.   : -1.0
 1st Qu.:11.00   1st Qu.:  6.30   Class :character   Class :character   Class :character   1st Qu.:165.0
 Median :12.00   Median : 14.20   Mode  :character   Mode  :character   Mode  :character   Median :171.0
 Mean   :12.04   Mean   : 20.36                                                            Mean   :171.1
 3rd Qu.:13.00   3rd Qu.: 30.00                                                            3rd Qu.:178.0
 Max.   :20.00   Max.   :108.60                                                            Max.   :199.0
 NA's   :821     NA's   :224                                                               NA's   :196
      IMC           Edad          PerCervical       Fumador            Roncador          Enfermedades
 Min.   :-1    Min.   :-1.00   Min.   :-1.00   Length:873         Length:873         Length:873
 1st Qu.:-1    1st Qu.:40.00   1st Qu.:38.00   Class :character   Class :character   Class :character
 Median :-1    Median :49.00   Median :41.00   Mode  :character   Mode  :character   Mode  :character
 Mean   :-1    Mean   :49.28   Mean   :40.19
 3rd Qu.:-1    3rd Qu.:59.00   3rd Qu.:43.00
 Max.   :-1    Max.   :88.00   Max.   :53.00
 NA's   :195   NA's   :195     NA's   :195
 Sala/Ruidos           Imagen            Dialecto         DIST EXT OJOS     DIST BARB-LOB        Cansancio
 Length:873         Length:873         Length:873         Min.   : 8.109   Length:873         Length:873
 Class :character   Class :character   Class :character   1st Qu.: 9.153   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Median : 9.444   Mode  :character   Mode  :character
                                                          Mean   : 9.510
                                                          3rd Qu.: 9.866
                                                          Max.   :11.000
                                                          NA's   :568
 Concentrarse       PerdRespNoche          HiperT           EstHOSP
 Length:873         Length:873         Min.   :0.0000   Min.   :0.0
 Class :character   Class :character   1st Qu.:0.0000   1st Qu.:0.0
 Mode  :character   Mode  :character   Median :1.0000   Median :1.0
                                       Mean   :0.8986   Mean   :0.6
                                       3rd Qu.:1.0000   3rd Qu.:1.0
                                       Max.   :9.6680   Max.   :1.0
                                       NA's   :850      NA's   :853
```

vis_dat(df_tmp) visdat provide a graphically understanding of all existing types within the dataset which each variable holds:

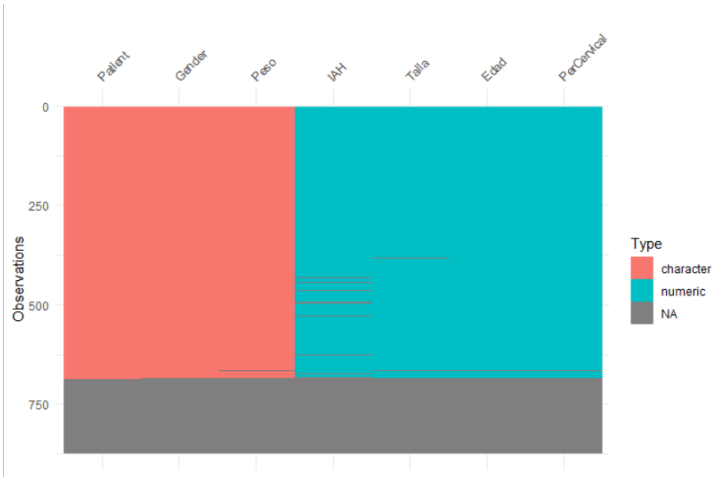


Figure 6 Visdat plot before dealing with NANs

In the data cleaning part a very controversial topic is on how to tackle missing values problem. The provided dataset had a few columns where missing values were assigned with a default value of -1. The following line of code replaces the missing values into NANs:

```
df_tmp <- replace_with_na_all(df_tmp1,condition = ~.x == -1)
```

With the purpose of removing useless samples all NAN samples which contained a NAN value was withdrawn by following line:

```
df_final <- df_tmp2 %>% drop_na()
```

```
vis_dat(df_tmp)
```

Figure 7 Visdat plot after removing NANs

A new BMI feature was added to the database. The feature takes the patient *height* and *weight* into

account as the formula is dependent on. The code snipplet below shows us on how to dynamically add a new column to a present database:

```
df_final$bmi <-
        with(df_final, Weight/(Height/100.0)^2)
```

The OSA severity response label also needs to be added to dataset because it's a vital parameter for our classification dilemma :

