Predictive and Descriptive Learning

Half-term Report

Sergio Pérez Morillo

5/11/2019

Table of Contents

1. Problem Description

The goal of this project is to utilize machine learning techniques to enhance the detection of obstructive sleep apnea (OSA). The first part of this kind of project is to learn about the problem and to gain specific knowledge. This improves both the machine learning modeling process and team communication with members from other fields.

The second part is to define a clear and concise machine learning methodology that will be implemented afterward. Selecting an already well-established methodology from a renowned source is considered good practice.

1.1. Framing the Problem

According to HealthLine and medically reviewed by the University of Illinois-Chicago [1], obstructive sleep apnea is a condition in which breathing stops involuntarily for brief periods of time during sleep. During normal sleep, air goes from mouth and nose to lungs. Intervals in which this flow abruptly stops are called apnea. In OSA cases, this occurs at night due to throat narrowing.

OSA symptoms include snoring, headaches, discontent, forgetfulness and somnolence. The segments of the population more vulnerable to OSA are older people and overweight people. Other than those, man collar sizes of more than 43cm and woman collar sizes of more than 40cm are also an important OSA risk factor.

OSA diagnosis includes polysomnograms that require patients staying overnight to monitor their sleep, during this time the medical staff might perform an electroencephalogram (EEG), an electrooculogram (EOM), an electromyogram (EMG), an electrocardiogram (ECG), a pulse oximetry test and an arterial blood gas analysis (ABG).

Some of the current OSA treatments are weight loss, Nasal decongestants, Continuous Positive Airway Pressure (CPAP), Bilevel Positive Airway Pressure (BPAP), positional therapy and, in the worst scenarios, surgery.

1.2. Machine Learning Methodology

I have based my methodology in one from a book I read last summer, *"Hands-On Machine Learning with Scikit-Learn & Tensorflow"* by Aurélien Géron [2]. It is a well-known book in the data science field that provides an excellent hands-on experience for developing machine learning and deep learning models to both newcomers and experts.

In this book, the author describes a comprehensive end-to-end methodology for machine learning business solutions. I decided to tweak it to fit this project's goals (the original methodology can be found in the MLLB report). The methodology of my project consists of six steps that coincide with the first six sections of this report.

1. **Framing the problem**: As mentioned in the opening paragraph, the first part of any machine learning project is to study the specific knowledge of the field. Depending on the problem complexity, it might take from a few days to even several weeks. This

newly attained knowledge contributes to developing intuition about the data that ultimately helps to achieve better results and speed up the whole process.

2. **Wrangling the data**: The next step is to collect the data that might be useful for the model considering the previous study. This time, the professors have already given the dataset to the students. After analyzing and describing the dataset features thoroughly, it must be cleaned to make it handier for the upcoming steps and to remove useless information beforehand.
3. **Exploratory Data Analysis:** Once the dataset is ready, a detailed analysis of the dataset is performed to gain new insights about its data. This step relies heavily on visualizations of distributions, correlations, and statistics. Visualizing the data helps to understand them at a deeper level. A possible outcome of this step is the filtering of irrelevant features and new ideas for augmenting the number of features.
4. **Preparing the data:** Now, the dataset is transformed to enhance the upcoming modeling process or just to meet certain models' requirements. This step includes data transformations such as *log(X+1)* and X^2, feature scaling methods such as *standard scaling* or *maximum-minimum scaling*, feature selection methods such as *filtering*, *wrapping* and *embedding*, and dimensionality reduction methods such as *principal component analysis* (PCA) and *manifold*.
5. **Model testing and fine-tuning**: The first part is to list several models to test them on the dataset. I have based mine on the ones presented in the book [2]. Once the list is ready, the process must be automated as much as possible taking into account aspects such as pipelining with data transformations, cross-validation to avoid overfitting and hence to produce more robust models, and hyperparameter tuning using grid search or random search to further improve the results. Bear in mind that the problem might be addressed as a classification model or a regression model. In this project, I developed models for both approaches.
6. **Results and model comparison**: Once the results are obtained, an analysis comparing the different models is performed. It is important to communicate the results to superiors or clients in an easy yet powerful manner. To do so, I have based my analysis in appealing and colorful visualizations to illustrate the results.

2. Data Wrangling

In this section, I cleanse the dataset provided by the teachers. Before doing so, I performed a data analysis in which I describe the different features found in the dataset and their usefulness at first glance.

2.1. Data Description

The features from the dataset are divided into categorical and numerical. The former includes the following ones:

1. ***Patient***: An ID for patient identification.
2. ***Comentarios***: Some comments related to patients and dataset instance and mostly referring to another index. It is mostly useless information and hence it was removed.
3. ***Audio tumbado***: Feature without any meaningful information, it just records whether there is an audio of the patient or not. It was removed.
4. ***Fotos***: Feature with a single value, thus useless information. It was removed.
5. ***Audio***: Same as 4. It was removed.
6. ***Gender***: Gender of the patient. It is really useful as seen in the problem description.
7. ***Fumador***: it records whether the patient does (or did) smoke tobacco. Useful at first glance.
8. ***Roncador***: it records whether the patient snores during sleep. Useful as seen in the problem description.
9. ***Enfermedades***: Disease registry of the patients. Useful as seen in the problem description.
10. ***Sala/Ruido***: Information about the room where the patient was filmed. It does not have much meaning nor relation with symptoms or patient attributes. It was removed.
11. ***Imagen***: It is filled almost with a unique value that does not provide useful info. It was removed.
12. ***Dialecto***: It records the patient's Spanish accent. As studied in the problem description, regional differences are no relevant to OSA diagnosis. It was removed.

On the other hand, the dataset includes the following numerical features:

1. ***IAH***: It is the target feature. It records the apnoea hypopnoea index (AHI).
2. ***Peso***: It is the patient's weight. It is really useful as seen in the problem description.
3. ***Talla***: It is the patient's height. It may be useful at first glance.
4. ***Edad***: It is the patient's age. It is really useful as seen in the problem description.
5. ***PerCervical***: It is the patient's collar size. It is really useful as seen in the problem description.

The rest of the dataset features: *EPWORTH*, *IAH Supino*, *IAH Lateral*, *IMC*, *DIST EXT OJOS*, *DIS BARB-LOB*, *Cansancio*, *Concentrarse*, *PerdRespNoche*, *HiperT*, and *EstHOSP* were removed due to their high density of *NaNs*. All of them had at least 50% of *NaNs*. To illustrate them, figure 1 shows the distributions of NaNs across the dataset.

The useful dataset features after the data description are *Patient, Gender, Fumador, Roncador, Enfermedades, IAH, Peso, Talla, Edad.*

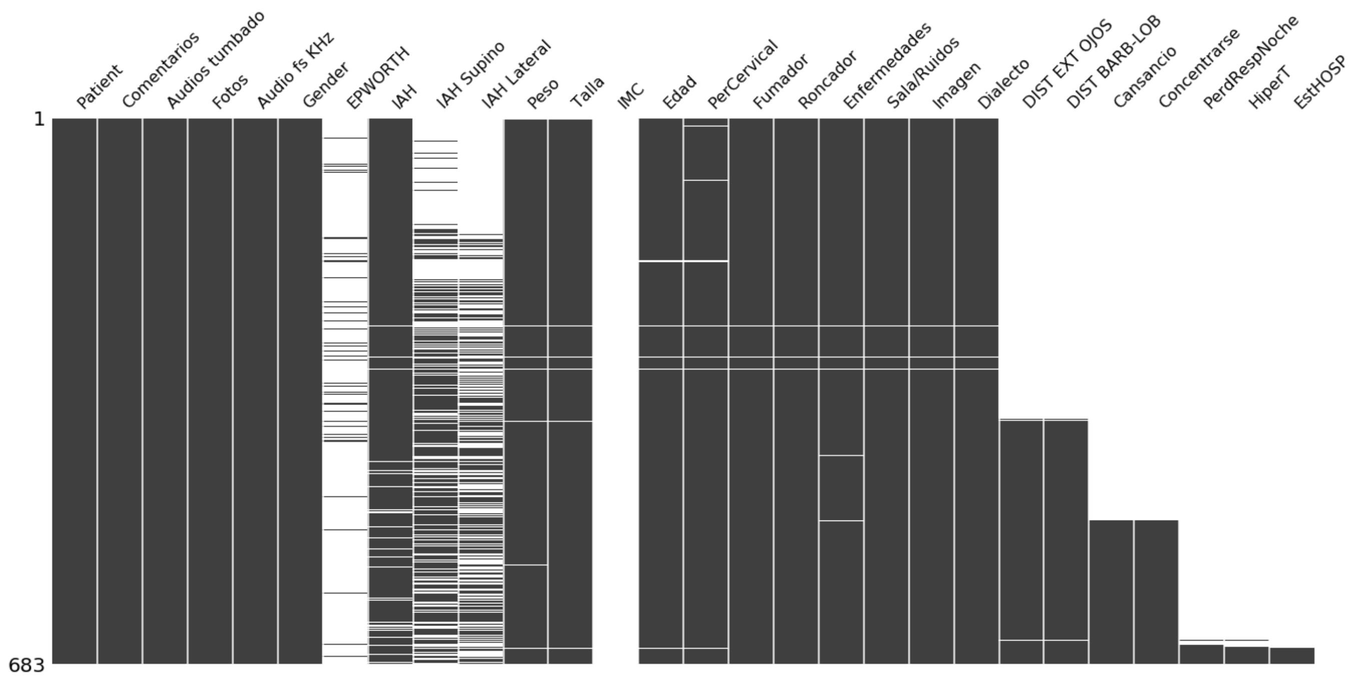


Figure 1: *NaNs* found in the dataset (whitespaces).

2.2. Data Cleansing

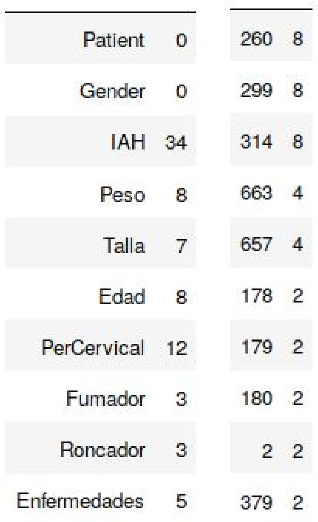
Even though most of the NaNs were already remove at this point, the useful features still had some of them. To decide whether to remove them or keep and assign them a value, I first looked at where they are located, both column-wise and row-wise.

Figure 2 shows these two distributions. My first action was to remove the *NaNs* from the target feature *IAH*. It was obvious as filling the target feature with estimations would cause problems predicting them later on. The ground truth must remain untouched.

Then, I removed rows with more than one *NaN* values since trying to guess several features of a column make their original information less useful. There were only twelve rows in this condition and even some of them had also *NaN* in the target feature, hence it was not a major loss.

After this filtering process, there were just seven *NaNs* in the dataset: one in *Peso,* four in *PerCervical* and two in *Enfermedades*. I decided to remove the numerical ones to avoid estimating them that might lead to noisy instances. For the categorical feature, I replace the *NaN* values with one of the categories 'ns' which means that a patient does not know.

My next step was to check possible outliers in numerical features. Figure 3 shows some simple statistics from the dataset such as mean, standard deviation, quartiles, minimum and maximum values for numerical features. From this information, I did not find any remarkable outliers but this is not a comprehensive method. The EDA from section 3 help to analyze the outliers better.



