ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MASTER OF SCIENCE THESIS

Person entity identification and matching using face recognition and machine learning algorithms

Author:
Enkeleda BOCAJ

Supervisor:
Dr. Vassilis VASSALOS

A thesis submitted in fulfilment of the requirements for the degree of Master of Science in Data Science

in the

Department of Informatics,
Athens University of Economics and Business

Abstract

The vast and rapidly increasing quantity of structured and semi-structured data sets has caused the Entity Resolution (ER) problem. The capability to extract and combine data from multiple sources or duplicated records, in order to create a single resolved entity is essential in order to leverage many aspects of the data. Indeed, several blocking techniques are typically employed to eliminate the problem of ER. However, the heterogeneity of structure and semi-structured data, multi dynamics, loose schema binding impose new challenges to the entity resolution and trigger the scientific community to create more efficient techniques. In the scope of this thesis, entity resolution problem and techniques are researched and studied upon, as well as software packages and frameworks. Furthermore, several machine learning and deep learning techniques has been implemented in order to extract knowledge from entities and provide an entity resolution solution approach. In this sense, attribute-agnostic mechanism has been used to extract feature from the values of entity profiles and assist on entity matching. Classification methods have been implemented and evaluated in order to create an efficient and effective predicted model.

Keywords: entity retrieval, classification, machine learning, augmented data, face recognition

Περίληψη

Η ραγδαία αύξηση παράγωγης δεδομένων, την τελευταία δεκαετία, έχει δημιουργήσει αρκετά προβλήματα, ένα από αυτά είναι και το ταίριασμα οντοτήτων. Λόγου του τεράστιου όγκου πληροφορίας δεν είναι εύκολο τα δεδομένα να ελέγχονται και για αυτό το λόγο πολλές φορές μπορεί η ίδια πληροφορία να είναι παραπάνω από μια φορά αποθηκευμένοι. Αυτό έχει σαν αποτέλεσμα να δαπανιούνται πολύ υπολογιστικοί πόροι άδικα όπως π.χ. χώρος αποθήκευσης. Στην παρούσα διπλωματική εργασία μελετάμε το φαινόμενο του ταιριάσματος οντοτήτων και προτείνουμε μια εναλλακτική λύση χρησιμοποιώντας τεχνικές μηχανικής μάθησης. Επιπροσθέτως, γίνεται ιστορική μελέτη του τι υπάρχει είδη και πως θα μπορούσαμε να συνεισφέρουμε στο τομέα αυτόν. Παράλληλα, παρουσιάζονται μέθοδοι ταξινόμησης και πώς αυτοί μπορούν να συνεισφέρουν στην επίλυση του προβλήματος. Στην συνέχεια έχουμε τον τρόπο υλοποίησης και τι χρησιμοποιούμε ώστε να επιτύχουμε τον στόχο μας. Τέλος, ξεδιπλώνονται τα αποτελέσματα των πειραμάτων, το τι δυσκολίες αντιμετωπίσαμε και τα συμπεράσματα μας. Για την υλοποίηση χρησιμοποιήθηκαν πολλές βιβλιοθήκες της παιθον, η οποία ήταν και η γλώσσα προγραμματισμού.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Vassilis Vassalos for the support that he continuously provided during my research work and thesis writing. Besides my supervisor, I would like to thank also Dimitris Kotsakos and Panagiotis Kasnesis for the stimulating discussions and insightful comments.

Last but not the least, I would like to thank Workable Inc. for giving me the chance to fulfill my master thesis under their supervision.

Contents

1. Introduction

During the last decades, remarkable progress has been made in important scientific and technological fields, including engineering and technology, medical and health science, and social science. Thus, the computational capabilities, cloud computing technologies, smart devices and the Internet of Things mentality has contributed to technological revolution. The intensity of technological innovations has led on having a data tsunami. Indeed, enormous amount of information is being produced every second thought all the smart devices which surround us. This information could be in any type of data form (such as text documents, images, audio clips etc) and could be classification in one of the three main data types, structured data, unstructured data and semi-structured data.

The Internet of Things, machine leaning (ML) and artificial intelligence (AI) have been momentously gaining grown in the society which will lead to enormous amounts of data flooding systems. In this sense, due to the sheer quantity of data, data quality issues, idiosyncratic data collection or schema variations, independently the storage mechanism or the storage type, makes the necessity of efficient prioritization and somehow organization methods and frameworks vital. Indeed, as the volume and velocity of the data grow, problems associated with entity match are being increased, making the entity resolution (ER) [1] and matching applications and algorithms crucial.

Thus, one of the major challenges of entity resolution is matching loosely structured (semi-structured) entities that arise from Web data and IoT or already existing information from data lakes and data warehouses. As a matter of fact, scientific community and several companies highlight and address the ER problems. For instance, the Yago [2] (a huge semantic knowledge base) features 10M entities in 350K classes and 120M facts (RDF triples) for 100 relations, while Freebase features 25M entities described by 2000 topics and 100M facts using 4000 properties.

In this work, an external study of entity resolution and machine learning classification algorithms are presented. Furthermore, our approach is described in combination with the material and methods were used. Then, we describe the implementation of the approach. Continually, there are discussions about the experiments and the limitations that exists. Finally, we have the conclusion and future work.

2. Literature Review

Since the beginning of 1959, when was the first time that entity matching was defined, a plethora of approaches, algorithms and frameworks have been proposed and implemented in the literature. Indeed, various facets has been considered from different scientific communities aiming to effectively identifying data which describe the same real-world entities. There are several works that overview the literature and the existing work in the domain [3,4], There is different type of approaches, some of them employs similarity methods using transformations [5] and relationships between the data [6] or others that span from the string similarity metrics [7]. The Sorted Neighborhood approach [8] sorts records based on their BKV and then slides a window of fixed size over them, comparing the records that it contains. On the other hand, String Map method [9] maps the BKV of each record to a multidimensional Euclidean space and employs suitable data structures for efficiently identifying pairs of similar records. Aiming to associate each record with a Blocking Key Value (BKV) [9] summarizing the values of selected attributes and then operate exclusively on it. Alternatively, the q-grams based blocking presented in [11] builds overlapping clusters of records that share at least one q-gram (i.e., sub-string of length q) of their BKV.

Moreover, Canopy clustering [12] employs a cheap string similarity metric for building high-dimensional overlapping blocks, whereas the Suffix Arrays approach, considers the suffixes of the BKV instead. As marked in [10], the performance of existing blocking methods depends on a wealth of application- and data-specific parameters. The error characteristics in the data to be linked, for example, affect the performance of the similarity metric used by canopy clustering, whereas the distribution of values affects the optimal window size of the Sorted Neighborhood approach. To facilitate parameter setting, several proposed methods automate this procedure with the help of machine learning. In this sense, [13] models this problem as learning disjunctive sets of conjunctions that consist of an attribute (used for blocking) and a method (used for comparing the corresponding values). A typical machine learning algorithm is then applied, yielding a significant improvement over the parameters manually set by experts. Similarly, [14] considers disjunctions of blocking predicates (i.e., conjunctions of attributes and methods) along with predicates combined in disjunctive normal form (DNF) and compares them against canopy clustering. The experimental evaluation suggests that DNF predicates have the best performance, with the disjunctive ones outperforming solely canopy clustering. Hence, these approaches are only applicable to data sets having a schema with a restricted number of distinct attributes. An alternative approach for improving blocking techniques is introduced in [15]. Where both the effectiveness and the efficiency of the ER process can be improved.

3. Materials and Methods

In order to accomplish and implement the tasks of the thesis we researched some main algorithms and methods that already exist and continually, we made use of them. Indeed, in the next subchapters are being explained briefly what material we used and witch methods.

3.1 Entity resolution (ER)

ER [1](also referred as entity matching, record linkage, duplicate identification etc. in the literature) is the process of identifying and disambiguating records and mentions that represent the same real-world entities across and within datasets by linking and grouping. For instance, in a mailing list, a physical address could be represented multiple entries which slightly differ, e.g. having some missing information, abbreviations or containing some spelling differentiations. ER can contribute on the redaction of the complexity by suggesting canonicalized references to particular entities by deduplicated or linking them or having an effective use of predictive analytics to optimize targeting, thresholding, and resource management.

The three of the main tasks that entity resolution includes are deduplication, record linkage, and canonicalization:

- Deduplication: aims to identify and eliminate the duplicated copies of repeated data by clustering the records or mentions that correspond to the same entity.
- Record linkage aims to identify records that are referenced to the same entity across different sources (particularly in already normalized data).
- Canonicalization aims to convert data with more than one possible representation form into a standard one, address to entity disambiguation.

However, there are several significant challenges, such as missing values, changing attributes or their formatting, poor data entry, language ambiguity or even abbreviations, that ER discipline should solve while addressing these tasks. Indeed, ER does not only include databases and information retrieval, but also contains natural language processing and machine learning techniques.

3.1.1 Entity resolution software and frameworks

ER is a challenging task that have been considered and researched from virous facets and different communities. Several approaches have been proposed and numerous frameworks have been implemented addressing to accomplish ER tasks. Having as common goal to achieve the high-level requirements and desiderata of an ER that are associated with the effectiveness, efficiency, genericity and low manual effort of the framework.

- Effectiveness represent the achieved matched results with respect to precision and recall (i.e., in the result should only be included the real corresponding entities).
- Efficiency is associated with the processing time of the framework, which should be performed fast even for large datasets.
- Genericity, offline/online matching is referred to the generalization of the model which should be performing well independently the variation of the data and data models. It should also be applicable on both online (interactive data integration steps) and offline (processing data from databases) matching tasks.
- Low manual effort/self-tuning is referred to the simplicity of the framework use. Indeed, the effort to employ the ER framework should be as low as possible, containing as many as self-tuning manners

Achieving all these goals simultaneously requires the flexible combination and customization of different machine learning techniques and match methods. However, it is very essential not to create conflicts between the above requirements and have a balanced framework. For instance, if the framework has high level effectiveness, efficient and generalized and on the hand, it needs high manual effort to use it, means that is not developed successfully. Furthermore, non-generic framework may thus incur a reduced manual effort to provide training or to find a suitable combination and customization of algorithms. Table 1 contains some of the main frameworks.

