Adult Income Prediction

Exploratory Data Analysis Project

Contents

1. Team Members
2. Introduction
3. Data preprocessing
4. Models and fine-tuning

4.1 Logistic Regression

4.2 Random Forest

4.3 SVM

4.4 XGBoost

1. Results
2. Team Members

This project is made by *Atudore Darius*, *Bouruc Liviu*, *Frâncu Richard* and *Tender Laura*. All of us follow the Data Science Master’s degree (411 group).

1. Introduction

The purpose of this project is to train and fine-tune four different models that can predict if an adult earns more than 50K per year given some characteristics. So, we try to solve a binary classification problem.

We use the “Adult Data Set”. For each adult we have the following list of 14 attributes: age, workclass, fnlwgt, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, native country.

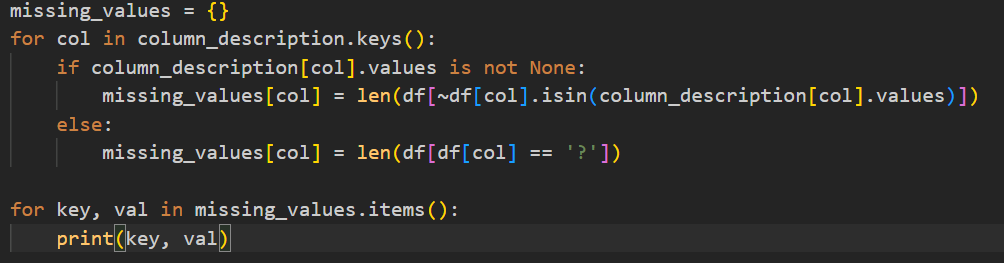
Our data is split in a training and a testing dataset. The training dataset has around 32K records, while the test dataset has around 16K. From our processed dataset 75.11% people win less than 50K while 24.89% more. So we are working with an unbalanced dataset.

Below we can see more details about each feature - if it is continuous or a category:

* age: continuous.
* workclass: 8 values (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked).
* fnlwgt: continuous.
* education: 16 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: continuous.
* marital-status: 7 values (Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse).
* occupation: 14 values (Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces).
* relationship: 6 values (Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried).
* race: 5 values (White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black).
* sex: binary (Female, Male).
* capital-gain: continuous.
* capital-loss: continuous.
* hours-per-week: continuous.
* native-country: 41 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, Trinidad Tobago, Peru, Hong, Holland-Netherlands).

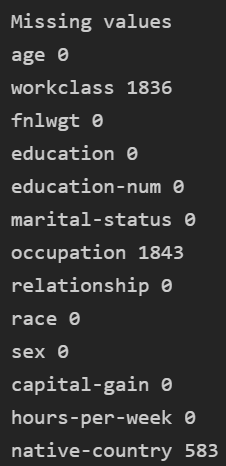
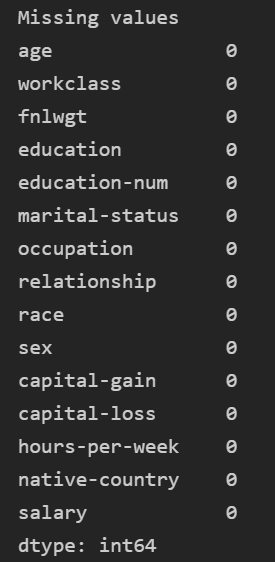
1. Data preprocessing

3.1 Data Cleaning

First we look at missing values using pandas built in function isnull(). As we can see there are no NaN values in the dataset. If we look into the dataset we can observe that there are many ‘?’ characters, which probably represent the Nan values, so we use this sign to check for missing values, and also we use the set of possible values for each column, listed in the dataset description, to clean the dataset.

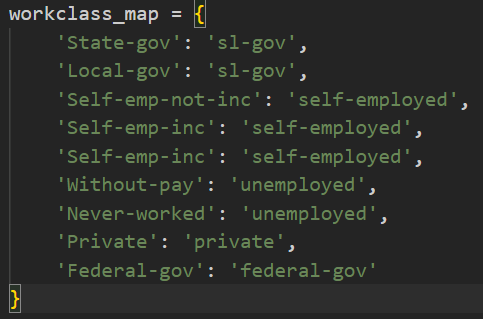
Because there are not so many rows with missing/inconsistent values we decided to remove those row:

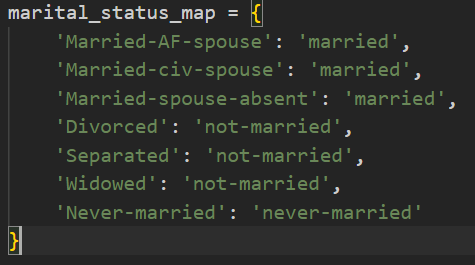
* Initial\_dataset\_size: 32561
* Cleaned\_dataset\_size: 30162



3.2 Data Processing

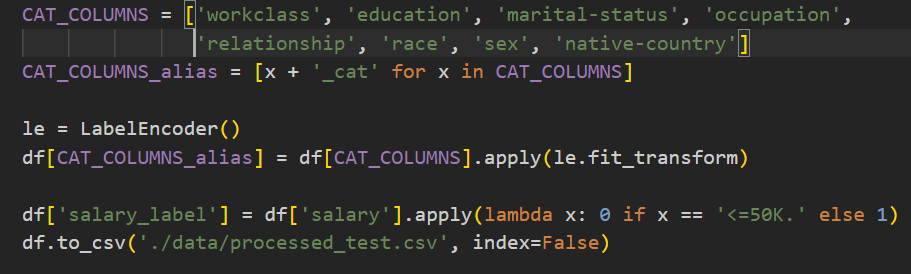
1. Column Values Mapping

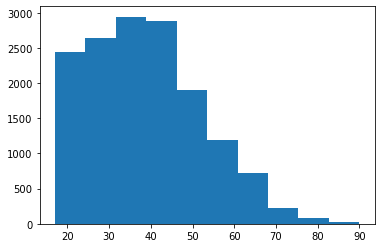
Work class  


Marital status  


Native country   


1. String Column Categorizer, because we can’t use string directly as input to the model we have to transform them to some categorical variables



1. Numerical Column Categorizer, because a high number of values can be confusing for the model we decide to create some new columns, where we map numerical columns to intervals, for example, for age, we divide the age axis into 10 intervals, according to this histogram.  
   
2. Models and fine-tuning

4.1 Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. It is used to predict a binary outcome (1/0, Yes/No, True/False).

We used Logistic Regression from Sklearn library. Sklearn makes available for us a large set of parameters that can be used to improve our training, we restrict our work only to two parameters:

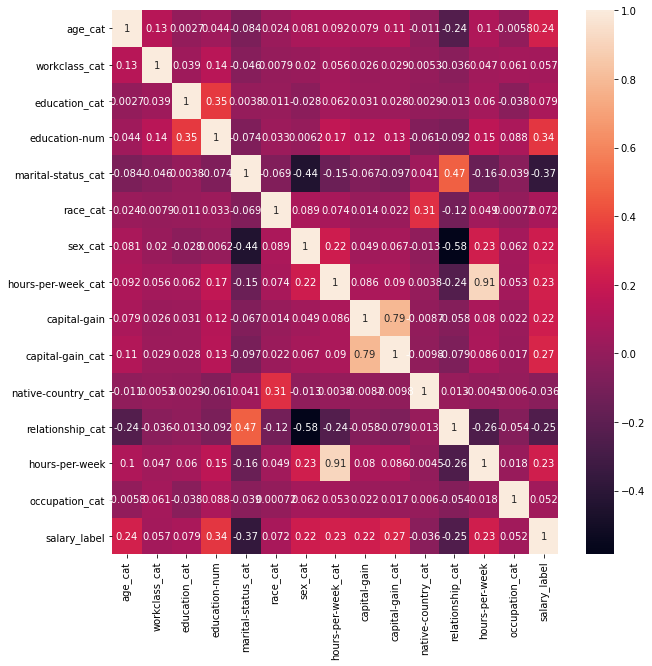
* C - the 'C' parameter is the inverse of regularization strength. Smaller values of C specify stronger regularization. Regularization is used to prevent overfitting, by adding a penalty term to the loss function.
* penalty: - the 'penalty' parameter specifies the type of regularization used in the model. The options are 'l1', 'l2', 'elasticnet' and 'none'. 'l1' penalty is L1 regularization, 'l2' penalty is L2 regularization and 'elasticnet' is a combination of both.

