Income Prediction
binary classification

By Babak Mehrabi

Outline

- Introduction
- Data Analysis
- Modelling (XGBoost, FNN)
- Next Steps

Introduction

- Task: Predict whether a person's income is over 50K/year or not based on different features
- 48842 records with 14 features and 1 response variable
- 66% taken as train data, 33% as test data
- Unbalanced binary classification problem

Introduction: A peek into the data

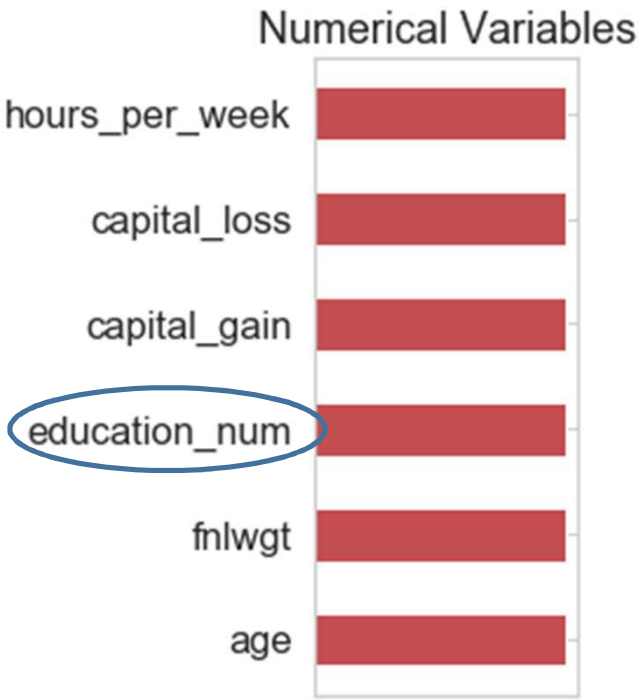
	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife

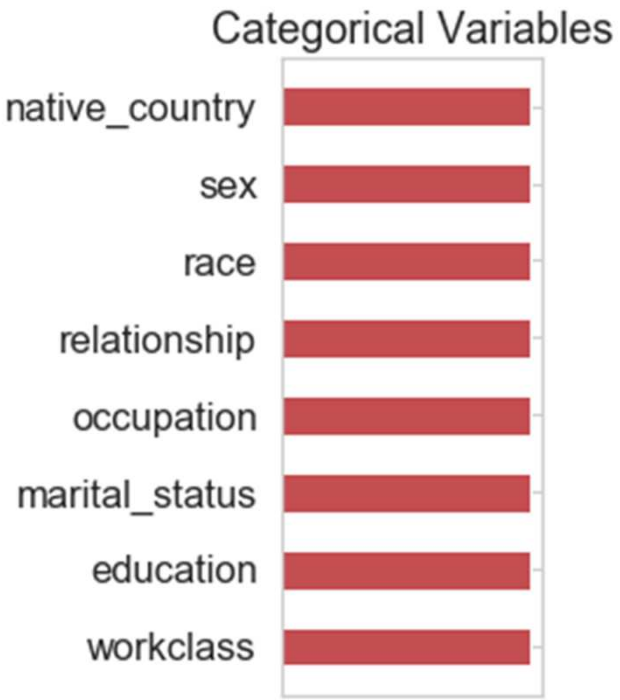
	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
0	White	Male	2174	0	40	United-States	<=50K
1	White	Male	0	0	13	United-States	<=50K
2	White	Male	0	0	40	United-States	<=50K
3	Black	Male	0	0	40	United-States	<=50K
4	Black	Female	0	0	40	Cuba	<=50K

Data Analysis (steps)

- Variable Identification (for ex. categorical or numerical etc.)
- Analysis of Missing Values (NaNs)
- Data Cleaning
- Univariate EDA (exploratory data analysis)
- Bivariate EDA (correlations, chi-2 test)
- Feature Engineering & Selection Ideas

Variable Identification





education_num can be considered as a categorical variable too, depending on the model whether it is tree-based (Random Forest, xgboost) or linear-based (logistic regression, neural networks)