Framework name	Description
MOMA (Mapping-based Object Matching) [16]	is a flexible and domain-independent framework for mapping-based entity matching providing an extensible library of matchers, both attribute value and context matchers (based on the COMA approach) Having as fundamental characteristic, the use of multiple match algorithms or workflows for any given matching problem. Hence, it is related to the notion of instance mappings. The output of each matcher and workflow step is represented as a so-called same-mapping (with semantically equivalent inputs) and can be refined by additional steps and matchers.
SERF (Stanford Entity Resolution Framework) [17]	Develops a genetic infrastructure focusing on improving the performers of ER solutions. The functions are expressed as black boxes which permits generic and extensible ER solutions by keeping tracks of previously compared values in order to avoid redundant comparisons.
MARLIN (Multiply Adaptive Record Linkage with Induction) [18]	It employs a two-level training-based approach combining multiple matchers by using SVM algorithm. At the first level, a set of similarity metrics are trained to appropriately determine the similarity for different database fields. At the second level, for each individual fields SVM is used to determine a combination of the tuned matchers from the previous level. MARLIN utilizes the canopies clustering method using Jaccard similarity for overlapping blocking.
Multiple Classifier System [19]	Proposed an approach where a variety of supervised learners (such as decision trees, 1-rule, Naive Bayes, linear and logistic regression, back propagation, neural network, and k-nearest neighbours) are employed as combining matchers. Moreover, meta-combination approaches for combining several supervised learners are supported, namely cascading, bagging, boosting, and stacking.
FEBRL (Freely Extensible Biomedical Record Linkage) [20]	is a hybrid framework supporting a training-based numerical combination approach utilizing the SVM as well as numerical approaches without training. FEBRL is the only one of the considered frameworks that is freely available on the web under an open source software license. It was originally developed for entity matching in the biomedical domain (hence the name). A large selection of 26 different similarity measures is available for attribute value matching. FEBRL supports one disjoint as well as three overlapping blocking methods. Besides manual training selection two strategies for automatic training are supported [17]: threshold- and nearest-based.
STEM (Self-Tuning Entity Matching) [21]	is a hybrid framework supporting the automatic construction of entity matching strategies. The framework addresses the problem of how to automatically configure and combine several matchers using

	numerical combination approaches. Several training-based numerical combination approaches are supported utilizing the SVM, decision trees, logistic regression and their combination. Different similarity measures are available for attribute value matching. Besides manual training selection two strategies for automatic training are supported: Threshold-Equal and Threshold-Random. With Threshold-Random entity pairs are randomly selected among the ones satisfying a given minimal threshold t applying a similarity measure m, whereas Threshold-Equal selects an equal number (n/2) of matching and non-matching pairs from the ones satisfying a given minimal threshold t applying a similarity measure m.
Context Based Framework [22]	The Context Based Framework [16] is a graph-based hybrid framework supporting a two-stage training-based numerical combination approach. The first stage tunes attribute value and context matchers using mainly SMOreg, a support vector regression approach. At the second level a second learner, e.g., logistic regression is applied to determine a combination of the tuned matchers from the previous step. The Context Based Framework utilizes a canopy-like technique for blocking.

Table 1. Entity Resolution frameworks and software

3.2 Classification methods and evaluation

3.2.1 Classification

In the terminology of machine learning and statistics, classification is considered as a supervised learning approach in which the model learns from the input data and then use this knowledge to classify new observations. The algorithms that implement classification, especially in a concrete implementation, is known as a classifier. Additionally, there are also ensemble classifiers [23] which are classifiers who have an external improvement in their final classification performance, such as AdaBoost or Extra Tree classifier.

3.2.1.1 Support vector machines (SVMs)

A Support Vector Machine (SVM) [24] is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. It is one of the most popular classifiers, due to the effectiveness and versatile. Indeed, it effective in high dimensional spaces and in the circumstance that the dimensions are greater than the number of the samples. Addressing on making it one of the most repeatedly used classifiers in entity resolution approaches. Moreover, in two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. Generally, by using the SVM the margin of the classifier is maximized.

3.2.1.2 Random Forest Classifier (RF)

Random Forests classifier has been proposed by Breiman [25] as an enhancement of Tree Bagging. To build a tree it uses a bootstrap replica of the learning sample, and the CART algorithm (without pruning) together with the modification used in the Random Subspace method. At each test node the optimal split is derived by searching a random subset of size K of candidate attributes (selected without replacement from the candidate attributes). Empirical studies have shown that Random Forests significantly outperform Tree Bagging and other random tree ensemble methods in terms of accuracy. In terms of degree of randomization, this algorithm is stronger than Tree Bagging, especially if K is small compared to the number of attributes, n. It is also stronger than Random Subspace since it combines this method with bootstrap sampling. Randomization is both implicit (attribute and cut-point) and explicit (attribute). We use the notation RFK, with K = d for the default setting, and K = ∗ for the best result over the range K = 1, ..., n.

3.2.1.3 Extra Trees Classifier

The Extra-Tree method [25](standing for extremely randomized trees) was proposed in [GEW06], with the main objective of further randomizing tree building in the context of numerical input features, where the choice of the optimal cut-point is responsible for a large proportion of the variance of the induced tree. With respect to random forests, the method drops the idea of using bootstrap copies of the learning sample, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node, it selects a cut-point at random.

3.2.1.4 Decision Tree Classifier

Decision Trees [25] are a type of Supervised Machine Learning algorithm, where the data is continuously splitting based on a certain parameter (creating a shape of tree). The tree can be consisted by two entities, the namely decision nodes and the leaves. The decisions or the final outcomes are represented by the leaves. On the other hand, the decision nodes are where the data is split. In the decision tree classifier, the decision variable is categorical (True or False).

3.2.1.5 AdaBoost Classifier

AdaBoost, short for "Adaptive Boosting" [25], is the first practical boosting algorithm proposed by Freund and Schapire in 1996. It an ensemble classifier which is focused on classification problems and aims to improve the performance of a set of weak classifiers into a strong one. Indeed, it combines the output of the learning algorithms ('weak learners') with a weighted sum which represents the final output of the boosted classifier. On the other hand, noisy data and outliners can affect negatively the performance of AdaBoost.

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

3.2.1.6 Bagging Classifier

Bagging [26] (was proposed by Breinman on 1996) is a ``bootstrap" ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples - where N is the size of the original training set; many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set. Breiman [26] showed that Bagging is effective on ``unstable" learning algorithms where small changes in the training set result in large changes in predictions. Breiman claimed that neural networks and decision trees are examples of unstable learning algorithms.

3.2.1.7 K Neighbor's Classifier

The main idea of KNN is that it treats all the samples as points in the m-dimensional space (where m is the number of variables) and given an unseen sample x, the algorithm classifies it by a vote of K-nearest training instances as determined by some distance metric, typically Euclidean distance [27].

3.2.2 Models Evaluation

Model evaluation is the process of selecting between different model types, features and tuning parameters. Thus, the model evaluation procedure is actually quantifying the quality of the prediction results of an unseen dataset, and it is suitable for judging the performance of a fitted classification model. Indeed, it assist on finding the best performed model that represents our data and well will perform the chosen model in the future. During the model evaluation processing the data that has been used for training is not acceptable because it may easily generate overoptimistic and overfitted models.

There are two main methods of evaluating models in data science, the Hold-Out and the Cross-Validation methods. In order to avoid overfitting, both methods use a unseen data set (test set) to evaluate model performance.

Hold-Out: In this method, the mostly large dataset is randomly divided to three subsets:

- Training set is a subset of the dataset used to build predictive models.
- Validation set is a subset of the dataset used to assess the performance of model built in the training phase. It provides a test platform for fine tuning model's parameters and selecting the best-performing model. Not all modelling algorithms need a validation set.
- Test set or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.

Cross-Validation

When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use k-fold cross-validation. In k-fold cross-validation, we divide the data into k subsets of equal size. We build models k times, each time leaving out one of the subsets from training and use it as the test set. If k equals the sample size, this is called "leave-one-out".

3.3 Face recognition

Face recognition is a method of identifying or verifying the identity of an individual using their face. The face recognition systems take advantage of computer algorithms in order to identify specific and distinctive details on an individual's face. For instance, they use the distance between the eyes or the shape of the chin, by converting it into a mathematical representation (array). The data about a particular face is often called a face template and is distinct from a photograph because it's designed to only include certain details that can be used to distinguish one face from another. These systems will offer up several potential matches, ranked in order of likelihood of correct identification, instead of just returning a single result. These systems could identify people in video, photos, gif or even in real-time streaming.

Face recognition systems vary in their ability to identify people under challenging conditions such as poor lighting, low quality image resolution, and suboptimal angle of view. Although, there are several times that face recognition data can mislead to false results, which can even implicate individuals for crimes which they did not commit. Exemplifying, African Americans and other minorities, young people and woman usually are misidentified, or the system fails to recognize them. When the systems are used in order to enforce the low or for security reasons, the misidentification could lean on dramatical results.

3.4 Feature extraction and evaluation

The problem of feature selection in terms of supervised inductive learning is defined as given a set of candidate features, select a subset. The main criteria for feature extraction is defined by one of the following three approaches:

a) the subset with a specified size that optimizes an evaluation measure,

b) the subset of smaller size that satisfies a certain restriction on the evaluation measure

c) the subset with the best commitment among its size and the value of its evaluation measure (general case).

In this sense, it is the process of finding and selecting the most useful, for the model, features in a dataset. The generic purpose pursued is the improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the representation. It is then possible to understand better the results obtained by the inducer, diminish its volume of storage, reduce the noise generated by irrelevant or redundant features and eliminate useless knowledge. Moreover, unnecessary features decrease training speed, decrease model interpretability, and, most importantly, decrease generalization performance on the test set.

3.5 Dataset and data preparatory steps

3.5.1 Data store types

The data arrives in diverse forms from diverse sources and they are defined on three main data types on structured data, unstructured data and semi-structured data.

Structured Data refers to data that has a high level of organization, usually resides in a fixed field, containing textual or numeric data, within a repository such as file or database. Indeed, the fields have usually an already defined size which in combination with a variety of rules that is concerned makes easier the data access and the information is rigidly arranged. Thus, its element can be easily searchable, addressing on a more effective processing and analysis. For instance, the information in an Excel files which could contain customer address, name, phone and email is considered as structured data. Generally, this type of data is stored in tables with various columns and rows.

Unstructured Data, on the other hand, refers to information that is not organized in any discernible manner, has no associated data model and generally does not reside in a traditional row-column database. The volume of unstructured data is rapidly growing, is estimated that around 80 to 90 percent of the data in any organization/company is unstructured. Thus, a wide range of enterprises have turned to several different technological solutions (hardware and software), in order to have a more efficient manage and store of their unstructured data (e.g. software designed to search unstructured data).Unstructured data form could generally be considered as books, e-mail messages, photos, video, audio files and many other significant amounts of digital material and documents that exists. In this sense, the information of the files is not organized but is placed into a repository (database or local file).

Semi-structured Data Is considered the data that contains semantic tags and associated information which has not been organized into a specialized repository. Indeed, semi- structured data falls between the structured and unstructured data. It includes aspects that are structured and other that are not. Although, it does not contain complex manner organization, the association between the information, such as metadata tagging, allows the sophisticated access and analysis. For instance, a word document could be considered as a semi-structured data set if it includes metadata tags in form of keywords.