For this project we also used:

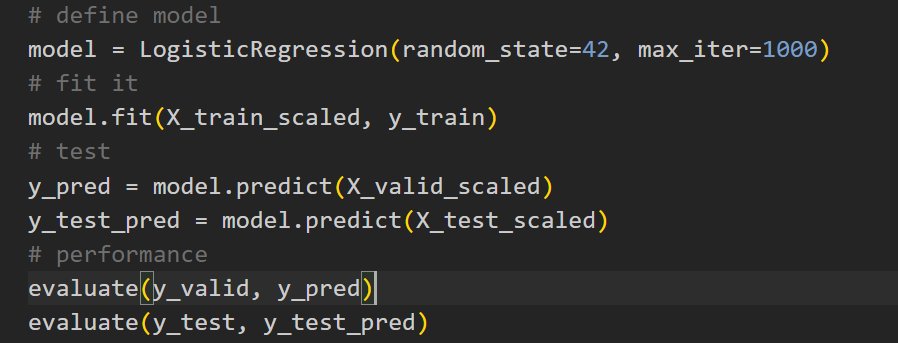
* GridSeachCV (sklearn) - function in the sklearn that is used to tune hyperparameters for a model by training the model multiple times with different combinations of parameter values. It takes a model, a set of parameter values to try, and a scoring method, and returns the best set of parameters as determined by the scoring method.
* StandardScaler (sklearn) - preprocessing method in sklearn that standardizes a feature by subtracting the mean and scaling to unit variance. It is used to transform the feature so that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one

4.1.1 Baseline Model

For the first experiment we used almost all of the features, categorized and not categorized. As we can see in the correlation matrix shown below, in the last row, some of these features are more correlated with the salary label that we have to predict, and some of them have a very low correlation score. In further steps we will use this correlation matrix along with other techniques to select a subset of features. Also, we need to specify that for a correct interpretation we have to consider the absolute value when selecting the highest correlation scores.



Given these features, we simply train our first model and we obtain the baseline.



And the results:

|  | Accuracy | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| Train | 83% | 53% | 72% | 61% |
| Test | 83% | 53% | 71% | 60% |

We have a very high accuracy score, we can be fooled into thinking that we already obtained a very good model, but this is not the case because of the class imbalancing, so, to have a better evaluation we introduced recall, precision and F1 Score which are more suited for this type of datasets because the take into account the False Positives as well as False Negatives.

4.1.2 Data Resampling

Data resampling is the process of manipulating the training data in a machine learning model to improve the model's performance. There are two main types of data resampling: oversampling and undersampling.

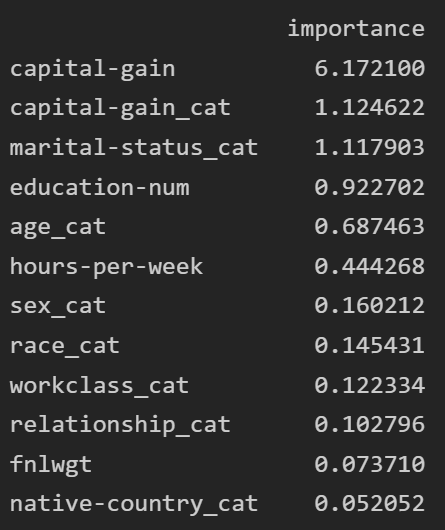
For this project we choose oversampling which involves increasing the number of instances in the minority class in the training data, while undersampling involves reducing the number of instances in the majority class.

To solve our problem we use a very useful class from imblearn library, called SMOTE (Synthetic Minority Over-sampling Technique). The technique works by synthesizing new minority instances between existing minority instances, rather than simply duplicating existing instances. SMOTE selects two or more similar minority instances and interpolates new synthetic examples along the line segments joining the selected examples.

|  | Accuracy | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| Train | 79% | 82% | 56% | 67% |
| Test | 79% | 82% | 54% | 65% |

Here, even though the accuracy has decreased a bit, we can see that the f1 score has improved, so we can conclude that oversampling is effective in our case.

4.1.3 Feature Selection



In logistic regression, the coefficient (coef) method is one way to perform feature selection. The coefficient of a feature represents the change in the log-odds of the outcome for a one-unit change in the feature, holding all other features constant. A larger coefficient indicates that the feature is more strongly associated with the outcome, and a smaller coefficient indicates that the feature is less strongly associated with the outcome.

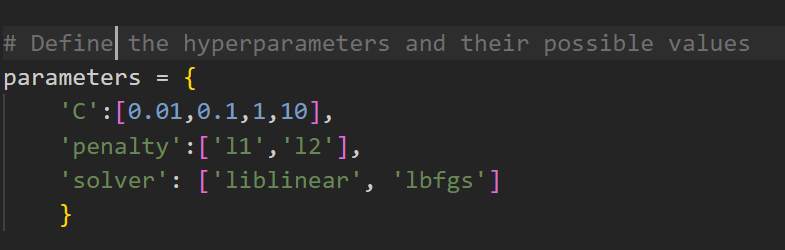
It's worth noting that this method only works well with a relatively small number of features, and it's not suitable for cases where there are many correlated features or when the number of features is large compared to the number of instances. Also, this method only considers linear association of features with the target variable.

In the picture shown we can see the importance of each feature. Further we took only the features with an importance bigger than 0.1 and retrained the model.

|  | Accuracy | Recall | Precision | F1 Score |
| --- | --- | --- | --- | --- |
| Train | 79% | 81% | 56% | 66% |
| Test | 79% | 82% | 54% | 65% |

4.1.4 Fine Tuning

To fine tune the model we used the GridSearchCV method, described in the beginning, with the following set of parameters but unfortunately we didn’t obtain better results.



4.2 Random Forest

In this project, we used Random Forest as a classification model. It consists of many decision trees whose low correlated predictions as a whole are more accurate than only one tree’s. A decision tree is an algorithm in which each node splits the data by a feature and each leaf node represents a class label.

We used Random Forest from the Sklearn library using two parameters: n\_estimators and max\_features. n\_estimators represents the number of trees in the forest (we tried two values: 1000 and 250 and the results were not very different). max\_features is the number of features to consider when looking for the best split.

The main approaches for this model were:

* trying more values for max\_features;
* removing less important features of the training data;
* training our models using both continuous and category features.