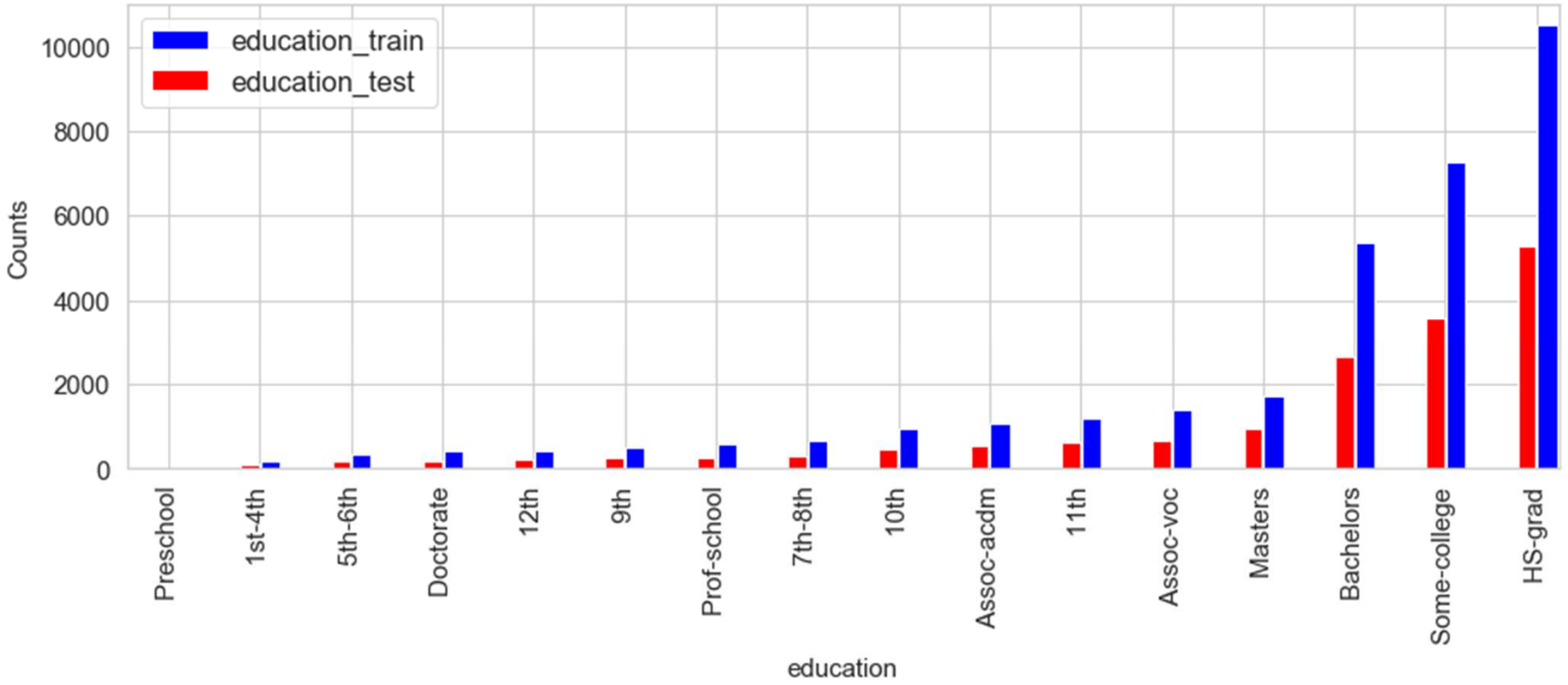
Missing values and data cleaning

- There were no missing values in form of NaNs, however there are missing values in form of “?”, which is not a problem for tree-based models as they can be treated as a level.
- For linear-based models we can dummy-encode them, and they still create no issue, as their lack is also a kind of information
- There were 30 duplicated rows in the whole data, which were dropped