Is estimated that the unstructured and semi-structured data represents 85% or more of the existing data. In our experiments, semi-structured data have been used.

3.5.2 Data Preparation

Data preparation is the process by which data analysts transform and organize the collected data into new data sets (file or data table) suitable for exploration and analysis. Data preparation process should be accurate and independent in order to prepare data for trusted reporting and analytics. Indeed, the need of formatting the data for a given software tool and make adequate for a given method, makes the data preparation crucial. Furthermore, the real-world data is usually “dirty” data by being incomplete, noisy or inconsistent. For instance, a common phenomenon on large datasets is lacking attribute values, lacking certain attributes of interest, or even containing only aggregate data. The is need of data quality improvement, such as enrich or eliminate specific attributes, before applying any analytical tools in order to improve the accuracy, the completeness of the data, the consistency and the believability. Typically, the major data preparation tasks are:

- Data Discretization
  Could be part of data reduction, is essential when using numerical data.
- Data Cleaning

Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.

- ❖ Data Integration
  Integration of multiple databases, data cubes, or files.
- ❖ Data Transformation
  Normalization and aggregation.
- ❖ Data Reduction
  Obtains reduced representation in volume but produces the same or similar analytical results.

For any type of data analysis, processing and management, the garbage-in garbage out principle holds. If the quality of the input data is low, then the output generated is normally not of high quality or accuracy either.

3.6 Data Augmentation

Data augmentation adds value to base data by adding information derived from internal and external sources within an enterprise. Data is one of the core assets for an enterprise, making data management essential. Data augmentation can be applied to any form of data, but may be especially useful for customer data, sales patterns, product sales, where additional information can help provide more in-depth insight. Data augmentation can help reduce the manual intervention required to developed meaningful information and insight of business data, as well as significantly enhance data quality.

Data augmentation is of the last steps done in enterprise data management after monitoring, profiling and integration. Some of the common techniques used in data augmentation include:

- ❖ Extrapolation Technique:
  Based on heuristics. The relevant fields are updated or provided with values.
- ❖ Tagging Technique:
  Common records are tagged to a group, making it easier to understand and differentiate for the group.
- ❖ Aggregation Technique:
  Using mathematical values of averages and means, values are estimated for relevant fields if needed
- ❖ Probability Technique:
  Based on heuristics and analytical statistics, values are populated based on the probability of events.

4. Our approach

The proposed approach (Image 1) intends to combine a wide range of the deep learning algorithms and machine learning techniques to promote an alternative way of JedAI Workflow [28]. JedAI is an open source library which implements state of the art methods for solving the entity resolution problem. Thus, it a toolkit for Entity Resolution that can be deployed in three main different ways: as an open-source Java library which implements domain-independent methods, as a workbench that facilitates the evaluation of their relative performance and also as a desktop application that offers out-of-the-box entity resolution solutions. It bridges the gap between the database and the Semantic Web communities, providing solutions that are applicable to both relational and RDF data.

JedAI combines the workflow presented on [29] which combines data reading, block building, block cleaning, comparison cleaning, entity matching, entity clustering, evaluation and storing. Table 2 illustrate both features of JedAI Workflow and the alternative proposed approach.

JedAI Workflow	**Alternative proposed approach**
Data Reading	Data Reading
Block Building	Entity Cleaning
Block Cleaning	Entity Matching
Comparison Cleaning	Entity Feature Extraction & Evaluation
Entity Matching	
Entity Clustering	Entity Classification
Evaluation & Storing	Evaluation & Storing

Table 2. JedAI and our approach workflow

Data Reading - in both approaches there is the data reading, although in our approach the loaded data could be structured (CSV) or semi-structure dataset (.json file) instead of structured as in the JedAI (e.g. CSV file).

Entity Cleaning – is responsible to process the entity row data and prepare it for the entity matching processing. In this point, the loaded data is evaluated and cleaned (e.g. candidate individuals that do not contain name and email are eliminated). Each loaded entity should achieve some particular specifications, in order to be consider are eligible entity for matching.

Entity Matching – in this component, different machine learning and deep learning tools have been implemented to match the elements of the two candidates. Indeed, each common element of both candidates, independently of the type (categorical e.g. name or numerical e.g. age), is compared and provide a similarity value, ranged from 0 to 1. For instance, the name of the candidate A is compared with the name of candidate B through the use of Fuzzywuzzy python library. Having as a result a value from 0 to 1, based on the similarity of the names by getting into consideration both letter of the name and letter's order.

Entity Feature Extraction & Evaluation – is responsible to manage and provide data analysis features to attend the entity's classifications need. The results of Entity Matching are evaluated and cleaned through several filters (e.g. eliminate the noise or impute missing values) and features evaluation algorithms (e.g., to obtain other attributes that could enhance the classification output). Moreover, data integration, aggregation and normalization procedures are processed. Thus, based on the matched data, integrate data from different elements, or eliminate them or scale the values of all the attributes in order to have the same weight in the data analysis process such as age.

Entity Classification – machine learning classifiers are used to build the model using the features from the previous step.

Evaluation & Storing – The built models should be evaluated using evaluation techniques (e.g., cross validation). In this sense, the output (recall, precision and f1-score) of the evaluation process will indicate which classification model will be stored.

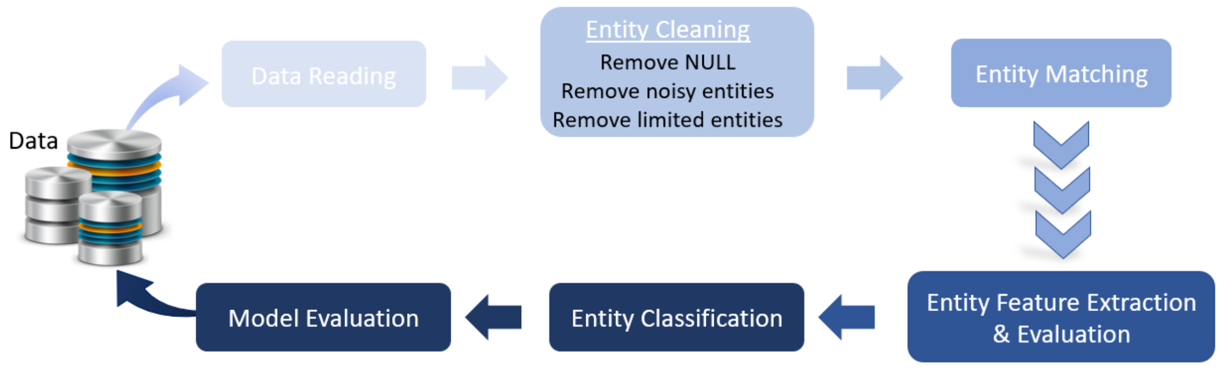


Image 1. Our entity resolution approach architecture

These components provided a workflow that solve the entity resolution problem in an alternative way.

5. Implementation

This chapter is focused on presenting in detailed the implementation of the proposed approach and describing the experiments we performed. Hence, to implement the approach we used a wide range of services, such as the FuzzyWuzzy python library or TensorFlow deep learning framework.

5.1 Our case study

The focus of this work is to illustrate the data analysis aspects of the proposed approach thus, it was considered a case study in the domain of entity resolution. In this sense, a data set with 1923 pairs of individual's profile data was used as input on the entity matching algorithms. Table 2 describes all the extracted features used in the experiments.

Raw Data	Extract Features		Description
First name	First_name		The results of the first name matching of both candidates (Float [0,1])
Last name	Last_name		The results of the last name matching of both candidates (Float [0,1])
Social Profiles	LinkedIn	URL vs first name	The existents of the first name in the URL for both candidates (Boolean 0 or 1)
		URL vs last name	The existents of the last name in the URL for both candidates (Boolean 0 or 1)
		Exist	The existents of the LinkedIn for both candidates (Boolean 0 or 1)

	Facebook	Exist	The existents of the Facebook for both candidates (Boolean 0 or 1)
	Twitter	Exist	The existents of the Twitter for both candidates (Boolean 0 or 1)
	Google plus	Exist	The existents of the Twitter for both candidates (Boolean 0 or 1)
Emails	Email vs first name		The existents of the first name in the email for both candidates (Boolean 0 or 1)
	Email vs last name		The existents of the last name in the email for both candidates (Boolean 0 or 1)
	Email vs email		Both candidates' emails comparison (Float [0,1])
Working experience	Company		Both candidates' company comparison (Float [0,1])
	Title		Both candidates' title comparison (Float [0,1])
	Current		Both candidates' current comparison (Float [0,1])
Location	Full location		Both candidates' full location comparison (Float [0,1])
	Full origin		Both candidates' full origin comparison (Float [0,1])
images	Face recognition		Face comparison of both candidates (Boolean 0 or 1)

Table 3. List of extracted features

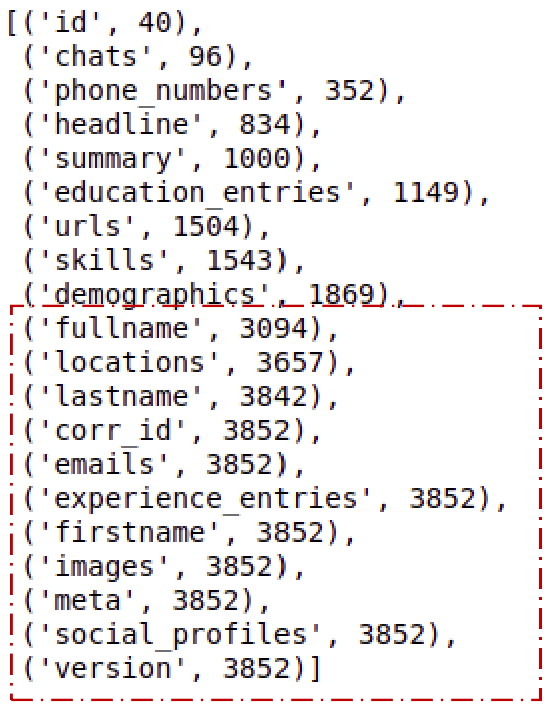
Table 3 was created after the implementation of three main steps (Image1) of the approach, the data loading, entity cleaning and the entity matching. Hence, each of the features contain the comparison information between the two candidates. In the following subchapters is described in detail how the process of feature extraction was implemented.

5.2 Experiments and analysis

The programming language that have been used is python and for the sake of simplicity everything was developed is Jupyter notebook IDE.

