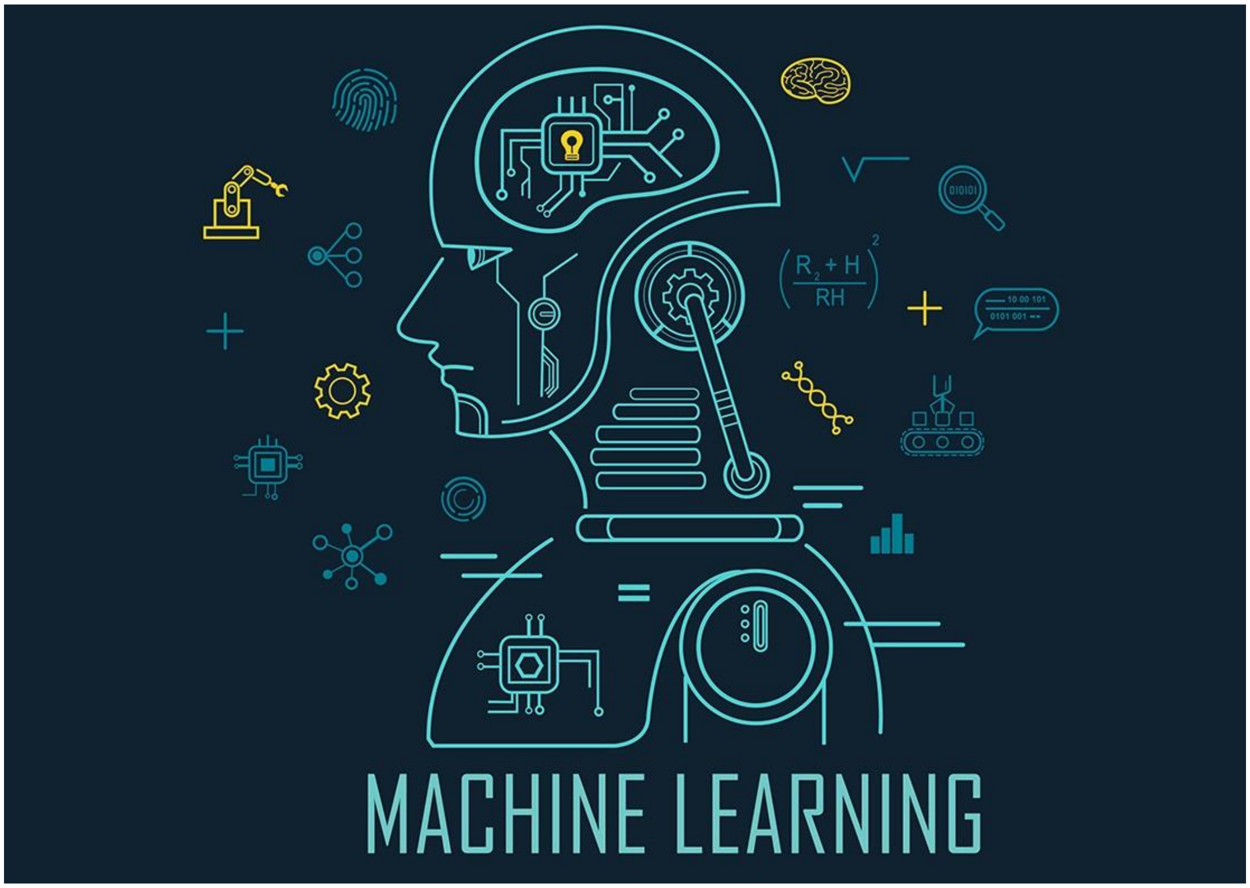
Machine Learning Report

CIE 417 /



Mohamed Abdelwahed	**201801978**
Omar Gaballah	**201801697**
Ahmed Tarek	**201801597**

Problem definition and motivation:

We are working on designing a model to help HRs hire some data scientists to their company. Many people are applying for training. So, companies would like to know which of applicant are suitable for the company , so building this model will help to reduce the cost and time for HRs.

