Prediction Job Acceptance Decision For Better Candidate Selection

Diogo Valente Polónia
MEGI - FEUP
Porto, Portugal
up201806473@up.pt

Jesse Purkamo
MEGI - FEUP
Porto, Portugal
up202111212@up.pt

João Silva
MEGI - FEUP
Porto, Portugal
up201806335@up.pt

Inês Santos
MEGI - FEUP
Porto, Portugal
up201806346@up.pt

Joana Pina
MEGI - FEUP
Porto, Portugal
up201806349@up.pt

Filipe Silva
MEGI - FEUP
Porto, Portugal
up201806315@up.pt

Otto Veijalainen
MEGI - FEUP
Porto, Portugal
up202111487@up.pt

Margarida Sá
MEGI - FEUP
Porto, Portugal
up201806662@up.pt

Abstract

This project addresses the analysis of a dataset containing factors that lead a person to want to leave a certain company. The goal is to help this company (looking for good data scientists to hire) select the right candidates, from a course the company makes, who really want to work for them and, as a result, reduce costs and time spent during this selection process and assess the quality of training and planning of the course.

The original dataset was analyzed and pre-processed through many stages: data cleaning (deal with noisy data and exploratory analysis), variable transformation (categorical encoding), deal with missing data and outliers, study of correlation between attributes and data balancing.

Predictive models were constructed using several classification methods: OneR, Logistic Regression, Decision Tree, Naïve-Bayes, kNN, Random Forest, XGBoost, LGBM and Neural Networks. These algorithms were applied with different imputation and feature selection methods. We also applied clustering into the datasets.

The models were evaluated using Accuracy, F1-Score and G-Mean, along with some observations about processing time and model "realism". Overall, it is recommended that the company uses the lightweight, yet very effective, XGBoost algorithm (with Accuracy of 79,99%, F1-Score of 87,11% and a G-Mean of 72,37%) for candidate classification.

Keywords: ***Predictive Modeling, Data Processing, Data Cleaning, Data Transformation, Missing Data Imputation, Clustering, Cross Validation, Parameter Tuning, Classification, Evaluation metrics***

I. Introduction & Problem definition

A company wants to hire data scientists among people who successfully pass some courses conducted by the company. They want to know which of these candidates really want to work for the company after training and who will look for another employment.

The motivation to carry out this type of study is essentially due to **its importance in cost and time reduction**, as well as the **quality of training or the planning of the courses** and **categorization of candidates**.

With that said, this dataset is designed to understand the factors that lead a person to leave current job for HR researchers too. Identifying those key features that are affecting employee decision (it includes current credentials, demographics, experience, among others) is vital to predict the probability of a candidate to look for a new job or will work for the company, as well as interpreting affected factors on employee decision.

Overall, the goal with this assignment is to **predict the job acceptance decision for better candidate selection**. To fulfill that purpose is necessary to create a predictive model using **several classification techniques** appropriated for the problem in hand (classification problem with imbalanced data) to better serve the company's objectives.

II. Background theory

Briefly describe how the algorithms you used work

A. Dataset characterization

The dataset selected was the "Job Change". It has 19 158 observations and 14 features, 3 of which are numerical and 11 are categorical and have **high cardinality**. Besides, the two other main challenges offered by the dataset were:

- The **number of missing values** (9% in total with only 46% of the observations being complete) (Annex 1).
- The **data being imbalanced** (Annex 2) across almost all features. In case of the target variable, there were only 25% of positive values (data scientists that are looking for a job change).

B. Data Pre-Processing

Due to the aforementioned factors, we spent a large majority of our time in the pre-processing stage, before starting to construct our prediction model. A summary of this can be seen in Annex 3.

i. *Data Cleaning*

a. *Noisy Data*

First, we conducted a consistency analysis to deal with possible noisy data. After looking at the meaning of all features, we decided to:

- **Erase 44 observations** with no logical sense (data scientists that are enrolled in university (full time and part time) despite only having been in Primary School).
- **Fill in 1195 missing values** in the major discipline feature with "No Major" for data scientists that had been only in Primary School or High School.

b. *Exploratory Analysis*

To explore the dataset, we used different graphs and created data tables of all variables. The main conclusions extracted from this analysis were:

- All categorical features (except for the company size) have imbalanced data.
- The city feature follows, approximately, a Pareto distribution as can be seen in annex (about 20% of the cities appear in 80% of the observations) as can be seen in Annex 4;
- When comparing the company size with the company type, it was possible to conclude that as the company increases in size, the most likely it is a Private Limited Company (Pvt Ltd) as can be seen in Annex 5.

c. *Categorical Encoding*

After exploring the dataset, we decided to perform some transformations according to each variable type to improve all future models.

For all ordinal features (education level, company size, experience, and last new job) we used **Label Encoding** which is an approach to convert the levels to integers.

For the city feature, since it had 123 levels, we decide to use **Target Encoding** which is an approach to transform the categorical column to a new numeric column based on the value of target variable.