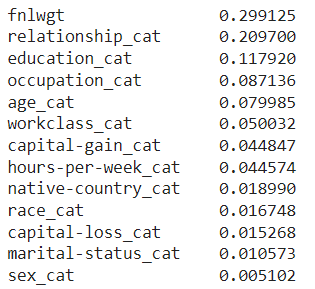
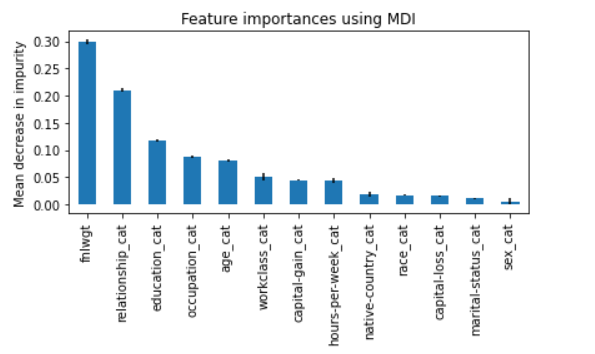
Our first approach was choosing all 14 attributes, 13 of them as categories. We used fnlwgt as continuous.

Bag = RandomForestClassifier(n\_estimators = 1000, max\_features = 13)

Bag = Bag.fit(X\_train, y\_train)

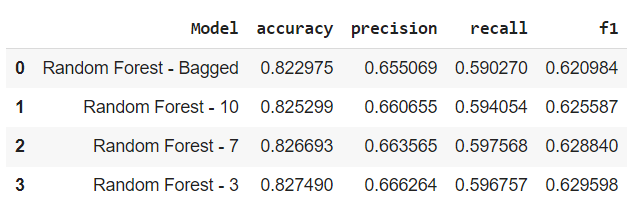
importances = Bag.feature\_importances\_

forest\_importances = pd.Series(importances, index = columns\_for\_training)

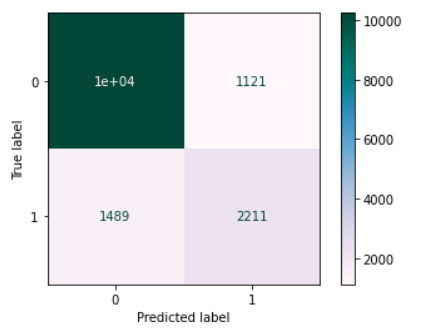


We notice that the most important feature is fnlwgt (which is the only continuous one). Only three features have more than 0.1. while six features have less than 0.05. These results encouraged me to replace the categories with continuous values where possible.

Main results of this approach:



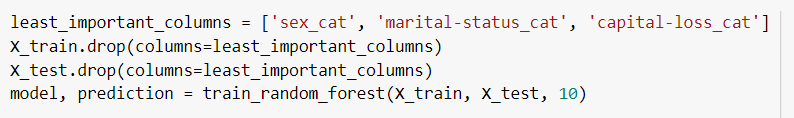
We notice that the results are similar, we’ll have a look at the confusion matrix for Random Forest - 7.



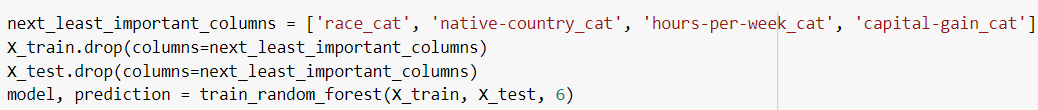
Out of 11360 people who earn less than 50K, 10239 (90,13%) were predicted correctly. Out of 3700 people who earn more, 2211 (59.75%) were predicted correctly.

I tried two other approaches using these columns: removing the top 3 less important features and keeping the top 6 most important ones.

First approach



Second approach



These were the results of the bagged random forests:

| Model | Accuracy | Precision | Recall | F1 Score |
| --- | --- | --- | --- | --- |
| 13 columns | 0.82 | 0.65 | 0.59 | 0.62 |
| 10 columns | 0.82 | 0.66 | 0.59 | 0.62 |
| 6 columns | 0.83 | 0.66 | 0.60 | 0.63 |

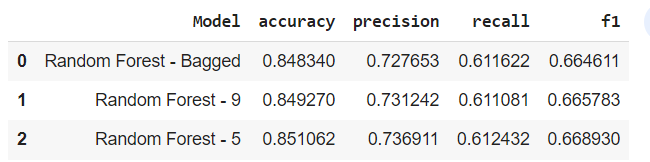
Looking at the result we can notice no significant improvement.

I decided training the model with different columns, using continuous values were possible:

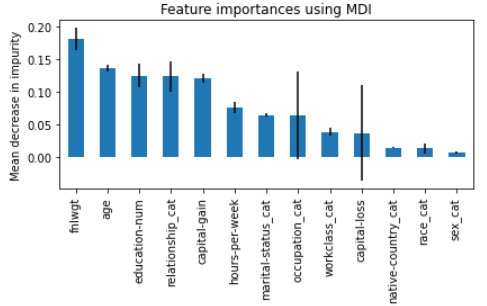
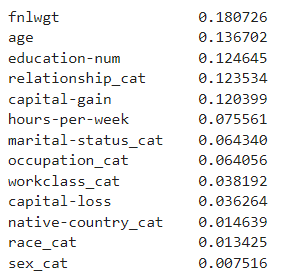
continuous\_columns\_for\_training = ['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']

category\_columns\_for\_training = ['workclass\_cat', 'marital-status\_cat', 'occupation\_cat', 'relationship\_cat', 'race\_cat', 'sex\_cat', 'native-country\_cat']

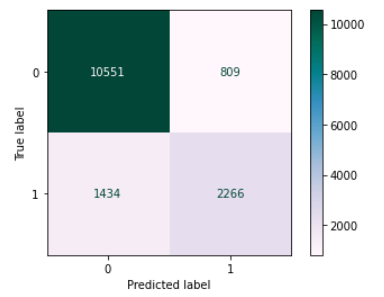
columns\_for\_training = continuous\_columns\_for\_training + category\_columns\_for\_training



In this case we can notice an improvement in our scores. Let’s have a look at the feature importances and confusion matrix for the best model (Random Forest - 5).

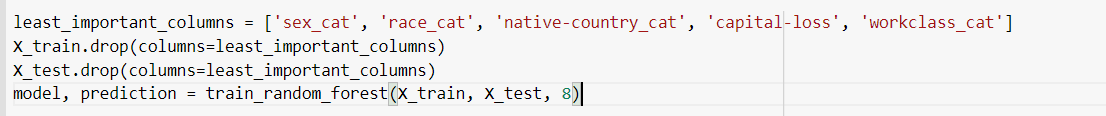
 

Comparing these with the previous ones we can notice a decrease in the importance of fnlwgt (from 0.29 to 0.18), while age, education and capital gain increased significantly.



Out of 11360 people who earn less than 50K, 10551 (92,87%) were predicted correctly. Out of 3700 people who earn more, 2266 (61.24%) were predicted correctly.

We can notice that 5 features are less important than 0.05 so we removed them and trained a new model.



In the results below we can notice the scores did not improve.

| Model | Accuracy | Precision | Recall | F1 Score |
| --- | --- | --- | --- | --- |
| Best model | 0.85 | 0.74 | 0.61 | 0.67 |
| 5 columns removed - Bagged | 0.85 | 0.73 | 0.61 | 0.66 |
| 5 columns removed - 5 max\_features | 0.85 | 0.74 | 0.61 | 0.67 |

4.3 SVM

Another machine learning classification model we used is SVM (Support Vector Machines). This is a binary classifier which tries to build a straight delimitation with two margins between two classes as such it tries to maximize the number of data points correctly classified. For the actual implementation, I used the SVC classifier from the SVM module in sklearn.

First of all, to solve our problem we had to make preprocessing operations on the dataset. On this matter, I used the *get\_dummies* method from pandas. It takes categorical variables and expands it into the table dataset as columns. Corresponding rows with that property will have 0 or 1 to the new columns. Making different experiments showed me that using this method the metrics used for evaluating the model improved.