Data Loading:

The information is being loaded from a local host json file which contains 1926 pairs of candidates, a unique identification key and their label (Table 3). The loaded data have been studied and evaluated. We realized that not all the candidates contain the same information. Image 2 represents all the elements for each candidate and their frequency on the data set.



```
[('id', 40),
 ('chats', 96),
 ('phone_numbers', 352),
 ('headline', 834),
 ('summary', 1000),
 ('education_entries', 1149),
 ('urls', 1504),
 ('skills', 1543),
 ('demographics', 1869),
 ('fullname', 3094),
 ('locations', 3657),
 ('lastname', 3842),
 ('corr_id', 3852),
 ('emails', 3852),
 ('experience_entries', 3852),
 ('firstname', 3852),
 ('images', 3852),
 ('meta', 3852),
 ('social_profiles', 3852),
 ('version', 3852)]
```

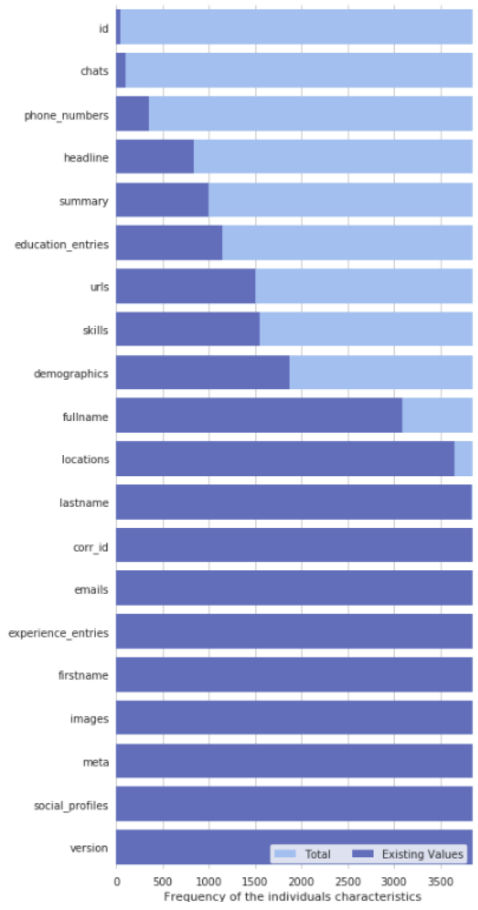


Image 2. Frequency of the individual's characteristics.

In the beginning, we considered a threshold close to 3000. Thus, we took into consideration only the elements that had frequency over 3000 (Image 2) which means 1500 pairs of candidates. Furthermore, during our data study we identify that not all the candidates have the same number of elements. Indeed, the min number of elements that a candidate had was 17 and the max was 20. This was also one of the reasons that we considered a threshold of 3000. Last but not least, each of the elements does not contain only one information. For instance, the social media element contains 4744 LinkedIn URLs even if our individuals are 3852. As a result, in some individuals there more than on LinkedIn URLs. In this sense, the elements are used for the matching and the feature extraction are 12.

Entity matching

In this point of the implementation, several python libraries have been used, individually or in combination, to achieve the entity matching. Its entity's element was employed in a different way. For instance, the email element was splitted into parts, the part before the "at" (which is the part that users are defining) and the part after the "at" (which is the company hosting the email).

Additionally, we also eliminated any symbols or numbers that were included in the before "at part". After having done some "pre-processing" for each element, we do the matching process.

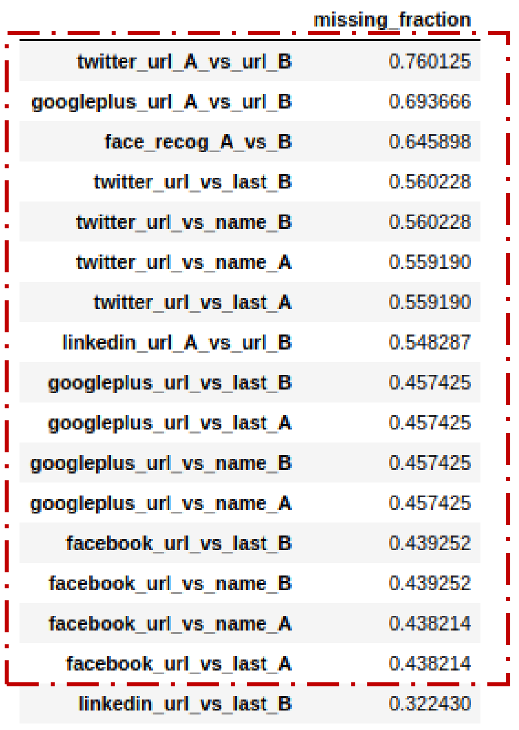
The Fuzzywuzzy, Difflib, Urltools, Regular expression operations, Email_split and Face recognition are some of the main libraries that have been used. The results of the matching could be Boolean (0 or 1), Float [0,1] and -1. The Boolean value indicates if it exists or not, the float the level of the similarity and -1 the missing values. In the end of the matching processing we had a data set with 37 columns (candidate features) and 1923 rows.

Feature extraction and evaluation.

This component gets as input the produced dataset from the previous step. Thus, we have a dataset with length 1923 and 37 columns without the labels. Each of the column is a candidate feature for our classification model. In order to, identify and select the most important and more efficient feature for the model we used some model evaluation techniques. The FeatureSelector and XGBoost python libraries have been employed. The main steps that we followed are:

- ❖ *Identify columns with more than 35% the missing values.*

Image _ contains all the columns with more than 35% of missing values.



	missing_fraction
twitter_url_A_vs_url_B	0.760125
googleplus_url_A_vs_url_B	0.693666
face_recog_A_vs_B	0.645898
twitter_url_vs_last_B	0.560228
twitter_url_vs_name_B	0.560228
twitter_url_vs_name_A	0.559190
twitter_url_vs_last_A	0.559190
linkedin_url_A_vs_url_B	0.548287
googleplus_url_vs_last_B	0.457425
googleplus_url_vs_last_A	0.457425
googleplus_url_vs_name_B	0.457425
googleplus_url_vs_name_A	0.457425
facebook_url_vs_last_B	0.439252
facebook_url_vs_name_B	0.439252
facebook_url_vs_name_A	0.438214
facebook_url_vs_last_A	0.438214
linkedin_url_vs_last_B	0.322430

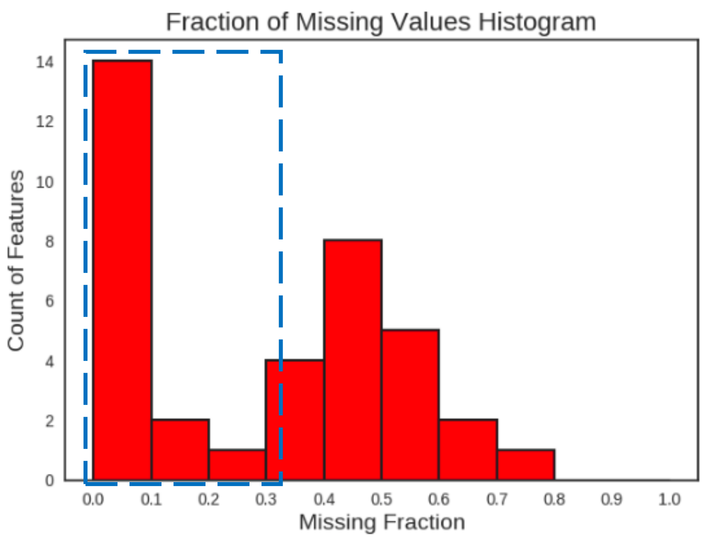
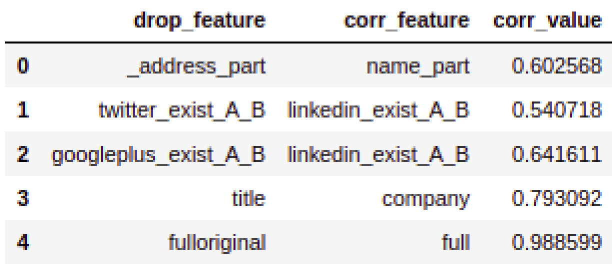


Image 3. Missing Values

Moreover, Image 3 describes the distribution of the missing values in all the columns. Thus, we can see 17 columns have less than 30% of missing values (Image 3 right).

❖ *Collinear Evaluation*

Continuously, we evaluated the collinear level (highly correlation) of the existing columns (Image 4). We eliminate or merge some of the columns which had more than 30% of missing values and greater than 50% of correlation (highly correlated pairs). For instance, we merge the information that we had form the social media element of LinkedIn into a more compact way, instead of having for each entity separate information associated with his LinkedIn and name we merged it for both entities. In the end of this process, we have a data set with same length and 16 columns.



	drop_feature	corr_feature	corr_value
0	_address_part	name_part	0.602568
1	twitter_exist_A_B	linkedin_exist_A_B	0.540718
2	googleplus_exist_A_B	linkedin_exist_A_B	0.641611
3	title	company	0.793092
4	fulloriginal	full	0.988599

Image 4. Colinear features

❖ Feature Importance

The previous methods are deterministic and can be applied to any dataset that have been structured. In this point, we will use a method designed only for supervised machine learning problems, like ours, and is one non-deterministic process. In both ways that we will describe, is used as input the 16 columns dataset which is the output of collinear evaluation step. The identify_zero_importance function finds features that have zero importance according to a gradient boosting machine (GBM) learning model. In this sense, a tree-based machine learning model is created where the columns with zero importance are not used to split any nodes, so they can be removed without affecting the model's performance.

