Handout

Final Project in Machine Learning - Fabrice Rossi

Alexandre Saint and Lubin Bénéteau

Github link to our source code: Click Here

2025 – first semester

Université Paris-Dauphine – PSL

Contents

1 Introduction

This project aims to develop a machine learning model able to predict an unobserved feature linked to the socio-economic status of a French individual. We are using their job occupation, the sport they practice and their geographic environment. Hence, we built a learning and a test set. We built machine learning models on the learning set, then we estimated performances on these data before producing predictions based on our best model. To select our best model we played with meta-parameters and used cross-validation, using as benchmark a basic random forest model.

2 Data Acquisition and Merging

2.1 Dataset Overview

The main files are `learn_dataset.csv` and `test_dataset.csv`. They describe the individuals to their ages, sex, family, diplomas, activities, student status, socio economics category, INSEE code, etc. On the top of that, in the `learn_dataset.csv` we find the `target_variable`.
`learn_dataset_EMP_CONTRACT.csv` and `test_dataset_EMP_CONTRACT.csv` with the variable `EMP_CONTRACT` describe the type of job. More precisely (and it is important to focus on this point), `EMP_CONTRACT` concerns all the occupied people (including those without job), that is why some cells without description. Hence these cells are not missing values but could indicate a certain job status.

Also, `learn_dataset_job.csv` and `test_dataset_job.csv` give features about jobs occupation (type of contract, sector, category of job, employment condition, hours worked, etc.). There are also files related to retired individuals which give us an overview of the previous socio-economic category, age of retirement, description of the latest job and the level of pensions. Finally we found a subset of files concerning sports and administrative data (town, region, localization, population, etc). The sport related dataset could lead to issues. Indeed, some individuals don't have any club, we treated this case writing "missing" as an information not as a "missing value".

2.2 Merging Strategy

We built a individual unique base merging, for each subset (`learn` and `test`), the main `.csv` file (`_dataset.csv`) with the other `.csv` files related to employment, retirement, sport using the key `Person_id`. All merges are done using `left join` to keep all the individuals from the main file and to integrate available information without canceling any observations.

Using the `INSEE` code using a string type to keep initial zero(s) we add information to our merging dataset. Then we run sanity test to identify any suspicious codes. Also, to avoid any inconsistency between files (for instance, it is the case for contracts types) we used a priority rule as it is said in your instructions. That is why the `EMP_CONTRACT` file has been chosen to be the most reliable.

3 Data Cleaning and Sanity Checks

3.1 Inconsistency Corrections

At this stage, we did a systematic cleaning to guarantee consistency in the merged dataset in order to avoid any error at the learning and training stage.

We set up two **sanity checks**, one concerning information both in `EMP_CONTRACT` and `_job.csv`, the other concerning the length of `INSEE` code which could begin by a zero. The first check identify inconsistency between datasets. To solve this we used as it has been said before, a priority rule consisting of considering `_dataset.csv` as the most reliable source then the `_EMP_CONTRACT.csv` file is the second more reliable. In other words, we normalized some contracts name to make it comparable. Also, we paid attention not to cancel any observation to keep consistence with the prediction step.

3.2 Missing Value Handling

Missing values are treated first in our pipeline during the step of cleaning (for both `cleaned_learn_dataset.csv` and `cleaned_test_dataset.csv`). These data are consistent between each other and are our input for following machine learning models. We pay attention to delete none of the observations. There are two types of missing values, the occasional (or incidental) missing values (unknown `Working_hours`), for which imputation may be appropriate and the structural missing values which cannot be imputed but encoded as a new type (unemployed, independent, no club, etc.).

Concerning categorical variables, missing or absent values (for instance those linked to employment for retired people) are encoded using a explicit category naming "`"Unknown"`". This approach permit to keep the absence of information as an information, which may be helpful for our prediction.

Concerning numerical variable, missing values are computed as median on the sample. We privilege this method rather than the mean since it is more robust to extreme values. The same process has been applied to the learning and test sets separately to avoid any data leakage (which would have been a major issue in our framework).

4 Feature Engineering and Encoding

4.1 Categorical Mapping

Some categorical variables have a huge number of modalities (more than a hundred). Encoded them using `get_dummies` would lead us to issues in term of computational capacity concerning our laptops. To reduce the dimension of data those modalities has been grouped by larger categories (at a N1 level for employment type for instance in `code_work_desc_map.csv`). For instance, in this file we grouped the not numerous categories into larger ones, at `N1` level and in some case at `N2`. The choice of the level is guided by a compromise between amount of information and reduction of modalities. `N1` make the merge robust while `N2` permit to keep diversity when it enhance the performance.

This permit us to keep a coherence economically speaking, limiting risks linked to overfitting due to a overwhelming number of dimensions. Also we reduced the time of training and facilitate the comparison of models.