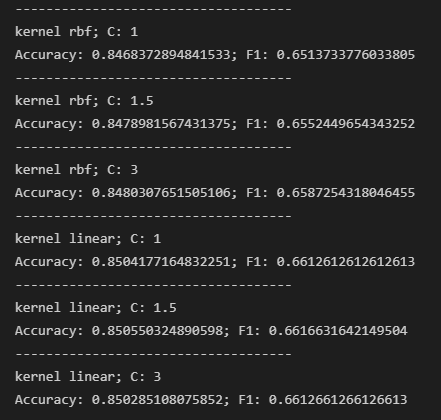


For the training itself, I used a tool named GridSearchCV which lets me make a dictionary with all the hyperparameters I want to test and it runs the model with every combination of them. It also has built-in cross-validation. Hyperparameters tested:

• Regularization parameter *C*: tells the model how much it can wrongly classify data; a large C reduces the learned margins (can lead to overfitting) / a small C is more permissive (can lead to underfitting)

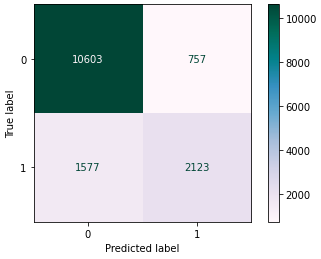
• *kernel* parameter which represents the type of kernel the model uses, his role being to project the data to linearly separable feature spaces

• *gamma* parameter: coefficient for kernel

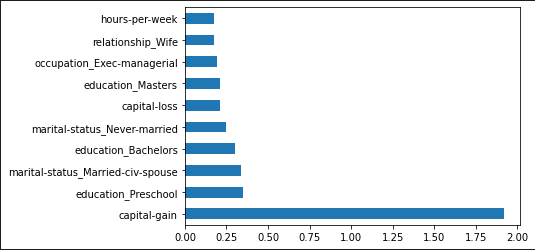


Because the dataset is very imbalanced, the accuracy isn’t the best metric to guide on, so we took into account the F1 Score. Analyzing the best result from above (kernel linear and C 1.5), we can observe the following:

Confusion matrix



Feature importance



4.4 XGBoost

One of the models used in this project is the XGBoost. Also known as Extreme Gradient Boosting algorithm, it is a decision tree based machine learning algorithm which uses a process called boosting to help improve performance. Since its introduction, it’s become one of the most effective machine learning algorithms and regularly produces results that outperform most other algorithms, such as logistic regression, the random forest model and regular decision trees.

We chose to work with the XGBoost library and its implementation of the XGBClassifier.XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on a major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

The main approaches of using this model were:

* getting a benchmark using as many features as possible
* removing some ot the less important features from the training data;
* training our models using both continuous and category features.

The first approach was to use all of the attributes, specifically 14 of them:

* 'age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week' as continuous values
* 'workclass\_cat', 'marital-status\_cat', 'occupation\_cat', 'relationship\_cat', 'race\_cat', 'sex\_cat', 'native-country\_cat' as categories

Our data contained 32 thousand entries for training, and 16 thousand for testing.

X\_train = train\_data[columns\_for\_training]

y\_train = train\_data['salary\_label']

X\_test = test\_data[columns\_for\_training]

y\_test = test\_data['salary\_label']

Then we took a look at all of the different features and their corresponding importance values.

importances = model.feature\_importances\_

importances = pd.Series(importances, index = columns\_for\_training)