For the remaining categorical features, we decided to vary the transformation type we use in order to create different datasets. In some, no transformation was made, while in others we used **One-Hot Encoding** (dummy variables with linear dependence) while dropping redundant variables and **Clustering One-Hot Encoding** (clustered dummy variables).

d. *Binary Variable Clustering*

Although useful, One-Hot Encoding has the drawback of increasing data dimensionality, which can hinder model performance. To overcome this, we used **K-Means Clustering** and used the clusters for dummy variables instead of the categorical labels. Using the **Elbow Method**, the optimal number of clusters was identified (11 or 12 depending on the dataset) allowing us to reduce the number of variables to input in the models. However, in the end, **the models did not show improvements with clustering.**

e. *Missing Data*

One of the major challenges was the large amount of NA's. We found that only 46% of the observations had all its columns filled. The features with the higher percentage of NA's were the **company type (32%)**, **company size (31%)** and **gender (23.5%)** as can be seen in Annex 7. When exploring the combination of features that are missing together (Annex 6), it became clear that the company size and the company type combination is, by far, the most frequent missing one.

f. *Outliers*

Before constructing our models is also extremely important to verify the existence of outliers in all the numeric features (city development index and training hours). **We used boxplots** since this type of plot is usually used to detect outliers. However, one of the limitations of using the standard boxplot outlier rule is that it's not appropriated for highly asymmetric data. Because of that, we decided first to **check the data symmetry** graphically, using histograms, and analytically, calculating the skewness values.

For the training hours feature, the data observed is highly asymmetric (skewness value of 1.82). That **means we should use an adjusted boxplot** to verify the existence of outliers. No outliers were detected in this feature.

For the city development feature, the data observed is also highly asymmetric (skewness value of -1). We also use the adjusted boxplot and **verified the existence of two outliers**. Since we don't have information that these outliers are due to incorrectly entered or measured data, we shouldn't drop them immediately. After analyzing these values, **we decided to keep them** since they had virtually no impact in the performance of our models.

g. *Correlated Attributes*

One of our main concerns was related to the correlations between among all the independent features and between the independent features and the target variable. This is extremely helpful in terms of data reduction, since it leads to a better **feature selection** and a **dimensionality reduction**. Independent features that are highly correlated can be combined, creating a whole new feature. The correlation tests between the independent features were the following:

- **Polychoric Correlation** (between ordinal features)
- **Cramer-V Test** (between nominal features)
- **Chi-Square Test** (between features of different types)
- **Pearson Correlation** (between numeric features)

Even though the null hypothesis in the Chi-Square Test was rejected almost every time (which means that the two variables in study are related), we decided that **no significant correlations between independent features were to be found in this dataset**. We conclude this because the Chi-

Square Test is sensible to imbalance data, which lead to inaccurate results. The values obtained for each test can be found in Annex 8.

To study the relationship between the independent features and the target variable, we used the following correlation tests:

- **Point-Biserial Correlation** (between numeric features and the target variable)
- **Chi-Square Test** (between categorical features and the target variable)

From the Point-Biserial Correlation it was possible to conclude that **the training hours feature had almost no correlation with the target**. The wrapper algorithm used in feature selection for the application of Random Forest also did not consider training hours as a significant feature. For that reason, in most of the models attempted, the training hours feature was discarded.

The Chi-Square Test concluded that all the independent features were related to the target. However, this test suffers when dealing with imbalanced data as previously mentioned.

Since we are using non-linear classifiers, the best way to proceed is to train and test the model, using some form of cross-validation with and without the feature. The best overall solution we found was to **visualize the feature importance** of each feature present in the model and extract them accordingly.

ii. Data Transformation

III. METHODOLOGY AND DATA

Firstly, the dataset preparation was conducted, having developed several datasets ready to be tested, with only the selected the features and the attributes normalized. Afterwards, the models were developed with **10-fold cross validation and parameter tunning**, when needed; and ready to be tested with the unbalanced and balanced samples developed for each model. Lastly, with the predictions, **the metrics were evaluated and registered in the ModelResults Excel spreadsheet**, to check what the next adjustments to the datasets and models were going to be. This project was essentially an iterative process since the output of the predictive models were the input of the new trials of datasets and models.

A. Imputation methods

Since there's only missing data in categorical features, replacing the missing values with the mean or the median of the column would not make any sense. Also, **since almost all features were imbalanced, replacing with the mode of the column would not be the best approach**. Two different imputation methods were used then across multiple datasets:

i. ***KNN:*** the k-nearest neighbor algorithm has proven to be generally effective, not only as a classification method but also as an imputation one. This method finds the observations that are similar and places the most frequent values among the number of neighbors chosen. The number of neighbors is a parameter that needs to be chosen and, from the research we made, we decided to set **a k that was equal to the square of the total number of samples (k=3).**

ii. ***MICE with KNN:*** Multiple Imputation By Chained Equations (MICE) is an iterative process, where there is a continuous regression based on the other attributes. At the end of a cycle, all missing values are replaced. This algorithm runs until a convergent solution is found. Based on the research we made, we decided that **if missing data for a certain feature or sample is more than 5% then we would leave that feature out**. In those features (gender, company type and company size) we used kNN Imputation.

