Census Income Data Eploration Analysis

Simone Dutto, s257348
Mathematics for Machine Learning
Politecnico di Torino

July 11, 2020

Contents

1 Introduction

This data exploration analysis is performed on Census Income dataset. Census is a method of collecting information about inhabitants of a nation. The rationale behind by work is that nowadays targetized marketing is the main business of many companies, to give an example the core business of Google is offering a platform (Google Ads) able to targetize people based on their interests to give them appropriate advertisements. Of course my analysis are just a toy example, but it is useful to explore concepts standing by using machine learning to extract useful information, like expected income, from census data.

2 Dataset

The dataset is the UCI Census Income dataset: it was extracted by the 1994 census in the US.

2.1 Dataset Inspection

The training dataset is a csv file containing up to 32561 and 15 features.
Missing values are indicated as '?', I decided to drop entries containing missing values because they are just 2399 entries and after a simple analysis with my baseline model the dropping made no impact.
The test set is composed by 16281 entries and it is treated in the same way as the train set for what concerns preprocessing.

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native	income	capital
0	39	State-gov	77516	Bachelors	13	Single	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	0	1

Figure 1: Example of an entry

```
age                0
workclass       1836
fnlwgt             0
education          0
education.num      0
marital.status     0
occupation      1843
relationship       0
race               0
sex                0
capital.gain       0
capital.loss       0
hours.per.week     0
native           583
income             0
dtype: int64
```

Figure 2: Number of missing values in the train set

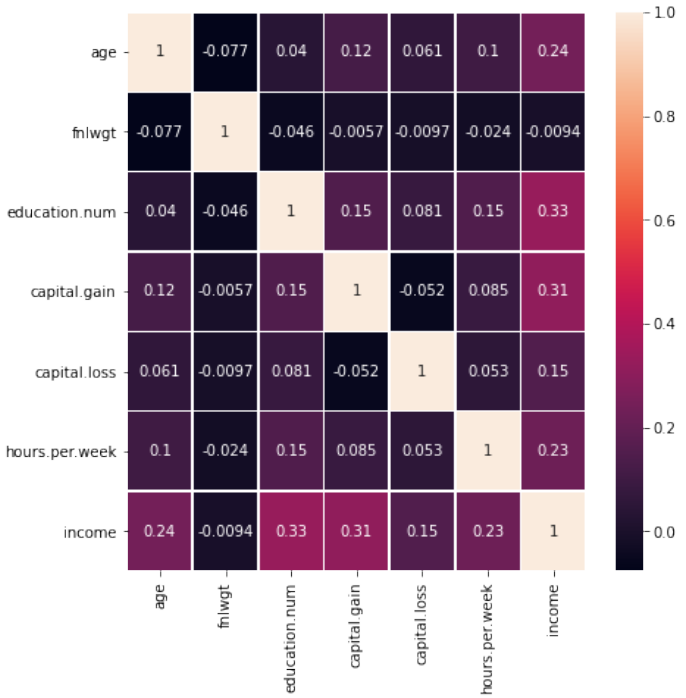
2.2 Features analysis

The dataset is described by 14 features(8 categorical and 6 numerical) and 1 label. In this section I will briefly describe and show graphically them and I will go through a process to motivate my features selection to feed the model I have chosen. In addition I will try to give some considerations.

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Figure 3: Statistics about numerical features

Correlation matrix.
Correlation between attributes expresses how much this attributes are linearly related. 0 means they are not correlated, |1| is the perfect linear relationship.
This correlation coefficients are useful to take decision in the features extraction process.



Income.
This is the label, the ultimate goal of this work is to build a binary classifier to predict it. The label is expressed as '<=50K' and '>50K', it would have been nice to have much more classes or even having the continuous values for this feature to build more sofisticated models.
However I have converted this label respectively in 0 and 1. The first consideration to make is that the dataset is unbalanced, I will go later into the balancing procedure.

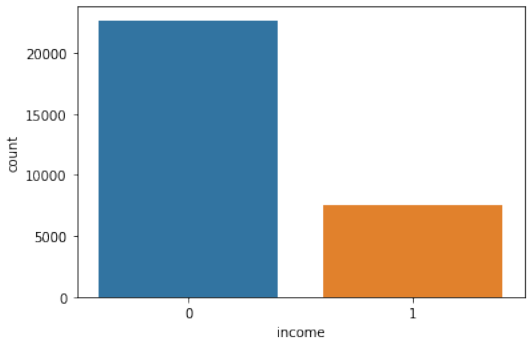


Figure 4: Number of entries with income 0 and 1

Age.
Age is self explanatory but it is interesting to see, using a box plot, that the median value with income >50K is higher (around 40-45 year).