```
df_final <- df_final %>%
 mutate(osa =ifelse(IAH<=10,"Healthy", ifelse(IAH>=30,"Severe","Mild")))
```

Now all data is pretty much cleaned up and ready to be applied to our EDA process.

4. EXPLORATORY DATA ANALYSIS (EDA)

EDA implies the critical process of performing initial investigations on data, to find patterns, testing hypothesis and checking assumptions with the aid of summary statistics and graphical representations. In order to analyze the data and present the results in a simple format for easy understanding. However, there are no simple guidelines on how to proceed with the task, it's up to the data scientist himself to find a pattern and analyze the given data so the representation is followed in a simple way. Good EDA representation should preferably be:

- Self-explanatory
- Anyone who reads the analysis should be able to understand the meaning of the concluded data.
- Less is more. Fewer words, more charts, diagrams and pictures

In our case study, we wanted to investigate the significant connection between each variable, and its correlation to the final AHI and OSA severity variable value. In the end, these two variables are the most interesting ones. Because the main goal of the project is to hopefully generate a good prediction model to ease the understanding of why a person is diagnosed with OSA, and what a person's AHI value is depended on.

Just like in previous section only R code and libraries have been utilized. Some libraries used for data visualization is: **ggplot2, corrplot, gridExtra, lattice.**

3.2 EDA for regression

EDA is mostly about focusing on generating charts, diagram and pictures and to find patterns. Code snippet below show creation of an histogram :

```
hist(df_OSA$IAH, col = "red", xlab ="IAH Value", main = "Distribution Histogram of IAH values", labels = TRUE, )
```

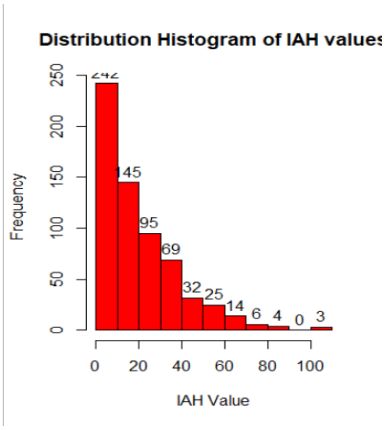


Figure 8 Generated histogram

And how to generate an scatterplot in R :

```
plot(df_OSA$Age,df_OSA$IAH)
```

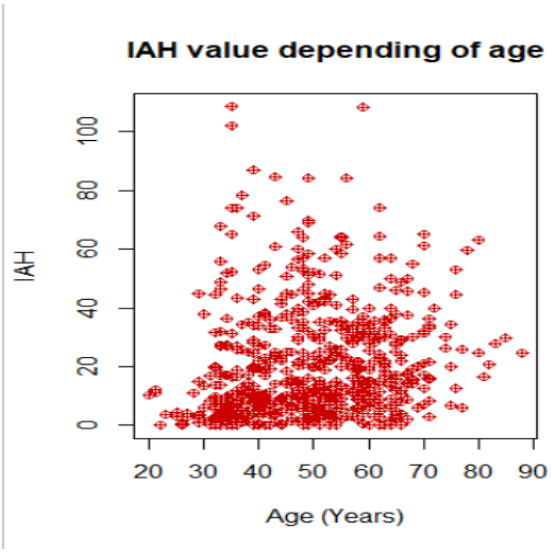
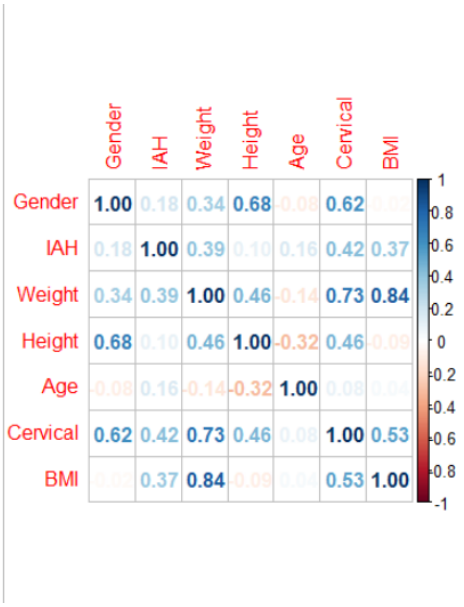


Figure 9 Scatterplot in R

Correlation coefficients is a great way on analyze whatever parameters have correlation or not. The code for plotting a correlation matrix on the clinical data variables:

corrplot(M, method="number")



Linear regression is a well-known regression model used to predict certain quantitative and continuous values. The code shown below written in R builds and fits a simple linear regression model with weight as the independent variable and IAH as the dependent response variable we wish

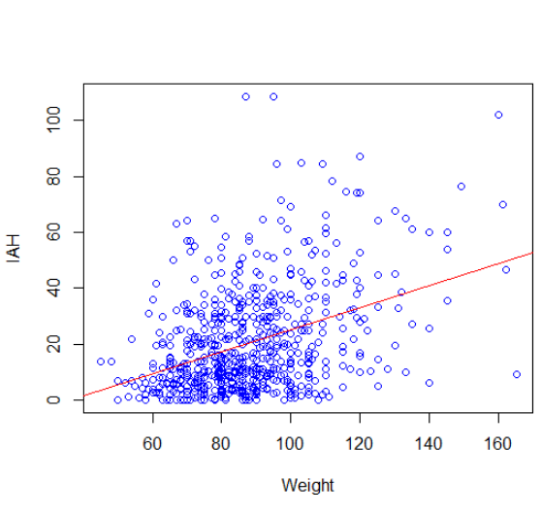
to predict:

*lm_one_feature = lm(IAH ~ Weight) (*lm is the syntax in R for fitting a linear regression model*)*

The code below creates a visual representation of a red fitted linear line:

plot(Weight,IAH,col = "blue")

abline(lm(IAH ~ Weight), col="red")



summary(lm_one_feature) (summary of the features of the linear model)