B. Training and test samples

Several datasets were tested for the different models performed. In the methods which required **parameter tuning**, (*NeuralNet, Decision Tree, kNN, XGBoost, Random Forest*), a division between training and test sample was also done. **The 10-fold cross validation method was applied** in all classification methods, to ensure the score of the predictive model does not depend on the selection of the train and test subsets, in parameter tuning cases, the cross-fold method was applied to the training subset.

C. Data Balancing

As previously mentioned, the original dataset was considerably unbalanced, thus some sampling methods were applied to the training sets to avoid performances of existing classifiers being **biased towards the majority class**.

i. ***Undersampling*****:** reduces substantially the dataset, by decreasing the majority class to even the dimension of the minority class, thus improving run times, but possibly losing information.

ii. ***Oversampling*****:** increases the dataset, by replicating random observations of the minority class to even the dimension of the majority class. This avoids information loss; however, it may lead to overfitting due to the repetition of observations of certain type.

iii. ***Both undersampling and oversampling:*** combines the two methods and reduces the impact of both.

iv. ***Rose*** **(Random Over-Sampling Examples):** draws artificial samples, through smoothed bootstrapping, from the feature space neighborhood around the minority class.

v. ***SMOTE*** **(Synthetic Minority Oversampling Technique):** creates new observations from the minority class synthetically, choosing points that lie in the line between rare observations and the nearest neighbor. This method only handles numerical and categorical variables, so we converted the non-numerical variables to factor and excluded the ones which had too many factors (SMOTE couldn't handle categorical variables with more than about 50 levels).

D. Classification methods

i. One Rule (OneR)

It is one of the simplest, yet quite accurate, algorithms to be applied. It basically **generates one rule for each predictor in the data and then, selects the rule with the smallest total error as its "one rule"**.

Regarding the Confusion Matrix obtained, this method shows significant predictability power. On the other hand,

OneR does not generate score or probability, which means evaluation charts (like ROC) are not applicable.

ii. Logistic Regression

The method of modeling probability of a discrete result given an input variable is called as logistic regression. It is a widely used categorizing method in the industry since it is an **efficient and simple classification model for binary and linear classification problems**. The most frequent logistic regression results have a binary outcome like, for example, true or false, yes or no. We made the logistical regression analysis so we could assess how different variables affect the target variable. This classification technique was used with multiple datasets and the results were all quite similar.

iii. Decision Tree

Logistical Decision Tree is a nonparametric statistical model that requires a large training dataset. This fact did not affect the normal samples used, however, datasets that suffered undersampling were affected. **The tree-shaped structure represents a set of decisions, in a certain path, that conduct to the creation of the classification rules**. There were three parameters to be tuned: the criteria to be used (**Gini** or **Information Gain**), the minimum number of observations in a leaf (minbucket) and the complex parameter (cp), which directly penalizes the complexity.

iv. Naïve-Bayes

Being a simple method, Naïve-Bayes is fast and easy to implement. It is applicable to binary dependent variables, as the variable "Target" is, according to the Bernoulli distribution. However, **it assumes the independence of variables**, which was proven not to be the case of the dataset with the chi-square test between different types of variables, thus the performance of this method may be hindered. No extra data preparation was needed.

v. K Nearest Neighbors (KNN)

K Nearest Neighbors develops its models around the similarity of data points, which means the proximity of them. Considering the variable "Target" is binary, the model delivers the value voted by the defined neighbors. The number of neighbors defines how similar the data is, and it may not be too small to avoid missing the most similar case and include outliers and also not too big not to include non-similar data. This way, this parameter was tuned to be more adequate to the dataset in question, having the maximum been defined to 50 neighbors and never been close to that number. However, kNN is computationally heavy and may be influenced by irrelevant attributes. No extra data preparation was needed, due to the categorical variables being already computed in dummy variables.

vi. Random Forest

This classification method consists of creating a **random set of trees in a random subset of the training data in order to obtain a more accurate and stable prediction**. In simple words, Random Forest searches for the best feature from a random subset of features providing more randomness to the model and results in a better and more accurate model.

Before applying this model, we did **parameter tuning** using the function tuneRF, which returns an optimized value of the parameter mtry (provided lowest error rate). We also tried the function tune (package "randomForestSRC") which returns a list of all Random Forest parameters optimized, but the results were not good enough as we expected them to be. Besides that, we used a **wrapper algorithm for feature selection called Boruta**, which is only used for Random Forest. The best overall solution we found for each dataset was a list of significant variables in the model to be used.