==== IMPORTANCES ====

age 0.032329

fnlwgt 0.015305

education-num 0.108458

capital-gain 0.157564

capital-loss 0.070002

hours-per-week 0.030752

workclass\_cat 0.022224

marital-status\_cat 0.099085

occupation\_cat 0.031554

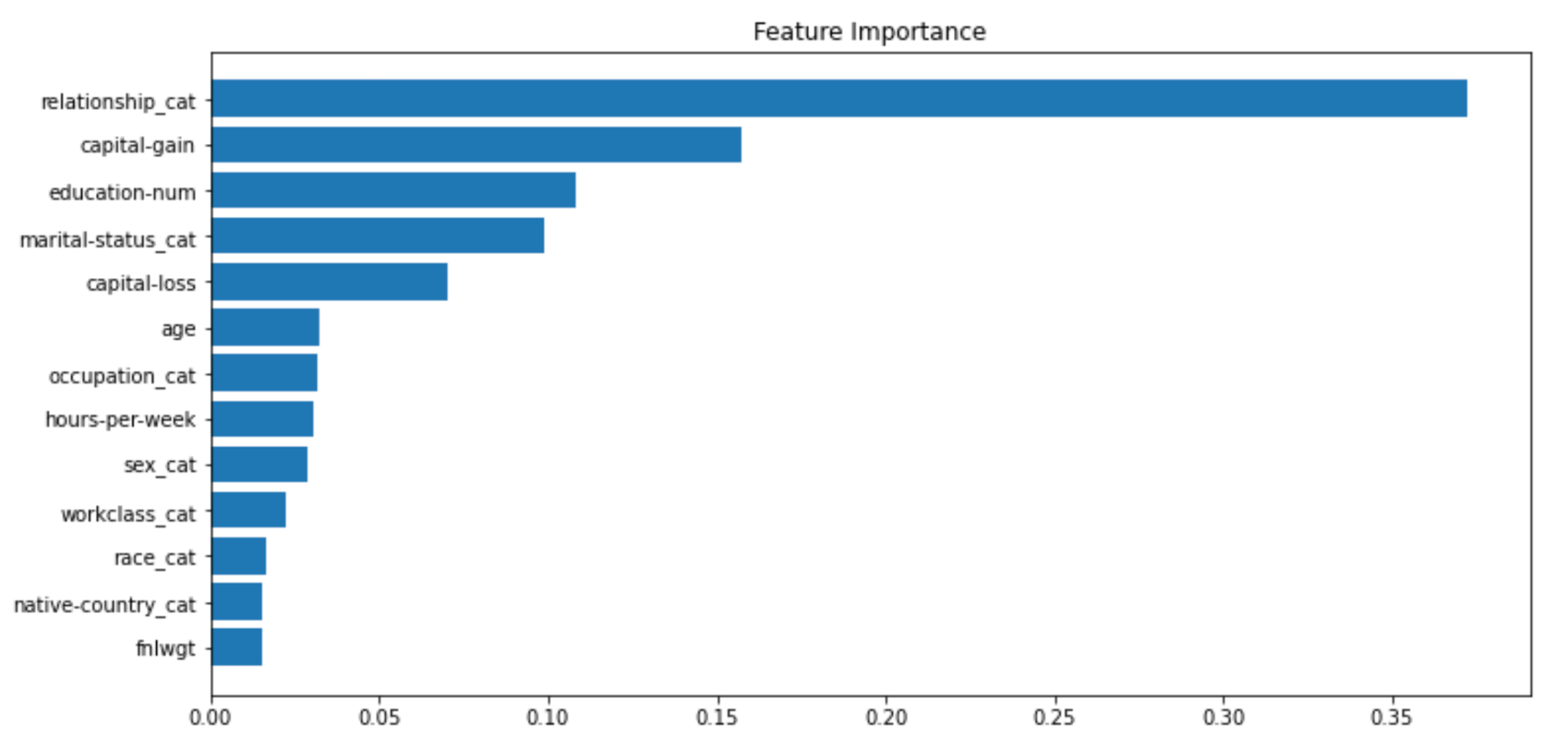
relationship\_cat 0.372461

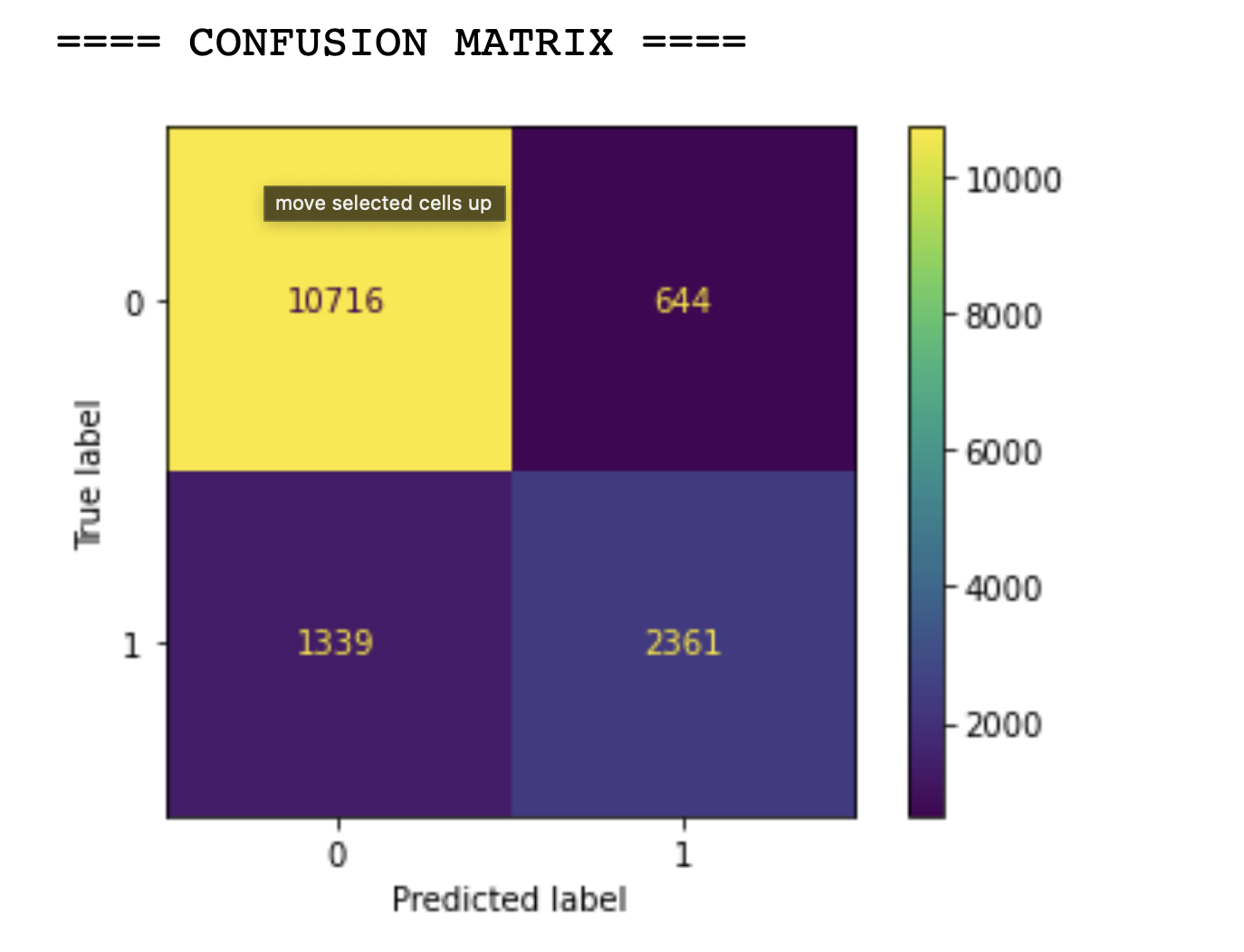
race\_cat 0.016256

sex\_cat 0.028634

native-country\_cat 0.015375

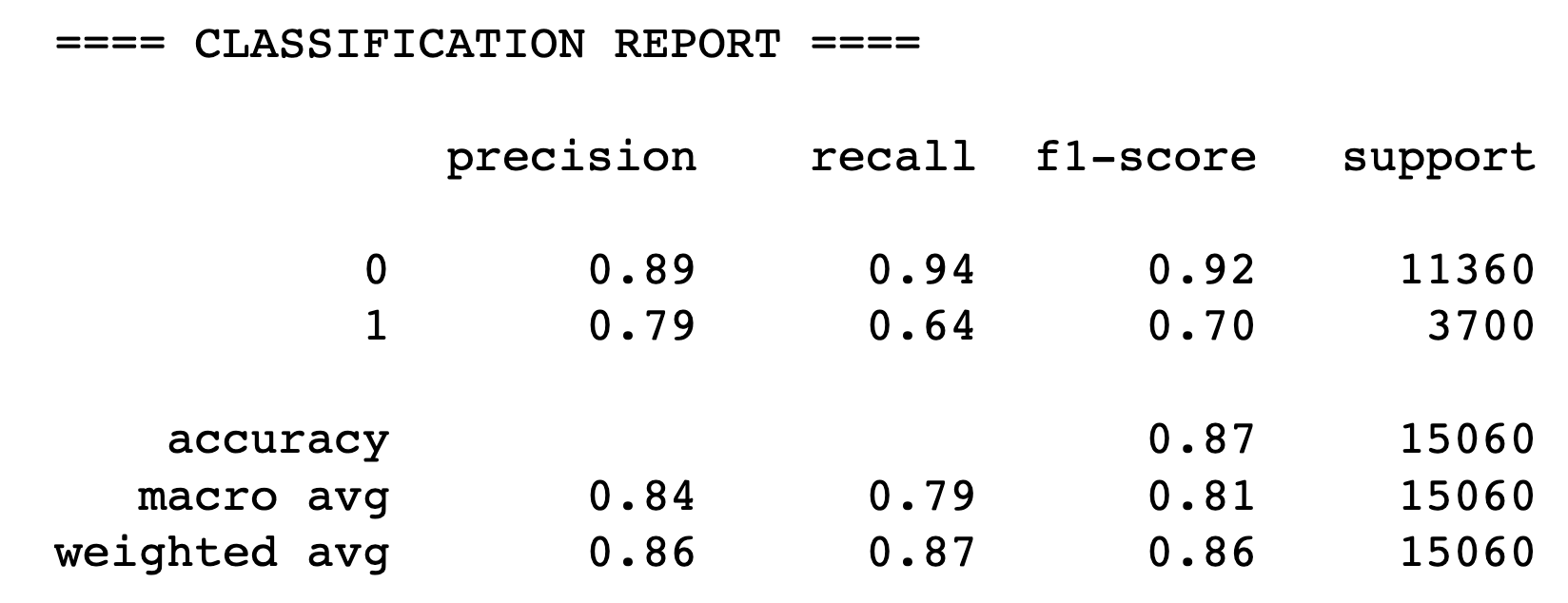
We notice that the most important features are the relationship category, the capital gain value and the education num. All three of these values are >= 1.0. The remaining 11 features have a value smaller than that.