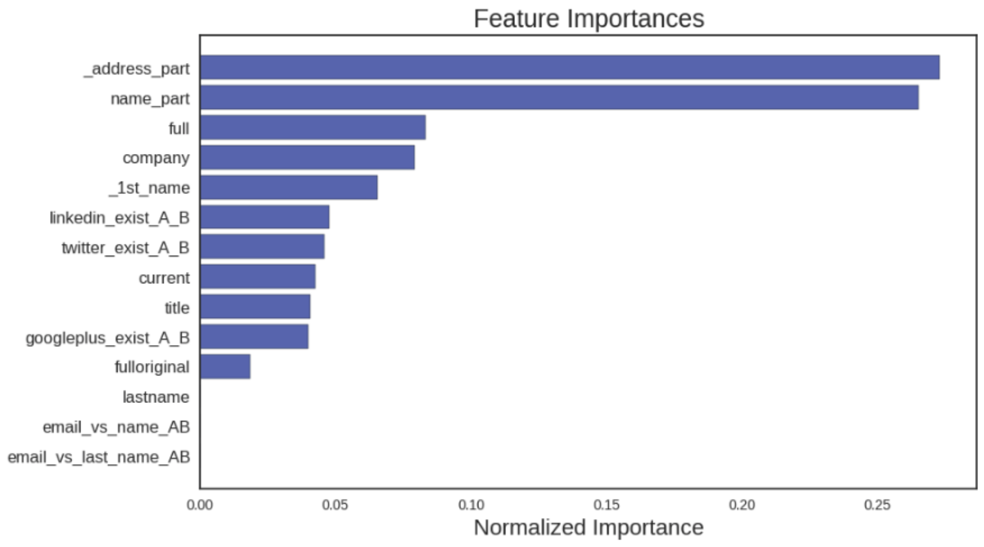


Image 5. Feature Importance

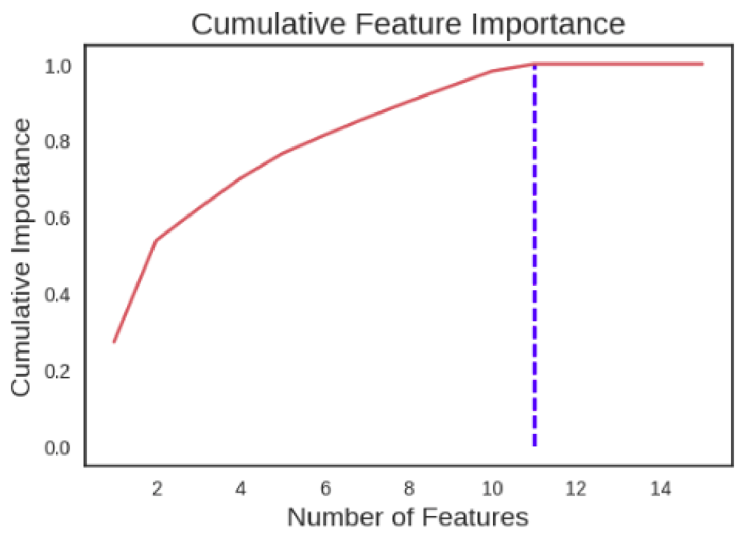
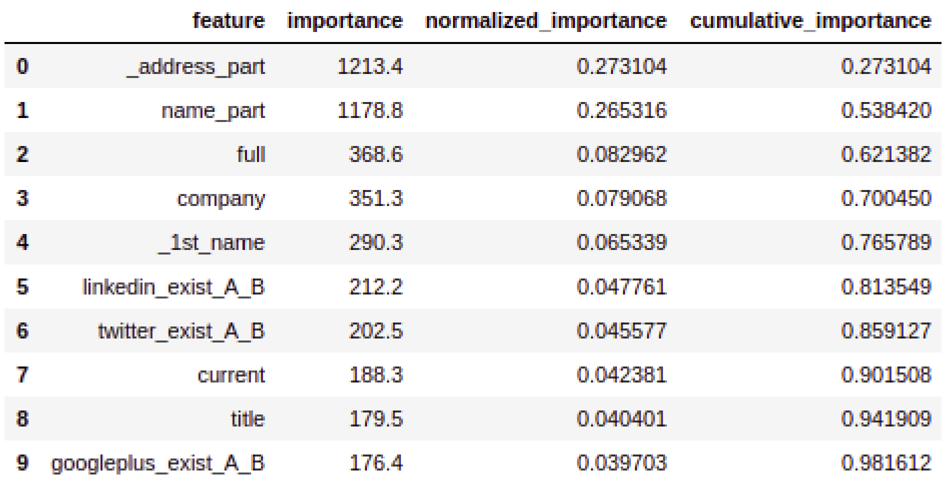


Image 6. Cumulative Feature Importance

Image 5 indicate that 11 columns required for 0.99 of cumulative importance. Furthermore, 10 features required for cumulative importance of 0.99 after one hot encoding and 5 features do not contribute to cumulative importance of 0.99 (Image 7).



	feature	importance	normalized_importance	cumulative_importance
0	_address_part	1213.4	0.273104	0.273104
1	name_part	1178.8	0.265316	0.538420
2	full	368.6	0.082962	0.621382
3	company	351.3	0.079068	0.700450
4	_1st_name	290.3	0.065339	0.765789
5	linkedin_exist_A_B	212.2	0.047761	0.813549
6	twitter_exist_A_B	202.5	0.045577	0.859127
7	current	188.3	0.042381	0.901508
8	title	179.5	0.040401	0.941909
9	googleplus_exist_A_B	176.4	0.039703	0.981612

Image 7. 10 cumulative importance features

Last but not least, we used XGBClassifier from the XGBoost library in order to identify the columns importance based on the f1-score. XGBoost is an ensemble meta-algorithm that iteratively trains sequences of weaker base learners and retrieve importance scores for each attribute. Indeed, the importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. As a result, the more an attribute is used to make key decisions with decision trees, the higher its relative importance. This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other. Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The feature importance is then averaged across all of the decision trees within the model.

In our experiments, the booster that was employed is gbtree which uses version of regression tree as a weak learner [31]. We did not modify the default parameters of the gbtree booster. Moreover, from sklearn.metrics module was import the function accuracy_score, which is being used in order to get the accuracy of prediction model on the test set (Table 4).

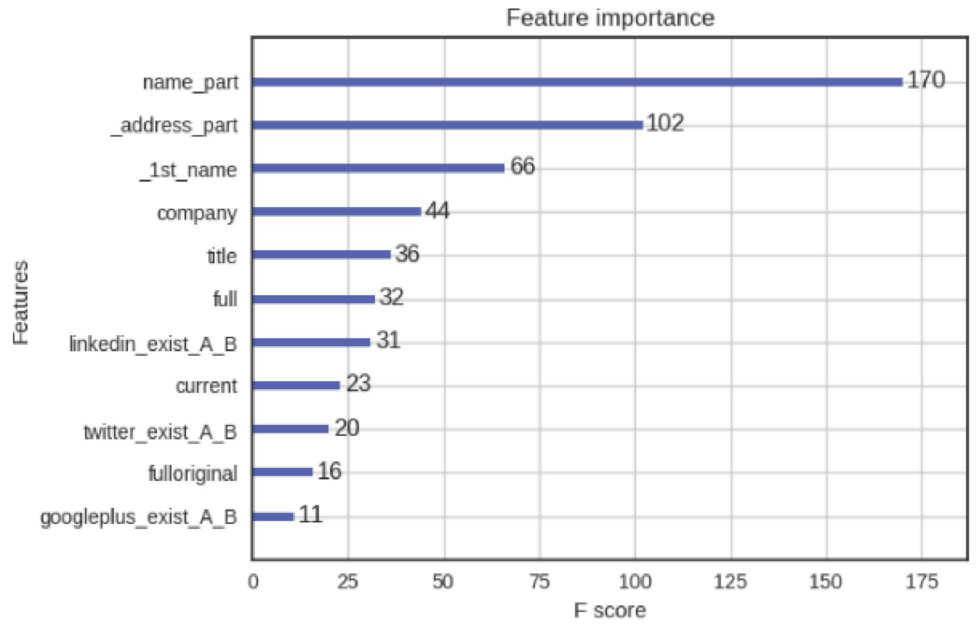


Image 8. Feature importance/ F score

Accuracy: 94.18%
Thresh=0.000, n=15, Accuracy: 94.18%
Thresh=0.000, n=14, Accuracy: 94.18%
Thresh=0.000, n=13, Accuracy: 94.18%
Thresh=0.000, n=12, Accuracy: 94.18%
Thresh=0.013, n=11, Accuracy: 94.18%
Thresh=0.019, n=10, Accuracy: 94.18%
Thresh=0.042, n=9, Accuracy: 94.18%
Thresh=0.048, n=8, Accuracy: 94.18%
Thresh=0.050, n=7, Accuracy: 94.34%
Thresh=0.056, n=6, Accuracy: 94.34%
Thresh=0.081, n=5, Accuracy: 94.34%
Thresh=0.083, n=4, Accuracy: 94.18%
Thresh=0.115, n=3, Accuracy: 94.34%
Thresh=0.227, n=2, Accuracy: 93.87%
Thresh=0.265, n=1, Accuracy: 92.45%

Table 4. Accuracy of the Features

We can see that the performance of the model generally decreases with the number of selected features. On this problem there is a trade-off of features to test set accuracy and we could decide to take a less complex model (fewer attributes such as n=4 or n=5) and accept a modest decrease in estimated accuracy from 94.18% down to 92.45%. This is likely to be a wash on such a small dataset but may be a more useful strategy on a larger dataset and using cross validation as the model evaluation scheme.

Having as assets all the previous results, we observed that both ways of feature importance selection include the same features in different importance order. Image 5,7 and 8 represent the same features with deferent order, as a result, both methods converge into the same feature importance with different score. Thus, we decided to maintain 15 more important and efficient columns from the dataset which were indicated from both importance feature selection methods (some of which do not appear in the previous plots due to the plot threshold).

- Model creation and evaluation

The classifiers that have been implemented in order to fulfil the purpose of this thesis are SVM, Decision Tree Classifier, AdaBoost, Random Forest, Extra Tree Classifier, K Neighbours, and Bagging Classifier.

To evaluate the performance of the classifiers, we create the confusion matrix and extract also the precision, recall and f1 score for each k model of the cross-validation. So, we get the True Positive (TP), False Positive (FP), False Negative (FN) and the True Negative (TN) for each class (0 or 1) on Table A and the precision, recall and f1 score in the Table B.
For instance, the first model (k=0) that SVM classifier created gave us 9 TP, 3 FP, 32 FN, and 342 TN. As a result, the performance of the model was awful (75% wrong) because it missed almost all the times that the class was 0.

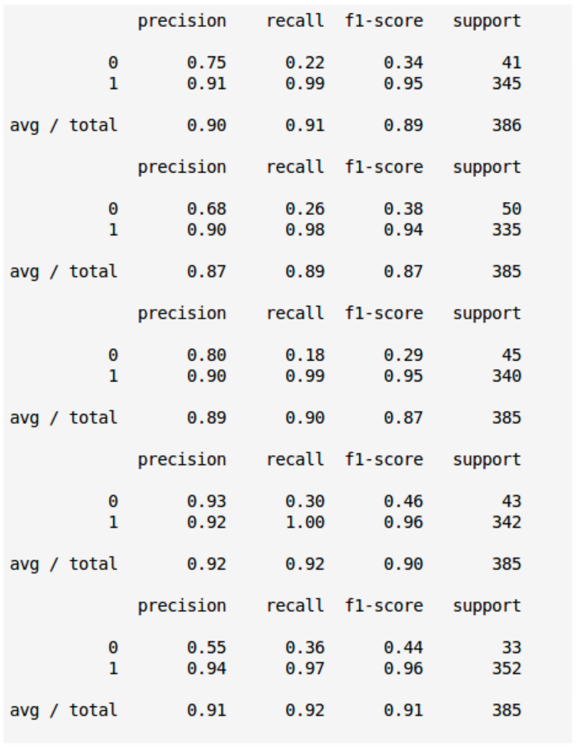
The follow tables contain the performance of each classifier while implementing cross validation method with k equal to 5.