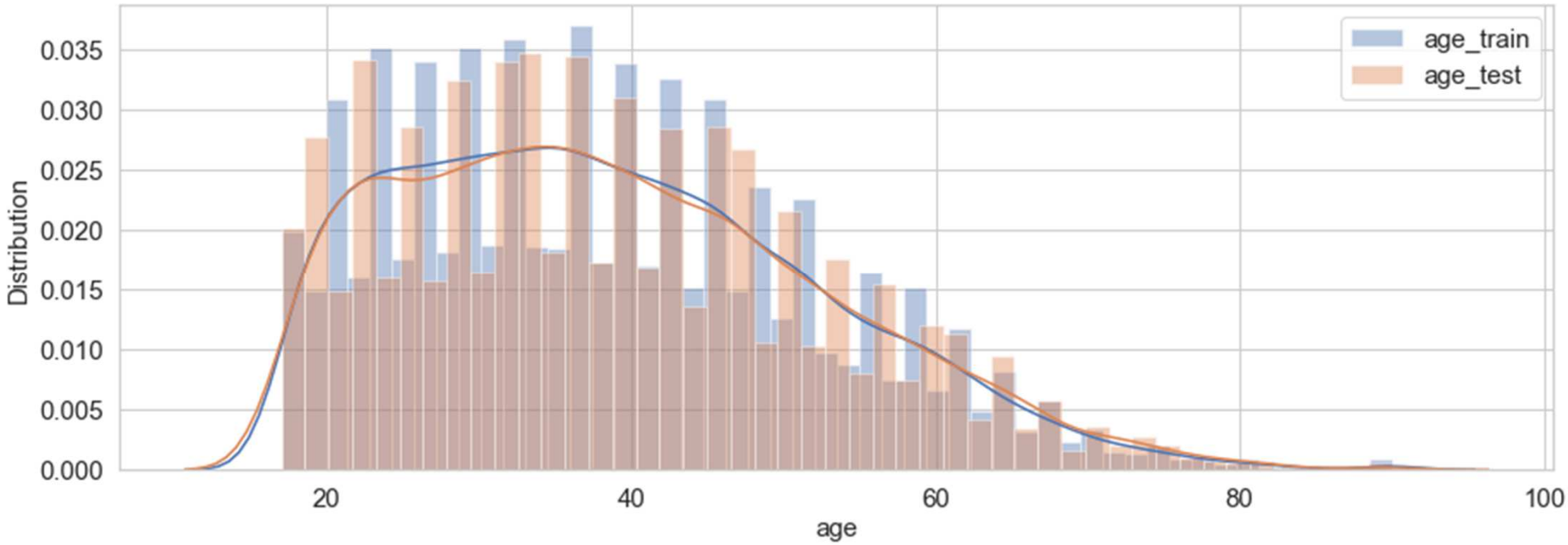
Univariate EDA

- Barplots for categorical features wherein counts for both train & test datasets are shown
  - This is important because we can find out if there is a major difference between the distribution of variables among train and test datasets
- Histograms and density functions for numerical features
- dispersion plot for numerical features vs the row index (in J notebook)
  - The data points are plotted versus the row. (Good for discovering irregularities)