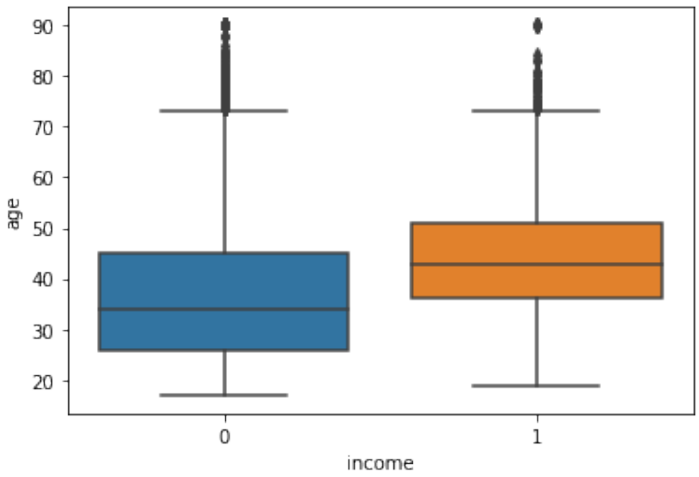


Figure 5: Age boxplot per income

Workclass and Occupation.
The first attribute express whether you work for a company, for the government or by yourself, the second is more precise but it is not much more meaningful than the first, so I dropped it and analyzed just the first. What is very interesting is that the most favourite workclass is 'self-emp-inc', which means people who work by themselves for a corporate entity rathen then people who work for themself.

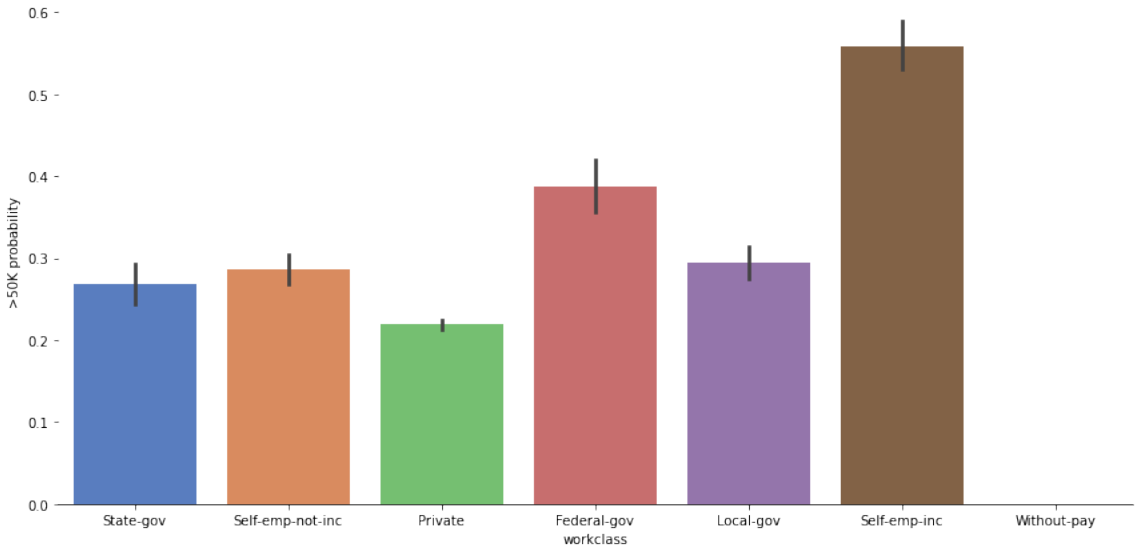


Figure 6: Income >50K probability per workclass

Fnlwgt.
Final weight represents the (estimated) number of people each row in the data represents. It is an indicator on how meaningful is each group but for my analysis is not useful, so to prove it is irrelevant I used hypothesis testing.
Hypothesis testing:
The Independent Samples t Test compares the means of two independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different.
Null Hypothesis: there is no difference in mean fnlwgt of income group >50k and income group <=50k.
Alternate Hypothesis: there is difference in Mean age of income group >50k and income group <=50k.
The p-value is the probability value is the probability of obtaining test results at least as extreme as the results actually observed, assuming that the null hypothesis is correct. So if the p-value is very low we reject the null hypothesis.
I've used `ttest_ind()` from SciPy, and the resulting p-value was 0.55, so we accept the null hypothesis and we can safely drop this attribute in the analysis.
Education and Education-num.
Education is the same attribute as education num but it is categorical, so I will use education-num for my analysis. Education-num is the number of year of schooling. This attributes is the most correlated to the expected income, and the following graph express how good was

in 1994 an investment into achieving higher level of education (14 represents the Master of Science degree, to give an idea).

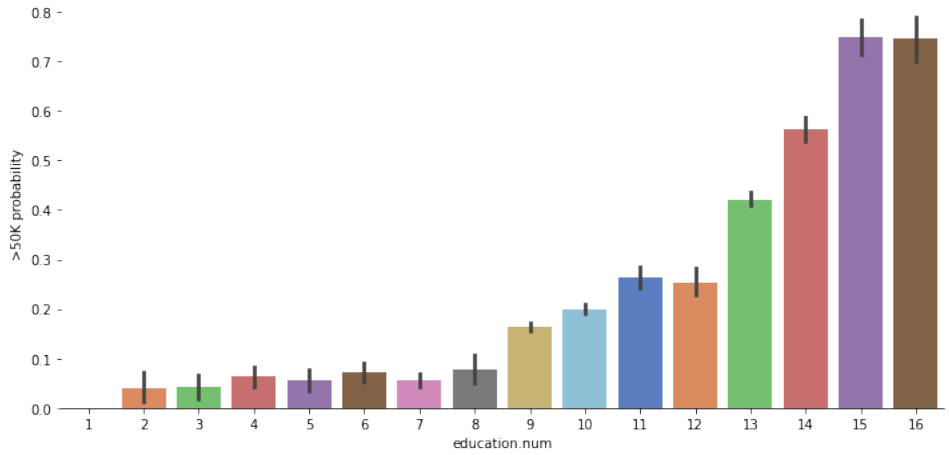


Figure 7: Income >50K probability per years of schooling

Marital-status and relationship

These two attributes are difficult to analyze by themselves, so I analyzed them together. What we clearly see from the graph is that having a baby strongly impact the expected income, and being single has a great impact for box sex but especially if you are female. In further analysis I simplify the analysis by reducing these two attributes to a single one expressing 'Single' or 'Married'.

