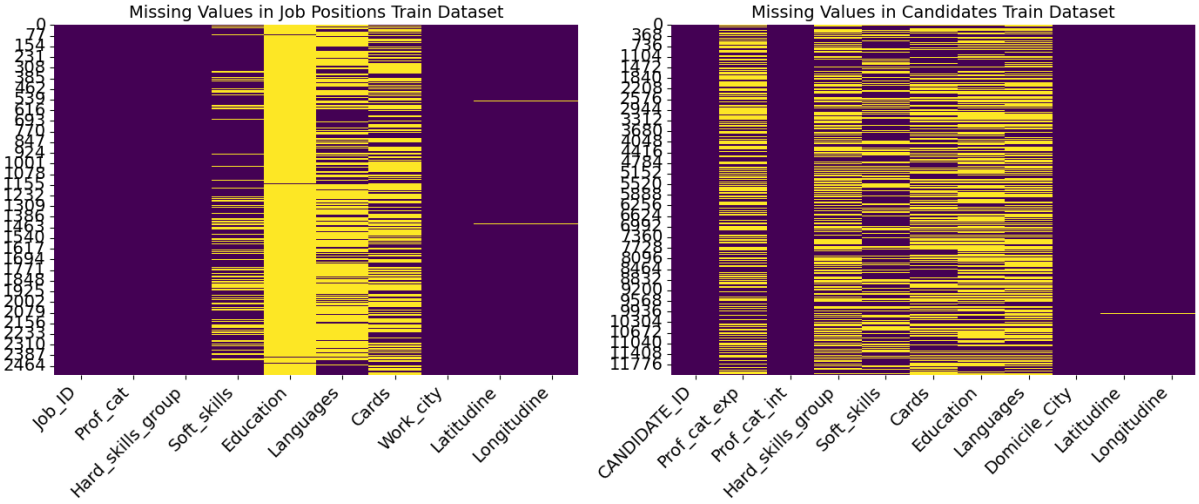
SUS7
Gli Imbruttiti
Hackathon Report

Giorgio Bertone, Filippo Parlanti, Gianluca Procopio, Himel Ghosh,
Sapienza Università di Roma

June 17, 2024

1 Exploratory Data Analysis

First of all we had to perform some pre-processing of the data because we noticed that there were many missing values. For example, 'Education', 'Cards' and 'Languages'.



We mainly focused on the Professional Categories, on Skills, Latitude and Longitude. We also noticed that for each *Job_ID* there were several *Candidates* who applied.

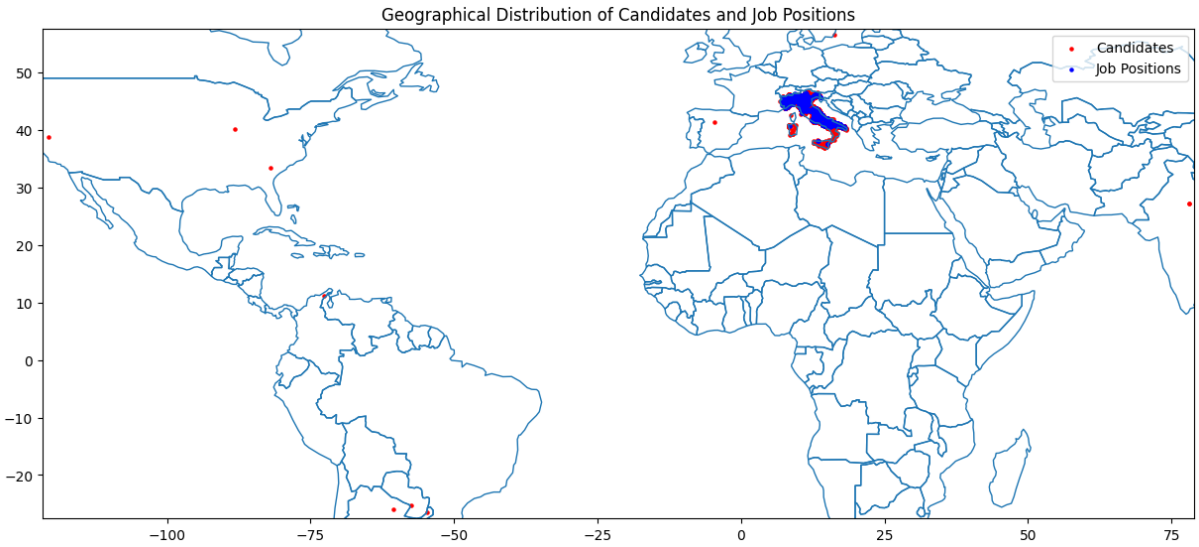
We also found that there are a majority of the candidates from Italy and a few others from other European countries and some also from India, USA and South America while the job positions are all available in Italy.