```
Residuals:
    Min      1Q  Median      3Q     Max
-41.692 -11.942  -4.510   9.512  88.337

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -14.43567    3.31991  -4.348  1.6e-05 ***
Weight        0.39653    0.03704  10.706  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 17.14 on 633 degrees of freedom
Multiple R-squared:  0.1533,    Adjusted R-squared:  0.152
F-statistic: 114.6 on 1 and 633 DF,  p-value: < 2.2e-16
```

Usually, Multiple linear regression is a more common approach building a model with specific features can be created using the same lm commando but with more variables to it:

lm.fit=lm(IAH~Weight+Height+Cervical+Age+Gender)

summary(lm.fit)

```
Residuals:
    Min      1Q  Median      3Q     Max
-37.640 -11.002  -3.342   8.502  84.189

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.52123   18.31341  -1.339 0.181062
Weight        0.28082    0.05873   4.782 2.17e-06 ***
Height       -0.22789    0.10586  -2.153 0.031715 *
Cervical      1.15975    0.31370   3.697 0.000237 ***
Age           0.21019    0.05930   3.545 0.000422 ***
Gender        1.01429    2.41410   0.420 0.674519
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.42 on 629 degrees of freedom
Multiple R-squared:  0.228,     Adjusted R-squared:  0.2219
F-statistic: 37.16 on 5 and 629 DF,  p-value: < 2.2e-16
```

Classification EDA

The second part of the project is a classification problem. With the goal of creating and fitting a model which will make it possible to predict whether a patient is suffering for Healthy or Severe OSA. Trying to fit a logistic regression model will be performed using following code:

x <-glm(formula = OSA ~ BMI + Age + Cervical, family = binomial, data =df_OSA_male)

(The glm function stands for generalized linear models)

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3665  -0.8823  -0.4005   0.9565   2.1032

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -12.06078    2.17783  -5.538 3.06e-08 ***
BMI           0.13319    0.04399   3.027  0.00247 **
Age           0.05293    0.01215   4.355 1.33e-05 ***
Cervical      0.13158    0.06493   2.026  0.04272 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 385.33  on 277  degrees of freedom
Residual deviance: 307.12  on 274  degrees of freedom
AIC: 315.12

Number of Fisher Scoring iterations: 4
```

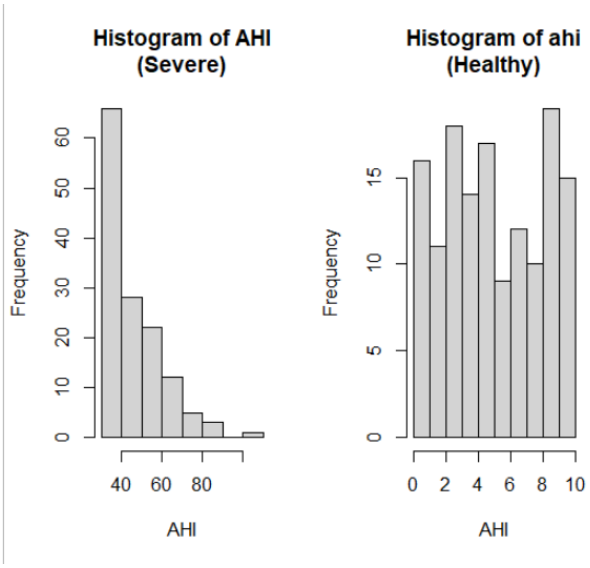
Through some analytic processing of the summary it is possible to pick out relevant parameters for given model.

For humans seeing two diagrams, tables side by side is in many cases very supportive in terms of comparison.