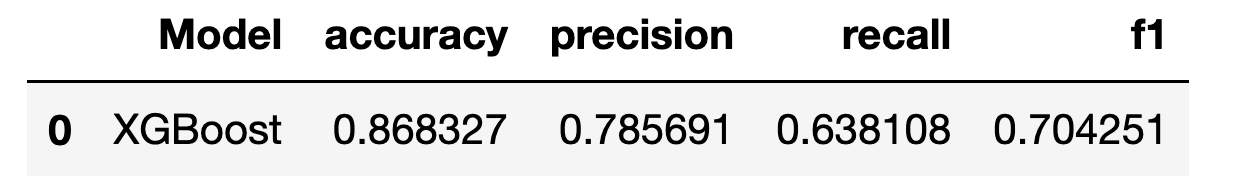


The confusion matrix of this model shows that out of the 11360 people that earn less than 50K/year, 10716 were predicted correctly, that means that 94.3% of the values were predicted correctly. And of the people who earn more, 2361 of the total 3700 people were predicted correctly (63.8%).

The classification report of this approach is the following:



We can see that the f1 accuracy score obtained was 0.704.



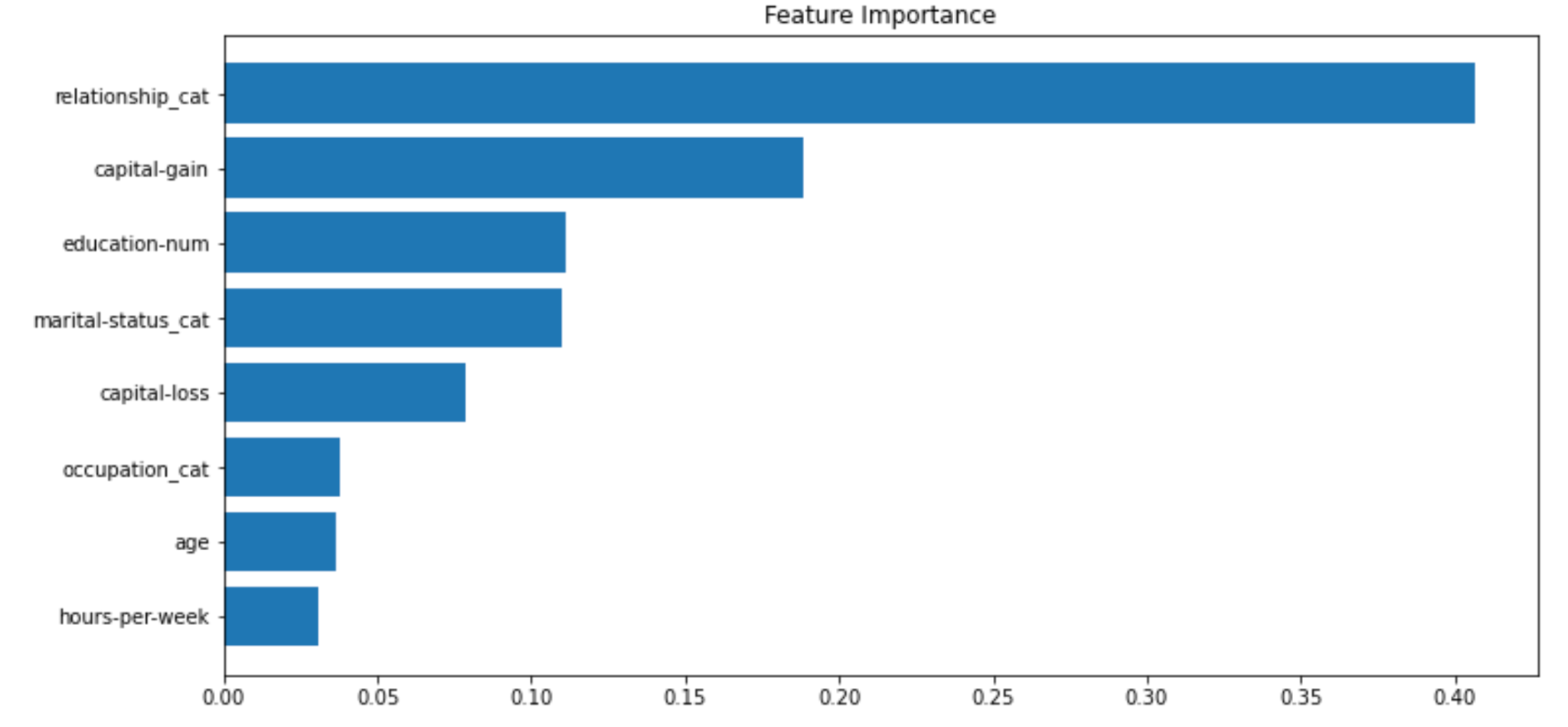
The next run of the model was made on training data that had the lowest rated features removed from the dataset (mainly the ones that were smaller than 0.03).

continuous\_columns\_for\_training = ['age', 'education-num', 'capital-gain', 'hours-per-week', 'capital-loss']

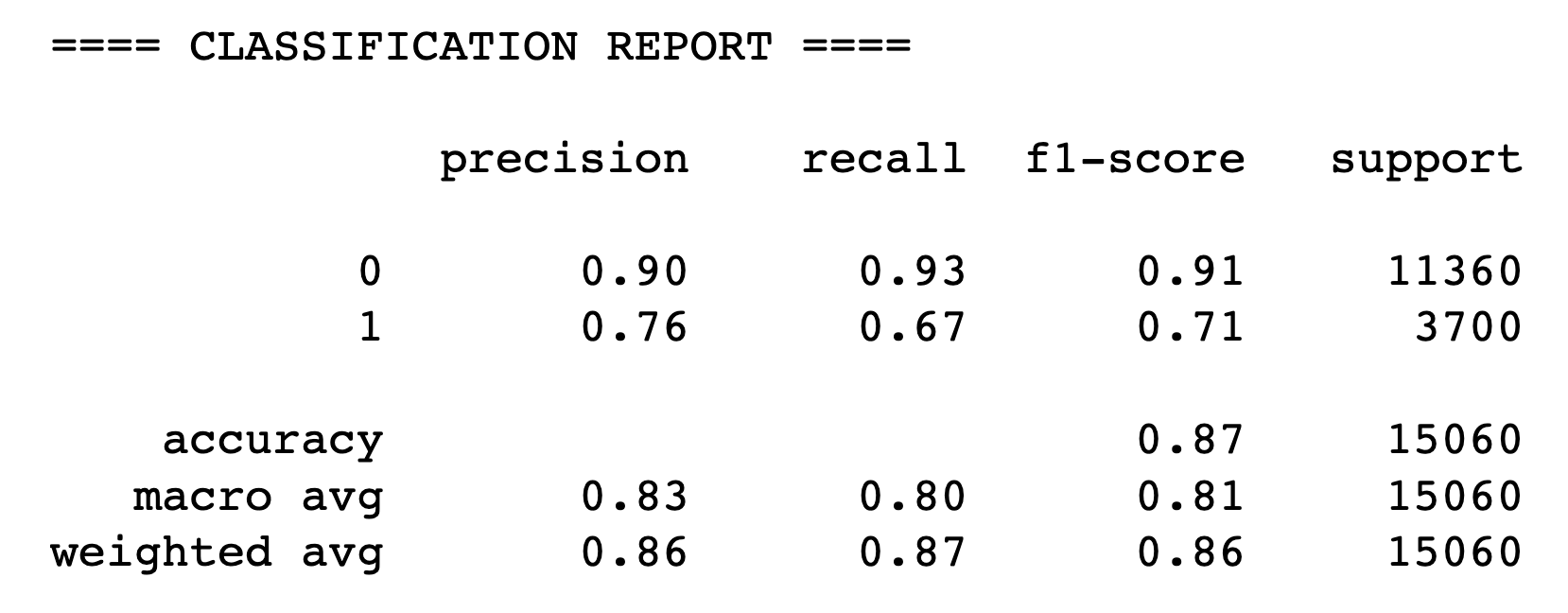
category\_columns\_for\_training = ['marital-status\_cat', 'occupation\_cat', 'relationship\_cat']