Univariate EDA: barplot for categorical features, example



Univariate EDA: histogram for numerical features, example



Univariate EDA: observations

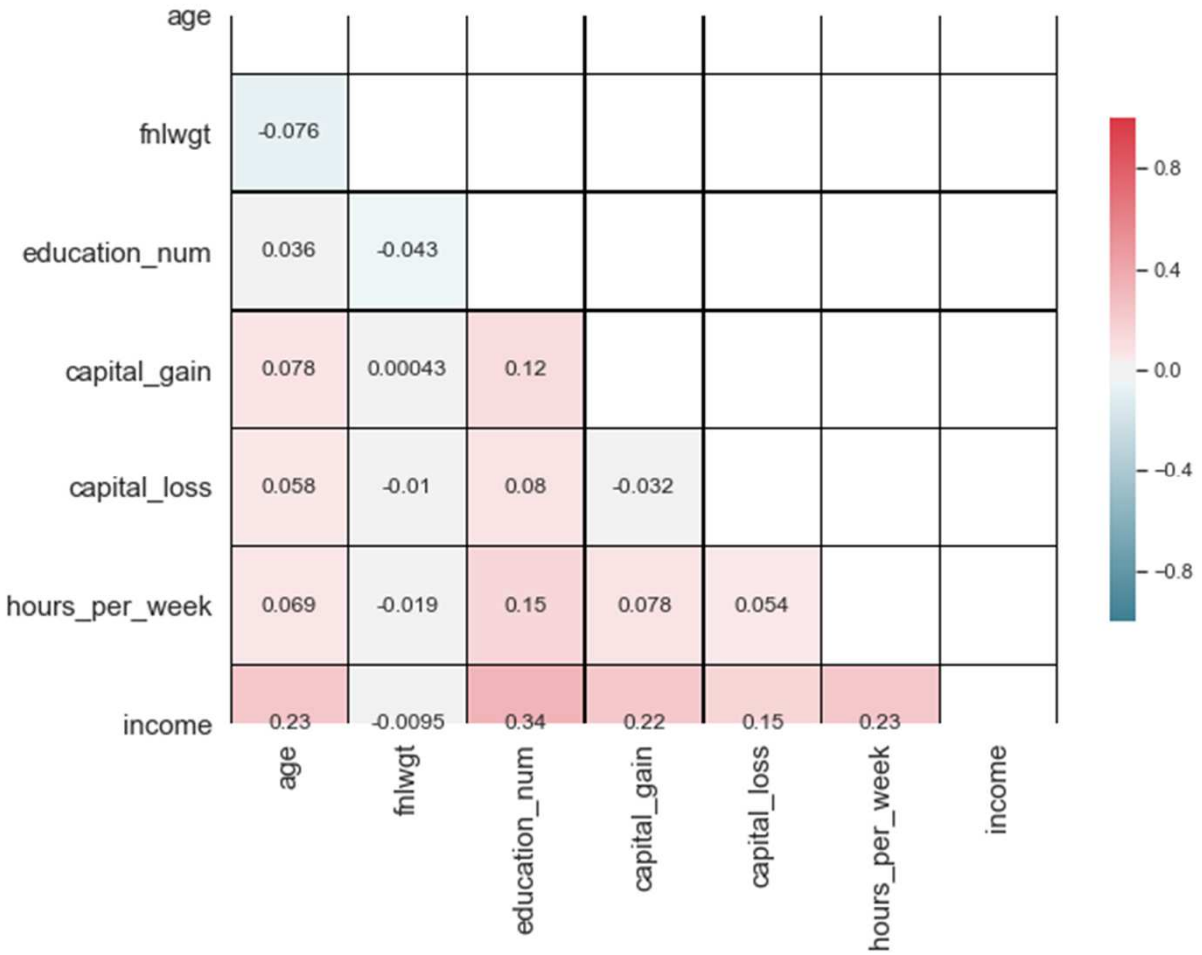
- **workclass**, **occupation** and **native_country** have missing values which is denoted by “?”. This is not a problem for tree-based models however!
- In variable **native_country**, one level "United-States" has the highest frequency (89.6%)!
- variable **fnlwgt** sounds to have outliers
- variables **capital_gain** and **capital_loss** mostly have zero as their value
- 40 hours is the most common level in hours_per_week variable

Bivariate EDA

- Heatmap plot of correlation among different variables (numerical)
  - Potential discovery of collinearity among variables
  - Response variable can be treated as a numerical feature by dummy-encoding, thus enabling us to calculate its correlation with other features (good for feature selection)
  - This heatmap is plotted only for numerical features (including response)

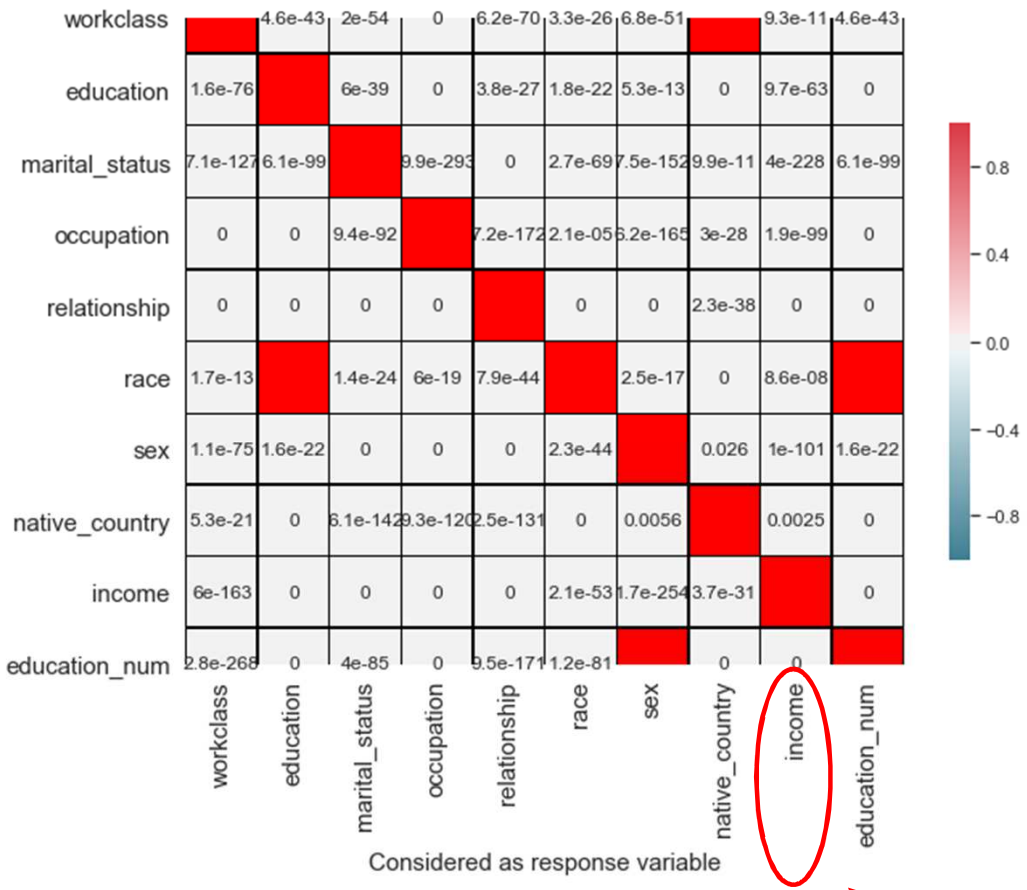
- Heatmap to plot chi-2 test results (categorical)
  - Good to find out dependency among the response variable and other features, to help us with feature selection