4.2 Data Transformation

Categorical variables are encoded through `LabelEncoder` from `FeatureEncoder`. This class guarantee a consistent encoding between learning and test sets. Unobserved categories in the learning set are attributed to a known class permitting the models to be robust at the prediction stage.

The target variable is also encoded using a specific encoder consistent with classification strategies from Scikit-learn (numerical labels to train the model, then transform it with `inverse_transform`

on the prediction to re-find the original labels in the final `predictions.csv` file). Finally, there is a methodological limit: `LabelEncoder` introduces an artificial order between modalities. `OneHotEncoder` would have been more suited but require more resources for our laptops. Either way, the encoding step is a part of our pipeline as it is the case for the merging and cleaning stages.

5 Model Development

5.1 Benchmark Model

Several models of classification are assessed. We used decision trees, Random Forest and boosting models. Among them, Random forest model is our benchmark/baseline. Boosting models are known to be better at the prediction stage, they can be better the Random Forest after being fine tuned. The choice of the models is consistent regarding the dataset which include a lot of data (categorical, but numerically encoded).

5.2 Hyperparameter Tuning

Model performances are evaluated using a 5-fold cross-validation, which provides a more reliable and more reproducible estimate of the expected performance on new data than a simple train/test split. The selection of the main hyperparameters is do ne through this resampling scheme: for each algorithm, we test a grid (or a random search) of hyperparameters, and we keep the configuration that maximizes a metric fixed in advance (for instance balanced accuracy or macro-F1 when classes are imbalanced). For the Random Forest, the key hyperparameters considered are typically `n_estimators`, `max_depth`, `max_features`, `min_samples_leaf` and, if needed, `class_weight`; for boosting models (including XGBoost), we mainly tune `n_estimators`, `learning_rate`, `max_depth`, `subsample` and `colsample_bytree`. Finally, to ensure reproducibility of the results, we set a common random seed (`random_state`) for both the cross-validation procedure and the training steps.

6 Results and Discussion

6.1 Performance Metrics

Model performances are evaluated using standard classification metrics, including overall *accuracy*, the confusion matrix, and detailed classification reports (precision, recall and F1-scores by class). However, to properly assess how the model should behave on unseen data, we do not rely only on a single train/validation split. Instead, we report an explicit estimate of generalization performance based on cross-validation, for example the mean CV score ± its standard deviation. The expected performances of our final model (XGBoost) on new data are `acc =` **0.7291**.

Moreover, since confusion matrices and accuracy are based on hard class predictions, we complement them with a score-based assessment. When available, we exploit predicted probabilities to plot ROC curves and compute AUC values in a one-vs-rest framework, which provides additional information on the ranking quality of the classifier and helps identify classes that are harder to separate. Finally, we explicitly state which metric is used to select the final model (accuracy vs balanced accuracy vs macro-F1), and we justify this choice, especially when the target distribution is unbalanced and accuracy alone can be misleading.

Figure 1: Comparative table

This barplot illustrates the performances consistent with our expectations, where boosting models are better than the other, especially for XGBoost and Hist Gradient Boosting. Scores are high for all the metrics used meaning that the method used to treat our data seems consistent. This display confirm the fact that XGBoost is our best model, hence our final one.

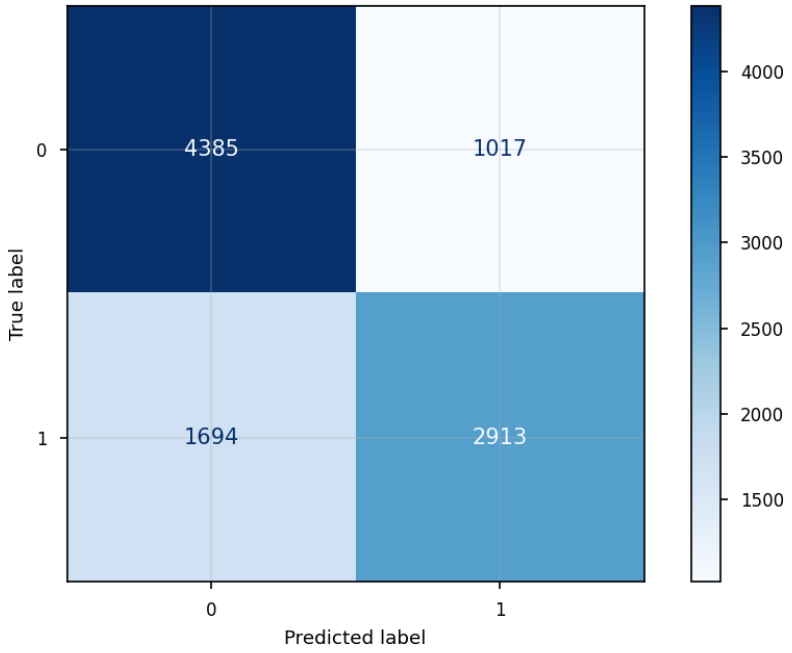

Figure 2: Confusion Matrix

This confusion matrix validates the performance of the model with 7 334 correct classifications. The identification of the 0 class is high, while the 1 one is slightly underestimated. Overall, the model demonstrate a solid capacity regarding the discrimination for the objective.