Finally, we ran the Random Forest model with the optimal mtry obtained with tuneRF, 200 trees (parameter ntree) and the default node size value applied to the dataset with only the significant features identified by Boruta.

vi. Extreme Gradient Boosting (XGBoost)

Having a parallelizable library (the core algorithm can run on clusters of GPUs or even across a network of computers) makes XGBoost popular on solving machine learning challenges. Moreover, this method is known for assuring fast and accurate results, so it consistently outperforms any other algorithms aimed at supervised learning tasks.

Besides that, **XGBoost supports missing values by default**, so it can handle them without an imputation preprocessing. Then, to reprocess the dataset, apart only from basic data cleaning operations, numeric features need to be scaled and categorical features encoded, since this method accepts only numeric features as input. For that, normalization and one-hot encoding were required.

To build and optimize the model, **hyper parameters were tuned using a grid search**. For that, some tuning parameters were held constant, like "eta", "min_child_weight" and some others by default values. The optimal values obtained were the number of trees and the depth.

vii. Light Gradient Boosting Machine (LGBM)

Light GBM is a fast (hence the word "light") and **high-performance gradient boosting framework based on decision tree algorithms**. It may produce slightly better accuracy results than other boosting models, however it has the big disadvantage of being sensitive to overfitting as it produces much complex trees. Thus, since the difference from XGBoost model results were not significant, the model was not developed much deeper.

viii. Neural Network

Neural Network method simulates the recognition pattern of the human brain, comprising node layers (the input one, the hidden layers and the output) and the nodes themselves (the neurons), which have an associated weight and threshold. Being the core principle of deep learning, it has promising performances, since it is **robust to noisy data**, and is **suitable for several types of dependent variables**, including binary, as Target, but is computationally complex to run. As mentioned previously, there were parameters to be tuned in this method: the hidden layers and the neurons.

E. Evaluation metrics

Choosing an appropriate metric is challenging generally in applied machine learning. For this specific dataset, the metrics which weighted more were the **F1 Score** and **G-mean**, since they are the most adequate. Besides, the accuracy was used, even though limitations were recognized considering the unbalance of the data.

i. Accuracy

The **accuracy** is one of the most typically reported model evaluation metric. Accuracy is a classification model metric that counts the number of correct predictions as a proportion of the total number of predictions made. **This can be reliable metric when the data is in balance, which is not the case of the dataset in question**.

ii. F1 Score

The **F1-score** takes the harmonic mean of a classifier's precision and recall on the positive class, creating a single statistic. It's mostly used to **compare two classifiers' performance** and is mostly adequate to be used for unbalanced datasets.

iii. G-Mean

The geometric mean, or **G-Mean**, is a **measure that combines sensitivity and specificity into a single value that balances both objectives**. The G-Mean assesses the balance between majority and minority classification performance. Even though the negative occurrences are accurately categorized, a low G-Mean indicates poor performance in the categorization of positive cases. This criterion is necessary to avoid underfitting the positive class and overfitting the negative class. This metric is once again adequate to this unbalanced dataset.

IV. Results & Discussion

A. Classification Results

i. Global Analysis

Following the metrics chosen to evaluate best the models (F1 Score, G-Mean and Accuracy), **the results obtained were relatively similar for all models**, with accuracies between 70-80%, F1 scores between 77-90% and G-Means being the attribute which varied most in between 50-72%.

KNN was the model with lower values of F1 (77,77%), but still reaching the usual values with some datasets. On the other hand, LGBM was the model with lower values of G-Means (61,67%), whereas XGBoost outperformed all the models in this metric (51,33%).

In terms of datasets, from all the combinations tried, the datasets with better performance were the ones which had the observations with more than 4NAs removed and **One Hot Encoding applied**, with a combination of KNN and MICE for missing data imputation. From this point, datasets were normalized and clustered, having originated also good results.

Although there is an issue with the unbalanced data, when processing the models, **the performance of oversampling, undersampling, and the other methods of sampling did not have a considerable impact**, having the unbalanced data better results, most of the times.

ii. Best models

XGBoost: For different datasets with different imputation methods, this model revealed to be one of the most promising, for combining fast and accurate results. - great values of accuracy (79.96%), F1 score (87.10%) and G-mean (72.49%) for the *dataset_2-MICE-Normalized*.

For its speed (less than 2 min), it became easy to test, iterate and find potential errors. In specific, the model was runned by using tunned hyperparameters and its output was *n_rounds* = 100 and *max_depth* = 7.

Neural Network: Considered as the slower model (running time more than 1h) but showed also great results. The accuracy for the best dataset *(dataset_2-MICE-Normalized)* was 80,76%, F1 of 87.17% and G-mean of 74.54%. Also, by doing the parameter tunning, the output of it showed a value of *hidden* = 3.

Decision tree: From this model, good results were also seen by using the dataset *dataset_1-MICE* and they were: accuracy of 79.10%, F1 of 86.44% and G-mean of 66.59%, with optimal values associated to the parameter tuning of *criteria* = "information", *cp* = 0 and *min_num_obs* = 27.