Bivariate EDA: heatmap of correlation



- The response variable “income” has a very low correlation with the numerical feature “fnlwgt”. This is the first hint that maybe we should drop this feature
- Apart from that, there is not much more finding

Bivariate EDA: heatmap for chi-2 test



- In chi-2 test, the dependency among an output variable and some features are calculated.
- Each variable is once considered as the output variable including our own response variable, and its dependency with other features are analyzed with p-values
- The values in cells are p-values

response variable: income

Bivariate EDA: heatmap for chi-2 test

- If p-value <= alpha:
  - significant result, reject null hypothesis (H0), dependent.
- If p-value > alpha:
  - not significant result, fail to reject null hypothesis (H0), independent.
- Assuming alpha= 0.05, most of p-values are way smaller than alpha
- Those p-values greater than alpha were colored with red. These are the non-diagonal red cells
- The most important finding is that the response variable is dependent on all features with 95 % confidence interval

Feature Engineering & Selection Ideas

- **General**
  - We have many categorical variables with many levels (8 vs 6). This might suggest that tree-based models (such as Random Forest, XGboost etc.) maybe be better candidates. But we could still try other models
  - Considering the Cartesian product of categorical features may improve the performance of the model
- **model specific**
  - For tree-based models, we will label-encode the categorical variables, although for XGBoost we could also consider the one-hot encoded features
  - For linear-based models, we will one_hot_encode them.