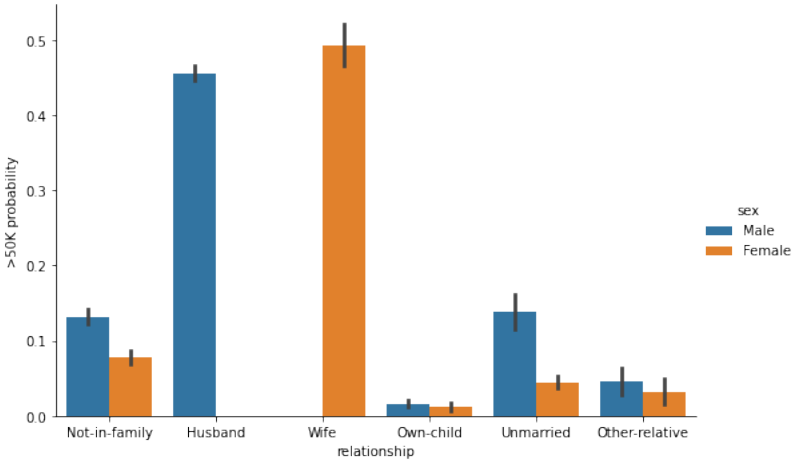


Figure 8: Income >50K probability per marital-status and sex

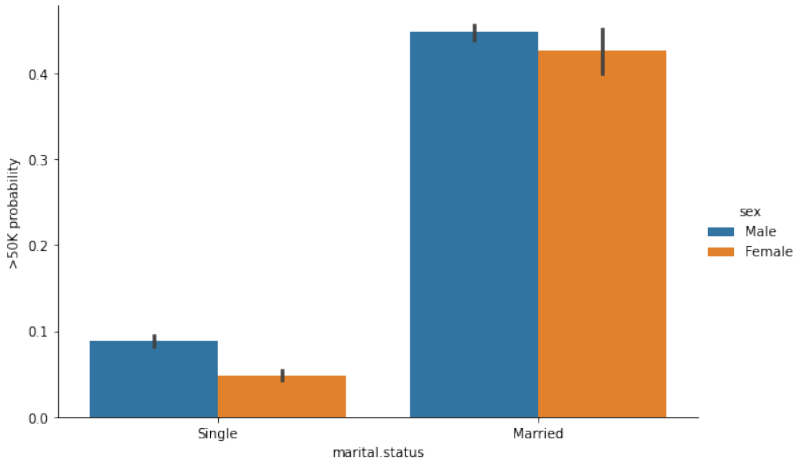


Figure 9: Income >50K probability per marital-status reduced to single or married

Race and native-country

Race express the ethnicity of a person, and we can see from the graph below on the left that is really impactful, therefore we could have concluded that the work environment had an important racial factor. However if we perform the same analysis but taking entries with an education level just above high school the difference are less important.

Of course I am not entitled to give social analysis but I think that it is interesting to notice that sometimes even without changing data it is possible to extract different point of views. The attribute 'native-country' is not relevant if we are already analyzing the race, so I dropped it.