Having all the above in consideration, we **recommend the usage of the XGBoost model**, due to its low processing needs and high-performance results. Besides, this last model is not very susceptible with changes on the attributes of the dataset, leading to low needs of preprocessing.

iii. Feature Importance

Looking to the **feature importance** returned by XGBoost in Annex 9, it's possible to see that, respectively, *city_dev_index*, *training_hours*, *experience*, *company_size* and *company_type_PvtLtd* were the variables that dominate the other features, clearly standing out as the most important predictors. This **feature importance analysis allows the company to understand better their candidates and pursue better recruitment strategies**.

V. Conclusions

After facing some issues with the missing data, unbalanced data, categorical variables with high cardinality, we managed to deal with the data, obtaining reasonable samples to be tested in the classification models, via continuous iteration and agile methods for dataset methods.

Considering the methodology adopted, more models could have been developed, such as **SVM** and **Extra Trees**, to seek for better results. Besides, more changes could have been done in the datasets, trying to **explore deeper the cardinality** issue and the junction of some levels of the factors.

Despite Neural Net having better performance metrics, the XGBoost is more promising due to the **high speed of obtaining similar results**. Considering this model and the results obtained with it, the company can improve its recruitment strategy through **city targeting**, with city-specific course marketing and advertising; also, by only

accepting candidates with a certain number of training hours and lastly by developing **job surveys in courses**, that inquire about attributes that had great impact: experience, years since last job and size/ type of current.

In the future, the company could start a **candidate database and increase the amount of data collected for this study**, such as course grades, time until the employee leaves the company and the age.

BIBLIOGRAPHY

V. Agarwal, "Outlier detection with Boxplots," medium, 30 11 2019. Retrieved May 12, 2022, from: https://medium.com/@agarwal.vishal819/outlier-detection-with-boxplots-1b6757fafa21.´

K. Grace-Martin, "Outliers: To Drop or Not to Drop," The Analysis Factor. Retrieved May 12, 2022, from: https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/

"Random Forest Approach for Classification in R Programming." GeeksforGeeks, 8 July 2020, https://www.geeksforgeeks.org/random-forest-approach-for-classification-in-r-programming/. Accessed 14 May 2022.

"How to find the optimal value of K in KNN?" Towards Data Science, 23 May 2020, https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb. Accessed 14 May 2022.

M. Alice, "Imputing Missing Data with R," datascience+, 14 05 2018. Retrieved May 12, 2022, from: https://datascienceplus.com/imputing-missing-data-with-r-mice-package/

unbalanced classes - ROSE and SMOTE oversampling methods - Cross Validated. (2021, December 3). Cross Validated. Retrieved May 13, 2022, from https://stats.stackexchange.com/questions/166458/rose-and-smote-oversampling-methods

Goyal, C. (2021, May 21). Importance of Cross Validation: Are Evaluation Metrics enough? Analytics Vidhya. Retrieved May 13, 2022, from https://www.analyticsvidhya.com/blog/2021/05/importance-of-cross-validation-are-evaluation-metrics-enough/

Vadapalli, P. (2020, December 11). Naive Bayes Classifier: Pros & Cons, Applications & Types Explained. upGrad. Retrieved May 13, 2022, from https://www.upgrad.com/blog/naive-bayes-classifier/

Harrison, O. (2018, September 10). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Towards Data Science. Retrieved May 13, 2022, from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

How to tune Hyper parameters using Grid Search in R? (2022, May 2). ProjectPro. Retrieved May 13, 2022, from https://www.projectpro.io/recipes/tune-hyper-parameters-grid-search-r

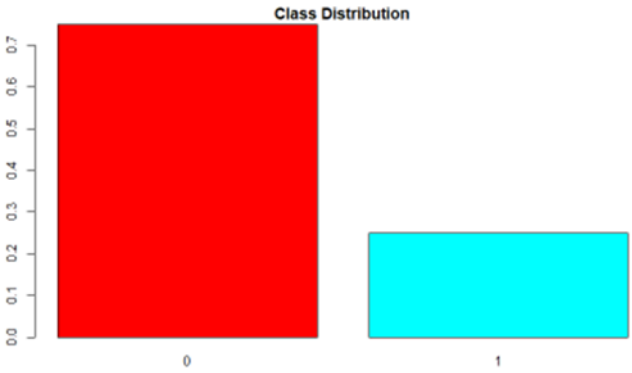
Gould, D. (2021, April 7). Beginner's Guide to XGBoost for Classification Problems. Towards Data Science. Retrieved May 13, 2022, from https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems-50f75aac5390

Surana, S. (n.d.). What is Light GBM? Advantages & Disadvantages? Light GBM vs XGBoost? | Data Science and Machine Learning. Kaggle. Retrieved May 13, 2022, from https://www.kaggle.com/general/264327 https://www.kaggle.com/general/264327

[1]APPENDIX

Number of NA's	Frequency %	
0	46,7%	Remain
1	66,1%	
2	85,2%	
3	95,4%	
4	98,7%	
5	99,6%	Delete
6	99,9%	