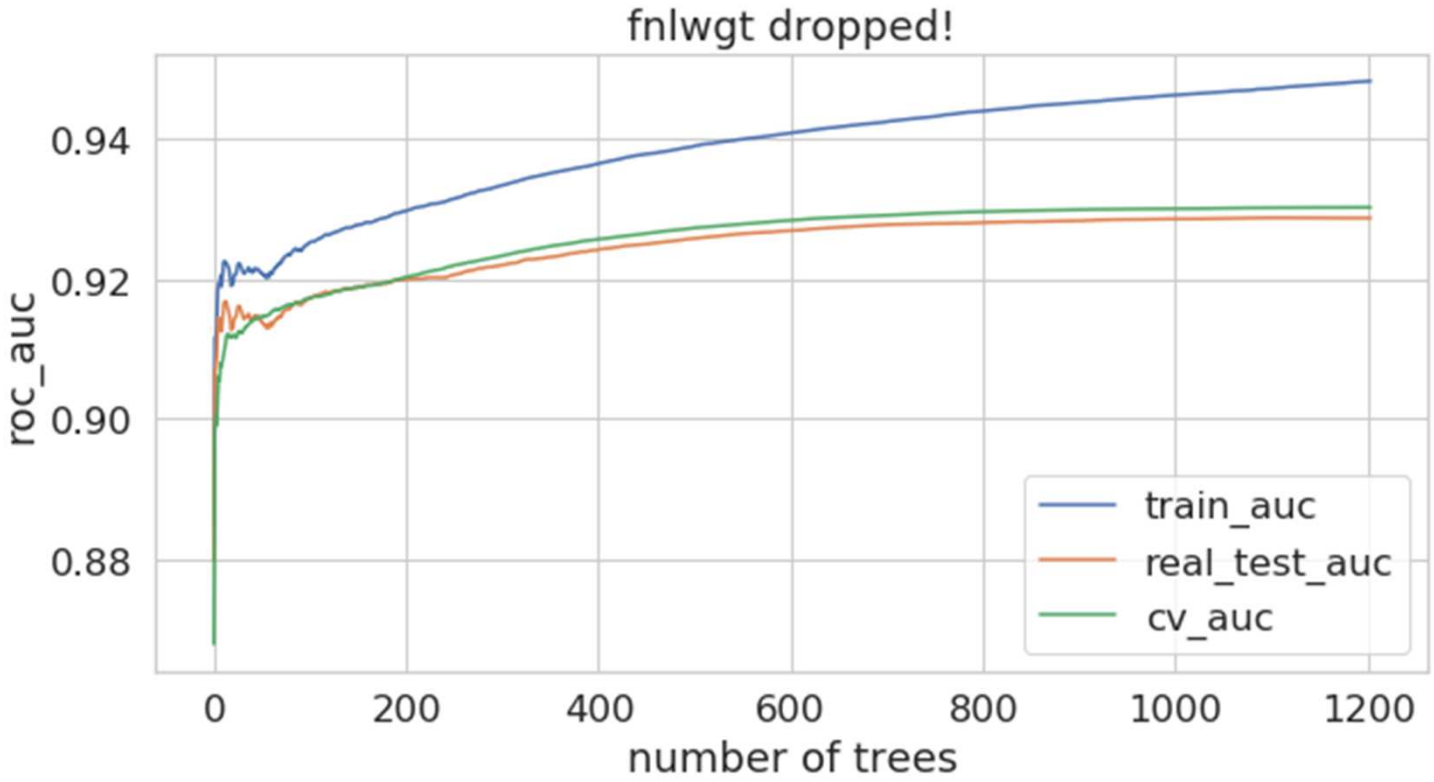
Modelling (XGBoost)

- Since the data is imbalanced, we choose roc_auc as our metric to be reported
- In XGBoost, hyperparameters must be optimized
- In theory, one would need to large grid search in order to find the optimal hyperparameters, but in practice this is impossible
- So we have a different approach to hyperparameter optimization

Modelling (XGBoost): Our approach to hyperparameter optimization

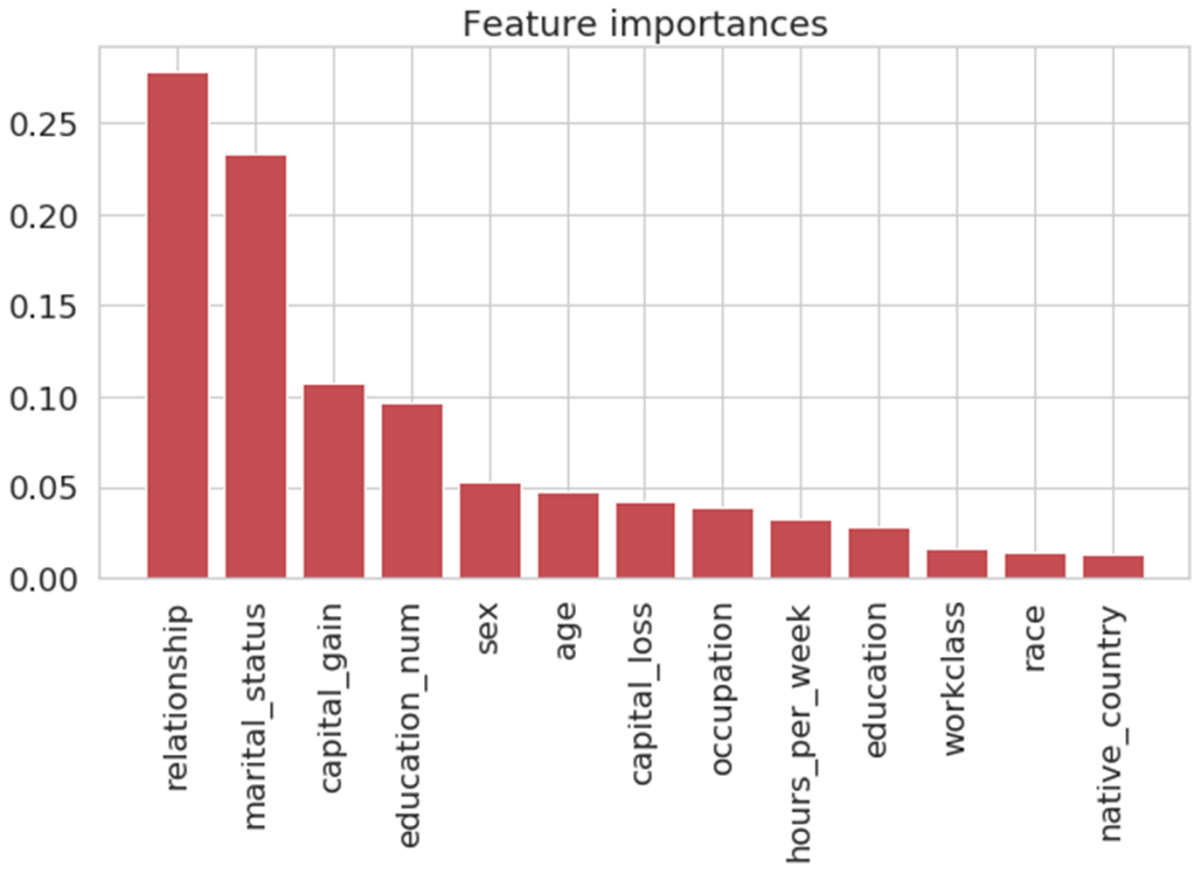
1. Choose a high learning rate
2. Determine the optimum number of trees for this learning rate, assuming an initial set of hyperparameters
3. Tune tree-specific parameters (max_depth and min_child_weight)
4. Tune gamma
5. Tune subsample and colsample_bytree
6. Tune regularization parameters (lambda, alpha)
7. Lower the learning rate with the found hyperparameters