2 Problem Approach

We decided to create new features by computing the similarities between variables relating jobs (*job_positions.csv*) to candidates (*candidates.csv*)

First of all we retrieved the list of unique tokens for each of the above mentioned features and we used these to transform the semicolon separated values through one-hot encoding (MultiLabelBinarizer).

Then, we used these representations to compute similarities leveraging **cosine** and **Jaccard** similarities for each couple of Job and Candidate specified in the *Match* dataset.



2.1 Similarities

We calculated the following features:

- **prof_cat_exp_similarity**, between 'prof_cat_exp' (candidates) and 'prof_cat' (job positions) to find relationships between the categories required from the job and the categories the candidate has worked in the past;
- **prof_cat_int_similariity**, between 'prof_cat_in' (candidates) and 'prof_cat' (job positions) to find relationships between the categories required from the job and the categories the candidate is interested into;
- **hard_skills_similarities**, between 'Hard_skills' the candidate has and the required 'Hard_skills' for the job;
- **soft_skills_similarity**, between 'Soft_skills' the candidate has and the required 'Soft_skills' for the job;
- **language_similarity**, between the languages the candidate knows and the required languages;
- **cards_similarity**, between the cards/patents the candidate has and the ones they need for the job;
- **education_similarity**, between the education degree of the candidate and the required degree for the job;
- **overall_similarity**, the overall vector wide similarity;

Subsequently, we took advantage of the Latitude and Longitude to compute the distances between the location of the job offering and the one of the candidate's domicile.

As a last step, to increase the training speed and the stability of the models, we scaled all the features using a standard scaler.

2.2 Models used

In all the models we trained, we wanted the model to predict the probability of a couple Job-Candidate to be a match ("1" label). Then, among all the candidates for the same job, we chose that one with the highest probability of matching. It's important to say that we trained all the following models with the dataset made by the covariates that we have created.

First of all we decided to train a simple Logistic Regression, observing that the accuracy was not so bad (~ 0.745). This made us think that the covariates we created were good for the purpose.

Then we tried to apply Random Forest Classifier and XGBoost, optimizing the parameters using Cross Validation with a number of folds given by 5. This means that we decided to split the train dataset with the ratio $80\%/20\%$ for training/testing. With these models we add an accuracy of ~ 0.73, a bit lower compared with the logistic regression.

Lastly, we tried using **Neural Networks**. We built a simple model using dropout, early stopping and a small number of parameters in order to avoid overfitting. We tried also using L2 regularization, but we obtained more stable results without using it. In fact the early stopping guarantees an halting mechanism before the model overfits the dataset.

2.3 Our best model

In the end, the model that performed best on our covariates was the Neural Networks, basically a Multy Layer Perceptron. In the following figure we can see the simple structure of the network.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_45 (Dense)             (None, 128)               1408

dropout_30 (Dropout)         (None, 128)               0

dense_46 (Dense)             (None, 64)                8256

dropout_31 (Dropout)         (None, 64)                0

dense_47 (Dense)             (None, 1)                 65

=================================================================
Total params: 9729 (38.00 KB)
Trainable params: 9729 (38.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

We were able to have an accuracy on the test dataset of ~ 0.785, which was stable among the epochs. Below we can also see how the accuracy behaves during the training and validation phase.