Introduction to machine learning: Census income classification

Maria Kałuska, Michał Komorowski, Marcelina Kurek
April 16, 2021

Abstract

The goal of this report is to show the results of trying to predict whether an individual's annual income is greater than $50 000 based on the census income dataset.

1 Introduction

First section of this report contains description of the data, which helps with the next step - preprocessing and feature engineering. In the next part, we compare a number of classifiers to find the best scoring ones and then tune their hyperparameters. In the last part, there is a summary of the best models that we could achieve.

2 Dataset

2.1 Description

The dataset contains 48842 records with 15 columns. Below, we shortly describe each one:

- **age** — positive integer, the age of an individual.
- **workclass** — string, describes employment status.
  Possible values: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- **fnlwgt** — positive integer, final sampling weight. Inverse of sampling fraction adjusted for non-response and over or under sampling of particular groups.
- **education** — string, education level of an individual.
  Possible values: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
- **education_num** — positive integer, same. as above but ordered ascending as numbers.
- **marital_status** — string, the current marital status of an individual.
  Possible values: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- **occupation** — string, describes the general field of work of an individual.
  Possible values: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial,

Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

- **relationship** — string, describes the relationship with an individual's family. Possible values: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- **race** — string, race of an individual. Possible values: White, Asian Pac Islander, Amer Indian Eskimo, Other, Black.
- **capital_gain** and **capital_loss** — integer, the capital gain/loss for and individual.
- **hours_per_week** — integer, total working hours per week for an individual.
- **native_country** — string, country of birth. Possible values: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
- **income_level** — string, predictor class if individual earns greater or less than $50000 per year. Possible values: <=50K, >50K.

2.2 Target variable distribution

The target variable describing income level is not evenly distributed. There is about 3 times more individuals that make no more than $50000 per year. We kept this ratio during training of the models. Additionally, the main metrics that we will use to compare the models will be accuracy and F1 score as it deals well with target imbalance.

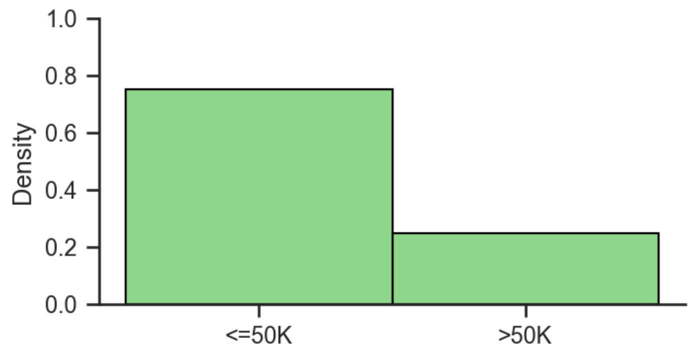

Figure 1: Target distribution

2.3 Missing values

Data contains missing values in some categorical features. They have been denoted as a question mark. In our assignment task, we were advised to reduce number of records, therefore we dropped all the rows containing at least one question mark. There were about 3000 of them, mainly in **workclass** and **native_country**,

3 Preprocessing

3.1 Columns to remove

We have decided to remove the following columns:

- **education** — the information about level of education is already present in column **education_num**
- **fnlwgt** — it does not have impact on a model performance, as it only contains a statistical parameter.

3.2 Data Encoding

We have transformed the following columns:

1. **age** - normalisation by using min-max scaler.
2. **native_country** - division into 5 categories:
   - United States
   - LA - (Latin America with Outlying-US),
   - ASIA
   - EU (Europe)
   - NA (North America)
3. **hours_per_week** - division into 3 categories (approximately half of the observations contain value 40 - distribution of this variable is not close to normal):
   - More than 40
   - Less than 40
   - Exactly 40

We have encoded the following columns by using **OneHotEncoder**:

- **workclass**
- **marital_status**
- **occupation**
- **relationship**
- **hours_per_week**
- **native_country**

The column 'race' was encoded with **TargetEncoder**.

We also considered applying Target Encoding to all categorical columns, but it slightly affected model performance. The table below shows the model performances with categorical features encoded by OneHotEncoder and TargetEncoder (on 5 fold cross validation on training dataset).

	recall	accuracy	roc auc	f1
AdaBoost (OneHotEncoded)	0.654318	0.870327	0.927199	0.714104
AdaBoost (TargetEncoded)	0.650030	0.869943	0.926928	0.712162
CatBoost (OneHotEncoded)	0.659321	0.870828	0.928343	0.716426
CatBoost (TargetEncoded)	0.655390	0.870150	0.927932	0.714155
XGBoost (OneHotEncoded)	0.664800	0.870209	0.927807	0.717162
XGBoost (TargetEncoded)	0.661584	0.870356	0.928151	0.716380

4 Model selection

We have tested many models with average accuracy above 84%. Below we describe all the tested models.