This dataset is designed to understand the factors that lead a person to apply for a job. By a model that uses the current credentials,demographics,experience data you will predict the probability of a candidate to look for a new job or will work for the company.

Dataset:

The dataset is collected from employees in a company which is active in Big Data and Data Science. It is collected from the employees data which is

[enrollee_id : Unique ID for candidate

city: City code

city_ development _index : Development index of the city (scaled)

gender: Gender of candidate

relevent_experience: Relevant experience of candidate

enrolled_university: Type of University course enrolled if any

education_level: Education level of candidate

major_discipline :Education major discipline of candidate

experience: Candidate total experience in years

company_size: No of employees in current employer's company

company_type : Type of current employer

lastnewjob: Difference in years between previous job and current job training_hours: training hours completed

target: 0 – Not looking for job change, 1 – Looking for a job change]

We noticed that these features are sufficient for training our model.

Pre-Processing:

Our data set was an imbalanced dataset with many missing values. We have handled the missing values in two different ways. In both ways we had dropped “company,& company_type” columns.

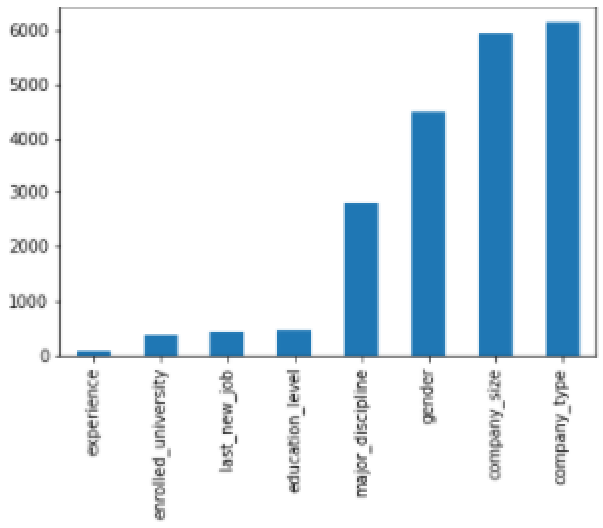


Figure 1: missing values sorted

First way

We had replaced the nulls in the categorical columns with a string value “missing” -which represent a dummy variable- while we replaced missing values in the numerical columns with the median as most of the columns were skewed and later on made one hot encoding, so we have dealt with the data without changing or assuming any features. The resulting data had the number of columns equal =28 which is much greater than the number of columns in the original data which was= 14.

Second way

We have imputed the missing values with KNN imputer, which is a multi-feature imputer which imputes the value with the mean of the nearest observations. The knn imputer has to have numerical inputs as a result, we had to assign numerical values to most of the categorical columns. In our case KNN imputer had many advantages

1. We had many columns that can be classified as ordinal categorical data such as "`education_level,education level`"
2. Other columns can be classified as binary data such as (gender, relevant experience) so now we don't have to add a category "missing" and make one hot encoder
3. The dimension of the data after imputing and cleaning has only 17 columns which is much less than the way we handled the missing values in the first way

After imputing the missing values and due to having lower dimensionality than the first methodology we have used "SMOTE" to oversample our data and increase the number of observations that have targeted the minor classifier.

Methodology: Model selection, model evaluation and model development

Approach; we have tried four models from which we have studied in the course which are logistic regression, Support Vector machine, Decision trees, and Random Forest Trees. Regarding model evaluation, we have split our data into train and test data with test percentage= 0.15.

Also in every model we have used a K-Fold cross validation with K = 10. And used grid search to tune our hyper parameters models.

A further discussion on every methodology is tackled below.