SVM classifier

Scikit learn library was used to implement the SVM algorithm. The kernel function that was used, was the linear (x,x').

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	9	32	13	37	8	37	13	30	12	21
1	3	342	6	329	2	338	1	341	10	342

Table 5A. Cross Validation (k=5)

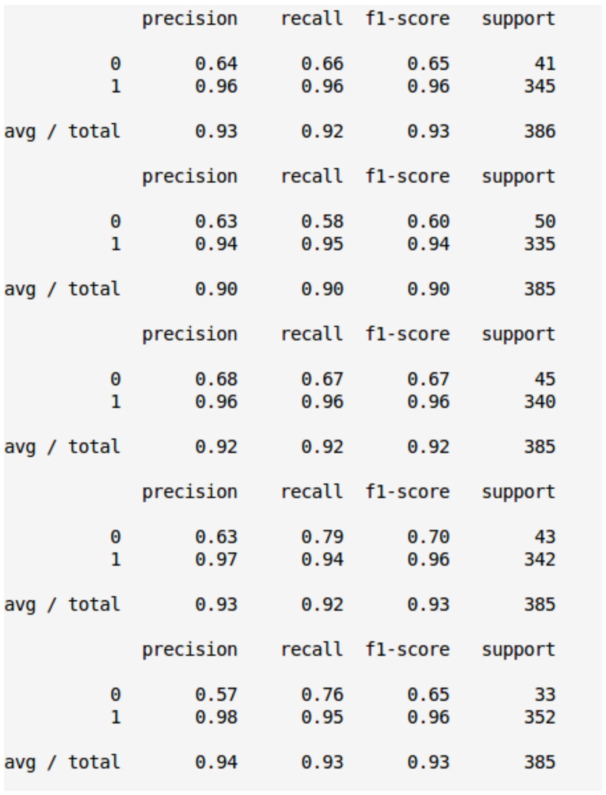

```
             precision    recall  f1-score   support

          0       0.75      0.22      0.34        41
          1       0.91      0.99      0.95       345

avg / total       0.90      0.91      0.89       386
             precision    recall  f1-score   support

          0       0.68      0.26      0.38        50
          1       0.90      0.98      0.94       335

avg / total       0.87      0.89      0.87       385
             precision    recall  f1-score   support

          0       0.80      0.18      0.29        45
          1       0.90      0.99      0.95       340

avg / total       0.89      0.90      0.87       385
             precision    recall  f1-score   support

          0       0.93      0.30      0.46        43
          1       0.92      1.00      0.96       342

avg / total       0.92      0.92      0.90       385
             precision    recall  f1-score   support

          0       0.55      0.36      0.44        33
          1       0.94      0.97      0.96       352

avg / total       0.91      0.92      0.91       385
```
Table 5B. Precision, recall and f1 score for each k

Decision Tree Classifier

We implemented decision Tree Classifier in two different way. In the first way, we employed the function from the library scikit learn and in the second way we implemented it from scratch following Jason Brownlee tutorial [29] by using basic python libraries such as numpy. Table A1 and A2 are the results from the scikit learn library from tree module using the default setting for the split [30].

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	27	14	29	21	30	15	34	9	25	8
1	15	330	17	318	14	326	20	322	19	333

Table 6A. Cross Validation (k=5)



```
             precision    recall  f1-score   support

          0       0.64      0.66      0.65        41
          1       0.96      0.96      0.96       345

avg / total       0.93      0.92      0.93       386
             precision    recall  f1-score   support

          0       0.63      0.58      0.60        50
          1       0.94      0.95      0.94       335

avg / total       0.90      0.90      0.90       385
             precision    recall  f1-score   support

          0       0.68      0.67      0.67        45
          1       0.96      0.96      0.96       340

avg / total       0.92      0.92      0.92       385
             precision    recall  f1-score   support

          0       0.63      0.79      0.70        43
          1       0.97      0.94      0.96       342

avg / total       0.93      0.92      0.93       385
             precision    recall  f1-score   support

          0       0.57      0.76      0.65        33
          1       0.98      0.95      0.96       352

avg / total       0.94      0.93      0.93       385
```

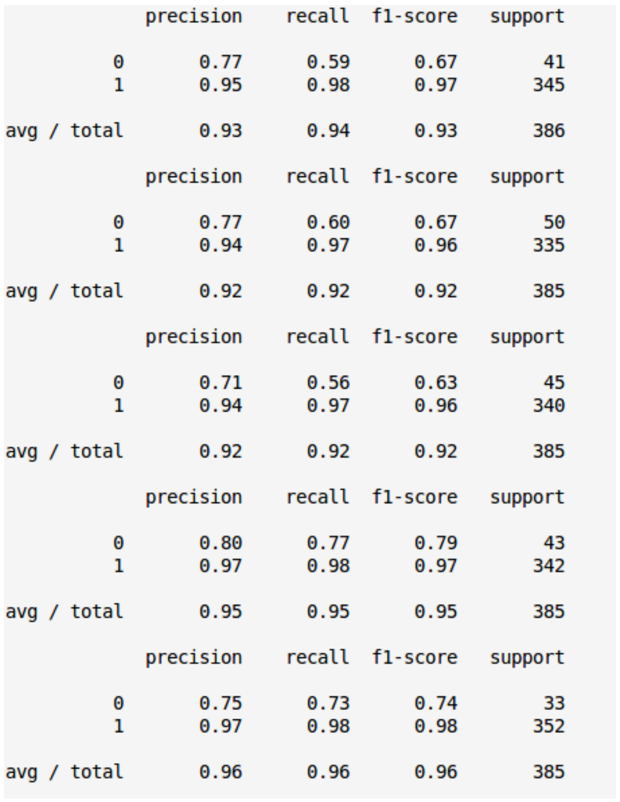
Table 6B. Precision, recall and f1 score for each k

<u>*AdaBoost Classifier*</u>

Is a *sequential* ensemble classifier where the base learners are generated sequentially. The overall performance can be boosted by weighing previously mislabelled examples with higher weight. We have used the AdaBoostClassifier function from the ensemble module of sklearn library. The base estimator that was used is the Division Tree classifier with max_depth equal to 1. Thus, we observed some similarities on the performance of both classifiers during the experiments.

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	24	17	30	20	25	20	33	10	24	9
1	7	338	9	326	10	330	8	334	8	344

Table 7A. Cross Validation (k=5)



```
             precision    recall  f1-score   support

          0       0.77      0.59      0.67        41
          1       0.95      0.98      0.97       345

avg / total       0.93      0.94      0.93       386
             precision    recall  f1-score   support

          0       0.77      0.60      0.67        50
          1       0.94      0.97      0.96       335

avg / total       0.92      0.92      0.92       385
             precision    recall  f1-score   support

          0       0.71      0.56      0.63        45
          1       0.94      0.97      0.96       340

avg / total       0.92      0.92      0.92       385
             precision    recall  f1-score   support

          0       0.80      0.77      0.79        43
          1       0.97      0.98      0.97       342

avg / total       0.95      0.95      0.95       385
             precision    recall  f1-score   support

          0       0.75      0.73      0.74        33
          1       0.97      0.98      0.98       352

avg / total       0.96      0.96      0.96       385
```

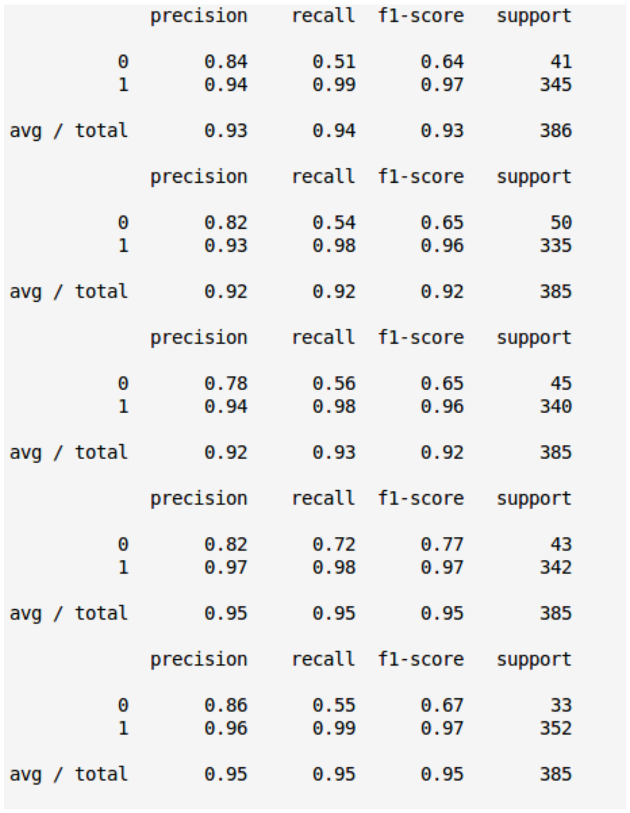
Table 7B. Precision, recall and f1 score for each k

Random Forest Classifier

On the other hand, Random Forest is a parallel ensemble classifier where his base learners are generated in parallel. The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging. In our implantation we used Randomforestclassifier function from the ensemble module of sklearn library. The numbers of trees that we used to build within the Random Forest before aggregation were 20 while the default is 10.