Patient	0
Gender	0
IAH	34
Peso	8
Talla	7
Edad	8
PerCervical	12
Fumador	3
Roncador	3
Enfermedades	5

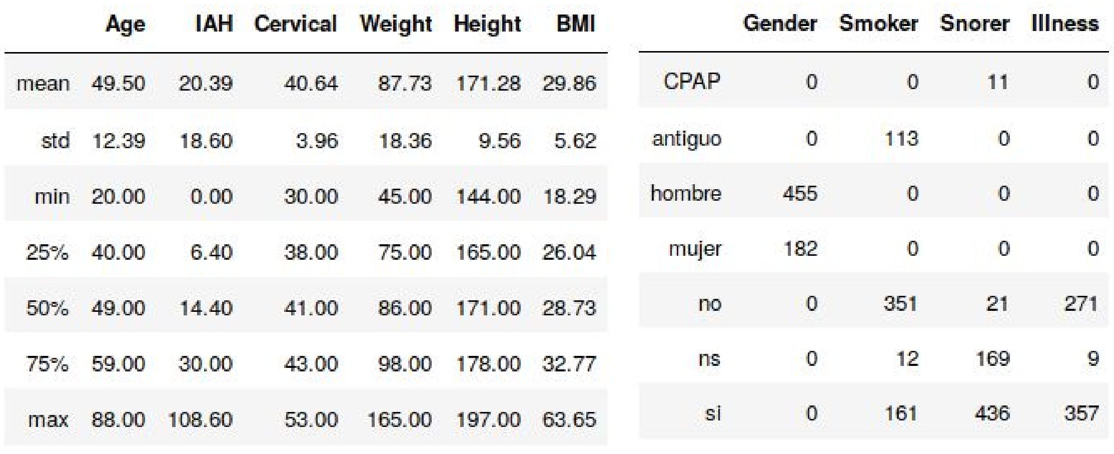
260	8
299	8
314	8
663	4
657	4
178	2
179	2
180	2
2	2
379	2

Figure 2: Remaining *NaNs* in is useful features by column and row.

The final step was to check the categories within the categorical features. First, *Gender* had only two values so no further processing was necessary. Second, *Smoker* had six categories but two of them were almost nonexistent and their meaning overlapped with another one, hence I merge the three as just one category.

Third, *Snorer* has eight categories but again four of them were almost nonexistent with overlapping meanings. After merging them, there were only four classes. And fourth, *Illness* has 249 categories. To reduce this enormous number, I followed a naive approach by aggregating all diseases as just one feature called ‘si’ or yes. In future work, better analysis must be performed. Thus, the number of categories was narrowed down to just three. Figure 3 shows the final distribution of these categories for each feature.

I performed also feature engineering by creating a new feature based on others. I computed the body max index (BMI) of each patient using the weight and height features. The final dataset has a total number of 637 instances and 11 features: an index, six numerical and four categorical.



	Age	IAH	Cervical	Weight	Height	BMI
mean	49.50	20.39	40.64	87.73	171.28	29.86
std	12.39	18.60	3.96	18.36	9.56	5.62
min	20.00	0.00	30.00	45.00	144.00	18.29
25%	40.00	6.40	38.00	75.00	165.00	26.04
50%	49.00	14.40	41.00	86.00	171.00	28.73
75%	59.00	30.00	43.00	98.00	178.00	32.77
max	88.00	108.60	53.00	165.00	197.00	63.65

	Gender	Smoker	Snorer	Illness
CPAP	0	0	11	0
antiguo	0	113	0	0
hombre	455	0	0	0
mujer	182	0	0	0
no	0	351	21	271
ns	0	12	169	9
si	0	161	436	357

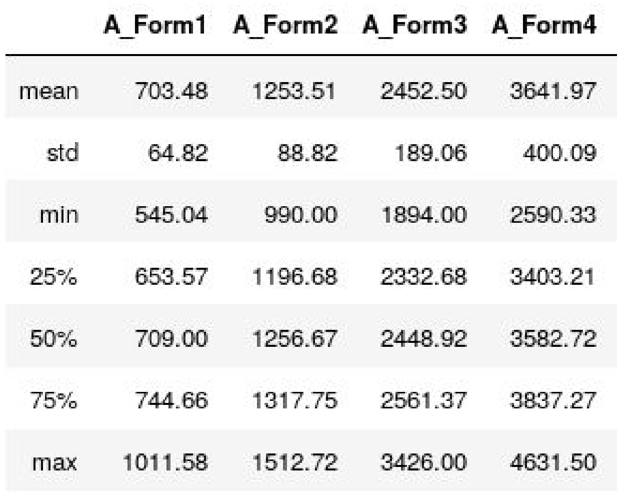
Figure 3: Simple statistics of the final dataset.

2.3. Classification Dataset

So far, this dataset is the same for both classification and regression modeling. Although for the former, a categorical feature is necessary to work properly. Thus, I added an extra feature based on the regression target feature *IAH* in which patients with less than a value of 10 are healthy and patients with more than a value of 30 are severe. Instances that did not meet these conditions were filtered. The feature was named *OSA.* Consequently, the regression target feature was removed from the classification dataset.

In addition to this new feature, another forty features were added by merging this dataset to another one. This new dataset includes voice records of male patients. These features are numerical and not much information can be inferred from them. Figure 4 shows some simple statistics of a feature sample.

I decided to perform an inner merge to avoid filling the resulting dataset with *NaNs*. The resulting classification dataset has 174 instances and 51 features: an index, forty-five numerical and five categorical.



	A_Form1	A_Form2	A_Form3	A_Form4
mean	703.48	1253.51	2452.50	3641.97
std	64.82	88.82	189.06	400.09
min	545.04	990.00	1894.00	2590.33
25%	653.57	1196.68	2332.68	3403.21
50%	709.00	1256.67	2448.92	3582.72
75%	744.66	1317.75	2561.37	3837.27
max	1011.58	1512.72	3426.00	4631.50

Figure 4: Simple statistics from a sample of voice record features.

3. Exploratory Data Analysis

In this section, I draw insights from an exploratory data analysis (EDA). It is heavily focused on visualizations as I considered them the best way of highlighting the relevant information. First I plotted the categorical features, then the numerical ones and finally I combined them.

3.1. Categorical Features

Figure 5 shows the distribution of categories for each categorical feature. In all of them, there is a clear dominant class so these features are imbalanced. However, they also have other significant options that represent at least roughly one-third of the feature.

Some options were almost empty and might have been removed. Nevertheless, they were kept to avoid filtering more instances from a dataset with already very little information.

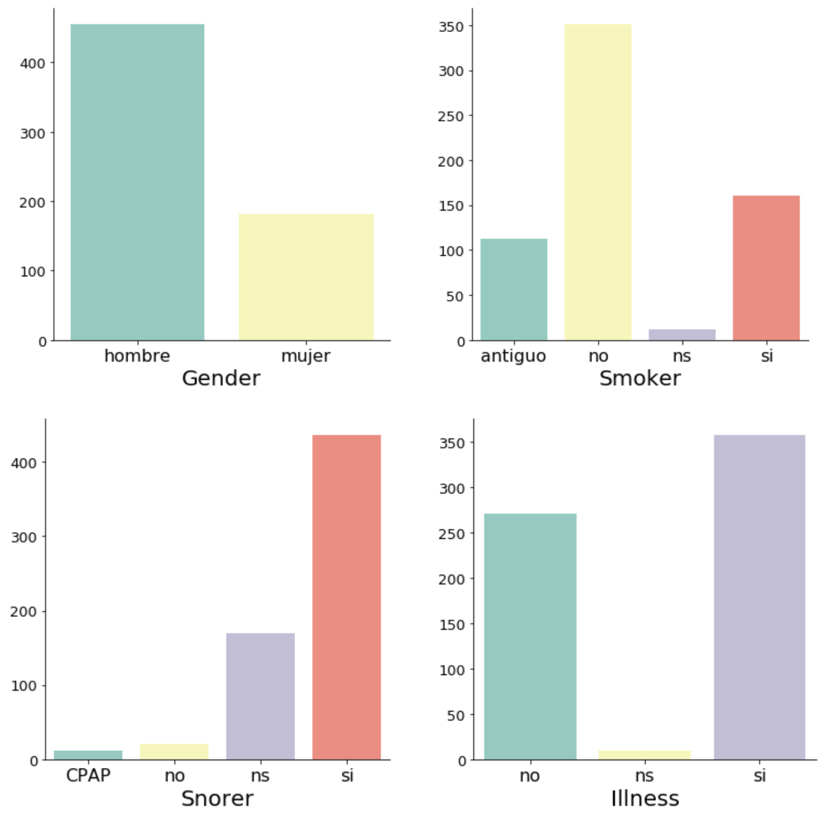


Figure 5: Categorical plots of the dataset.

3.2. Numerical Features

Regarding numerical features, my first approach was to plot scatters, correlations (via heatmap) and distributions (via histogram). Figure 6 shows the resulting plots. Looking first at the left figure, most of the feature histograms depict a normal distribution without many outliers. The only one that is different is the target feature *IAH* which has smaller values than larger values. This might cause an overfitting problem in the regression model as it would prefer predicting smaller values rather than larger values.

As for correlations, there are some strongly correlated features with mixed results regarding their relation with OSA specific knowledge. On the one hand, *Cervical*, *Weight*, and *IAH* are strongly correlated as the study on OSA denoted in section 1. On the other hand, *Age* and *IAH* do not show much correlation, thus contradicting the OSA study whose results showed a correlation between these two.

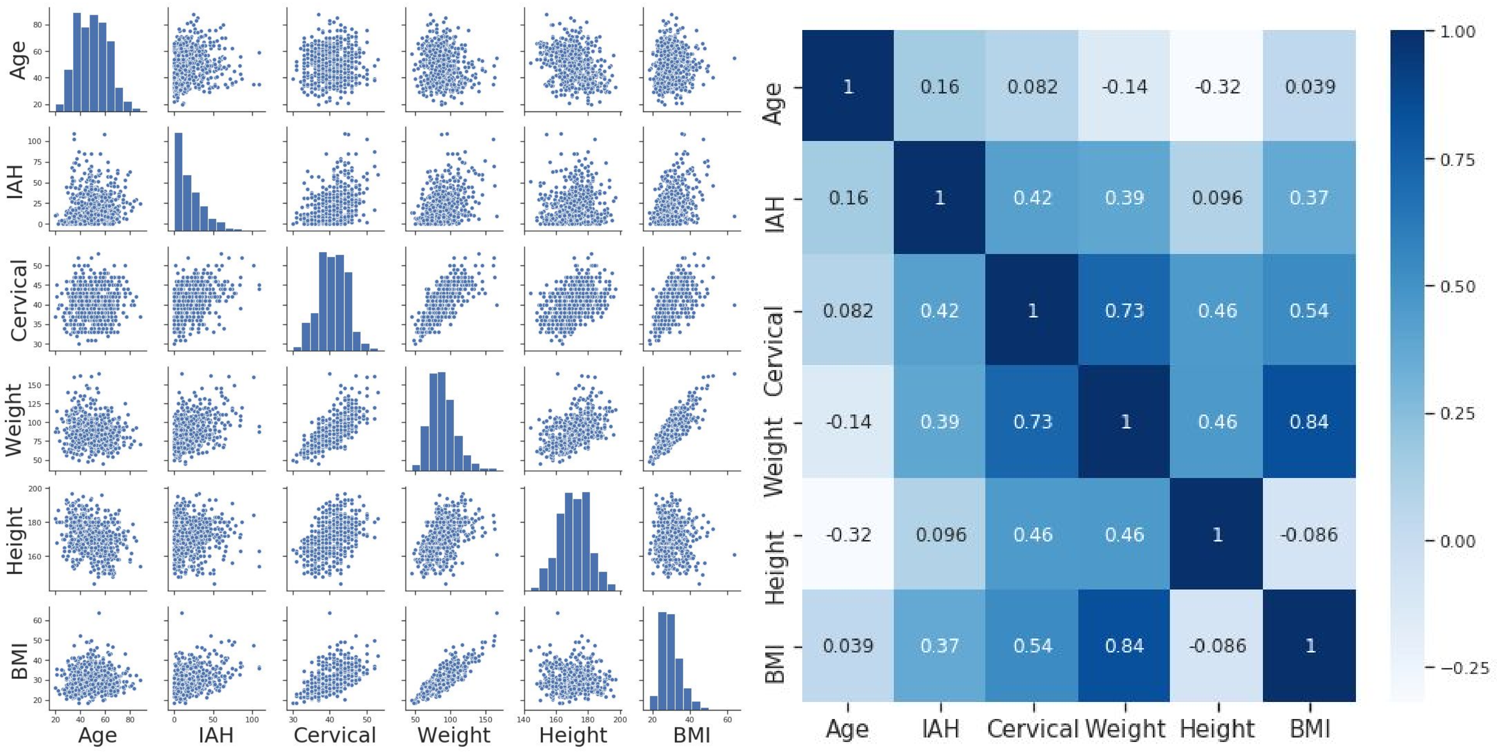


Figure 6: Pair plots and heatmap of numerical features.