Support vector machines:

This model creates a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

Hyper-parameters: a grid search was used to tune the parameters based on the best results; the parameters considered for model are:

- C
- Gamma
- kernal

For evaluating for the support vector machine

Support Vector machine's best score using the knn imputer was 0.784 . And without the knn imputer was 0.793

	KNN imputed data+smote	Data encoded with First way+smote
Accuracy	0.784	0.793
Recall	0.644	0.665
Precision	0.755	0.765
F1- score	0.865	0.884

Decision Trees and Random forest:

Hyper-parameters: a grid search was used to tune the parameters based on the best results; the parameters considered for both models are:

- Criterion
- Minimum sample split
- Maximum depth

For the decision trees

	Data encoded with First way+smote	KNN imputed data + smote
Accuracy	0.83524	0.8442
Recall	0.86430	0.8448
Precision	0.83975	0.8442
F1- score	0.8165680	0.843

Random Forest:

Random Forest's best score using the knn imputer was 0.8576395595904964. And without the knn imputer was 0.8587985319683213

	Data encoded with First way+smote
Accuracy	**0.85366**
Recall	**0.86221**
Precision	**0.8547**
F1- score	**0.8474692**

Logic Regression:

It is a classification algorithm that can be used to find the probability of even success and failure.

This model is used when the dependent variable (our target) is categorical or binary.

The parameters considered for this model are:

- Solver
- Penality
- C

Logistic Regression's best score using the knn imputer was 0.7313502028201661 . And without the knn imputer was 0.7441761638014295

	KNN imputed data + smote	Data encoded with First way+smote
Accuracy Score	73.31%	75.22%
Recall Score	0.74	0.76
F1 Score	0.73	0.75
Precision Score	0.7289	0.746

Implementation:

From sklearn we used for

1. Model selection: Grid search and it is used for hyperparameter tuning to choose the best with the best accuracy.
2. Trees: Decision Tree classifier
3. Ensemble : Random Forest
4. Imputers: KNNimputer
5. Preprocessing: One hot encoder, Standard scaler and column transformation
6. Imblearn: used SMOTE, Oversampler and undersampling - but later on we used only smote in the submitted code, however we used the other two in backup notebooks -
7. SVC: for The support vector machine models.

We used other libraries such as

1. Pandas and numpy for reading and manipulating data in csv files
2. Matplotlib and seaborn for visualization

Performance of the model:

The best model used for our dataset is the Random forest. Random forest tree and decision tree: use different methodology than the logistic and use the entropy to get the information gain and start to divide the feature space into regions and start classifying the data.

The random forest shows an 0.8547 for precision.

Conclusion:

A. Final evaluation of approach

Compared to the model submitted on kaggle, our model adds the idea of margin and tolerates some amount of errors which in return give us a more generalizable mode. Our model ranges in the same evaluation scores as the models submitted except for the models that use XGBOOST, in that case it is a bit better.

B. What we have learnt

- **Dealing with imbalanced data**

We have for the first time tried to deal with imbalanced dataset, we knew there are three approaches to deal with the imbalance data:

1. Pre-processing: so before training the data we may do either undersampling or over sampling; in our project we have come across different techniques which are SMOTE. However, only the SMOTE technique gave us reliable results so we disregarded the other techniques and did not include them in the submitted code
2. Cost-sensitivity: during the training we may focus on decreasing the error in a certain target class; In our project we have not used this methodology as we did not know whether the low number of the minor class was due to the fact that the chosen sample was not representative or due to the fact that the lower numbers in the target variable yes“ -which is the minor class- was predicted as not all the people who would take the training would be interested in joining the company.

Boosting: after training the models we may use boosting for treating the imbalance; however we have not searched a lot in that case how it is done.

Handling missing values

We have looked for how generally data scientists deal with the missing values; we have learnt that there are univariate and multivariate imputers. We have read that statistically multivariate imputers are more efficient and there are two common models

KNN imputer: in that imputer we impute the missing value with the mean of the k nearest observations. It has an important advantage as it is less complex than the iterative imputer

Iterative imputer: -we had a qualitative understanding for how it works -

it tries to use the input features to draw a regression that we use later to impute the missing values.

C. Ways to improve our model

In case of improving the models we have used; We have tried many variations in the hyper-parameters of the models that we have used. So, we don't have anything on our minds on how to improve the scores in the models that we have used.

In the case of trying new models; We may try to use other models than that we have taken in the course. When we investigated the other notebooks to see their accuracy, we found that they get high scores only when they use “XGBOOST” and “LightGBM”. However, on using the models that we have used we have got higher scores.