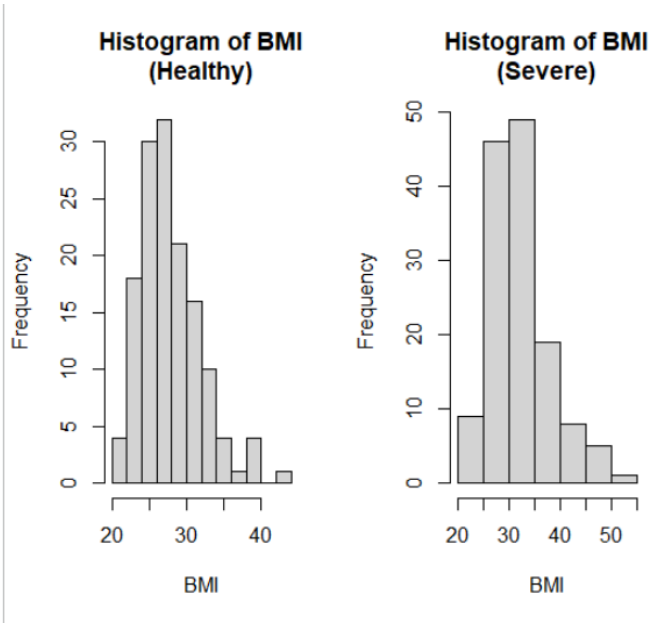
par(mfrow=c(1,2)) (Code to make it possible for two plots to be on the same row, an 1x2 array)

hist(subset(df_OSA_male, OSA=="Healthy")$IAH) (Left Histogram)

hist(subset(df_OSA_male, OSA=="Severe")$IAH) (Right Histogram)



```
hist(subset(df_OSA_male, OSA=="Healthy")$BMI, main = paste("Histogram of BMI
                                                           (Healthy)"), xlab= "BMI")
hist(subset(df_OSA_male, OSA=="Severe")$BMI, main = paste("Histogram of BMI (Severe)"), xlab=
"BMI")
```