3.3. Combining Features

After analyzing both types of features separately, I combined them to gain more insights. Figure 7 shows box plots of the target feature *IAH* for each categorical feature. Here, it shows that men are more prone to OSA problems than women, as it was expected.

Besides that, patient smoking status does not show much correlation as all categories share similar values. The outliers even show contradictory results since they indicate worse OSA conditions for non-smokers and former smokers.

Regarding snorers, it shows a clear correlation between non-snorers and small *IAH* values. This is really useful namely for classification modeling as the values of this category are really concentrated in the OSA class *healthy*.

Finally, patients with any type of disease show smaller *IAH* values than patients with none, which seems to contradict the OSA study. One reason might be the naive aggregation technique that I used to reduce the number of categories. This shows that the type of disease is important to determine the severity of OSA cases. On the bright side, these categories differ from each other more than in other features like *Smoker* and *Snorer*. Thus, further analysis was carried out combining the two feature with most variability.

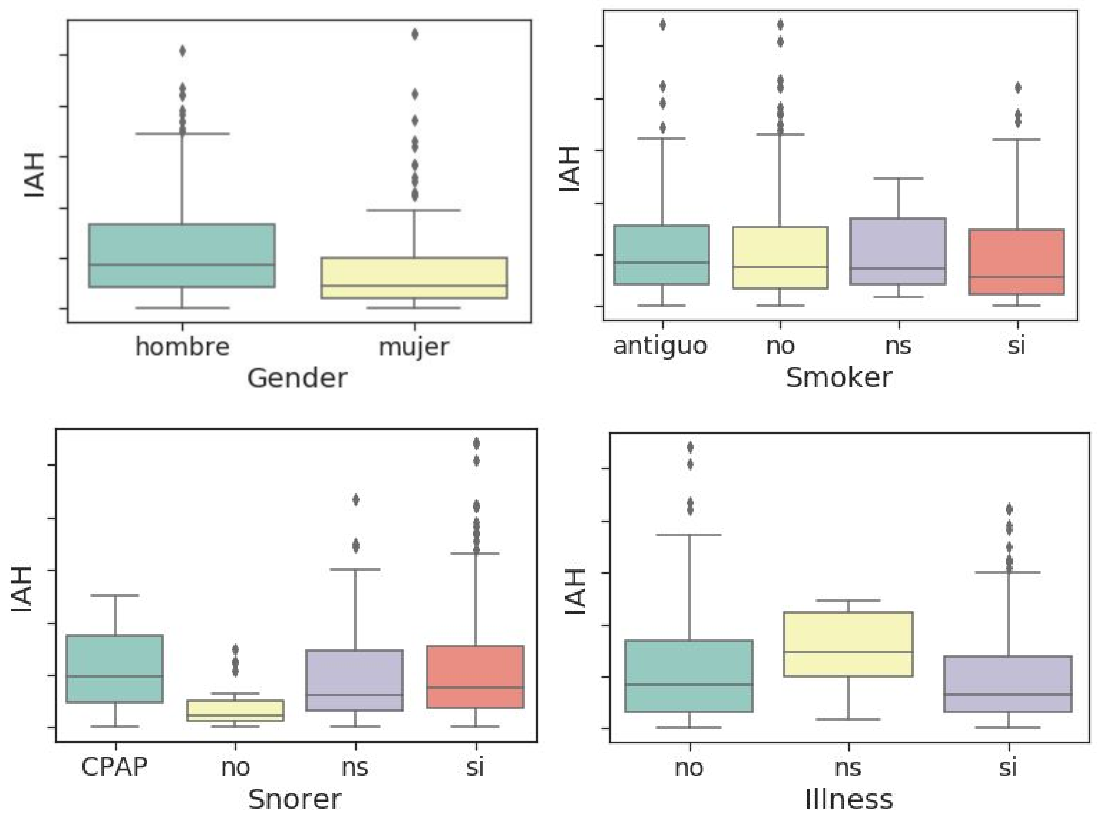


Figure 7: Box plots of target feature for each categorical feature.

For combining the most variable categorical features (*Illness* and *Gender*) with each numerical feature, I utilized violin plots. Figure 8 shows the plots for the most relevant features besides the target feature.

Regarding the patient's age, it shows that both men and women are equally distributed by this feature. Most patients are middle-aged. The patient's cervical perimeter shows more concentrated values than the previous one. It makes sense considering that this feature has a smaller range of values. It also shows that women have a smaller perimeter and it is oddly smaller when the patient has a disease.

As for the patient's weight, it shows that men are heavier, yet it does not show that they are particularly overweight. Thus, this dataset does not feature many overweight patients. More data from patients with this condition would have been helpful as the OSA study shows a strong relation for this case. The patient's height shows that men are taller than women, as it was expected. Other than that, it does not show any relevant information.

Finally, I added categorical features as hue in the pair plots and distribution plots to verify my assumptions about the data and better grasp them. Figure 9 shows two of these plots, for the *Gender* feature and for the *Smoker* feature. Comparing those two, it is really clear that for *Gender* which has more variability there are more evident data clusters than for *Smoker* whose data seem to overlap in all scatter and distribution plots.

Here, I have presented the most relevant plots of the EDA. To see the full implementation, please visit my Github account **[3]**.

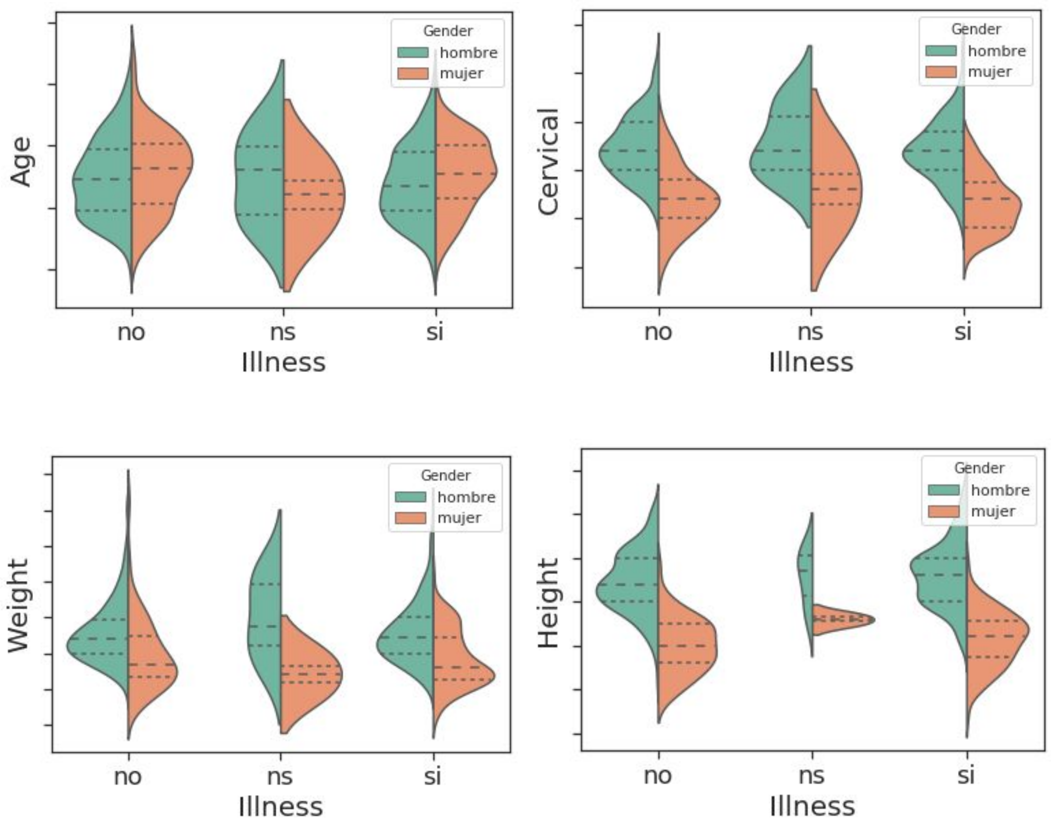


Figure 8: Violin plots of numerical features by *Gender* and *Smoker*.

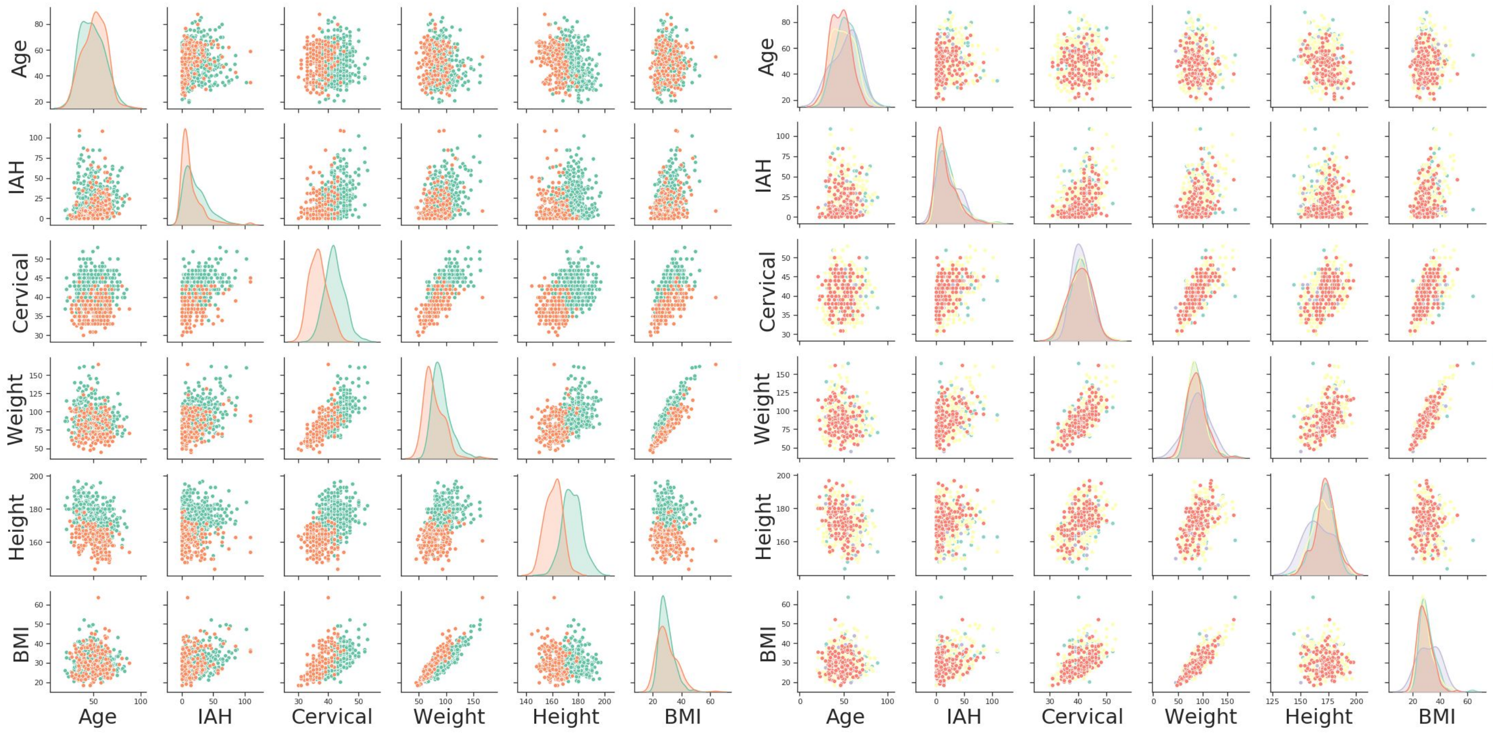


Figure 9: Pair plots by *Gender* (left) and *Smoker* (right).
Gender plot: men (Green) and women (Orange).
Smoker plot: former (Green), no (yellow), yes (red), and DK/NA (purple).

4. Data Preparation

In this section, I present a wide range of techniques that I used to prepare the dataset before fitting it with a model in order to enhance the final results. This section includes techniques such as data transformation, feature scaling, dimensionality reduction, and feature selection.

For the latter, I did also a complete study of the available techniques before implementing some of them in the final classification modeling. I decided to carry out this analysis because at my research lab I had not tried any of these selection techniques before in my deep learning models.