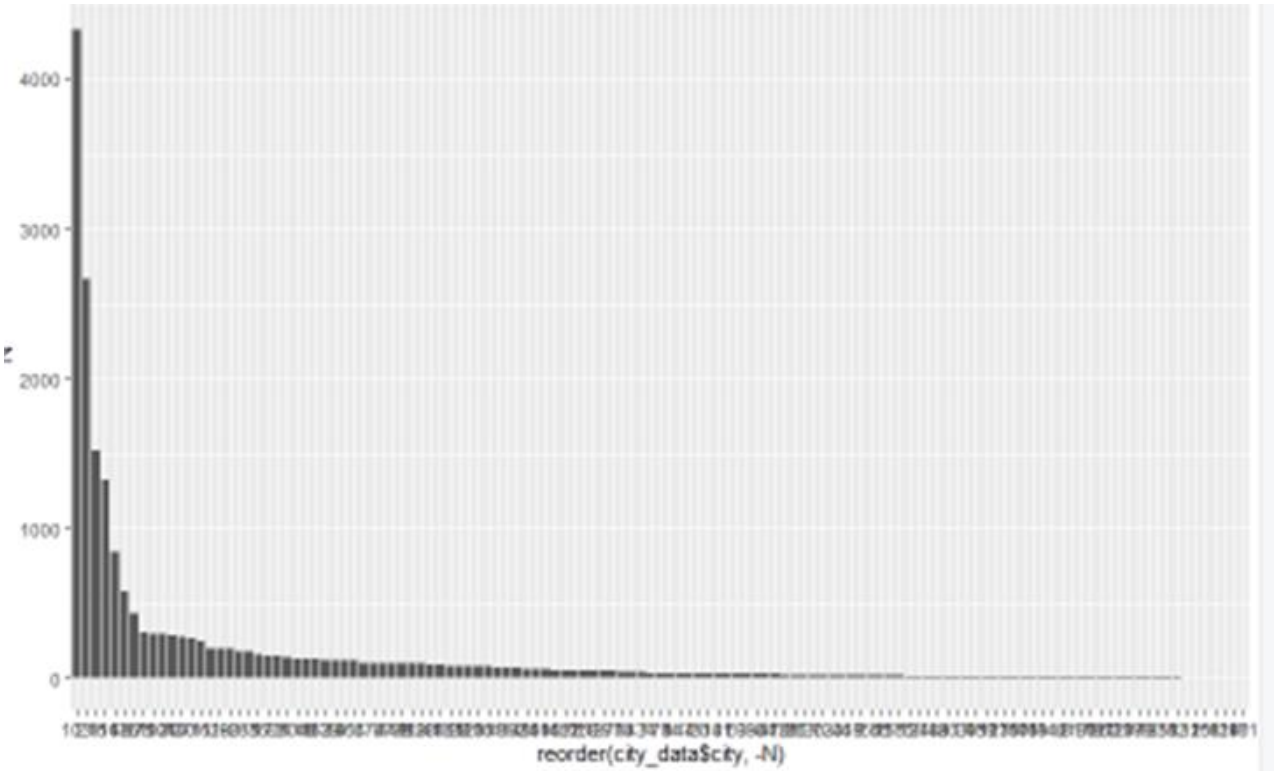
Annex 1 - Distribution of NA's per row and criteria for elimination