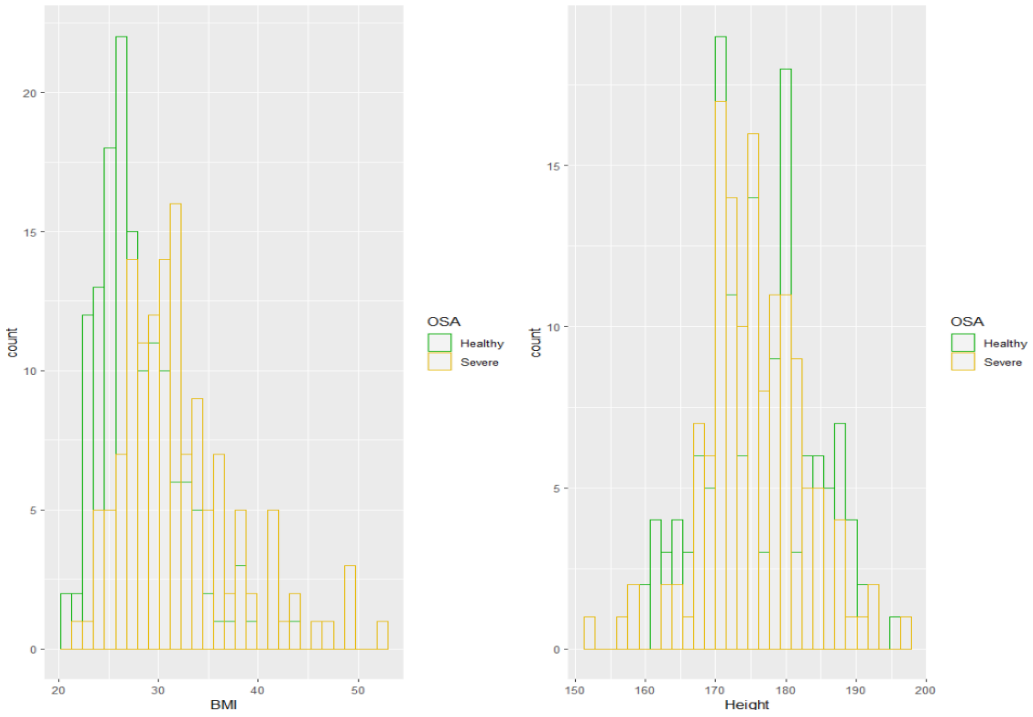


Another great visualization library is **ggplot2** and **gridextra.** The code below generates a great visual representation of two histogram which makes it easy to compare and find and understand certain relationships in a graphical way:

```
p1 <- ggplot(df_OSA_male, aes(x = BMI)) +
  geom_histogram(aes(color = OSA), fill = "white",
                 position = "identity", bins = 30, alpha = 0.1) +
```

```
  scale_color_manual(values = c("#00AF00", "#E7B800")) +
  scale_fill_manual(values = c("#00AF00", "#E7B800"))


p2 <- ggplot(df_OSA_male, aes(x = Height)) +
  geom_histogram(aes(color = OSA), fill = "white",
                 position = "identity", bins = 30, alpha = 0.1) +
  scale_color_manual(values = c("#00AF00", "#E7B800")) +
  scale_fill_manual(values = c("#00AF00", "#E7B800"))
library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```

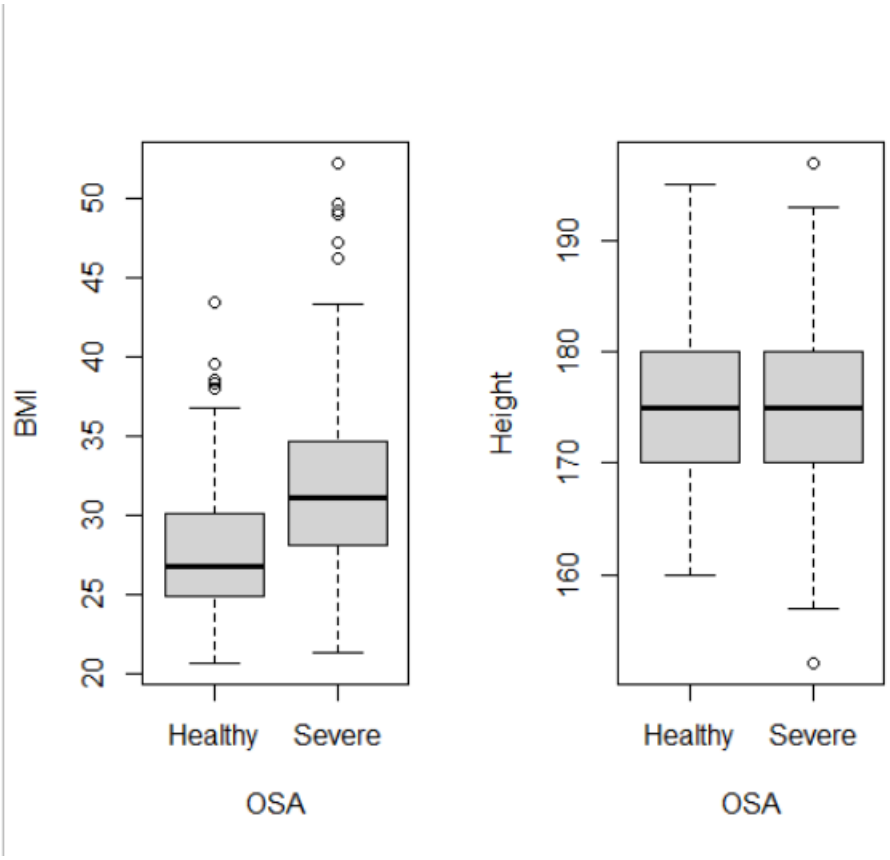


Boxplots have for many years been tool for understanding data. R provides easily generated boxplots which tent to be veryuseful. E.g. when it comes to observe relationship between features. The statistical data OSA based on BMI value and height value below is a useful tool for pull conclusions

on whatever the feature has anything to do with healthy or severe OSA.

boxplot(BMI ~ OSA)

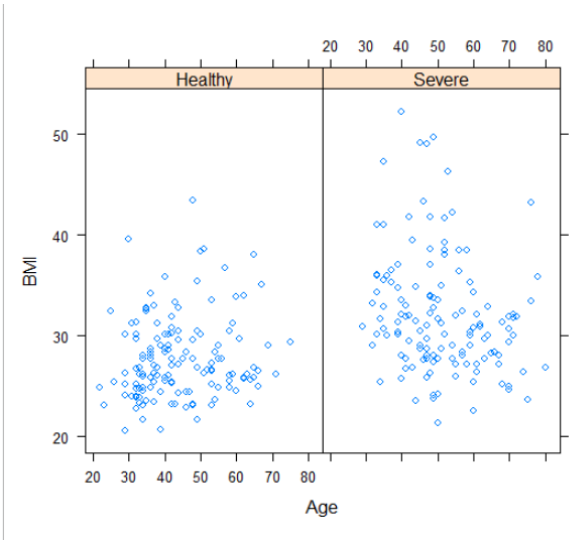
boxplot(Height ~ OSA)



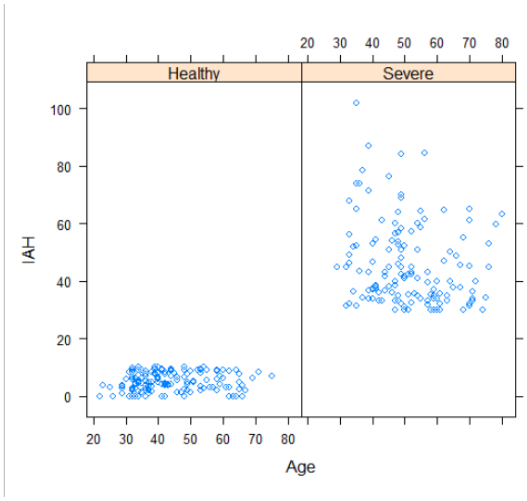
The **lattice** add-on package plot the association of two variables for two specific groups. Code snippet below give us two groups:

1) OSA depending on BMI and Age:

   xyplot(BMI ~ Age | OSA, data = df_OSA_male) :



2) OSA depending on AHI and age:



5. DEVELOPING MACHINE LEARNING MODELS

After performing exploratory data analysis on given data, the next step is to develop and compare different machine learning models to achieve our desired goal. Even though each subgoal of the project is supervised machine learning problems, both the regression and classification problem have two different end goals.

- **Regression goal:** To find the optimal model for predicting IAH value based on clinical data features.
- **Classification goal:** to find the optimal predictive classifier model based on IAH value the model may categorize into qualitative classes: Healthy if IAH value < 10, and Severe if IAH value >= 30.

Up until now R has been the only programming language utilized in the project, for representing useful tools and script. However, in the final section of the project *Python* programming language is going to be used for both training and testing different machine learning models.

The syntax for training and testing a specific model is similar even though the training process may differentiate of each other. The procedure syntax for picking a model python is presented below:

4.1 Model selection

In this part we will pick out a some of the most well-known, and commonly used machine learning algorithms. Primarily the open-source data analysis library **scikit-learn** is going to be utilized for the ease of develop and use of the models. It also includes packages like KFold and GridSearchCV will be used for find optimal hyperparameters for each model. In the end different Metrics

In **Scikit-learn /sklearn** following libraries and package include:

- **Numpy**
- **Pandas**
- **SciPy**
- **Matplotlib**
- **Ipython**
- **symPy**

4.1.1 Regression models

Now it's time to tackle on what basic syntax we may use for correctly implement procedure for training and testing and evaluation a complete regression model. Depending on what model being tested, different librarys and packages may be used. Initially we separate the predictors and the response variable with following line of code :