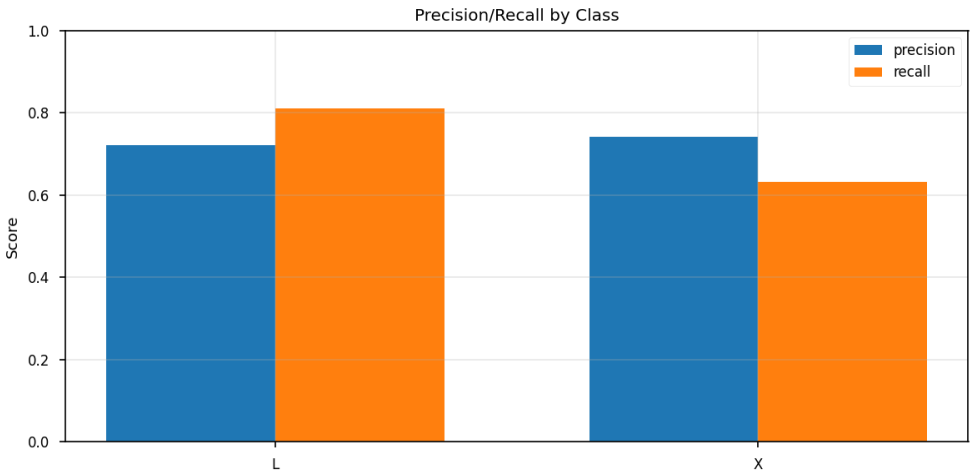

Figure 3: Prediction and Recall by class

This graph show that the model is both a good predictor and balanced: `L` class is well predicted while the other is predicted with a higher precision. Coupled to an the confusion matrix, this balance confirm that our final model is robust to estimate our target.

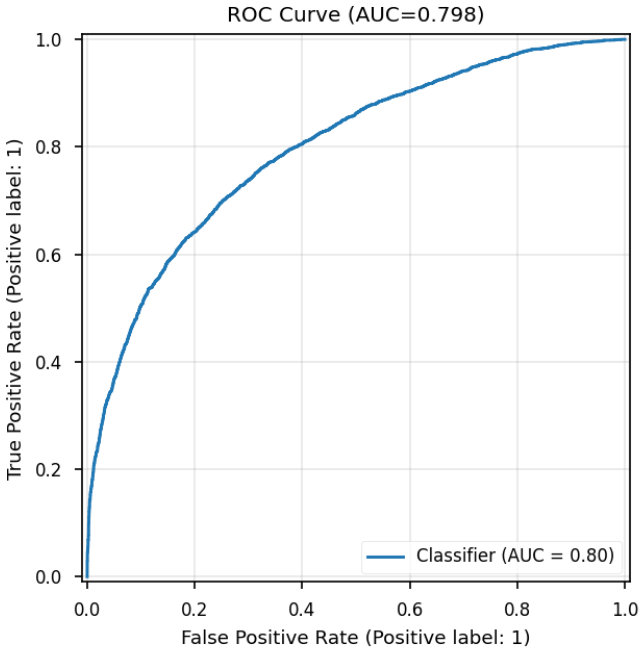

Figure 4: ROC Curve

The ROC curve displays a robust AUC rounded to 0,8 which is a sign of great capacity of the model to distinguish the target variables. this result confirm that the classifier maintains good compromises between sensibility and specificity.

6.2 Error Analysis

A confusion matrix is used to analyze the main patterns of misclassifications between socio-demographic categories. It helps us identify which classes are frequently confused with each other and therefore highlights the limits of the model, especially when some categories have similar profiles in terms of observed features. To go further than this global overview, we add two complementary diagnostics, as recommended. First, we provide a feature importance analysis (for instance permutation importance, or model-based importance for Random Forest / boosting) in order to interpret the predictions and to check that the model effectively uses the information coming from the different tables (job characteristics, retirement history, sport membership, and geographical context). Second, we investigate a set of "very wrong" predictions: when probability scores are available, we look at individuals for which the model is highly confident but still predicts the wrong class. These cases are particularly informative, since they often reveal systematic issues such as structural missing information (e.g., no detailed job description because the person is not employed), rare categories that are poorly learned, or remaining inconsistencies across data sources. Overall, these diagnostics provide a clearer understanding of where the pipeline performs well and where it can be improved.

7 Conclusion

This project shows that the combination of individual features, professional characteristics, sports tastes and geographical context permit to predict more or less precisely socio-economic categories. We used a robust and reproducible machine learning pipeline from the pre-processing of the data to the assessment of the models. Our final model is used to generated predictions on the test set. Those predictions are exported in the file `predictions.csv`.