Modelling (XGBoost): train, test, cross-validation auc



- As it can be seen from the plots, we have approximated the real test auc very well with the cross-validation auc
- Real test-auc is always lower than the approximated cv auc

Modelling (XGBoost): feature importance



- As it can be seen, native_country does not have much importance. Besides, it has too many levels.
- Removing native_country has a very minimal effect on the test auc (tested)

Modelling (XGBoost): model performance

Confusion matrix for train data

0	1
20742	779
3978	7062

Confusion matrix for test data

0	1
10292	534
2143	3312

auc

train auc	test auc
0.948283	0.928779

Modelling (XGBoost): categorical features dummy encoded

- We expect that the performance of the model improves because now features have spread out in the features space, from a label encoded to a dummy encoded state

- This gives the XGBoost algorithm the flexibility to learn more complex relationships

Modelling (XGBoost): categorical features dummy encoded

Confusion matrix for train data

0	1
20621	778
4099	7063

Confusion matrix for test data

0	1
10249	524
2186	3322

auc

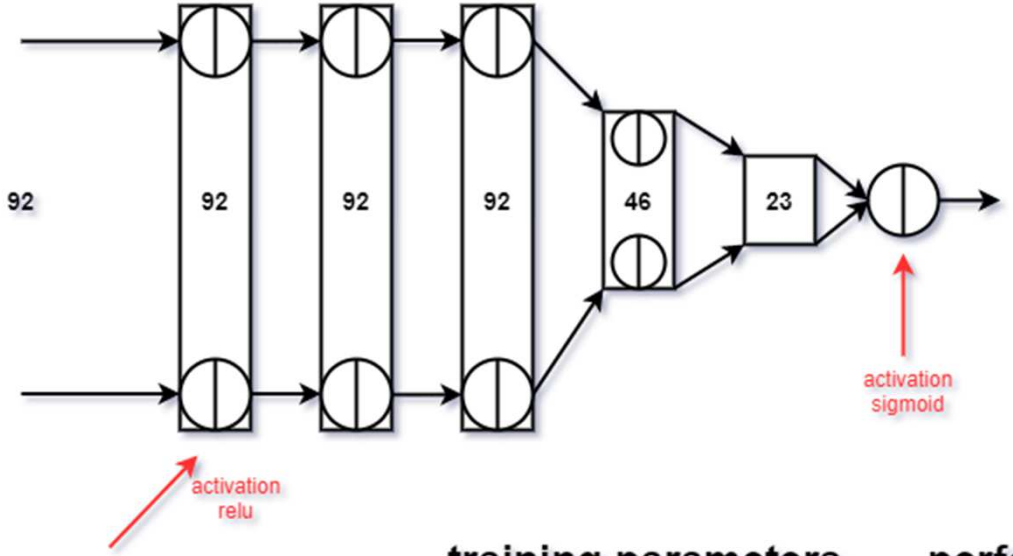
train auc	test auc
0.946151	0.929234

Comparing to when the features were only label encoded, the performance slightly improved.