```
features = ['Weight', 'Height', 'Cervical', 'BMI', 'Age']
```

```
X = np.array(OSA_df[features])

y = np.array(OSA_df['IAH'])
```

Then we import all the models we wish to train and test from the scikit-learn library, except for the XGBoost regressor model which may be fetched by a unique library called xgboost :

```
from sklearn import linear_model

from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

4.1.2 Classification models

The first thing we do is to pick out the desired predictor columns and target variable.:

```
columns = df_OSA_male_extreme.columns.tolist()

# Filter the columns to remove ones we don't want.
columns = [c for c in columns if c not in ["Patient","Gender","IAH","OSA"]]

# Store the variable we'll be predicting on.
target = "OSA"
```

Next thing to do is import all needed classification models that was we wish to try out :

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
```

4.2 Model Tuning

Almost every model has its unique parameters and hyperparameters. A model parameter in machine learning and coding terms is a configuration variable that is learnt during machine learning process. The hyperparameter is however a parameter that is manually specified at the start. An example for an hyperparameter is the total number of trees within a decision tree model. Obviously tuning and assigning the optimal values are crucial for getting final optimal results.

4.2.1 Hyperparameter Tuning

One famous procedure to retrieve and tune the most optimal hyperparameters is a technique so called cross-validation. In simple terms the method is based around splitting up the data into K large batches and iteratively validate each batch on where the K-1 batches are training the model and one batch is used as validation set. times. One major flaw Cross Validation suffering from is since we train and test out with specific hyperparameters on all k subset, and therefore becomes biased.

This problem may be solved by implementing an external loop and one internal loop. The outer loop is splitting the data into K-subsets and will estimate the quality of the models trained on the inner layer. While the inner loop is being used for selection the optimal hyperparameters.

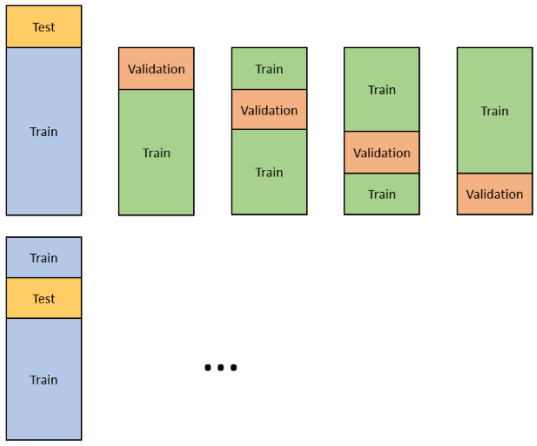


Figure 10 Nested cross-validation illustration

Main two head tools will be used for the model tuning part.

1) The **Kfold** model from **sklibrary** is randomly splitting k number of subsets of the main set for testing and training each subpart.

