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 through their ages, sex, family, diplomas, activities, student status, socio economics category, INSEE code, etc. We find the `target_variable` in `learn_dataset.csv`.
The datasets `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), which is why some cells are without description. Hence, these cells are not missing values but may rather indicate some job status.

Moreover, `learn_dataset_job.csv` and `test_dataset_job.csv` give features about job occupations (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 their previous socio-economic category, age of retirement, latest job description and pension level. Eventually, we found a subset of files on sports and administrative data (town, region, localization, population, etc). The sport related dataset could lead to issues, as some individuals don't have any club. We thus treated this case by writing "missing" as an information rather than as a "missing value".

2.2 Merging Strategy

We built a unique consolidated database for each subset (`learn` and `test`), by merging the main `.csv` file (`_dataset.csv`) with the other `.csv` files related to employment, retirement, sport using the key `Person_id`. All merges were done using `left join` to keep all the individuals from the main file, while integrating all available external information.

We used the `INSEE` code as a string type to keep the initial zero(s) to add information to our merging data set. Then we ran sanity tests to identify any suspicious codes. Moreover, to avoid any inconsistency between files (which we found for contracts types) we used a priority rule as mentioned in the project 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 regarding information both in `EMP_CONTRACT` and `_job.csv`, and the other the length of the `INSEE` code which could begin by a zero. The first check identified inconsistency between datasets. To solve it we used, as mentioned before, a priority rule by considering `_dataset.csv` as the most reliable source and `_EMP_CONTRACT.csv` second most reliable. In other words, we normalized some contract names to make it comparable. We also made sure not to remove any observation to ensure consistency with the prediction step.

3.2 Missing Value Handling

Missing values are treated first in our pipeline during the cleaning phase (for both `cleaned_learn_dataset.csv` and `cleaned_test_dataset.csv`). These datasets are consistent and used as input for the following machine learning models. We paid attention not to remove any 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.).

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

Regarding numerical variables, 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). Encoding them using `get_dummies` would lead us to issues in terms of computational capacity. To reduce the dimension of data those modalities have 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 cases at `N2`. The choice of level is guided by a compromise between the amount of information and the reduction of modalities. `N1` make the merge robust while `N2` enable to keep diversity when it enhance the performance.

This enables us to keep consistency economically speaking, limiting risks linked to overfitting due to a overwhelming number of dimensions. We also reduced training time and facilitated the comparison of models.

4.2 Data Transformation

Categorical variables are encoded through `LabelEncoder` from `FeatureEncoder`. This class guarantees a consistent encoding between learning and test sets. Unobserved categories in the learning set are attributed to a known class enabling 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 (we used numerical labels to train the model, before transforming it with `inverse_transform` on the prediction to find back 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 requires more resources for our laptops. Either way, the encoding step is part of our pipeline as it is the case for the merging and cleaning stages.

5 Model Development

5.1 Benchmark Model

Several classification models 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 prediction stage, even better than Random Forest after being fine tuned. We chose our model consistently with the dataset, which includes a lot of data (categorical, but numerically encoded).

5.2 Hyperparameter Tuning

Model performance is 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 done 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 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 confirms the fact that XGBoost is our best model, hence our final one.