4.1. Data Transformation

I used two data transformations in this project, *log(x+1)* and *polynomial features*. The first one is a well-known transformation in data science that helps to alleviate overfitting problems due to features with wide value ranges. I have always used the *x+1* version at my research lab since it avoids problems computing the logarithm of zero values.

The second one works like kernels in support vector machines. It transforms the linear features into polynomial features to transform models like linear regressions into polynomial regressions. There is a trade-off with its parameter, polynomial degree. Usually, higher degrees might fit better the model but it also might produce overfitting and high variance. On the other hand, lower degrees do not overfit the data but it still might underfit them producing high bias. I decided to implement this transformation because it was suggested in the book **[1]**, as an alternative in linear regression models.

Other transformations were considered in this project such as the natural logarithm *ln(x)* and the square function X^2. They were eventually discarded to not increase the number of modeling variables that would ramp up the total number of combination exponentially.

4.2. Feature Scaling

Feature scaling is in most cases beneficial to the modeling process. It improves the numerical stability of a model and also reduces the training time. Models such as knn, k-means, logistic regression, SVM, perceptron, neural networks. On the other hand, distance-based methods like clustering might sometimes not benefit them as these methods assume that all features contribute equally to the model.

I implemented two feature scaling methods in this project, *minimum-maximum scaling* and *standard scaling*. I decided to select them since they are well-known in this field and usually are the two that I used at my research lab.

Minimum-maximum scaling scales data to a predefined range. In this case, I set this range to [0,1] which is the most typical one. This method limits the standard deviation so outliers do not have large values. However, just one outlier might lead to poor scaling. The formula is as follows:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Standard Scaling scales data by making values in each feature have zero-mean and unit variance. The standard deviation is less constraint than in the previous method. Moreover, it does not suffer from one isolated outlier in the data. Standardization is usually preferred over normalization. Its formula is as follows:

$$z = \frac{x - \mu}{\sigma}$$

Other techniques were considered such as scaling to unit length or mean normalization but they were discarded. Future works should also consider performing an exploratory data analysis with several visualizations of the standardized dataset.

4.3. Dimensionality Reduction

Dimensionality reduction is an unsupervised field that aims to solve the so-called *curse of dimensionality.* Datasets might contain hundreds or even thousands of features. Increasing the number of features leads to lower processing speeds exponentially. To solve this, these techniques reduce the number of features while trying to keep as much information as possible. Aside from that, they are also used in visualizations to detect patterns such as clusters in high dimensional data.

I have implemented two dimensionality reduction techniques, *principal component analysis* (PCA) and *t-distributed stochastic neighbor embedding* (t-SNE). The former was used in the modeling process for reducing the number of features and for visualizing the dataset. The latter was used just for visualizing the dataset.

PCA is the most popular technique in this field. It identifies a hyperplane that preserves as much variance as possible and then projects the dataset onto it. To define a PCA model, the number of components (or dimensions) must be set. Instead of arbitrary choosing this number, best practices set the least amount of required cumulative variance to choose the components. Figure 10 shows the evolution of the cumulative variance as a function of the number of components. From this graph, I decided to set the required cumulative variance to 80% in the modeling of section 5. Bear in mind that I always standardize data first for PCA.

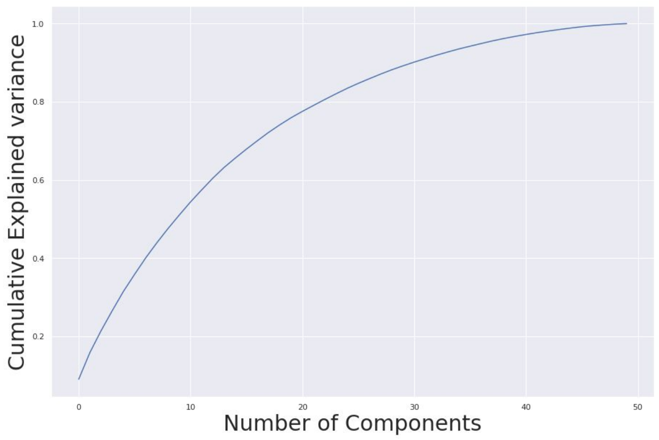


Figure 10: Evolution of the cumulative variance by PCA components.

Regarding visualization, figure 11 shows the dataset projection for a PCA of two components and three components and colored using the classification target feature *OSA*. It is clear that there is just one cluster in which data points cannot be clearly separated by *OSA* categories.

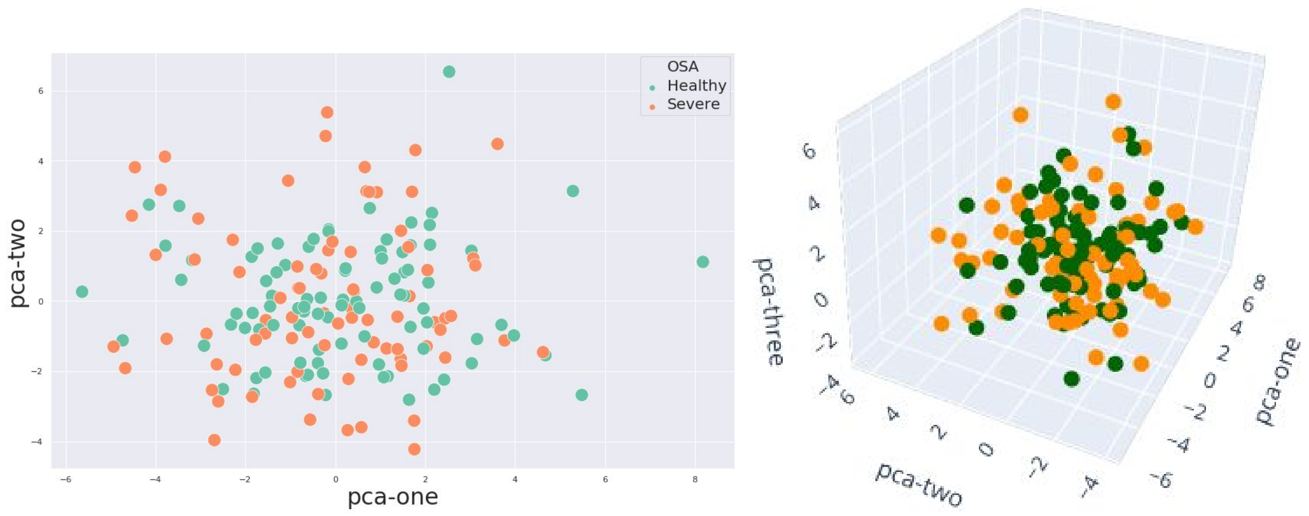


Figure 11: PCA 2D and 3D dataset projections

As I mentioned, I implemented t-SNE to try improving the visualizations. This method is part of the manifold sub-branch. It tries to keep similar instances close and not similar ones apart. This method is not usually utilized as an input of the model because it relies on inferences. Figure 12 shows the dataset projection for a two-components t-SNE and three-components t-SNE. It has concentrated even more the cluster data points so it was useful as it was expected. This technique might work better with larger datasets.

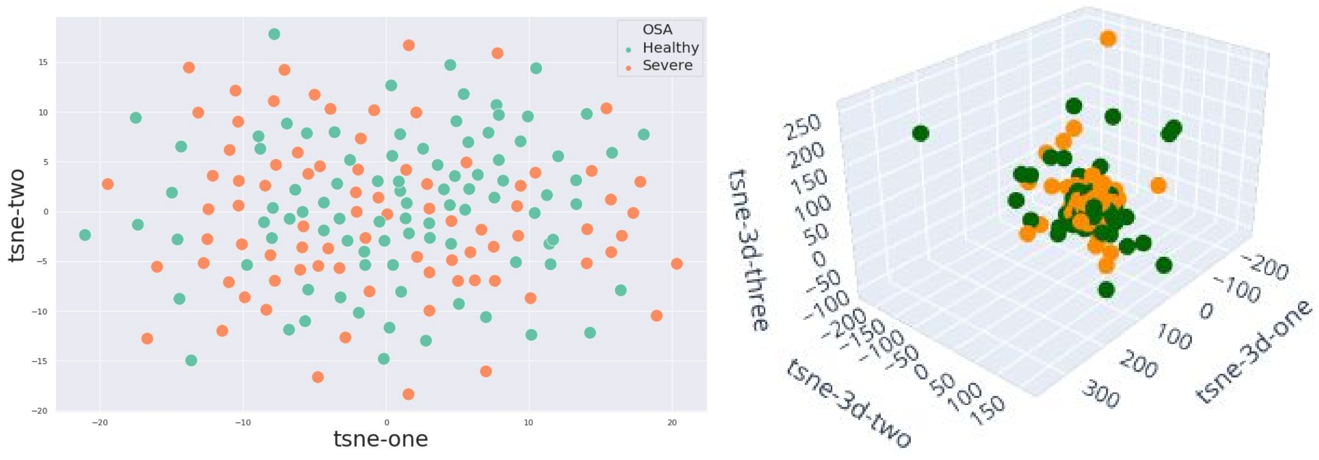


Figure 12: t-SNE 2D and 3D dataset projections

4.4. Feature Selection

As stated in the opening paragraph of this section, I performed an analysis of several feature selection techniques, even though not all of them were introduced in the modeling process. Note that, this analysis utilized the classification dataset as it has more than 50 features.

4.4.1 Filtering Techniques

The first filtering technique is the Pearson correlation using heatmaps. Its goal is to filter strongly correlated features. This time, I used the classification dataset. Figure 12 shows just a sample of the columns because it was complicated to present the whole dataset properly in this report. This sample shows that almost all of them do not share any correlation. So this method might not be the best one for this dataset.

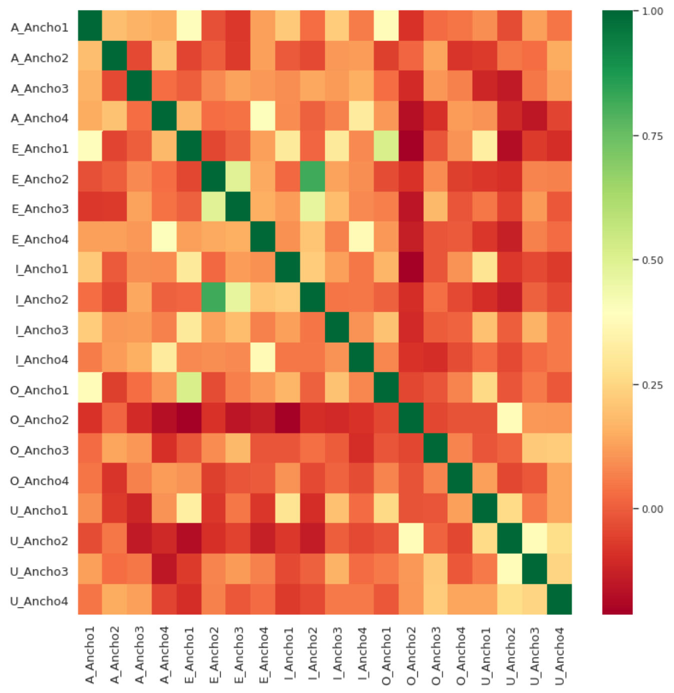
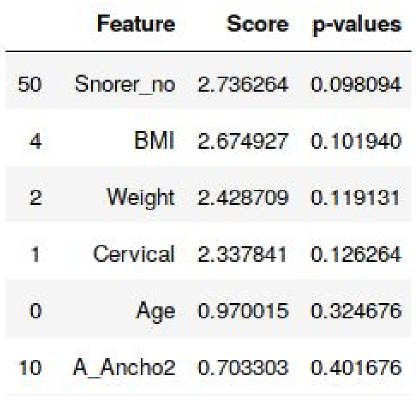


Figure 12: Heatmap from a column sample of the classification dataset.

The second filtering technique is univariate selection. It examines individually all the features to compare each of them with the target feature. Figure 13 shows the results of applying this technique to the standardized dataset with the score function chi2. As anticipated in the EDA, *non-snorer*, *Weight*, and *Cervical* are three major factors to predict the *OSA* feature.