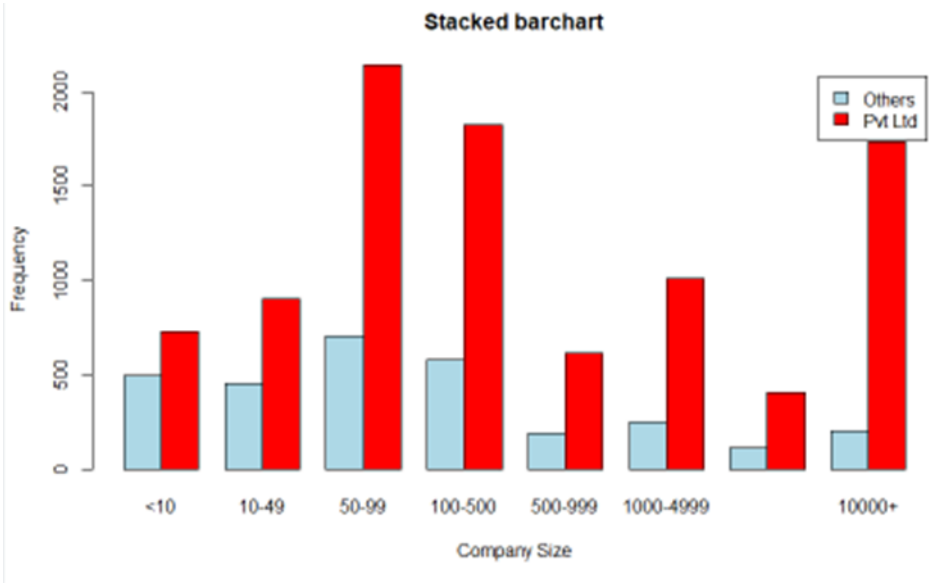
Annex 2 - Data Balance

Pre-Processing	Sections of Analysis	What was done
Data Cleaning	Noisy Data	Fill in NA with "No Major" for Primary School, and High School with no enrollement (1195 obs.) Erase observations: Enrolled University with Primary School (44 obs.)
	Exploratory Analysis	Data Tables and Barplots of each feature
	Categorical Encoding	Ordinal Features: Label Encoding Nominal Features: One-Hot Encoding (dummy variables with linear dependence) City Feature: Target Encoding
	Missing Data	Removing observations with at least k unknown values Filling in the Unknowns by Exploring Similarities between Cases (kNN) Filling in the Unknowns by Using Multivariate Imputation by Chained Equations (MICE)
	Outliers	Adjusted Box Plots for numeric features with high skewness Standard Box Plots for numeric features with normal skewness
Transformation	Feature Selection	Correlation Analysis: Point-Biserial Correlation between Numeric Features and Target Correlation Analysis: Chi-Square Test between Categorical Features and Target Correlation Analysis: Polychoric Correlation between Ordinal Features Correlation Analysis: Cramer-V Test between Nominal Features Boruta for Random Forest
Data Balancing	Dealing with Imbalanced Data	Oversampling, Undersampling, Both, ROSE and SMOTE
Dataset Preparation	Factors to Take in Consideration	Removing observations with at least 4 or 5 Unknown Values Before and after Categorical Encoding Knn imputation for missing values or MICE + kNN Imputation for missing values Removing observations with missing values from the Experience Feature (65 obs.) Data Min-Max Normalization