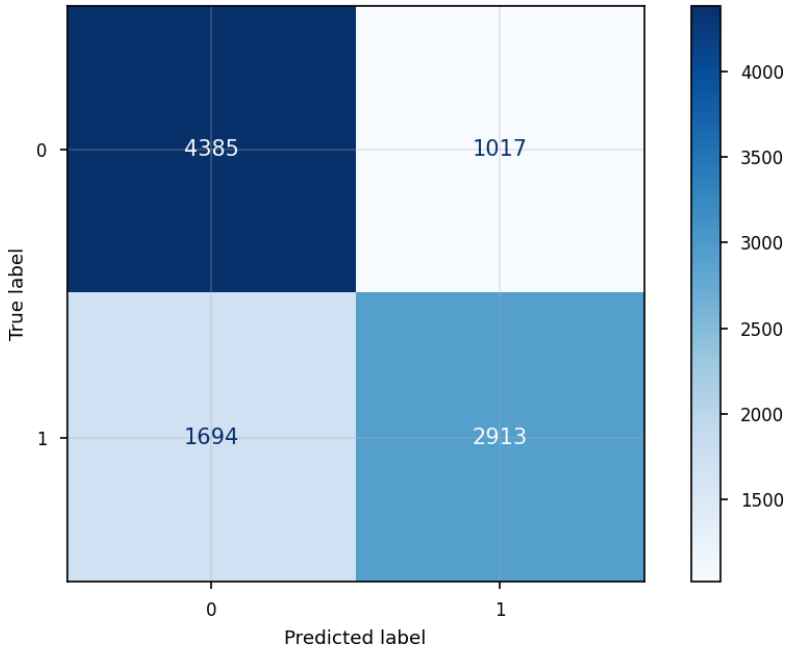

Figure 2: Confusion Matrix

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.

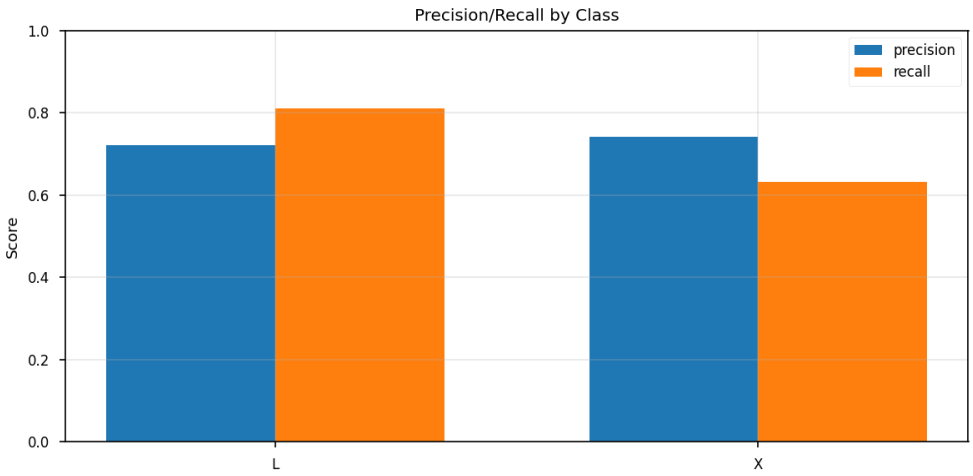

Figure 3: Prediction and Recall by class

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

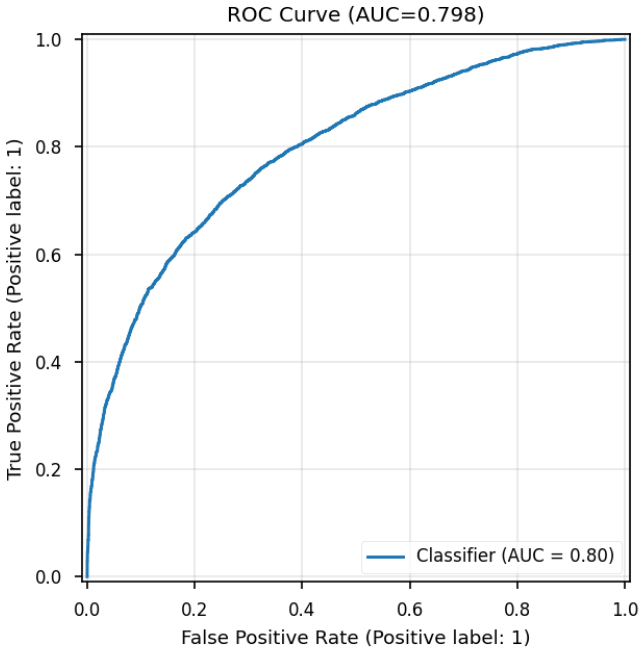

Figure 4: ROC Curve

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 confirms 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, sport tastes and geographical context enables 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. We used our final model to generate predictions on the test set. Those predictions have been exported in the file `predictions.csv`.