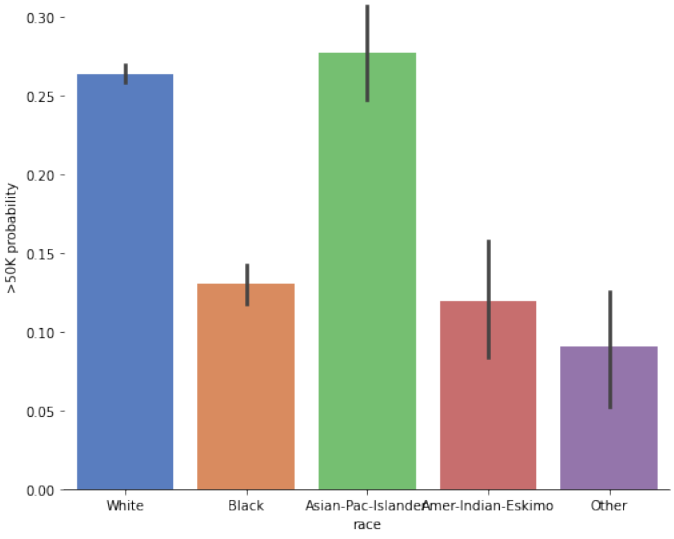


Figure 10: Income >50K probability per race

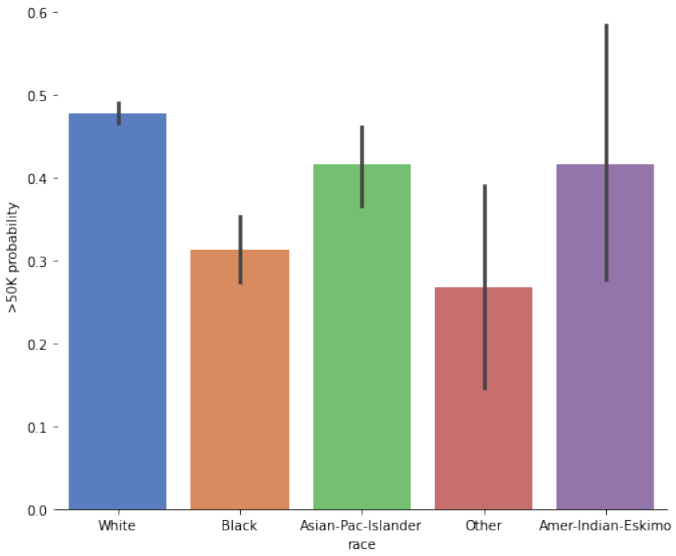


Figure 11: Income >50K probability per race and education-num¿11

Sex

Sex is an important factor in the expected income, I tried to search, as I did with the race, other factors that can help contextualizing this difference but I did not find any. The differences remained with higher level of education, different marital status and relationship as we can see from the preceding graphs.

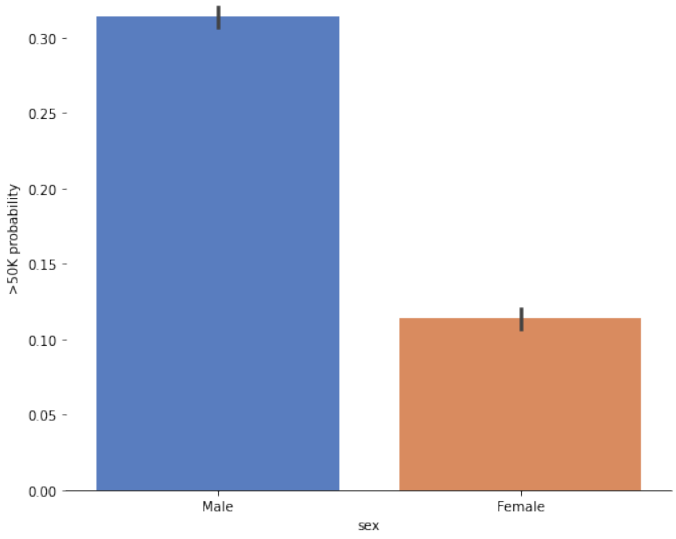


Figure 12: Income >50K probability per sex

Capital Gain and Capital Loss

Capital gain and capital loss refer to profit that results from a sale of a capital asset, such as stock, bond or real estate, where the sale price exceeds the purchase price. Since the median of capital gain and capital loss is 0 for both income classes but the average is different I

thought would be better to convert this two variable into a 'has invested any capital?', the result is that I get a binary attribute which is simpler.

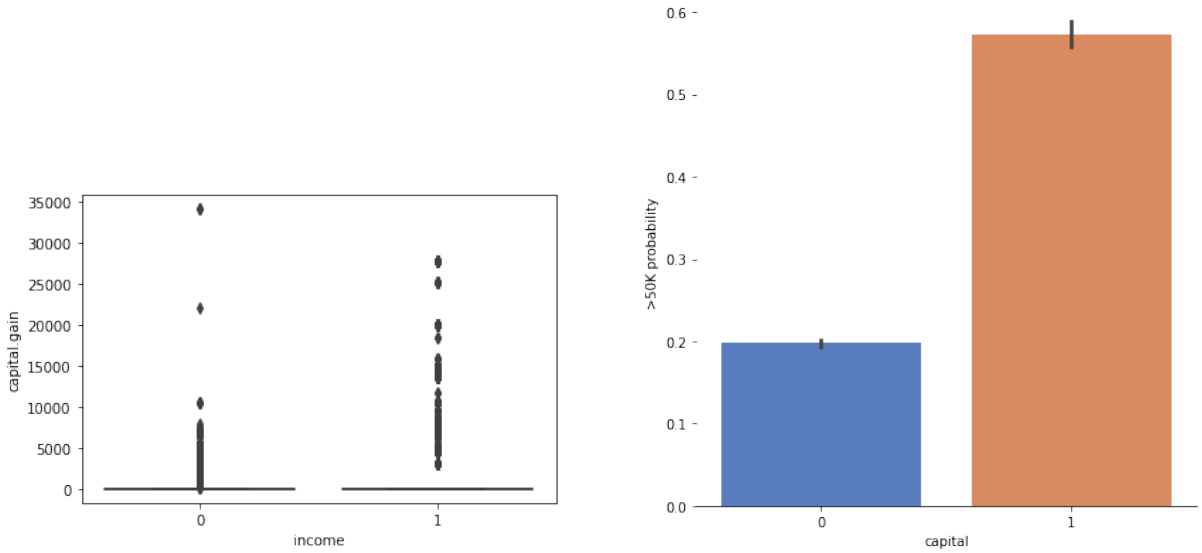


Figure 13: Distribution of capital gain per income and probability of > 50K income with capital binary variable

Hours-per-week

Number of hours per week worked, it would be intuitive to think the greater the number the greater the income. This is true, but we can see that with higher income the range 15%-to-75% percentile is much wider, so there is much more variance in the higher income class.

