# Report for Machine Learning with Python Labs

## 1. Tools:

Initially, we install packages for visualization (matplotlib, seaborn), for numerical operations (numpy), dataframe manipulation (pandas), and machine learning tools (scikit-learn).

## 2. Data analysis:

We start checking for missing values, typos or uncertain values. The main point is to identify if can be taking into account, it they are proportionally important, and in further analysis if we can skip them.

For three of the attributes in the dataframe, there are three that present unknown values: bmi, smoking_status, and gender. Respectively, NA, unknown, and other. Based on this information, we plot histograms in order to verify their proportional importance in the dataframe. Smoking_status is the one that present more problem due to be 30.2% of all instances.

We check how it influences the stroke attribute, our target variable, based in the conditional probability. In the influence is evident. In a first moment, we decide to skip this attribute, and look for less problematic ones, or at least, that does not present relevant missing data.

First, we look encode the categorical variables, and build a heatmap that can allow us to identify the correlation between the different attributes. A evident linear relationship is that between the 'age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi', and the target attribute 'stroke'.

From these five attributes, we choose first to plot histograms for the numerical ones, 'bmi', 'age', and 'avg_glucose_level'. With the exception of age, the other variables present outliers that we need to analyse further. For such case, we plot histograms where the different colors represent the proportion for one value of 'stroke' for that specific bin. For the case of 'avg_glucose_level', the proportion of 'stroke' equal to 1 is more concentrated in the outliers, where there is less dense the data points. And for some bins, it is possible to notice that there are no values for stroke equal to 1. This a fact to be explored soon when selecting features.

For the categorial variables we do a similar analysis using bar plots. Analogously as in the previous case, the proportion of positive values for stroke is much less. In the case of gender equal to 'Other', and 'work_type' equal to 'Never_worked', there is not even samples containing for positive cases in 'stroke'.

## 3. Cleaning and training:

As a first try, we train a linear algorithm just for , 'age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi'. Since they present a linear relationship, it is expected that the use of all these variables is redundant. We start with a logistic regression model, and consider just the accuracy to evaluate the performance.

The variables are encoded to numerical ones, and rescaled based in the active attributes for training. Our algorithm prints just the best results for accuracy for test and training, as the confusion matrix, precision and recall, and F1-score both for training and test.

The result shows how accuracy can be misleading about the performance of an algorithm. The best result for such metric is provided just using the attribute 'bmi' with 96% and 94.6% of accuracy for the training and test set respectively. However, checking the precision, recall, and F1-score, all of these metrics present 0. What means that True positives, stroke equal to 1, is not detected for training or test.

A scatter plot 'bmi' by 'stroke' shows why this happens: both cluster for the two values of stroke, are not linearly separable. It means that there are intersection between values in bmi that are in both clusters. If we consider that the data is highly unbalanced over the negative value of stroke, our linear model, selects everything as negative. And recalling the plots before, true positives for stroke, corresponds exactly to the lack of accuracy in this case.

Based in the previous histograms, we perform this filter:

1. age < 40;
2. 20 < bmi < 50;
3. 150 < avg_glucose_level < 170;
4. 'gender' different of 'other';
5. 'work_type' different of 'Never_worked'

Checking for the new proportion, there is not still much difference even removing such instances. Just an increase of 0.4%. Then we face the same situation: just a slight change for F1-score for training.

Then, we try to use the logistic regression model using weighted classes, trying to minimize the influence of the unbalance. Now, we evidently notice the difference, however, while for training and test, accuracy continues high, 92% and 90% respectively. For precision and recall, if it is not great for training, and relatively good for test. For training, precision and recall correspond to 13% and 17% respectively, while for test, we have 82% and 79%, respectively. The F1-score reflects such behavior, 22% and 28%, respectively. For different simulations, the same behavior can happen in an inverse way.

The results up to here allows again to face a similar situation as in the previous cases: unbalance makes the algorithm to focus more in false negatives, then in true positives.

We move to decision trees in order to evaluate what a nonlinear model can provide in such situation. For a same distribution between training and test set, here, the algorithm shows excellent accuracy for training (100 %) and for test (93 %). For precision, we have 100 % in both cases, but for recall, the performance could not reach such plateau. We have for precision, recall, and F1-score, 33 %, 24 %, and 28 %, respectively. These results shows that with decision tree, we continue to have the problem with the proportion of false negatives.

In a last attempt, we try random forest, but the performance is even below the decision trees, even if it corresponds to the same pattern.