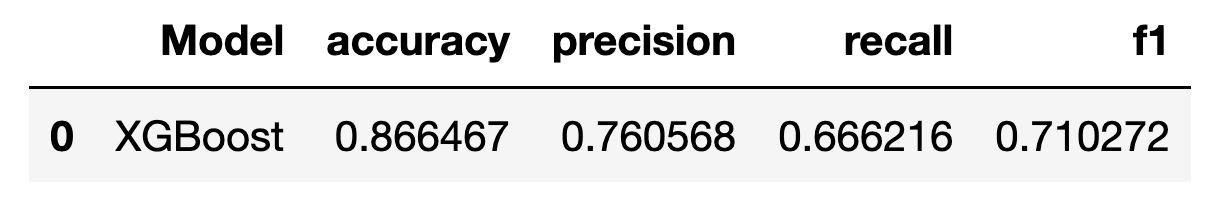
columns\_for\_training = continuous\_columns\_for\_training + category\_columns\_for\_training

Leaving us with 8 total features.

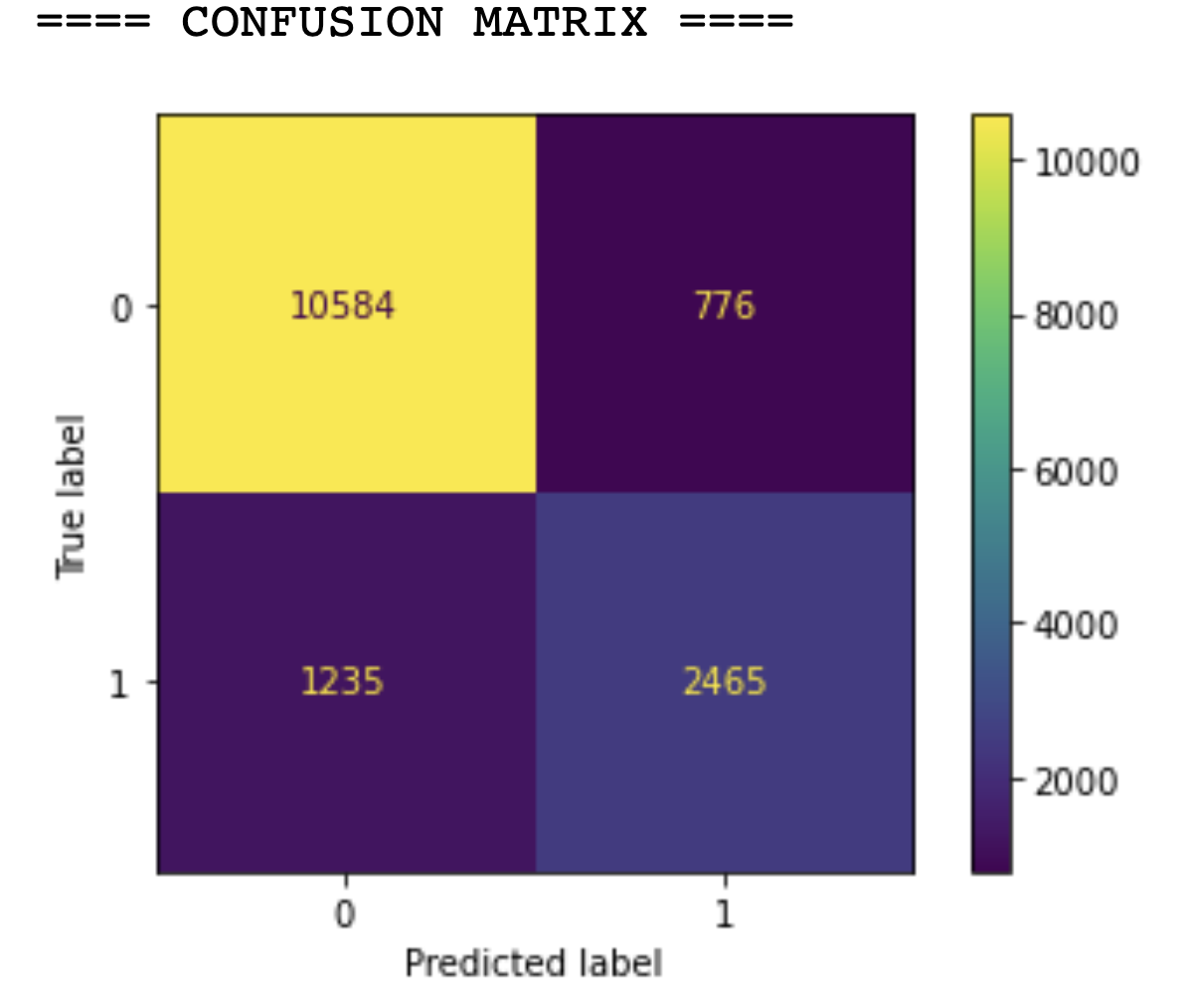


The most important features remained relationship, category, capital gain and education num.





As we can see, the results were expected. This approach leaving as with a slightly better f1 score than before (71.0%, before obtaining 70.4%)



From the confusion matrix of this model we can see that 10584 of the people that had a salary smaller than 50K were predicted correctly (93.1%). And 2465 of the 3700 of the people that earn over 50k were predicted correctly (66.6%).

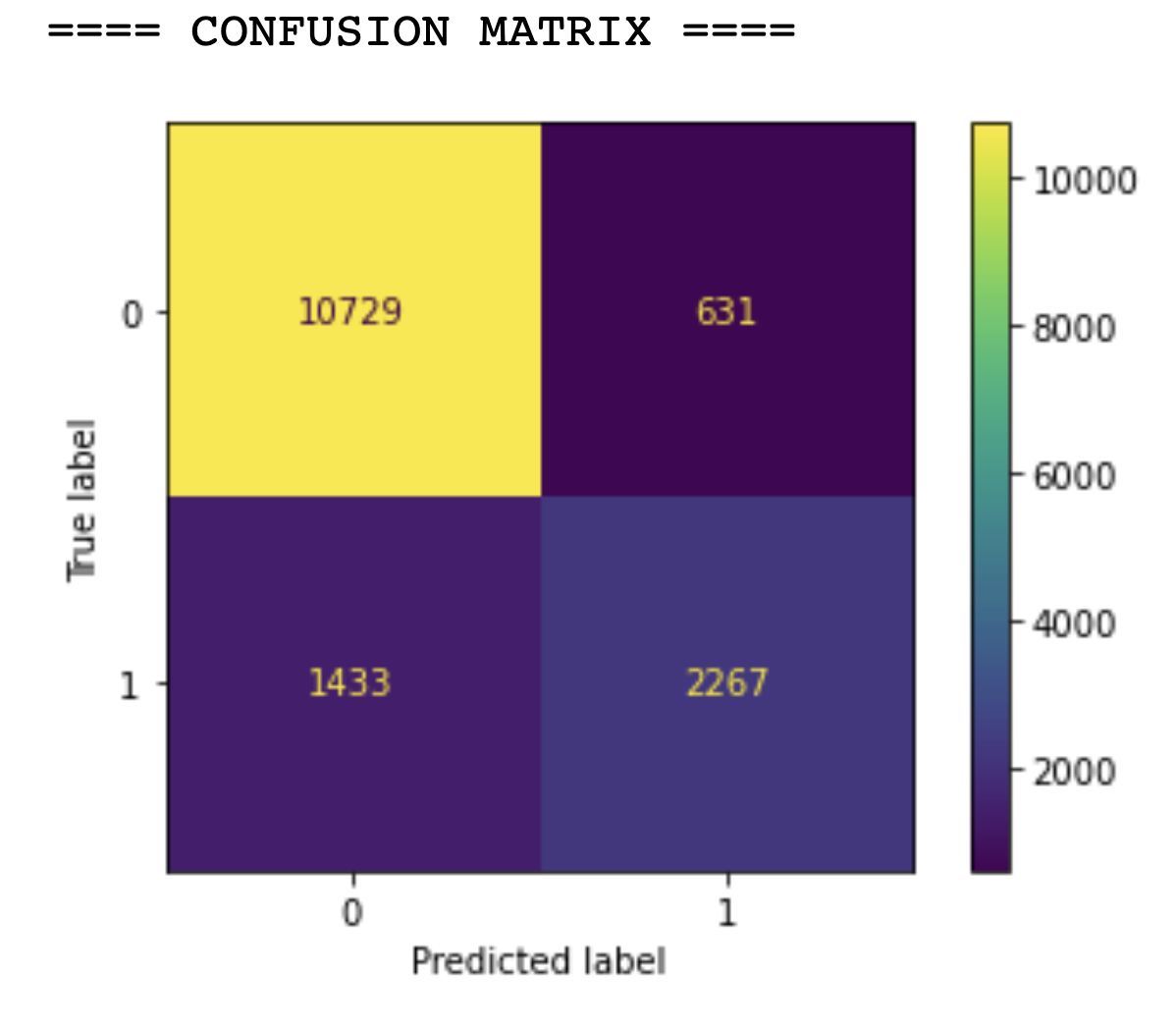
The third approach that we used with XGBoost was to remove even more of the features. Mainly the ones that had an importance lower than 0.07. Leaving us with 6 total features.