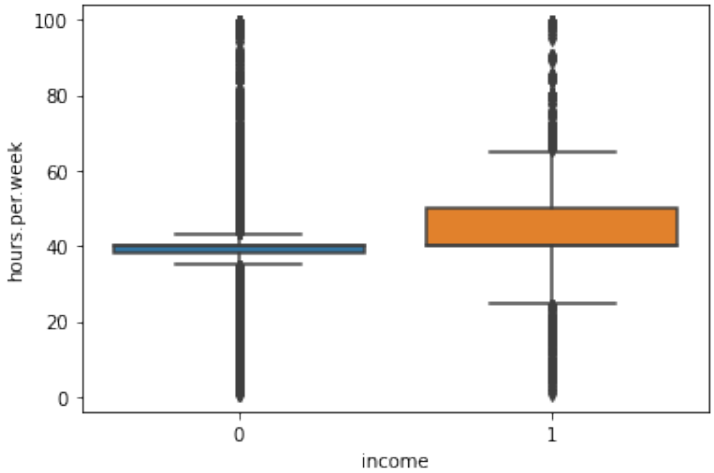


Figure 14: Income >50K probability per years of schooling

2.3 Features preprocessing

In this paragraph I will go through my preprocessing pipeline to prepare data to be used by my models.

Balancing procedure.

Since most machine learning algorithms are built to minimize error over the train set, if we have unbalance train set we encounter the problem of having high accuracy overall but low accuracy over the minority class as we can see in the following graph. For balancing my dataset I tried `SMOTE()`, which is more sophisticated oversampling technique than the random resample used by the `resample()` of Pandas. In fact, SMOTE works by selecting examples that are close in the feature space, drawing a line between them and creating a new sample at a point along that line. However, at the end it was not beneficial for my dataset because it brought down the precision a lot in favor of recall, so I decided to leave the train set as it is.

Best model on training set without oversampling has {'n_neighbors': 9} with recall 0.5820

	precision	recall	f1-score	support
0	0.87	0.90	0.88	11360
1	0.66	0.57	0.61	3700
accuracy			0.82	15060
macro avg	0.76	0.74	0.75	15060
weighted avg	0.81	0.82	0.82	15060

Best model on training set wit oversampling has {'n_neighbors': 9} with recall 0.8861

	precision	recall	f1-score	support
0	0.92	0.76	0.83	11360
1	0.52	0.80	0.63	3700
accuracy			0.77	15060
macro avg	0.72	0.78	0.73	15060
weighted avg	0.82	0.77	0.78	15060

Figure 15: Comparing with using and not using oversampling with imbalanced classes

Handling categorical data.

Models does not accept string as inputs, so we need to encode categorical variables into numbers. There are 2 ways: the first is the label encoder, each unique key is associated with a unique number progressively higher (Ex. $'Cat' \rightarrow' 0', Dog \rightarrow' 1',' Duck' \rightarrow' 2'$). This method implies 'Cat' and 'Dog' are closer in the features dimension than 'Cat' and 'Duck', and this could lead to problems.

So, the second method was invented, which is called OneHot Encoding. What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

I used OneHot Encoding for my only categorical column left, 'workclass'.

Standardize data.

Since most models I tried are distance-based (like KNN, SVM), it is better to have features which are standardized. This is to avoid having a feature prevail the others.

`StandardScaler()` does this operation by transforming each feature by removing the mean and scaling to unit variance.

PCA.

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
In particular, each principal component is perpendicular to the preceding and it points toward the maximum variance and minimum reconstruction error. I choose to keep components

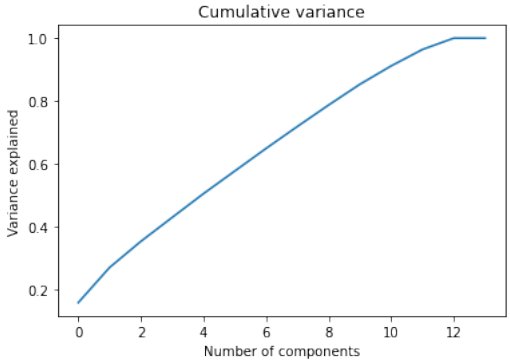


Figure 16: Cumulative variance by number of components

to express 90% of total variance, which corresponds to 10 features from 14 of the original data.
PCA can be also used to plot the data points in 2 pca-component dimensions and see if they are linearly separable. This is interesting because it gives an idea on how data is organized in the space.
From the graph it is easy to see that the datapoints are not linearly separable in 2 dimen-