Modelling (XGBoost): categorical features dummy encoded, selecting features

- Considering all the dummy features, we have around 107 features
- After dropping those ones whose importance are zero according to XGBoost algorithm, the number of remaining features reduced to 92
- In other words, XGBoost helped us select features
- Interestingly, these non-important features are mostly different levels of “native_country”, but nothing surprising as we already knew it is not an important feature comparing to other features.

Neural Network: Dummy features with omitted features from the XGBoost results



training parameters

eopchs= 1200

batchsize= 512

optimizer= adam

loss: binary-cross entropy

metric: auc

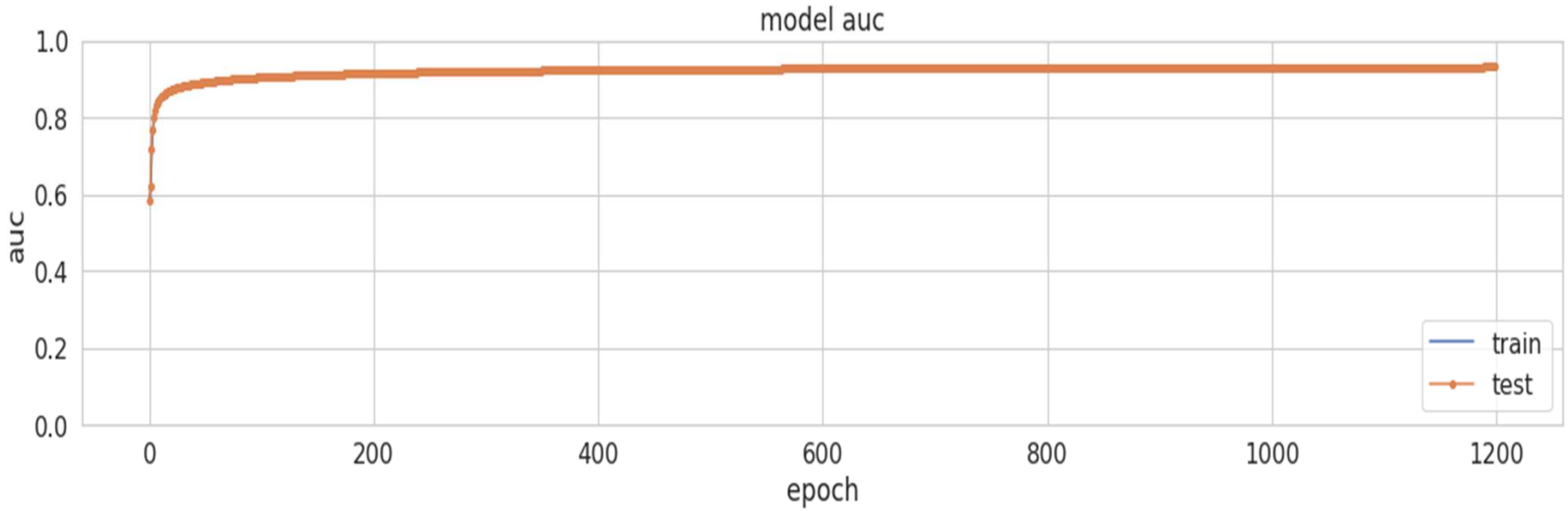
performance

train auc: 0.9330

test auc: 0.9327

- Here we see the architecture of the feed forward neural network we used for training.
- The architecture is not unique and it is itself a hyperparameter
- We could already improve our auc metric comparing to XGBoot by almost 0.003

Neural Network: Dummy features with omitted features from the XGBoost results



Neural Network: Dummy features with omitted features from the XGBoost results

Confusion matrix for train data

0	1
23742	978
831	7010

Confusion matrix for test data

0	1
10892	1543
1442	2404

auc

train auc	test auc
0.9327	0.9327

Performance improved w.r.t XGBoost

Next steps

- Train other models such as SVM, random forest, naive Bayes etc.
- Build an stacking of models to improve the performance
- Try to optimize models for other metrics such as precision, recall or F1-score if needed
- More feature engineering based on the domain knowledge