```
• from sklearn.model_selection import KFold
```

2) The **Grid search** method will be used as inner loop and hyperparameter selector. By

sending in a grid of parameters each and every one gets tested with eachother for picking optimal hyperparameters.

- `from sklearn.model_selection import GridSearchCV`

The nested cross-validation technique is implemented for tuning the optimal hyperparameter for the Support vector machine model with an hyperparameter searchgrid of:

```
SVMparam_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
```

with following syntax build whole nested cross-validation method :

```
# configure the cross-validation procedure
cv_outer = KFold(n_splits=10, shuffle=True, random_state=1)
# enumerate splits
outer_results = list()
for train_ix, test_ix in cv_outer.split(X):
  # split data
  X_train, X_test = X[train_ix, :], X[test_ix, :]
  y_train, y_test = y[train_ix], y[test_ix]
  # configure the cross-validation procedure
  cv_inner = KFold(n_splits=3, shuffle=True, random_state=1)
  # define the model
  model = SVC(random_state=1)
  # define search space
  #space = dict()
  #space['n_estimators'] = [100, 250, 500, 600]  #random forest para
  #space['max_features'] = [2, 3, 4]
  #tree_para = {'criterion':['gini','entropy'],'max_depth':[4,5,6,7,8,9
,10,11,12,15,20,30,40,50,70,90,120,150]} # decision tree para
  #SVM PARAM
  SVMparam_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
  # define search
  search = GridSearchCV(model, SVMparam_grid, scoring='accuracy', cv=cv
_inner,refit = True, verbose = 3)
  # execute search
  result = search.fit(X_train, y_train)
  # get the best performing model fit on the whole training set
  best_model = result.best_estimator_
  # evaluate model on the hold out dataset
```

```
  yhat = best_model.predict(X_test)
  # evaluate the model
  acc = accuracy_score(y_test, yhat)
  # store the result
  outer_results.append(acc)
  # report progress
  print('>acc=%.3f, est=%.3f, cfg=%s' % (acc, result.best_score_, resul
t.best_params_))
# summarize the estimated performance of the model
print('Accuracy: %.3f (%.3f)' % (mean(outer_results), std(outer_results
)))
```

where cv outer is the external loop and gridsearch cs is the internal one.

4.3 Model Evaluation

Various types of metrices and measurement for comparing the evaluation that have been computed.

- **For classification:**

In order to check the accuracy of each model following line is executed:

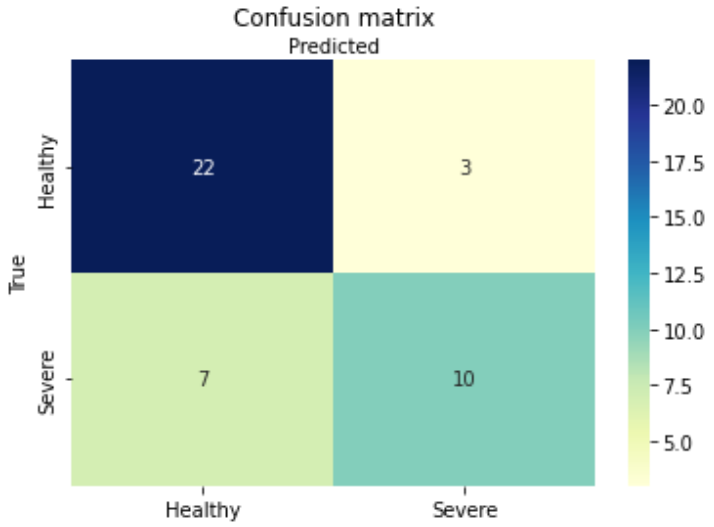
```
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

A great way to get a better understanding of each predicted resulst is by using creating and analyzing a confusion matrix :

```
import matplotlib.pyplot as plt

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
import seaborn as sns

labels = [0, 1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(labels))
plt.xticks(tick_marks, labels)
plt.yticks(tick_marks, labels)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g
')
ax.xaxis.set_label_position("top")
plt.title('Confusion matrix', y=1.1)
plt.ylabel('True')
plt.xlabel('Predicted')
ax.xaxis.set_ticklabels(['Healthy', 'Severe']); ax.yaxis.set_ticklabels
(['Healthy', 'Severe']);
```



An classification report of the confusion matric can be generated by using following python syntax:

```
import sklearn
print(sklearn.metrics.classification_report(y_test, y_pred))
```

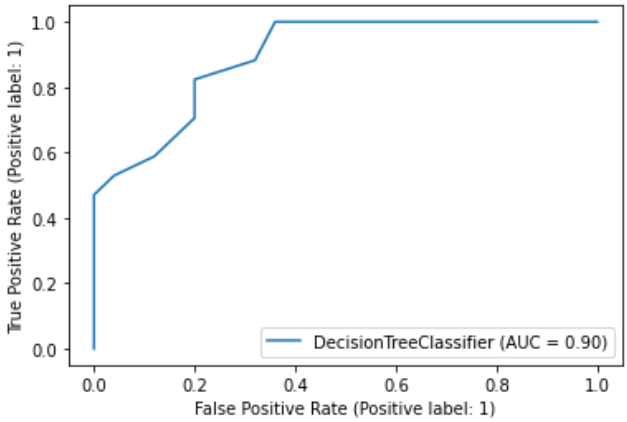
```
              precision    recall  f1-score   support

           0       0.76      0.88      0.81        25
           1       0.77      0.59      0.67        17

    accuracy                           0.76        42
   macro avg       0.76      0.73      0.74        42
weighted avg       0.76      0.76      0.75        42
```

