Census income
Report

Adrian Stańdo, Mateusz Stączek, Paweł Wojciechowski

20 April 2021

Introduction

In this task we were working on data about census income available here:
`https://www.apispreadsheets.com/datasets/106`.

There were 14 features telling different things about each individual such as age, sex, education, relationship, country of origin, number of hours spent working per week and a few others. The goal was to predict whether a person makes more than or less than $50k per year.

Some of the 48842 rows contained missing values in 3 columns, however less than 8% of records were missing in each such column. This dataset was unbalanced in terms of the predicted label since only 23% values in the target column were marked as people with income level above $50k. EDA is available here.

Preprocessing

First, missing values were imputed using *SimpleImputer* from *sklearn*. Then, redundant 3 columns were dropped (fnlwgt, education, because it was already encoded in education_num column, and relationship, because similar information was in maritial status column). Because of finely-grained origin feature, records were grouped by country of origin and divided into 2 groups: one with higher average income than the USA, other with lower. The average income was calculated from the available data therefore for rare countries the following results may not represent the reality well.

Finally, categorical columns were one-hot encoded. On the other hand, numerical columns didn't need any modifications because models chosen for hyperparameters tuning were tree-based only.

Fails

To keep this section short, we would like to describe one mistake we made during choosing final model. During preprocessing, last step was preparing data for training and validation (not testing) and included oversampling minority class. However models trained on training set with equal target class performed worse on unbalanced test set compared to models trained on unbalanced data. We believe that lower scores were caused by models overfitting and being trained to fit to balanced data.

Model selection

For each candidate model, GridSearchCV was run to find optimal hyperparameters. Candidate models were: Decision Tree Classifier, Random Forest Classifier and XGBoost Classifier. Different parameters where tested, these include: max depth of a tree, number of estimators and criterion (where applicable). Each GridSearchCV was run with 3-fold cross validation and scoring set to ROC AUC score.

After selection and evaluation phase, we checked feature importance and confusion matrix for every model. All 3 models selected 'being married' as the most important factor. The bigger part of other features is redundant because they had very low scores on feature importance plots.

Model	Precision	Recall	F1 Score
DecisionTreeClassifier	0.88	0.94	0.91
RandomForestClassifier	0.86	0.96	0.91
XGBoostClassifier	0.89	0.95	0.92

Tablica 1: Metrics scores for test set for $<= \$50k$ class.

Model	Precision	Recall	F1 Score
DecisionTreeClassifier	0.76	0.59	0.67
RandomForestClassifier	0.82	0.53	0.64
XGBoostClassifier	0.79	0.64	0.71

Tablica 2: Metrics scores for test set for $> \$50k$ class.

Model	For test set	For train set
DecisionTreeClassifier	0.90	0.91
RandomForestClassifier	0.91	0.93
XGBoostClassifier	0.93	0.94

Tablica 3: ROC AUC scores.

Summary

To conclude, we decided that XGBoost Classifier is the best model. It's overall scores were were slightly higher than others presented in the tables above. Another result is that very few features are required by the model to actually train it and the most important among them is marital status. Moreover, it turned out that the most important feature in each model was martial status.

One last important thing - parameters for the selected model are: max depth is 6, number of estimators is 50, objective is binary:logistic and the rest is default. As presented in the notebook, best version of every model is saved and ready to be loaded, and in this way parameters of other models can be found.