4.1 Tested models' scores

	recall	accuracy	roc auc	f1
CatBoost	0.656049	0.870196	0.928526	0.714705
XGBoost	0.660126	0.869590	0.927138	0.715033
Gradient Boosting	0.650013	0.869118	0.925858	0.711144
AdaBoost	0.648734	0.868692	0.927065	0.710042
Random Forest	0.518023	0.850427	0.903807	0.631938
Voting Classifier	0.557579	0.848891	0.905078	0.646528
Logistic Regression	0.605460	0.848813	0.903840	0.665006

From listed models, we have chosen CatBoost, XGBoost for further analysis.

4.2 XGBoost

First XGBClassifier had the parameters:
learning_rate=0.4, booster='gbtree', max_depth=4 eval_metric="logloss", use_label_encoder=False.

Next we tried to tune the hyperparameters with GridSearch. The tested parameters were: gamma, booster, max_depth, learning_rate, min_child_weight and eval_metric.
The best possible model had the following parameters: learning_rate=0.4, booster='gbtree', max_depth=3, eval_metric="rmse", use_label_encoder=False, gamma=0.5, min_child_weight=1.

Then we moved on to feature selection. Starting with SelectKBest algorithm.

1. At first only 30 features were left, but it slightly worsened the performance.
2. Leaving 40 features had similar result.
3. Leaving 46 features had almost no negative result.

Furthermore we decided to use Recursive Feature Elimination before dropping any columns.

1. Our starting point was leaving 40 features and similarly to SelectKBest dropping these columns did not improve the performance.
2. Leaving 45 features made almost no difference.
3. Next we selected 47 features and after dropping the remainder we obtained a slightly better result.

	test accuracy
XGBoost initial	0.8691
XGBoost best	0.8712
XGBoost best 47 RFE	0.8712
XGBoost best 45 RFE	0.8710

To sum up our final selection was XGBoost with parameters mentioned above and with 47 features left. We dropped columns: sex_Female and occupation_Armed-Forces

4.3 CatBoost

Initial CatBoost model had the parameters set to: learning_rate=0.04, depth=6.

Optimization performed with Grid Search verified the following parameters: l2_leaf_reg, min_data_in_leaf, grow_policy, learning_rate and depth.
The best model had the hyperparameters: loss_function='Logloss', silent=True, depth=3, grow_policy='Depthwise', l2_leaf_reg=2, learning_rate=0.04, min_data_in_leaf=2.

Then, the significance of the features was verified. We dropped 5 columns selected by SelectKBest, but the result worsened a little. Therefore we chose not to drop any features for this model.

	test accuracy
CatBoost initial	0.8709
CatBoost best	0.8710
CatBoost best 44 SKB	0.8698

5 Final model

Having compared **XGBoost** and **CatBoost** models, it is visible that the differences are negligible. Probably on different combination of train data set and test data set, the performances would change slightly. As mentioned before, the main metrics used to compare models were accuracy and F1 score. **CatBoost** did the best according to F1 Score, but apart from **XGBoost** with 30 best scoring features, the scores are basically the same so most of the models on graph below are good. Taking into account accuracy **XGBoost** and **CatBoost** have roughly the same scores. The decisive factor apart from metrics was model efficiency. Hence we have chosen **XGBoost best 47 RFE**, which is faster than CatBoost.

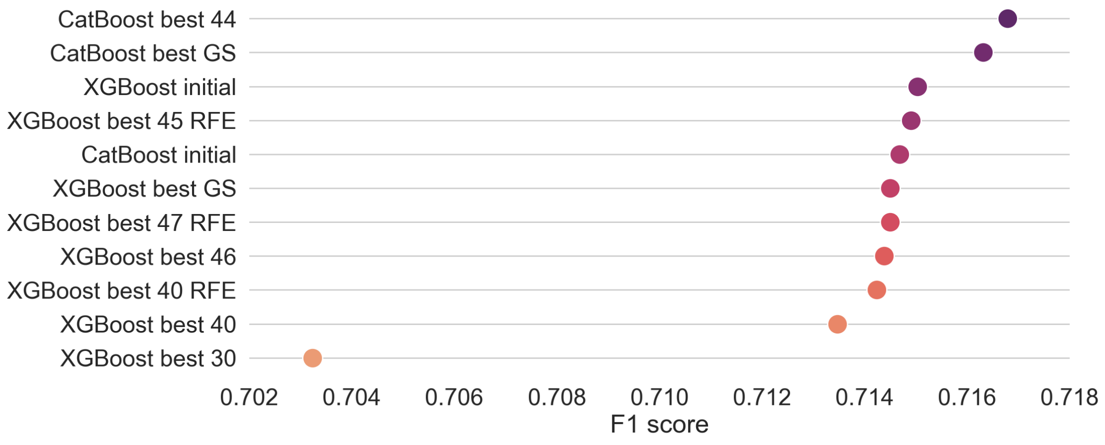

Figure 2: F1 score comparision