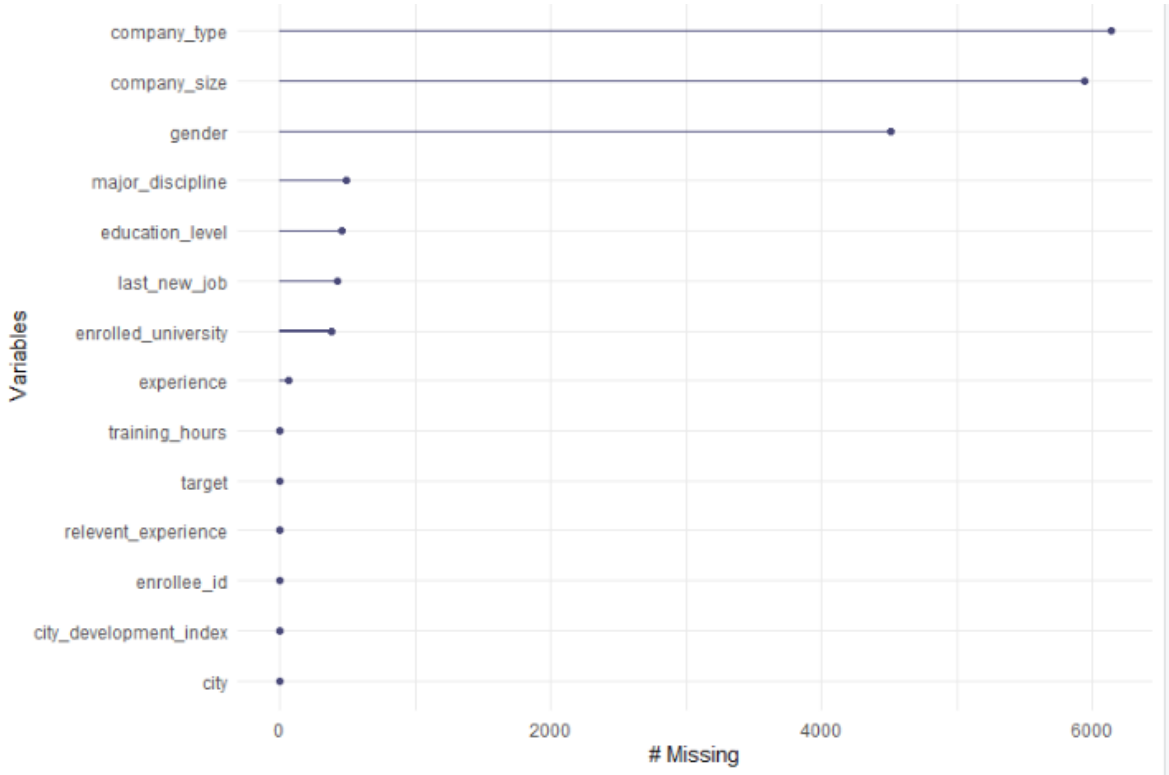
Annex 3 - Data Pre-Processing



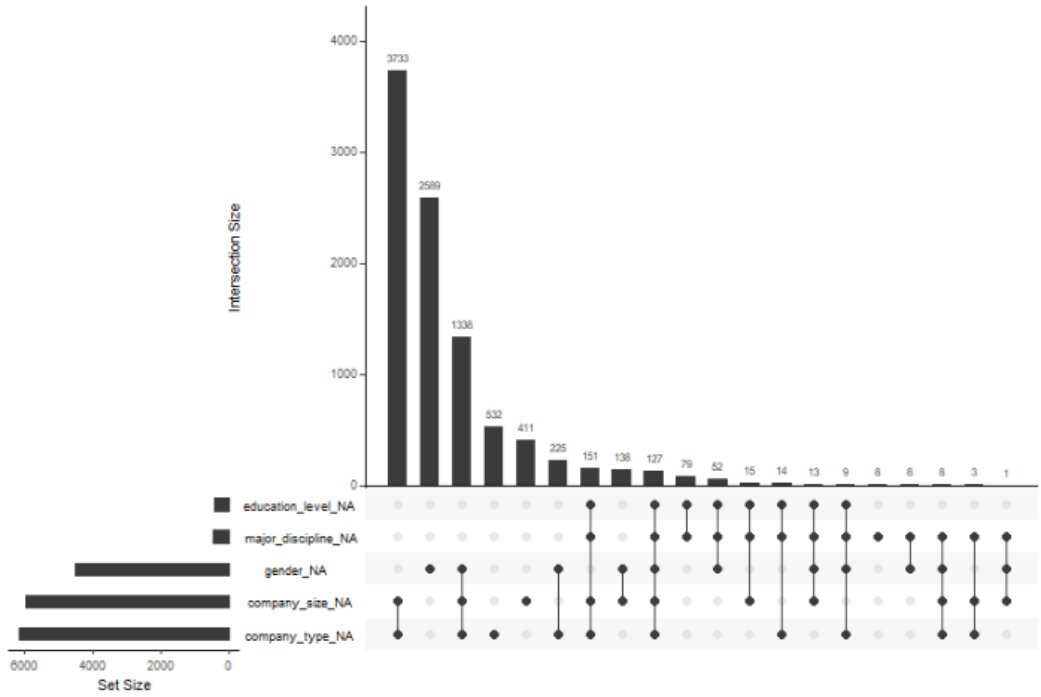
Annex 4 - Feature 'City' Distribution



Annex 5 - Relationship between Company Type and Company Size (Company Type was transformed into a two-level factor)



Annex 6 – Number of NA's per variable



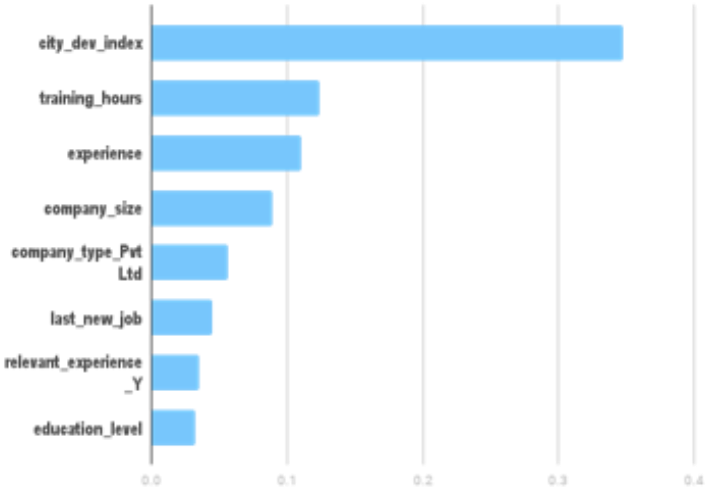
Annex 7 - Combination of variables that occur to be missing together

	Pearson Correlation
	Point-Biserial Correlation
	Cramer-V Test
	Polychoric Correlation
	Chi-Square Test

CORRELATION ANALYSIS

FEATURES	City	City Development Index	Gender	Relevant Experience	Enrolled University	Education Level	Major Discipline	Experience	Company Size	Company Type	Last New Job	Training Hours	**Target**
City		0	0,0864	0,1469	0,1665	0	0,1162	0	0	0,1013	0	0	0
City Development Index			0	0	0	0	0	0	0	0	0	0,0013	0,34
Gender				0,0514	0,0181	0	0,0607	0	0,3758	0,0314	0,0001	0,0044	0,0128
Relevant Experience					0,3824	0	0,2964	0	0	0,1704	0	0,2643	0
Enrolled University						0	0,103	0	0	0,0666	0	0,4015	0
Education Level							0	0,317	0,1068	0	0,2805	0,04117	0
Major Discipline								0	0	0,0288	0	0,5544	0
Experience									0,1212	0	0,5123	0,6476	0
Company Size										0	0,1094	0,6618	0
Company Type											0	0,6005	0
Last New Job												0,6543	0
Training Hours													0,02
Target													

Annex 8 – Correlation Table



Annex 9 – Feature Importance returned by XGBoost