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	21	20	27	23	25	20	31	12	18	15
1	4	341	6	329	7	333	7	335	3	349

Table 8A. Cross Validation (k=5)



```
             precision    recall  f1-score   support

          0       0.84      0.51      0.64        41
          1       0.94      0.99      0.97       345

avg / total       0.93      0.94      0.93       386
             precision    recall  f1-score   support

          0       0.82      0.54      0.65        50
          1       0.93      0.98      0.96       335

avg / total       0.92      0.92      0.92       385
             precision    recall  f1-score   support

          0       0.78      0.56      0.65        45
          1       0.94      0.98      0.96       340

avg / total       0.92      0.93      0.92       385
             precision    recall  f1-score   support

          0       0.82      0.72      0.77        43
          1       0.97      0.98      0.97       342

avg / total       0.95      0.95      0.95       385
             precision    recall  f1-score   support

          0       0.86      0.55      0.67        33
          1       0.96      0.99      0.97       352

avg / total       0.95      0.95      0.95       385
```

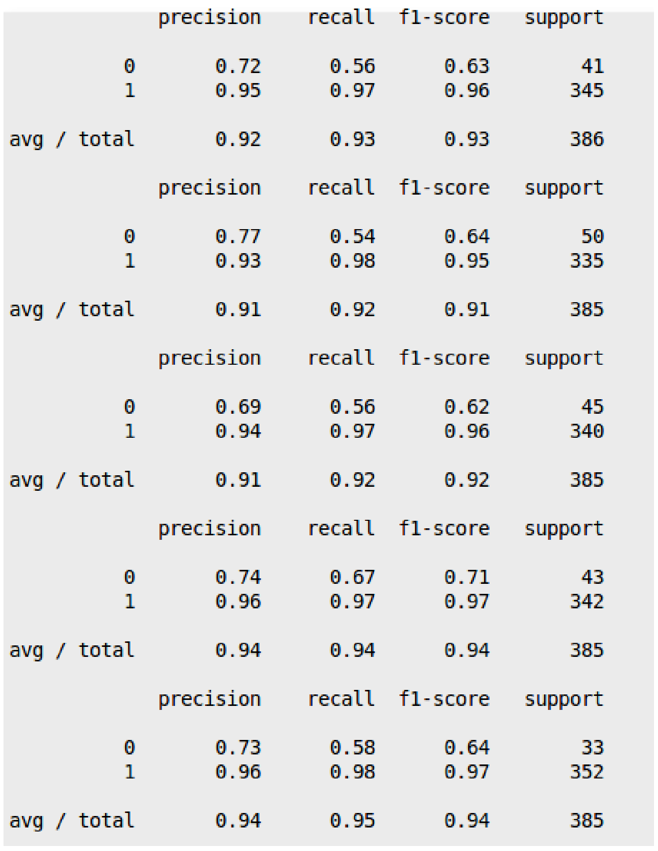
Table 8B. Precision, recall and f1 score for each k

Extra Trees Classifier

Is a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sklearn.ensemble module was used in order to implement the extra-trees classifier. Having done some experiments, we decided that the number of trees in the forest should be 20 instead of 10 which is the default.

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	23	18	27	23	25	20	29	14	19	14
1	9	336	8	327	11	329	10	332	7	345

Table 9A. Cross Validation (k=5)



```
             precision    recall  f1-score   support

          0       0.72      0.56      0.63        41
          1       0.95      0.97      0.96       345

avg / total       0.92      0.93      0.93       386
             precision    recall  f1-score   support

          0       0.77      0.54      0.64        50
          1       0.93      0.98      0.95       335

avg / total       0.91      0.92      0.91       385
             precision    recall  f1-score   support

          0       0.69      0.56      0.62        45
          1       0.94      0.97      0.96       340

avg / total       0.91      0.92      0.92       385
             precision    recall  f1-score   support

          0       0.74      0.67      0.71        43
          1       0.96      0.97      0.97       342

avg / total       0.94      0.94      0.94       385
             precision    recall  f1-score   support

          0       0.73      0.58      0.64        33
          1       0.96      0.98      0.97       352

avg / total       0.94      0.95      0.94       385
```

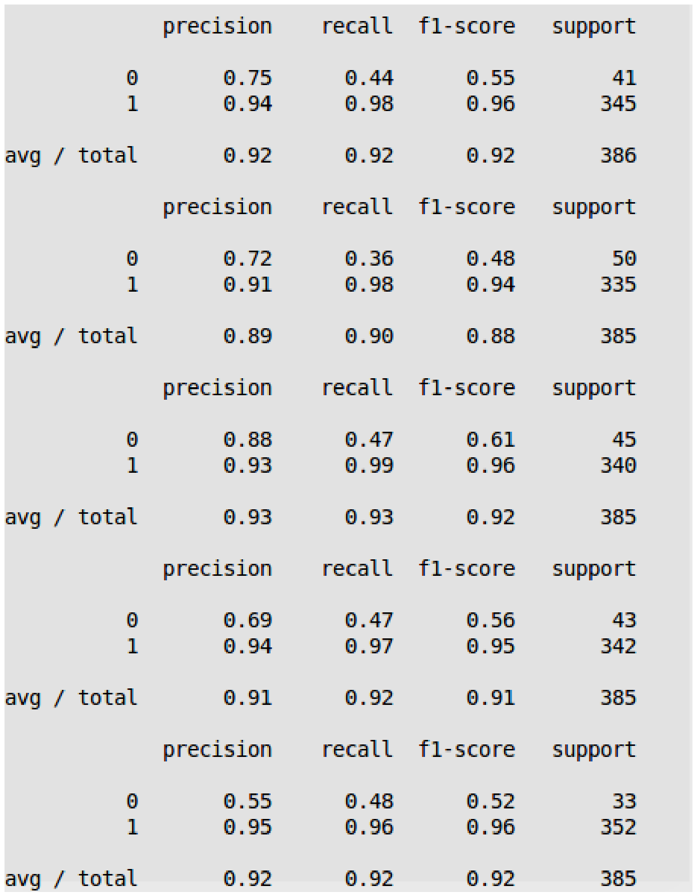
Table 9B. Precision, recall and f1 score for each k

K Neighbors Classifier

In order to implement K Neighbors, we employed the sklearn.neighbors module and used the kNeighborsClssifier. After some experiments, the number of neighbors we used is 10 instead of 5 which is the default.

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	18	23	18	32	21	24	20	23	16	17
1	6	339	7	338	3	337	9	333	13	339

Table 10A. Cross Validation (k=5)



```
             precision    recall  f1-score   support

          0       0.75      0.44      0.55        41
          1       0.94      0.98      0.96       345

avg / total       0.92      0.92      0.92       386
             precision    recall  f1-score   support

          0       0.72      0.36      0.48        50
          1       0.91      0.98      0.94       335

avg / total       0.89      0.90      0.88       385
             precision    recall  f1-score   support

          0       0.88      0.47      0.61        45
          1       0.93      0.99      0.96       340

avg / total       0.93      0.93      0.92       385
             precision    recall  f1-score   support

          0       0.69      0.47      0.56        43
          1       0.94      0.97      0.95       342

avg / total       0.91      0.92      0.91       385
             precision    recall  f1-score   support

          0       0.55      0.48      0.52        33
          1       0.95      0.96      0.96       352

avg / total       0.92      0.92      0.92       385
```

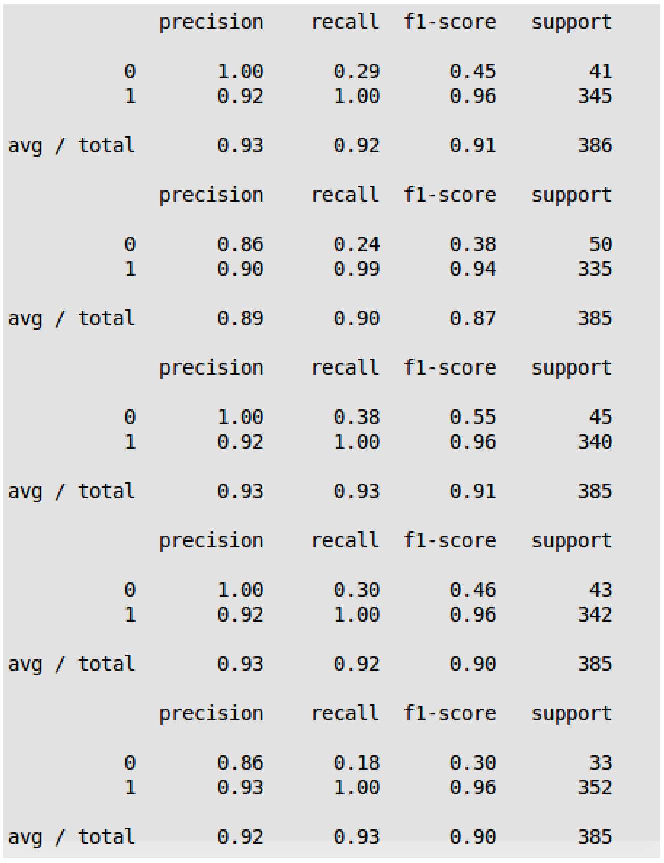
Table 10B. Precision, recall and f1 score for each k

Bagging Classifier

Is also a sequential ensemble classifier, like AdaBoost. We used the BaggingClassifier from the ensemble module of sklearn. Having as base estimator the KNeighborsClassifier, max_samples=0.5, max_features=0.5 and number of trees before the before aggregation were 20 (instead of 10 which is the default).

Confusion Matrix for each k										
	k=0		k=1		k=2		k=3		k=4	
	0	1	0	1	0	1	0	1	0	1
0	12	29	12	38	17	28	13	30	6	27
1	0	345	2	333	0	340	0	342	1	351

Table 11A. Cross Validation (k=5)



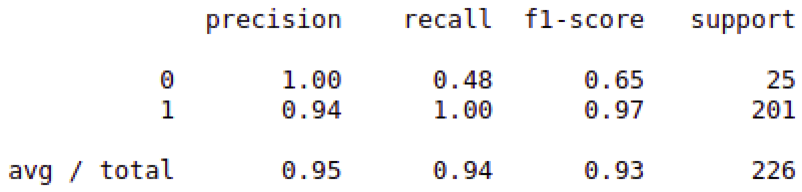
```
             precision    recall  f1-score   support

          0       1.00      0.29      0.45        41
          1       0.92      1.00      0.96       345

avg / total       0.93      0.92      0.91       386
             precision    recall  f1-score   support

          0       0.86      0.24      0.38        50
          1       0.90      0.99      0.94       335

avg / total       0.89      0.90      0.87       385
             precision    recall  f1-score   support

          0       1.00      0.38      0.55        45
          1       0.92      1.00      0.96       340

avg / total       0.93      0.93      0.91       385
             precision    recall  f1-score   support

          0       1.00      0.30      0.46        43
          1       0.92      1.00      0.96       342

avg / total       0.93      0.92      0.90       385
             precision    recall  f1-score   support

          0       0.86      0.18      0.30        33
          1       0.93      1.00      0.96       352

avg / total       0.92      0.93      0.90       385
```

Table 11B. Precision, recall and f1 score for each k

In this point, we create and evaluate the classification model using the split Train/ Test method. Indeed, 90% (1700 pairs) of the dataset is considered as training set and 10% (226 pairs) as test set. The performance of each classifier is as followed.