	Feature	Score	p-values
50	Snorer_no	2.736264	0.098094
4	BMI	2.674927	0.101940
2	Weight	2.428709	0.119131
1	Cervical	2.337841	0.126264
0	Age	0.970015	0.324676
10	A_Ancho2	0.703303	0.401676

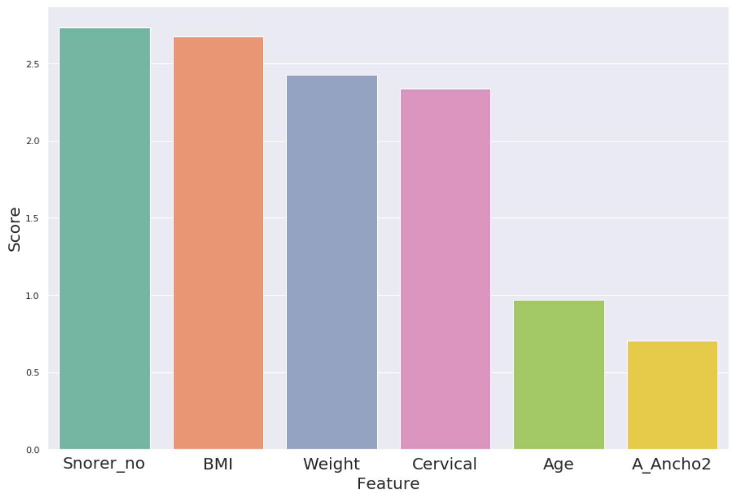


Figure 13: Univariate selection model results.

4.4.2 Wrapping Techniques

The selected wrapping technique is recursive feature elimination (RFE) and its variant with cross-validation (RFE-CV). Its goal is to recursively remove the weakest features until it gets to a specific number. By using the variant RFE-CV, the score of selecting from one to the maximum number of features is computed along with the name of the features used. I implemented this technique in the classification modeling process as it gives both a number and a list of features. These two parameters improve the interpretability of both datasets and models as shown in section 6, the results. Figure 14 shows an example of this technique.

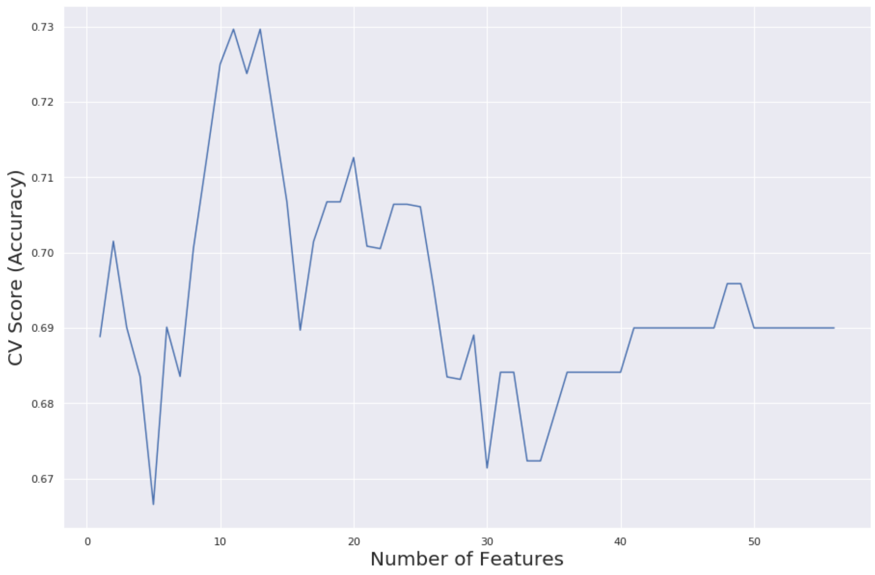
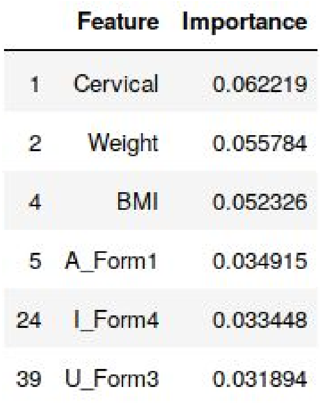


Figure 14: RFE-CV model result using a logistic regression model.

4.4.3 Embedded Techniques

The first embedded technique is feature importance. It uses built-in parameters from some models that weight the importance of each feature such as trees and ensemble of trees. Figure 15 shows an example in which again the features *Cervical* and *Weight* are important.



	Feature	Importance
1	Cervical	0.062219
2	Weight	0.055784
4	BMI	0.052326
5	A_Form1	0.034915
24	I_Form4	0.033448
39	U_Form3	0.031894

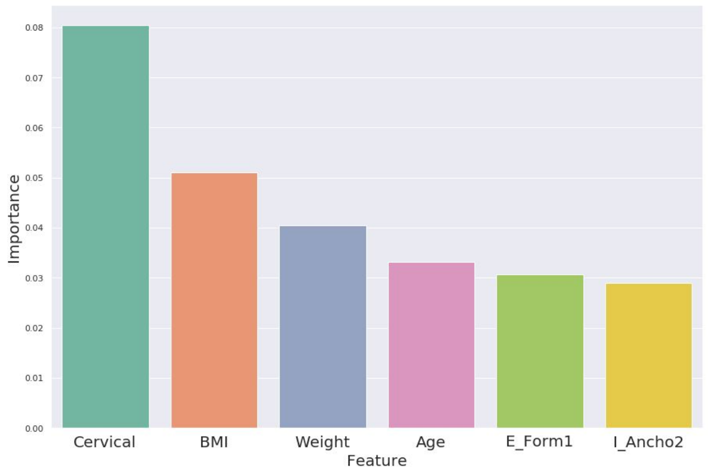


Figure 15: Feature importance model results using an extra tree classifier model.

The second embedded technique is ‘select from model’. As in the previous model, it uses built-in parameters from models. Although this time, it is based on models such as Lasso that force many features to zero weights.

For instance, I used a logistic regression model with Lasso as the penalty function for regularizing it. Then, I set the number of features that I wanted to keep to five and run the model. The output features were *Age*, *Cervical*, *Weight*, *A_Ancho2*, and *Snorer_no*. It is worth noting all these features have already shown up in other feature selection models.

Once I tried all these techniques, the main conclusion drawn was that all of them tend to output the same features, and even some of them were already highlighted in the EDA such as *non-snorer*, *Cervical* and *Weight*.

5. Machine Learning Modeling

In this section, I discuss the machine learning modeling process for both classification and regression models. I do not enter in too much detail on how each model works. I prefer to focus on the methodology that I followed to test the models as a whole. First. I explain general characteristics for both classification and regression modeling such as data splitting, cross-validation, and hyperparameter-tuning techniques. Then, I explain the metrics, list of models and their hyperparameters for each modeling approach.

Data splitting is always the first step. Initially, I thought of splitting both datasets into train and test and save the latter just for the last part of the results, when I would have already selected the best model. This is a common practice but I eventually discarded due to the small data sample that both datasets have. In order to avoid overfitting from not splitting the dataset in this way, I implemented cross-validation for all the tested models and for the RFE-CV feature selection technique.

As for the cross-validation, I used a stratified 10-fold whose output was the prediction for each of the ten folds. These folds were concatenated into just one prediction array which I used to compute the metrics. On the other hand, I implemented a stratified 5-fold for the RFE-CV feature selection. Stratified K-fold is a common choice in cross-validation and usually considered best practices. This method tries always to balance the generated dataset samples folds.

Regarding hyperparameter-tuning, I implemented grid-search manually, that is to say, I changed the hyperparameter by hand following a simple pattern. If it was numerical I either multiplied it by 10 or divided it by 10 as suggested in [1]. If it was categorical, I chose some of the available options. In my research lab, I usually automate this process but this time there were lots of models with many diverse hyperparameters. I thought it was going to be really time-consuming and just not worth it.

5.1. Regression Modeling

5.1.1 Regression Metrics

I decided to utilize the same regression metrics that I use in my scientific papers: *Mean Absolute Error* (MAE), *Root Square Mean Deviation* (RSMD), and *the coefficient of determination* or *R Squared* (R^2). These are well-known metrics in state-of-the-art regression problems. **MAE** is the simplest one, the absolute difference between predicted and real values. **RMSD** squares this difference to penalize bigger errors. **R^2** measures the proportion of variance that is predictable among two or more variables, one dependent and the others independent. The formulas of the three metrics are as follows:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \quad RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \quad \mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{xy}) - (\sum \mathbf{x})(\sum \mathbf{y})}{\sqrt{[\mathbf{n}\sum \mathbf{x}^2 - (\sum \mathbf{x})^2][\mathbf{n}\sum \mathbf{y}^2 - (\sum \mathbf{y})^2]}}$$

5.1.2 Regression Models

The list of regression models was based on the ones suggested in **[1]** plus some additions such as k-nearest neighbors, Xgboost, and multi-layer perceptron. I grouped them in families of algorithms to ease their interpretation. I discuss the hyperparameters of each group to analyze how they influenced the modeling process.

Classical models - Linear Regression, and Stochastic Gradient Descent (SGD)

These are the Machine Learning base-models. As for linear regression, I trained it with all the implemented techniques in section 4 and also tried its multinomial (multiple features) and polynomial (non-linear) variants. As for SGD, I trained it with different loss functions and penalties (Lasso, Ridge, ElasticNet). It was necessary to scale this model first to avoid convergence issues.

Regularizers - Lasso, Ridge, and ElasticNet

These algorithms apply regularization to avoid overfitting the model. As for Lasso, I trained it for different *alpha* values which is its regularization parameter. The same parameter was tweaked for Ridge but I also changed the solver for this one. Lastly, I played with the parameters *alpha* and *L1-ratio* of the ElasticNet, that weight the influence of Lasso and Ridge in a combined model.

K-Nearest Neighbors (kNN) - Nearest Neighbors, and Radius Neighbors

These are distance-based algorithms. As for the Nearest Neighbors, I tweaked the number of neighbors. On the other hand, I tweaked the radius for the Radius Neighbors. This last one gave many problems as it was difficult to set a valid radius.

Tree-based models - Decision Tree, and Extra Trees

These methods compute a graph of decisions based on the data that are described with nodes and leaves. They work great even without feature scaling. For both algorithms, I trained them unconstrained and then constrained (or regularized) to avoid overfitting. For doing so, I tweaked the maximum depth and leaf nodes, and the minimum leaf and nodes samples. I also set a random state and left untouched their criterion and splitter. Figure 16 shows an unconstrained and constrained tree respectively.

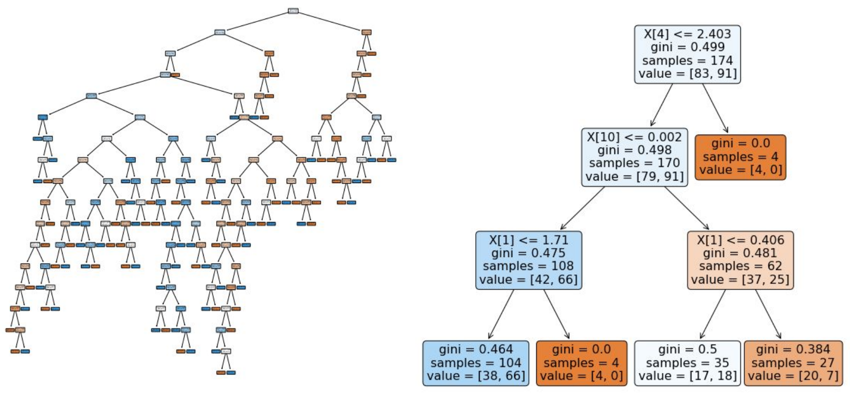


Figure 15: Unconstrained and constrained trees.

Ensemble models - Bagging, Random Forest, Adaboost, GradientBoosting, Xgboost

These are powerful algorithms that utilize hundreds and even thousands of the previous estimators using different approaches. For all of them, I tweaked the number of estimators and depending on the model I tweaked the parameters of their estimator (that were mostly trees and linear regressions).

Support Vector Machines (SVM) - Suppor Vector Linear and Nonlinear Regressor

These are highly interpretable models that used to be the state of the art until the uprise of Deep Learning and Neural Networks. For the linear case, I tweaked the parameters *C* and *epsilon* as suggested in **[1]**. These two are related to the speed and regularization of the model. For the nonlinear case, I tried also different *kernels* such as polynomial or RBF. The kernel functions transform the data to make the model nonlinear.