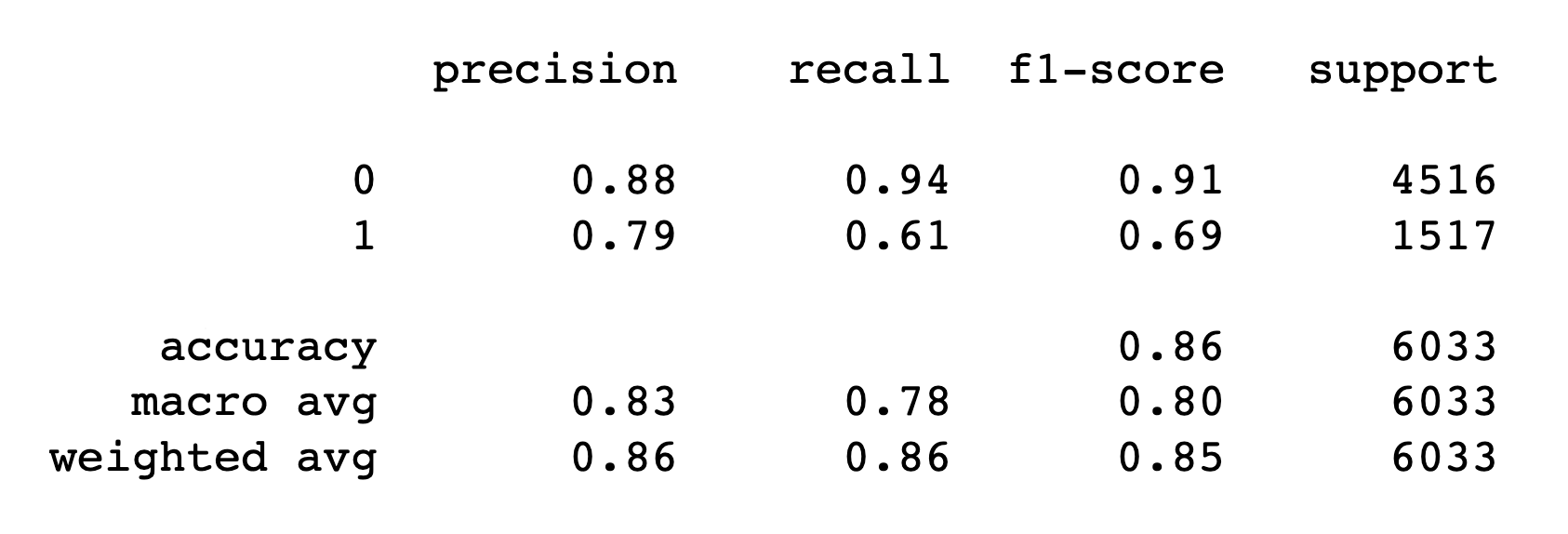
continuous\_columns\_for\_training = ['education-num', 'capital-gain', 'capital-loss']

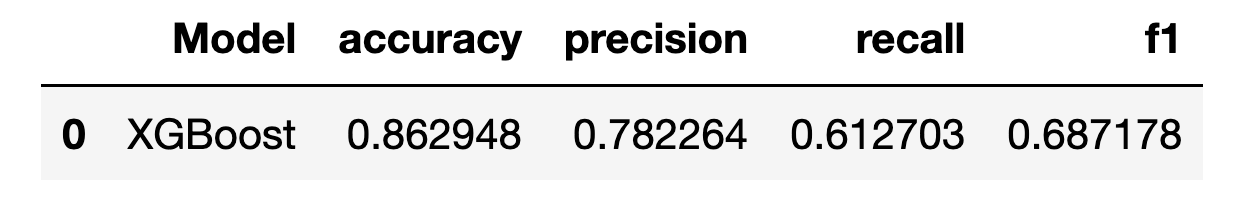
category\_columns\_for\_training = ['marital-status\_cat', 'occupation\_cat', 'relationship\_cat']

columns\_for\_training = continuous\_columns\_for\_training + category\_columns\_for\_training

The results of this approach being even worse than the previous one, as we can see from the plots listed below:







The results obtained by the three models are the following:

| **Nr of Features** | **Accuracy** | **Precision** | **Recall** | **F1** |
| --- | --- | --- | --- | --- |
| **13** | 0.868327 | 0.785691 | 0.638108 | 0.704251 |
| **8** | 0.866467 | 0.760568 | 0.666216 | 0.710272 |
| **6** | 0.862948 | 0.782264 | 0.612703 | 0.687178 |

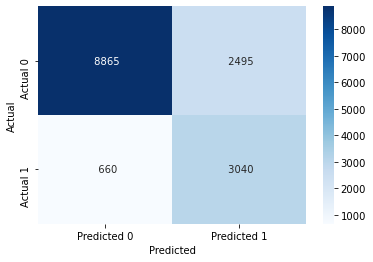
1. Results

Summary of our best models’ results on the test data:

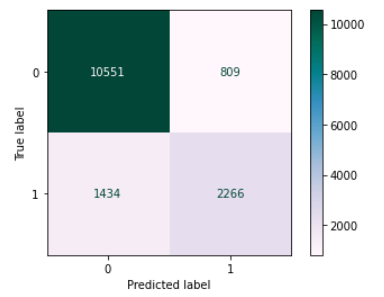
| Model | Accuracy | Precision | Recall | F1 Score |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.79 | 0.55 | **0.82** | 0.65 |
| Random Forest | 0.85 | 0.73 | 0.61 | 0.66 |
| SVM | 0.84 | 0.73 | 0.58 | 0.66 |
| XGBoost | **0.86** | **0.76** | 0.66 | **0.71** |

Confusion matrices:

1. Logistic Regression



2. Random Forest



4. XGBoost