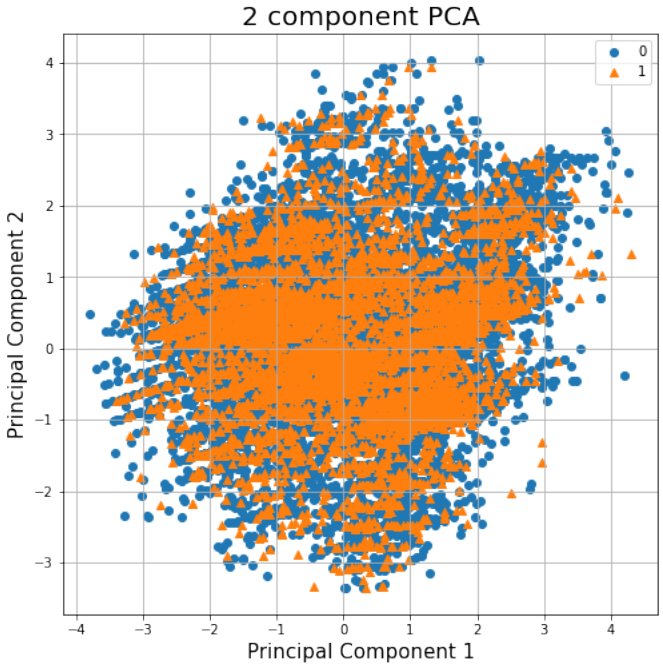


Figure 17: Cumulative variance by number of components

sions, moreover 2 pca-components represents just the 56% of the variance. So, it is not a good projection to take into consideration.

3 Models

In this section I will present the models I used and their rationale, giving some mathematical details and their differences.

3.1 KNN

K nearest neighbors is a supervised machine learning algorithm often used in classification problems. It works on the simple assumption that same class point are close in the future space. This algorithm works by classifying the data points based on how the neighbors are classified. Any new case is classified based on a similarity measure of all the available cases. K in KNN is the number of neighboors to take into consideration when assigning the label. It is the only hyperparameter to tune, together with the distances calculation method(L2, L1, Minkowski).
I used L2 method to calculate distance between two points X and Y with n coordinates:

$$d = \sqrt{\sum_{i}^{n}(X_i - Y_i)^2}$$

It is a discrimitive approach, it usually leads to non linear decision boundaries. However it is memory and computationally expensive, because every point needs to be stored to evaluate a new entry.

3.2 Logistic Regression

Logistic Regression is a learning algorithm based on the so-called Logistic Function to evaluate the probability of belonging to a class:

$$f(X) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i * X_i)}}$$

Then, we use the result from $f(x)$ and a threshold to decide which class the point belongs to (usually >=0.5 is 1, and <0.5 is 0).
We use Gradient Ascent to minimize the cost function and update weights accordingly:

$$J(\mathbf{w}) = \sum_{i=1}^{m} -y^{(i)} log\Big(f\big(z^{(i)}\big)\Big) - \big(1 - y^{(i)}\big) log\Big(1 - f\big(z^{(i)}\big)\Big)$$

$$\Delta w_j = -\eta\left(\frac{\partial J}{\partial w_j}\right)$$

$$\mathbf{w} := \mathbf{w} + \Delta\mathbf{w}$$

Logistic Regression is a discriminitive approach, and it is possible to update the model online each time we want to add an entry. In fact, it is an optimization problem. It is fast to train and fast to evaluate, and it reaches good performances in the limit.

3.3 SVM

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum margin.

$$y_i(x_i^T w + b) \geq 1(i = 1...m)$$

$$min||w||$$

Hard margins work only when data are linearly separable, so we relax the constrictions by adding a term in the optimization problem. C expresses how soft is the margin. The higher the softer, so some data points are allowed to cross the margin.

$$y_i(x_i^T w + b) \geq 1 - \epsilon_i,\, \epsilon > 0\, and\, (i = 1...m)$$

$$min||w|| + C\sum_{i}^{m} \epsilon_i$$

If data points are not linear separable it is possible to still use SVM, but we need to apply the so-called Kernel Trick. The Kernel Trick aims to substitute the dot product in the optimization problem, to implicitly map the data points into an higher dimensional space. So SVM can be adapted to work with non-linear separable data points class. SVM is a discriminative approach, it is the slowest both at training and evaluation time because the fit time scales at least quadratically with the number of samples.

3.4 Random Forest

A random forest is an ensemble model that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. To create the rules the algorithms usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best". I used Gini Impurity to evaluate how good is a split:

$$Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

Random Forest is usually really accurate, and it can shape non-linear boundaries really well. However it is really slow both to train and to evaluate, because it is an ensemble method.

4 Experiment

In the section before I will go through the results I obtained for my experiment, but before analyzing the results I will explain my validation process.

4.1 K-Fold Cross-Validation

Cross-validation is a resampling procedure used to evaluate machine learning models, such as it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split. The dataset is splitted in k folds (5 in my experiments), the model is trained upon k-1 folds and tested on the one-out. This is done for k times, and the results are averaged.

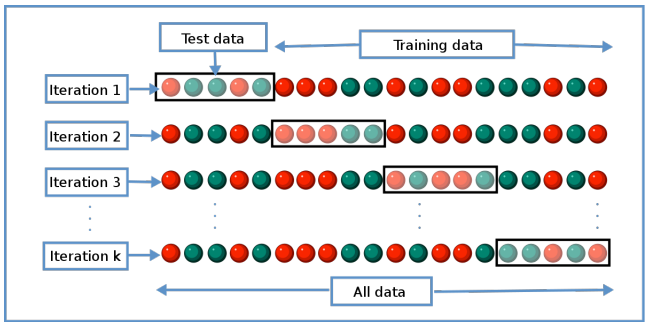


Figure 18: k-Fold Cross Validation [1]

4.2 GridSearch

For a fine-tuned selection of the models I used a GridSearch Approach. It generates all the possible combination from a list of hyperparameters. In this way I can choose properly the hyperparameters and test the best performing model on the test set to evaluate performances. With GridSearch it is possible to choose the scoring mechanism to find the best model, I have choosen to maximize precision on the high income class, because I thought it was interesting to see if the models were able to generalize enough high income class.

$$F_1 = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Note. The function *GridSearchCV()* of *sklearn* implements both the strategy, GridSearch in combination with k-fold cross-validation. In the next section I will go though the hyperparameters I decided to tune and how they affect the models. The results I will show are the recall on the income >50K class.