Finally a famous and good graphical evlatuion metric is the ROC curve:

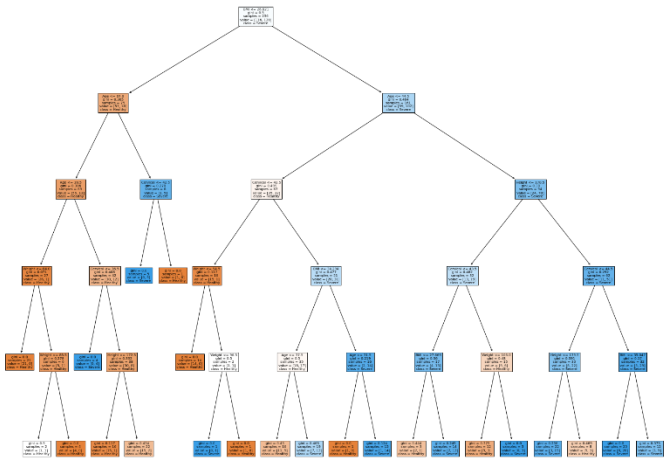
```
metrics.plot_roc_curve(model, X_test, y_test)
```



In order to get a good graphical representation of a generated decision code snipplet below gives us a plotted one:

```
import matplotlib.pyplot as plt
from sklearn import tree

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model,
                   feature_names = columns,
                   class_names = ['Healthy', 'Severe'],
                   filled=True)
```



- **For regression:**

All you gonna have to do to obtain metrics and measurement values for MSE,MAE,R-Squared and RMS is implement metrics from sckit-learn:

```
#Metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2
_score
```

And then use following code to retirve all the result values:

```
MSE=mean_squared_error(predictions, y_test)

RMSE=math.sqrt(mean_squared_error(predictions, y_test))

MAE=mean_absolute_error(predictions, y_test)

Rsquared = r2_score(y_test, predictions)

print('Mean squared Error: ', MSE,
      '\nMean Absolute Error: ', MAE,
```

```
      '\nR-squarred', Rsquared,
      '\nRMS', RMSE)
```

6. CONCLUSIONS

In this project, the head focus was to do some research with the purpose of getting a deeper understanding of the reasoning why people tend to develop the OSA breath condition. The disorder is considered as the most common sleep-related breathing disorder among humans. Hence it is a vital and hot topic around most doctors around the world. As a data scientist the doctors assigned us an object with the goal of developing a machine learning model which would ease the understanding out why people tend to develop the sickness, and therefore try to find the root of why development of the OSA disorder is happening. The Machine learning process goal is to predict patients AHI values and classify the degree of severity of OSA with help of clinical data collected by “Quiron Hospital of Malaga”

I’ve retrieved a great understand regarding the pipeline work within a data science project. It can be described as following: The absolut first thing is to get a good grip of the actual assignment. After that, the first step after maintaining the dataset is to do data cleaning of the provided data. By removing or adjusting irrelevant data with the intention of building a clean and useful database. The next segment is to understand and find patterns by performing exploratory data analysis. For finding relevant features that may be applied to our future machine learning model. After analyzing all the clinical data, the most significant variables that seems to have relations of having high AHI turned out to be the BMI, Cervical and to some extent the age of the patient.

During the whole project two of the best programming languages R and Python have been used for analyzing and managing big data. Both language has its unique tools and libraries used for distinct purposes.

The most challenging part was without any doubts the model tuning procedure. To fully understand all methods and models hyperparameters was also a very time consuming section.

Present a final summary on two main aspects:

Include your own critical comments on the potentials and difficulties when working with the different tools you have tried.

IT IS VERY IMPORTANT*: that you finish your Report with your personal conclusions related to what you have learned while working in this mid-term project*

AGAIN REMEMBER that together with the MLLB Report you MUST upload to Moodle ALL the code you have used.

- NOTE that you must be able to execute the code as **we will have an interview with you to check your experimental skills**.

7. REFERENCES

[1] TowardsDataScience, "Machine Learning general process," [Online]. Available: https://towardsdatascience.com/machine-learning-general-process-8f1b510bd8af. [Accessed 4 September 2019].

[2] MachineLearningBlog, "Developing Machine Learning models," [Online]. Available: https://machinelearningblogcom.files.wordpress.com/2017/11/1_kzmiuypmxgehhxx7slbp4w.jpeg?w=1400. [Accessed 4 September 2019].

[3] M. &. B. P. Staniak, "The Landscape of R Packages for Automated Exploratory Data Analysis," 2019. [Online]. Available: https://arxiv.org/pdf/1904.02101.pdf. [Accessed 4 9 2019].

APPENDIX A

You can include appendixes with any additional information on activities related to this subject. For example:

- Your activities after enrolling into an on-line course, for example on DL using PyTorch, (please give certificates and materials, as code, etc., to support the skills you have learned in every activity)