Neural Networks Model - Multi-Layer Perceptron (MLP)

These are the most powerful algorithms within this list. I just tried its feedforward version or MLP, but there are many variations such as CNNs, RNNs, GANs, Deep-RL. This algorithm might overfit the data as is better suited for largest datasets. I tweaked its number of neurons, layers, activation function, and solver function. I always scaled data first.

5.2. Classification Modeling

5.1.1 Classification Metrics

Regarding classification metrics, I selected again state-of-the-art ones. The first set of metrics is based on the confusion matrix of a binary class with positive statements (e.g the cat is black) and negative statements (the cat is not black) that are predicted correctly (or true) or wrongly (or false). Figure 16 shows this matrix.

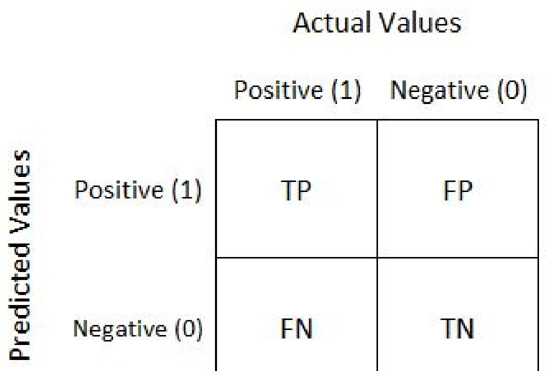


Figure 16: Confusion matrix.

1. True positives (TP): The predicted value is positive and the real value is 1 positive.
2. True negatives (TN): The predicted value is negative and the real value is negative.
3. False positives (FP): The predicted value is positive and the real value is negative.
4. False negatives (FN): The predicted value is negative and the real value is positive.

From this matrix, I computed the ***accuracy*** (total number of true predictions divided by the total number of prediction), ***precision*** (total number of true positive predictions divided by the total number of positive predictions), ***recall*** (total number of true positives predictions divided by the total number of positive statements), ***f1-score*** (aggregation of precision and recall, I used the weighted average variant).

Apart from these metrics, I plotted for different decision thresholds the ***precision-recall*** curve and the ***ROC-AUC*** curve (computing also its score). Figure 17 shows the resulting graphs of testing a model.

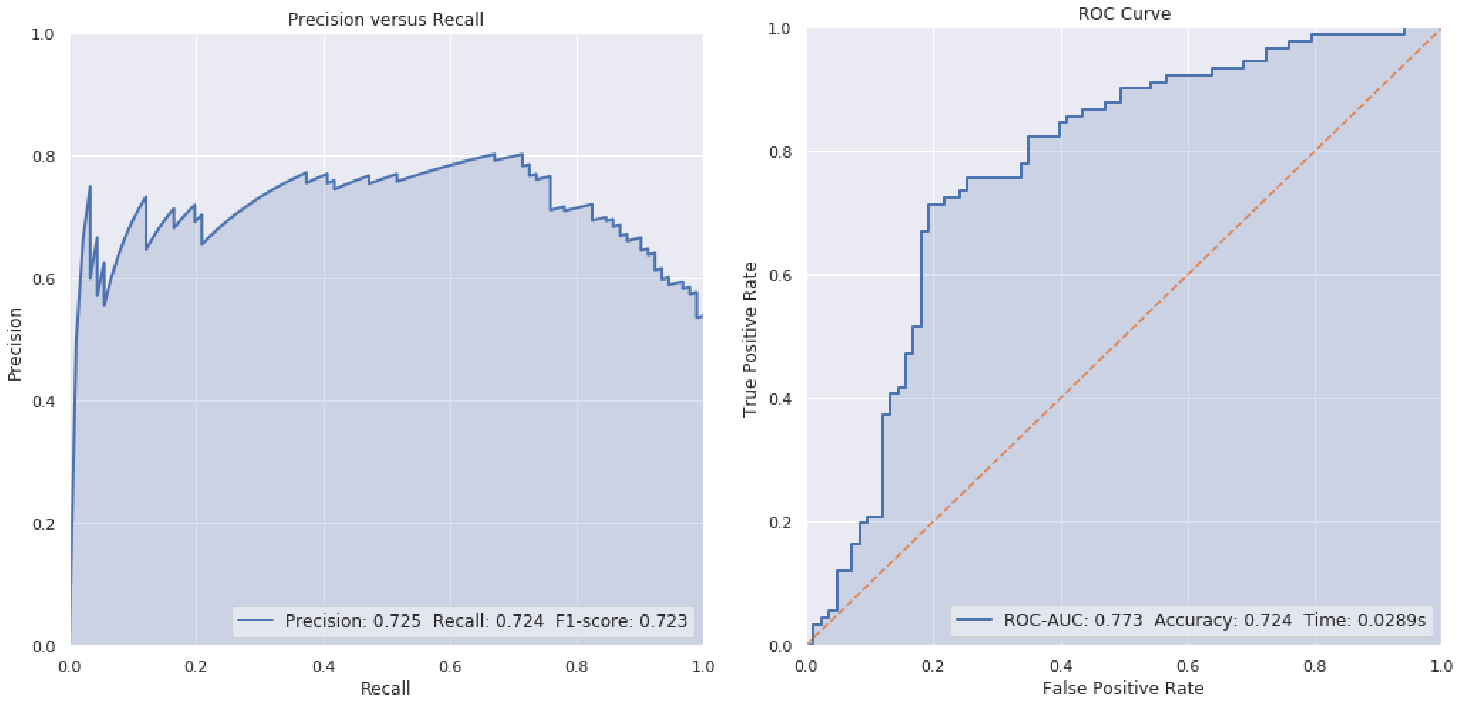


Figure 17: Precision-recall and ROC-AUC curves.

5.2.2 Classification Models

As in the regression modeling, I based my list of classification models on the one suggested in **[1]** and then I added other algorithms such as Naive Bayes models and perceptrons. This time, I list the models and comment them only if there are differences with their regression counterparts (aside from their goal which is classification) to avoid repetition. As mentioned before, I used RFE-CV feature selection for all the classification models that supported it.

Classical models - Logistic Regression, and Stochastic Gradient Descent (SGD)

This time, I used logistic regression which is the classification version of linear regression, so to speak. I tweaked its regularization parameter *C*, the penalty function and the solver function.

Naive Bayes models - Bernoulli and Gaussian

These Bayesian-based algorithms might have also been considered classical models. They are not powerful nor have many parameters. I just tweaked *alpha* in the Bernoulli model.

Regularizers - Ridge

I utilized just Ridge since it was the only one implemented for classification. Same parameters as in the regression model.

K-Nearest Neighbors (kNN) - Nearest Neighbors

Same reason as before, I used only nearest neighbors since it was the only one implemented for classification. I tweaked the same hyperparameters as in the regression model.

Tree-based models - Decision Tree, and Extra Trees

Same parameters and algorithms as in the regression model.

Ensemble models - Bagging, Random Forest, Adaboost, GradientBoosting, Xgboost

Same parameters and algorithms as in the regression model.

Support Vector Machines (SVM) - Suppor Vector Linear and Nonlinear Classifier

Same algorithms and parameters as in the regression model except for the parameter *epsilon*.

Neural Networks Model - Perceptron and Multi-Layer Perceptron (MLP)

The perceptron model is just a neuron of an MLP. It was only available for classification. I tweaked its hyperparameters penalty and *early stopping*.

6. Results and Model Comparison

In this section, I discuss the results and compare the models of both classification and regression. I do not compare only their accuracy, but also their elapsed training time (time constraint), and the features that they require when using RFE-CV (size constraint).

I decided to utilize these approaches since state-of-the-solution in data science are claiming now that quickness, size, and energy consumption are as important as accuracy. Data scientists now focus on developing more sustainable and scalable models.

As explained in the MLLB report, I just recorded the results of roughly 50 models for each modeling process but I tried many others. These are just the ones I found more interesting. To make the model comparison more appealing, I relied heavily on visualizations. For each subsection, I discuss different plots from which I draw some conclusions.

As an overall conclusion of the results, I would like to point out that the dataset was too small to really take the results seriously in a real-world scenario. It is not enough sample size. So I considered this section more useful for designing a viable methodology for assessing results in a future scaled scenario than for assessing a real-world scenario right now with just this data.

Note that I used one page per plot for better visualization. Most of them needed at least one page anyway.

6.1. Regression Results

MAE vs. RMSE - Model Comparison

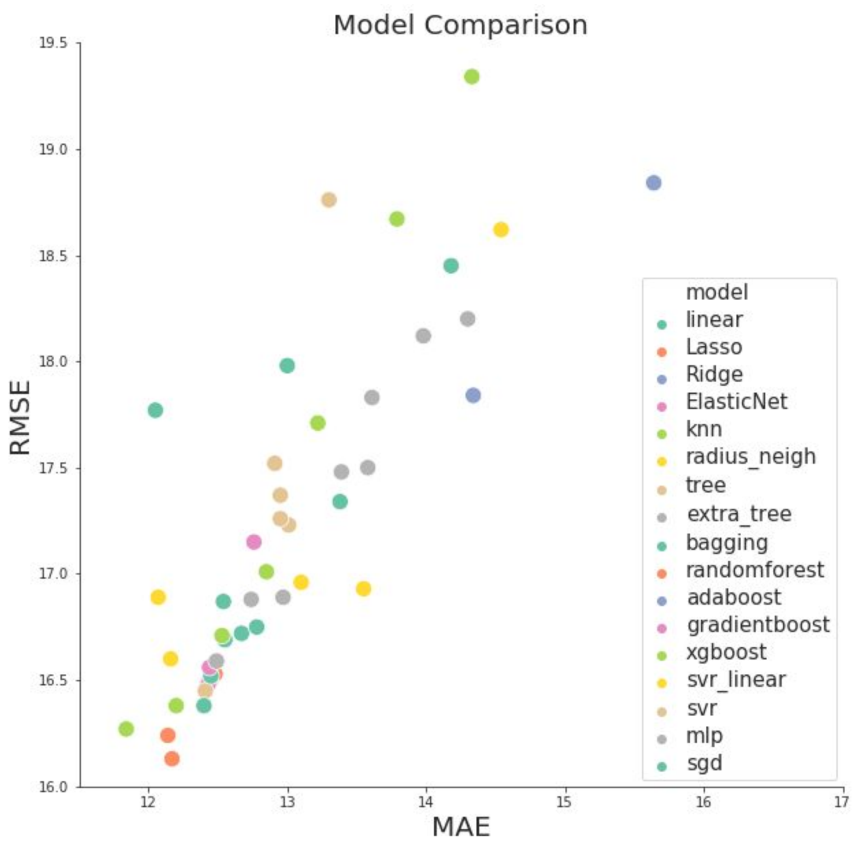


Figure 18: MAE vs. RMSE model comparison.

Figure 18 depicts a scatter plot of MAE and RMSE by algorithms. The best models are random forest and xgboost, perhaps the most powerful ensemble methods. On the other hand, the worst models are kNN-based. The best models roughly get a minimum MAE value of 11 and a minimum RMSE value of 16.