4.3 KNN

The only parameter to tune in KNN is the K. KNN is able to delineate complex boundaries. The higher is K and the less precise is the model in the limit, but it is less sensible to outliers.

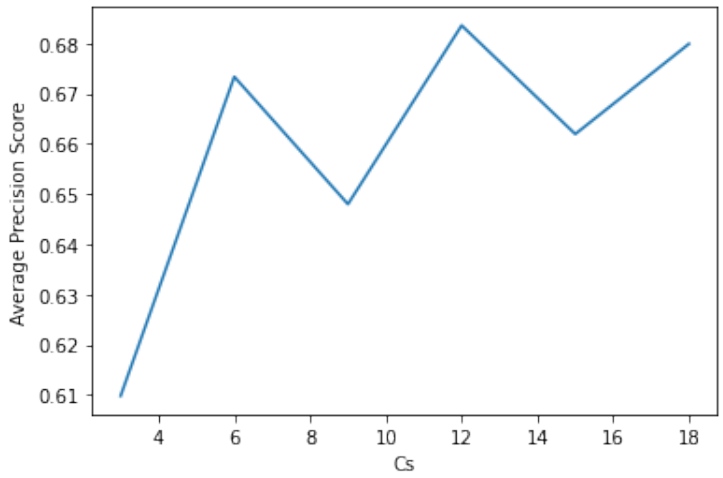


Figure 19: KNN Cross Validation results

With K equals to 12 on test set I obtain 83% accuracy and 70% precision on high income.

4.4 Logistic Regression

For Logistic Regression I choose to optimize C. It is a regulation parameter, which is a term used to enforce or diminish the weight update to avoid overfitting and underfitting.

$$L2 : \frac{\lambda}{2}\|\mathbf{w}\|_2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

C is the inverse of λ and it is basically a "balance" to the learning process - between regulating too much (forcing convergence to local parameter space or extremely precise area) and regulating too little (fail to converge to a optima point). The higher C the lower is the regularization.

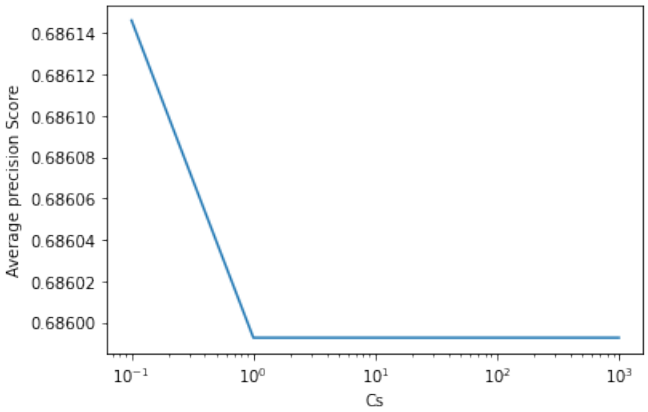


Figure 20: Logistic Regression Cross Validation

It is easy to see that a little bit of regularization helps the optimization to overcome local minima.
Finally this model, it is able to peak at 82% accuracy and 68% precision on high income.

4.5 SVM

For SVM I choose two kernels, linear and rbf. For the linear and rbf I tried different Cs, C expresses how soft is the margin.

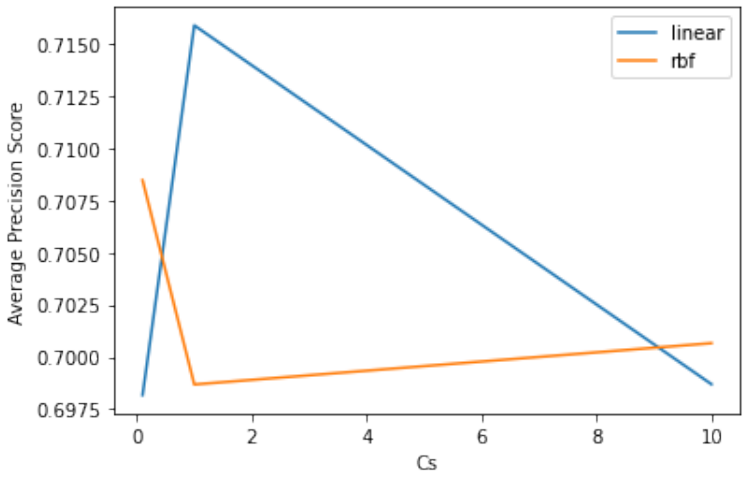


Figure 21: SVM linear kernel Cross Validation

For the rbf kernel there is another hyperparameter, which is gamma. Gamma is the hyperparameter which express how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

$$rbf\ kernel : K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma||\mathbf{x} - \mathbf{x}'||^2\right)$$

SVM is very slow to train and evaluate, and it reach 83% accuracy and 71% precision with rbf kernel.

4.6 Random Forest

To optimize the Random Forest I choose 2 hyperparameters:

- Number of estimators: The number of tree in the forest, the higher the noisy is the model and the classifier performs really well
- Maximum depth: It expresses how deep each tree can be build, the deeper the higher the risk of overfitting but the higher is the accuracy

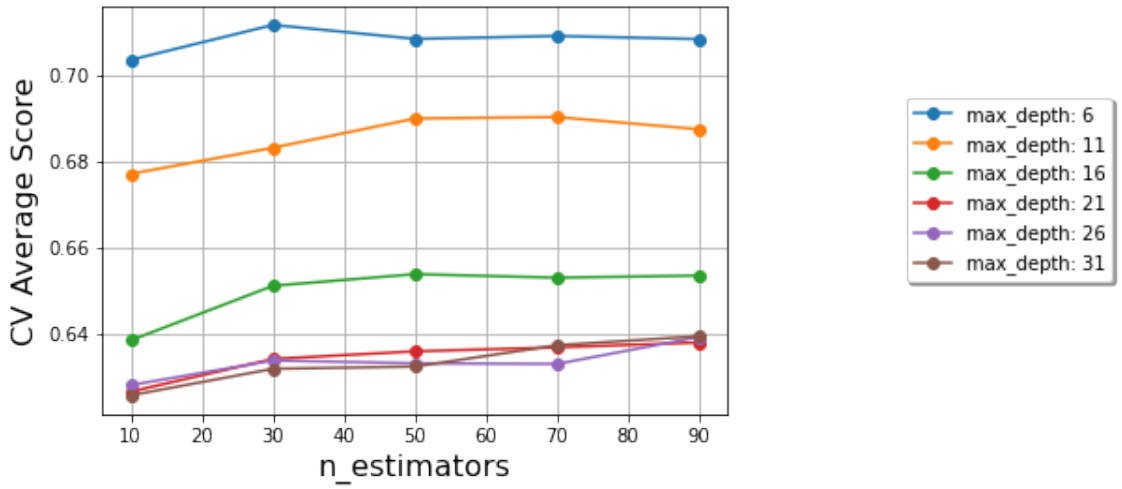


Figure 22: Random Forest Cross Validation

The random forest is the only model which was able to maintain an high accuracy (83%) and a good precision on the high income class(71%), but the recall is low (50%).

4.7 Performances on Test Set

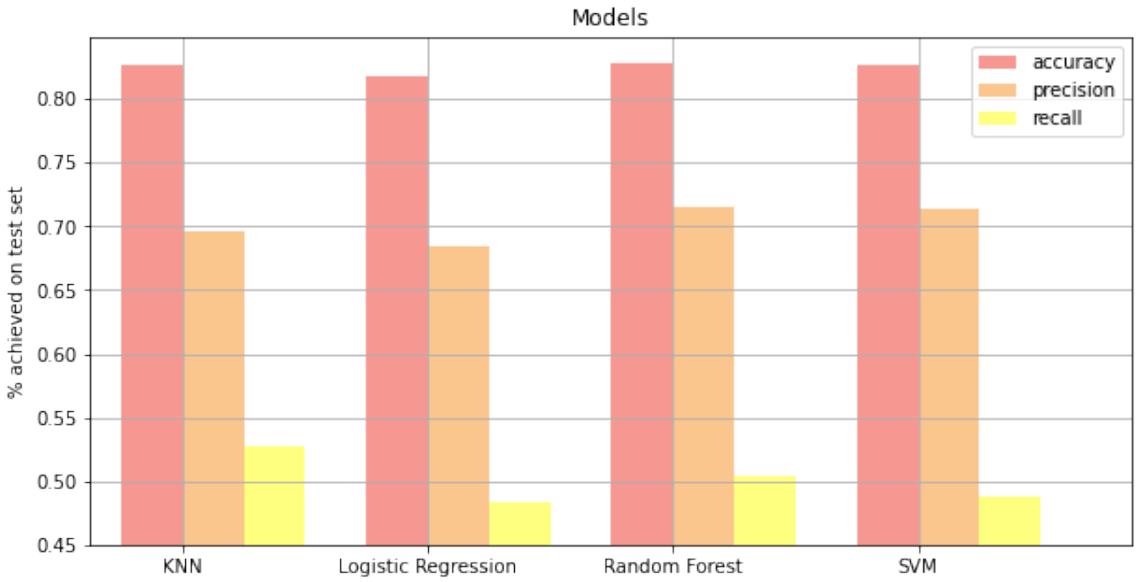


Figure 23: Performance on Test Set

5 Conclusion

In conclusion, the only model which was able to generalize a bit the high income class was the Random Forest, because of its abilities of defining non-linear decision boundaries. The

other models even though they reached high accuracy value they are not very good because their recall is very low (50%), so basically they cannot find properly all of the high income data point.
For these reasons I think that it would be better to use more complex models, or perform a better features extraction, to reach higher recall score on high income class.
Final Note on the Code. I developed my models and made my experiments using Google Colaboratory together with Python and sklearn library [2] . The Python Notebook can be found at *this link*.

References

[1] Wikipedia. Cross-validation. https://en.wikipedia.org/wiki/File:K-fold_cross_validation_EN.svg.

[2] sklearn. sklearn. https://scikit-learn.org/.