MAE vs. R^2 - Family of Algorithm Comparison

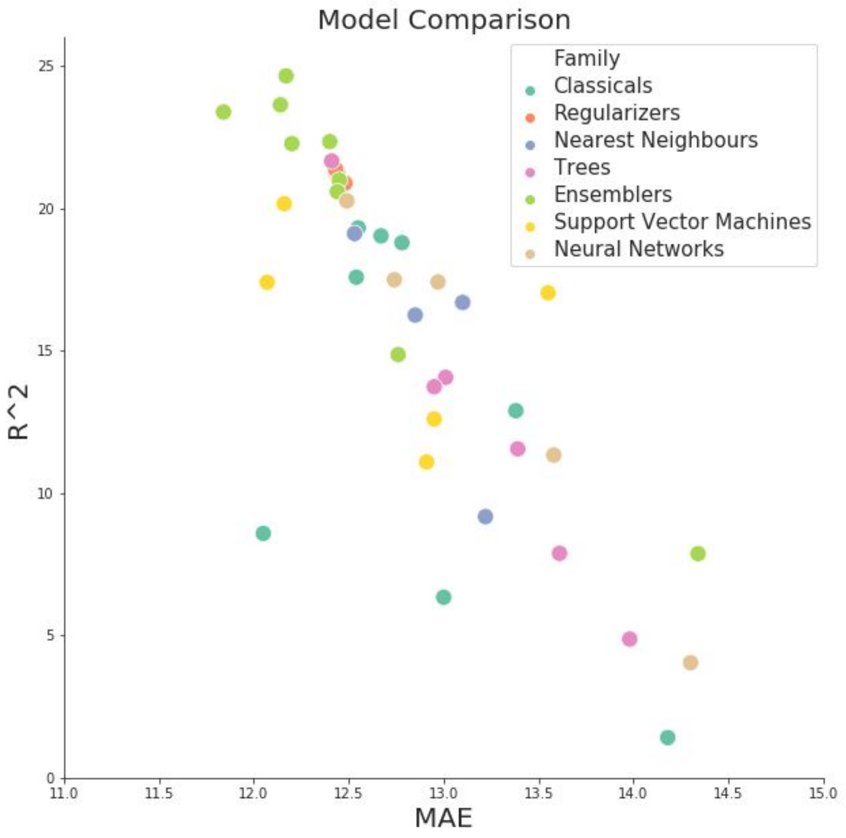
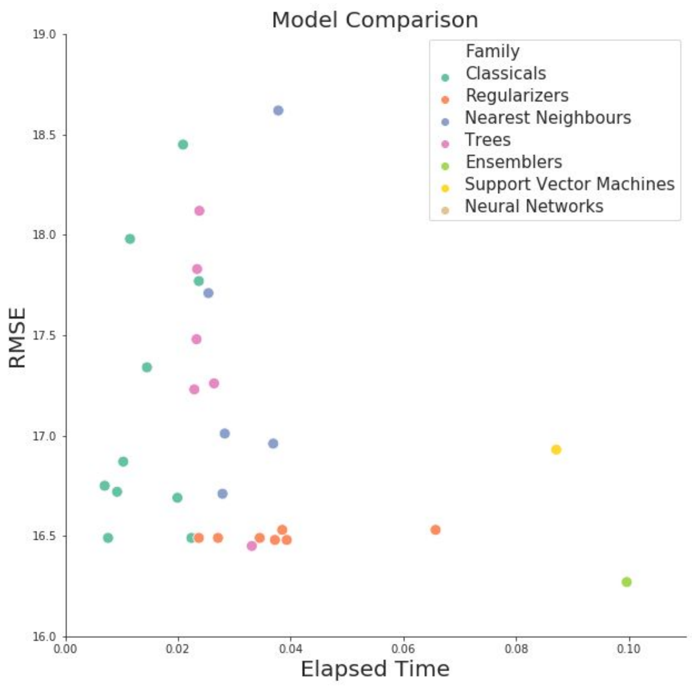


Figure 19: MAE vs. R^2 family comparison.

I grouped the algorithms by their family to better visualize the results. Figure 19 depicts a scatter plot of MAE and R^2 using these families. Here, it is clear that the best models belong to the ensemble family followed by some trees and regularizers. The worst models are a mixture of different families. The maximum value of R^2 is roughly 25%.

Elapsed time vs. RMSE - Family of Algorithm Comparison and Data Preparation



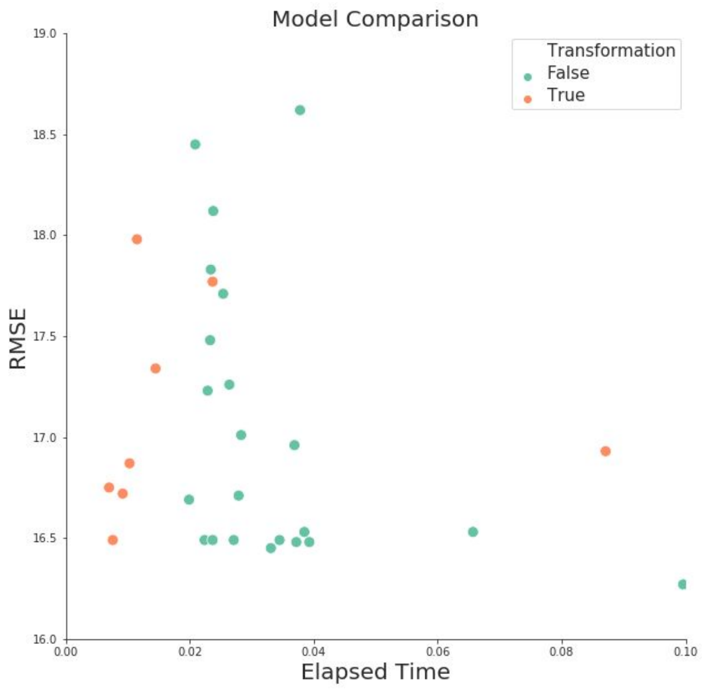


Figure 20: Elapsed time vs. R^2 family comparison and data preparation.

Thus far, I took only into account the accuracy of the models measured by standard regression metrics. As I mentioned before, I wanted to add a time constraint since it is relevant in data science nowadays.

The left plot from figure 20 depicts the RMSE of the quickest models by their family. Here, it is clear that ensemble methods were really slow and not necessary for achieving a good accuracy as classical methods performed almost as good as them (~+0.5 RMSE) and were at least more than one order of magnitude quicker.

Moreover, looking now at the right plot from figure 20, it represents the same distribution but colored by whether they used any data preparation technique from section 4 or not. Here, it is again very clear that the quickest and best models are the ones that underwent some kind of pre-processing.

Elapsed time by Family of Algorithm and Data Preparation

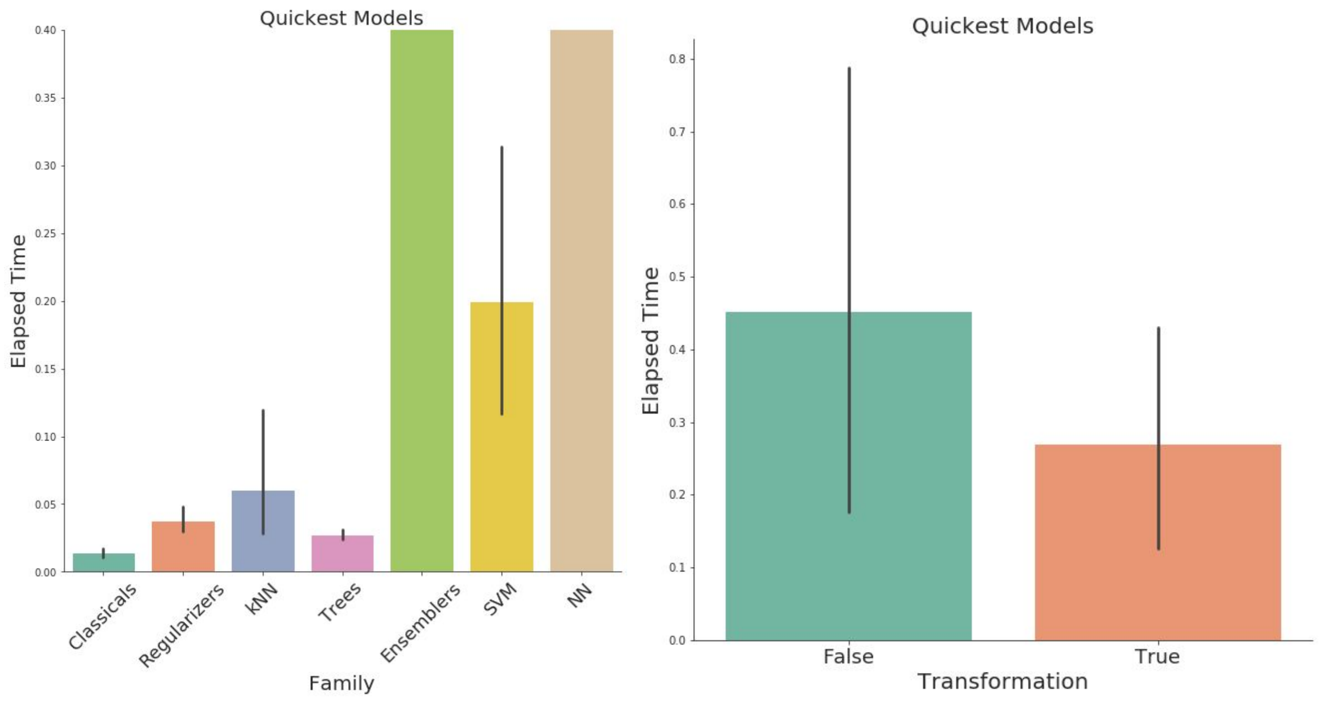


Figure 21: Time elapsed by family and data transformation.

Figure 21 depicts how rapid were the models to clarify the previous graphs. In the left plot, it is clear that the most powerful methods are also the slowest one, by more than one order of magnitude comparing them with the rest of the models. SVMs are mid-tier power-wise and also time-wise. Consequently, the least powerful options are also the quickest.

The right graph shows that preprocessing the data not only makes better and more robust models, but it also reduces the training time. From this analysis, it is inferred that for such a small dataset, the best models do not need to be powerful but do need some kind of pre-processing first.

6.2. Classification Results

Before diving into the classification results, there is a caveat. The metrics accuracy, recall, precision, and f1-score almost always had the same value within a model, hence I use just the first one to compare it with the other metrics.

Accuracy vs. ROC-AUC - Model and Feature Selection Comparison

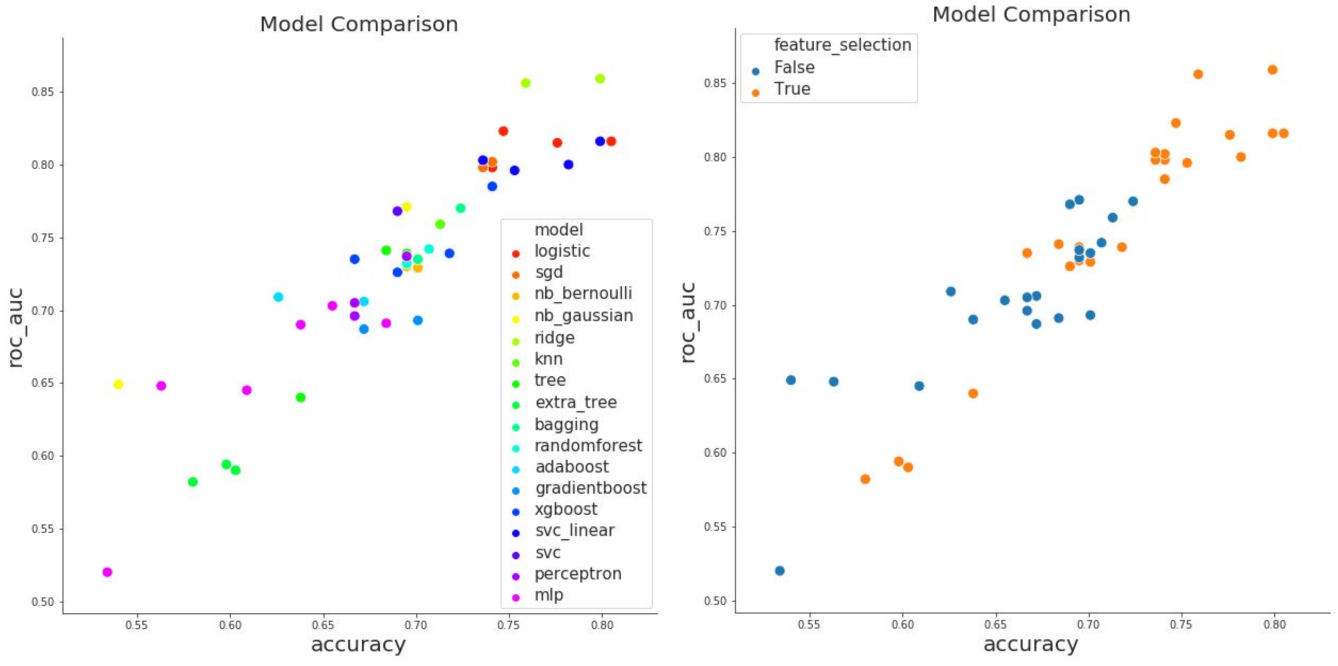


Figure 22: Accuracy vs. ROC-AUC model and feature selection comparison.

The left plot from figure 22 depicts a scatter plot of the ROC-AUC score and accuracy metrics colored by algorithms. Less powerful models like ridge classifiers, logistic regressions, naive Bayes approaches, and linear support vector classifiers were the best ones. This was odd as regression models showed that powerful methods were better when there are no time constraints. So, I wanted to measure how important was applying the feature selection technique RFE-CV.

The right plot from figure 22 depicts the same distribution but colored by whether they used RFE-CV. Here, It is very clear that feature selection was the most decisive factor for the performance of the classification models. Top-right, there is a cluster that only contains models that used RFE-CV.

Aside from that, the maximum ROC-AUC score was roughly 85% and the maximum accuracy was roughly 81%.

Total Number of Fitted Features vs. Accuracy - Feature Selection Comparison

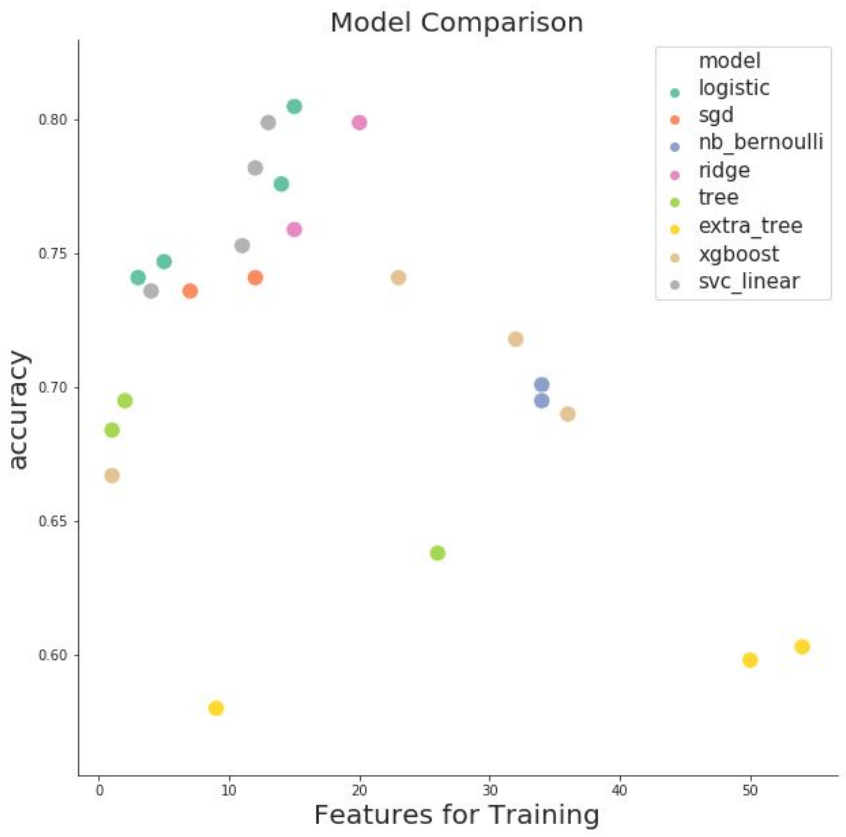


Figure 23: Total number of features used in training vs. accuracy comparison.

Figure 23 depicts the accuracy of models that have used RFE-CV by the number of features that they utilized. This helps to asses the model size and minimum number of features, another relevant factor in today’s data science world.

From more than 50 features that the classification dataset contains, best models use only between 10 and 20 features. The accuracy drops by ~6% when the number of features is between 1 and 10.

Selected Features by the RFE-CV Feature Selection Technique

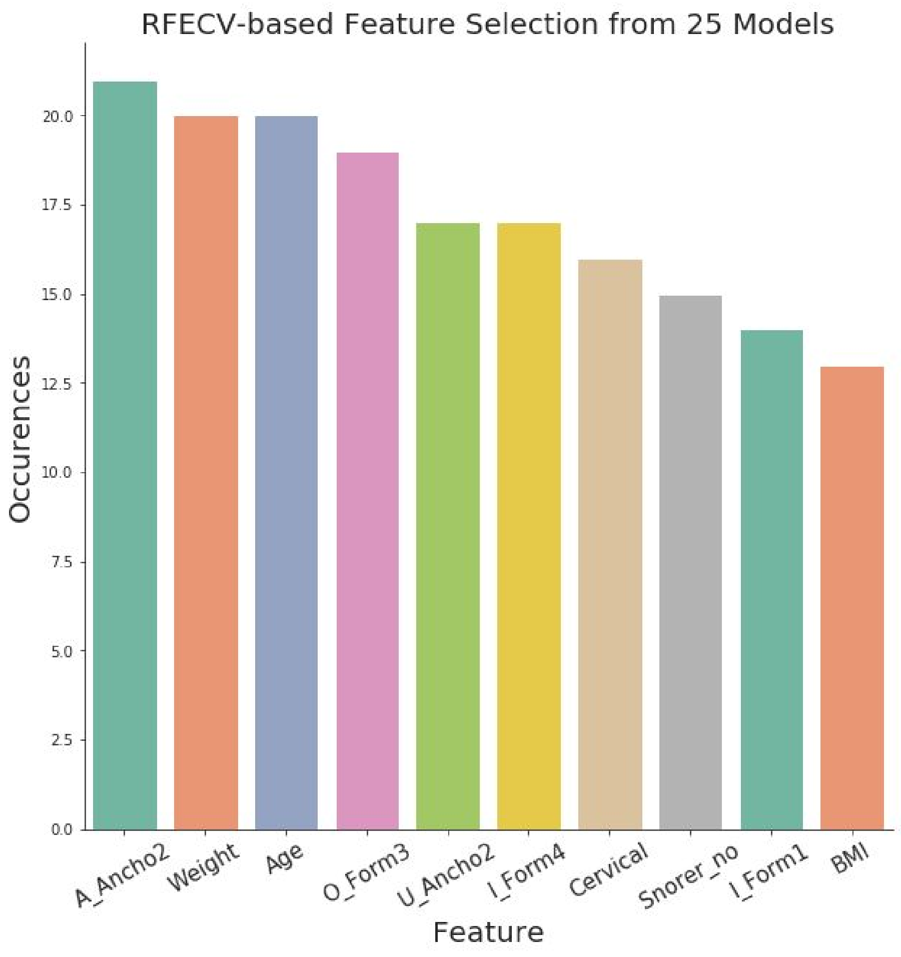


Figure 24: Selected features by the RFE-CV feature selection technique.

Bearing in mind the importance of feature selection, which are the most important, and thus selected, features in this technique? Figure 24 depicts the number of occurrences for each feature in 25 models that used RCE-CV. It only shows the top-ten most selected features.

On the one hand, it uses features that were already highlighted in the OSA study of section 1, in the EDA of section 3 and in the feature selection techniques such as *Weight*, *Age*, *Cervical* and *non-snorer*. On the other hand, there are five out of ten features from the voice records that were not highlighted before.

Moreover, the best feature is one of them appearing in 21 of the 25 models. This seems odd and might indicate that the models are overfitted as these features would help to find local minima that would fit for such a small sample size, but would not in a real-world scenario. It is difficult to draw meaningful conclusions from so little data (roughly 175 instances).

Elapsed Time by Family of Algorithms

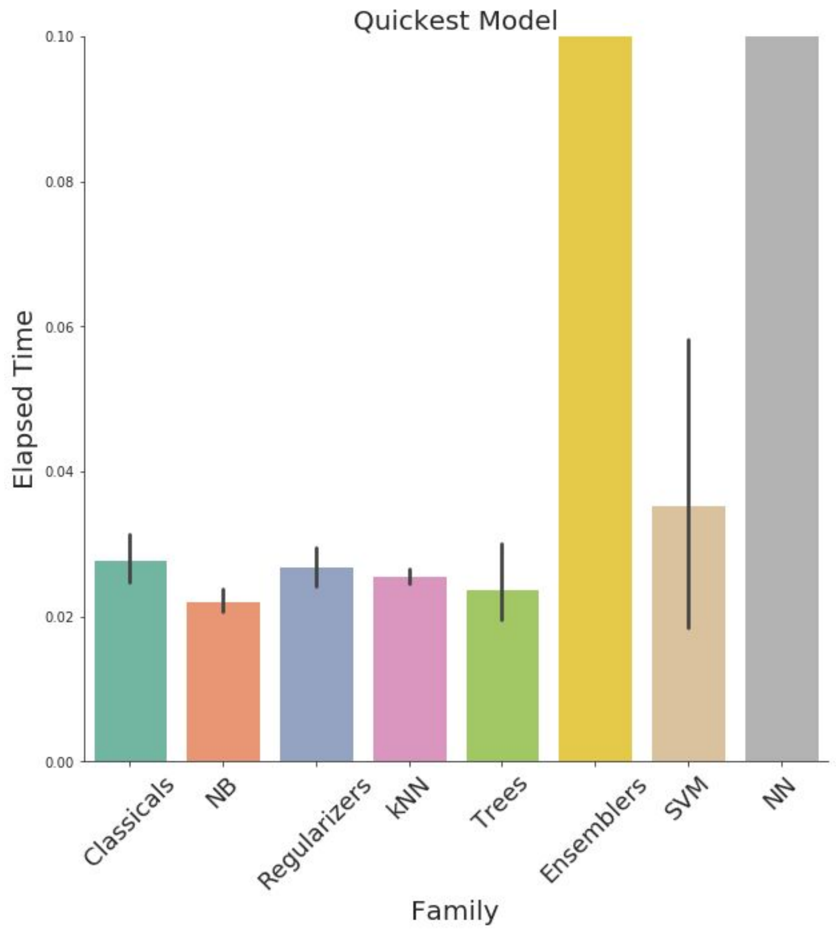


Figure 25: Elapsed time by algorithm family.

As in the regression results, I plotted the elapsed time by algorithm family to asses how quick the models are for this case. It follows the same time distribution as in the regression case. The most powerful methods are the slowest and vice-versa. It is interesting to note that perhaps the less powerful family, Naive Bayes, are also the quickest, confirming my hypothesis.

Contrary to the regression case, time was not a factor to select the best models. These already were simple models such as logistic regressions, regularizers, and linear support vector classifiers.

7. Conclusions

In this project, I developed a comprehensive and detailed methodology to predict obstructive sleep apnea (OSA) cases using machine learning. This methodology contains six clear and precise steps that contribute significantly to the final results. The main disadvantage was the quality of the dataset. It included so few instances that drawing meaningful conclusions for a real-world scenario is cumbersome.

As I already had experience in this field due to my job in a research lab, I found most of the steps familiar and easy to follow. I want to highlight the feature selection techniques as I had never used them before. They were the most important part of the classification model's accuracy. Moreover, they were really useful to asses feature importance. I am looking forward to utilizing them in other problems.

On the other hand, I considered that it was not necessary to pinpoint the best model for both classification and regression as probably it is overfitted due to the quality and size of this dataset. Instead, I found more interesting to group the models and compare them to evaluate how they evolve and behave while adding constraints such as time and size which are really relevant in today's data science field.

This evaluation method proved to be useful as it showed that models with a little less accuracy were much faster, and thus preferred. It also showed that data pre-processing not only helps improving the accuracy and avoiding overfitting, it also reduces the elapsed training time considerably.

If the reader still wants to have a look at the models individually, I will upload to my Github account **[3]** more than 250 plots from all the models presented in section 6, that were not shown in this report. These graphs contain box plots, histograms, scatters of regression models; ROC-AUC, precision-recall curves for classification models and tree plots for tree-based models I will also upload the eight notebooks with all the code I have implemented in this project. As of today, they are just partly documented. Finally, I will also update the excel files needed to run the notebooks.

If the reader wants to know more about my research, please do not hesitate to visit my ResearchGate account **[4]**.

8. References

[1] HealthLine, "Obstructive Sleep Apnea," [Online]. Available:
https://www.healthline.com/health/sleep/obstructive-sleep-apnea.
[Accessed 3 November 2019]
[2] *"Hands-On Machine Learning with Scikit-Learn & Tensorflow,"* Aurélien Géron, O'Reilly
[Accessed 3 November 2019]
[3] Github, "Sergio Pérez Morillo Profile," [Online]. Available:
https://github.com/spmorillo
[Accessed 4 November 2019]
[4] ResearchGate, "Sergio Pérez Morillo Profile," [Online]. Available:
https://www.researchgate.net/profile/Sergio_Perez77
[Accessed 4 November 2019]