SVM

	0	1
0	12	13
1	0	201



```
             precision    recall  f1-score   support

          0       1.00      0.48      0.65        25
          1       0.94      1.00      0.97       201

avg / total       0.95      0.94      0.93       226
```

Table 12. Confusion Matrix, precision, recall and f1 score for SVM

Decision Tree Classifier

	0	1
0	16	9
1	4	197

```
             precision    recall  f1-score   support

          0       0.80      0.64      0.71        25
          1       0.96      0.98      0.97       201

avg / total       0.94      0.94      0.94       226
```

Table 13. Confusion Matrix, precision, recall and f1 score for Decision Tree Classifier

AdaBoost Classifier

	0	1
0	16	9
1	2	199

```
             precision    recall  f1-score   support

          0       0.89      0.64      0.74        25
          1       0.96      0.99      0.97       201

avg / total       0.95      0.95      0.95       226
```

Table 14. Confusion Matrix, precision, recall and f1 score for AdaBoost Classifier

Random Forest Classifier

	0	1
0	17	8
1	2	199

```
             precision    recall  f1-score   support

          0       0.89      0.68      0.77        25
          1       0.96      0.99      0.98       201

avg / total       0.95      0.96      0.95       226
```

Table 15. Confusion Matrix, precision, recall and f1 score for Random Forest Classifier

Extra Trees Classifier

	0	1
0	18	7
1	2	199

```
             precision    recall  f1-score   support

          0       0.90      0.72      0.80        25
          1       0.97      0.99      0.98       201

avg / total       0.96      0.96      0.96       226
```

Table 16. Confusion Matrix, precision, recall and f1 score for Extra Trees Classifier

K Neighbors Classifier

	0	1
0	16	9
1	1	200

```
             precision    recall  f1-score   support

          0       0.94      0.64      0.76        25
          1       0.96      1.00      0.98       201

avg / total       0.96      0.96      0.95       226
```

Table 17. Confusion Matrix, precision, recall and f1 score for K Neighbors Classifier

Bagging Classifier

	0	1
0	12	13
1	0	201

```
             precision    recall  f1-score   support

          0       1.00      0.48      0.65        25
          1       0.94      1.00      0.97       201

avg / total       0.95      0.94      0.93       226
```

Table 18. Confusion Matrix, precision, recall and f1 score for Bagging Classifier

The prediction results of both validation ways were not satisfied. The False class had very bad percentage of recall and f1-score in almost all classification algorithms we used which was something that we were expecting. Our dataset, since the beginning was imbalanced which a special case for classification problem where the class distribution is not uniform among the classes. Approximately, 90% (1714 pairs) of our dataset is composed the majority True class and the 10% (212 pairs) minority is the negative class.

There are a few ways you can deal with imbalanced datasets. One of them is under sampling which involves removal of some of data from the majority class (True) to result in a balanced distribution of both classes. However, our dataset has a quite small size, so removing part of it is not an option. On the other hand, there are also methods of oversampling which can assist on increasing the numbers of your minority class (False). In this way, augmented data are created in order to increase the amount of the data of the minority class. This was also the solution that we select to implement in order to improve the performance of the model and eliminate the problem of imbalance dataset. We implement the following steps to create the new data set:

1. Eliminate 300-length "A and B candidates" from the original dataset (244 of class 1 and 56 of class 0) to use as unseen test set for model evaluation
2. From the 1626-length "A and B candidates" (1470 of class 1 and 156 of class 0) remained data set, the one that were in class 0 where eliminated (new dataset contains 1470-length "A and B candidates" of class 1)
3. We randomly select 400-length "A and B candidates" from the new data set in order to create the augmented data.
4. We modify the contents of the some of the elements of the 400-length dataset. The modifications are based on the study of the already existing entities of class 0. For instance, in some situations the deference between A and B candidates' email was that the one of them had some numbers of extra letters on it. Thus, we randomly import some extra letters on the email of one of the two candidate.
5. Merge and shuffle the augmented dataset with the remain one, having a new data set of 2026 row.
6. Create the features 32 features from the new dataset.
7. Feature evaluation and keep only the 15.

Having the new dataset (2026*15) we recreate our classification model. In the first part of the test, we present how was the performance of three best classifiers using cross validation on the new dataset. Is important to mention that the cross validation is using both original and augmented data. In the second part, we considered the new dataset as training set and use the 300 left out dataset as test set.

1st Part

The sklearn.model_selection have been used in order to evaluate the model by building the learning curve plot function. The implemented function creates a simple plot of the test and training set. We use as parameters the estimator which is the classifier for the model creation, the data set, the number k of the cross-validation method and the number of jobs that will be run in parallel.

Each learning curve image represents with red the performance of training set and green the performance of cross validation. The field around the line presents the mean value minus the standard deviation of the f1-score. The absolute number of training examples that is used to generate the learning curve is (0.01, 1) and the classification sampling is 50 (the o in the line). Furthermore, the k value of cross-validation is 5 and the number of jobs that run in parallel are 4.

Decision Tree Classifier

	precision	recall	f1-score	support
0	0.95	0.85	0.90	111
1	0.94	0.98	0.97	294
avg / total	0.95	0.92	0.93	405

*the table contains the mean of the precision, recall, f1-score and support of k=5

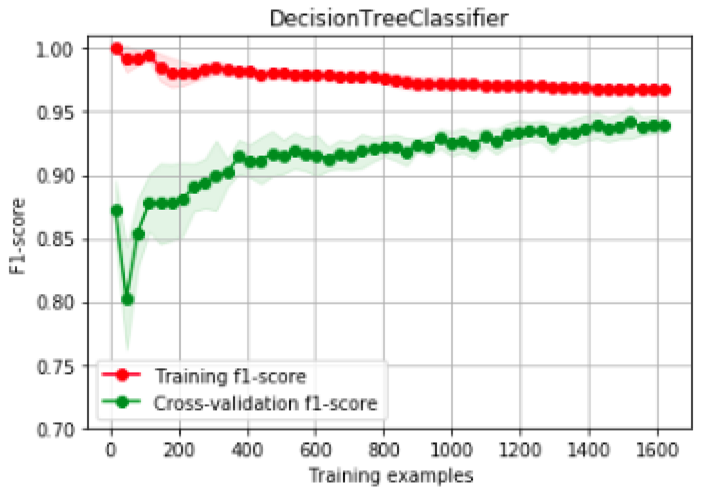


Image 9. Learning curve of the Decision Tree Classifier

Ada Boost Classifier

	precision	recall	f1-score	support
0	0.95	0.86	0.90	111
1	0.95	0.98	0.97	294
avg / total	0.95	0.92	0.93	405

*the table contains the mean of the precision, recall, f1-score and support of k=5

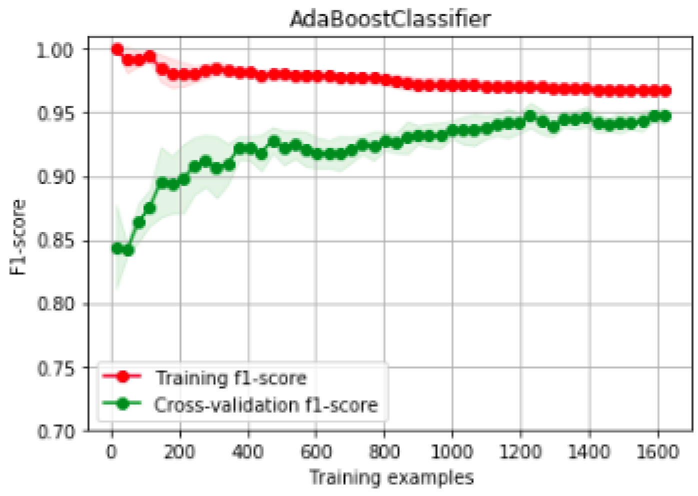


Image 10. Learning curve of the Ada Boost Classifier.

Random Forest Classifier

	precision	recall	f1-score	support
0	0.94	0.86	0.90	111
1	0.95	0.98	0.97	294
avg / total	0.95	0.92	0.93	405

*the table contains the mean of the precision, recall, f1-score and support of k=5

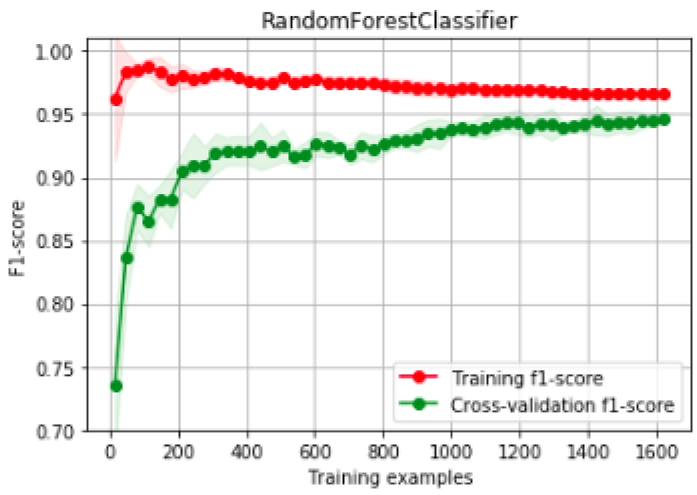


Image 11. Learning curve of the Random Forest Classifier.

Extra Trees Classifier

	precision	recall	f1-score	support
0	0.94	0.84	0.89	111
1	0.94	0.98	0.96	294
avg / total	0.94	0.91	0.93	405

*the table contains the mean of the precision, recall, f1-score and support of k=5

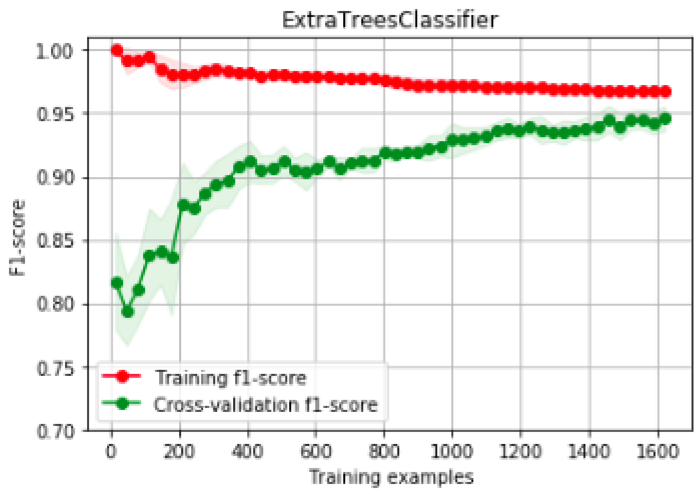


Image 12. Learning curve of the Extra Trees Classifier.

2nd Part

In this part, we used all the new dataset to train the model and for model evaluation the test dataset (300 “A and B candidates” from the original dataset). Even if there were significant improvements on the performants of all models, the Ada Boost, Decision Tree, Random Forest and Extra Trees Classifiers have still the optimal results.

For each classifier we are plotting the ROC curve and the learning curve. From skplt.metrics module the function plot_roc was used to plot the ROC while for the learning curve was used manually made above learning curve plot function.

Ada Boost Classifier

	precision	recall	f1-score	support
0	0.90	0.93	0.91	56
1	0.98	0.98	0.98	244
avg / total	0.94	0.96	0.95	300

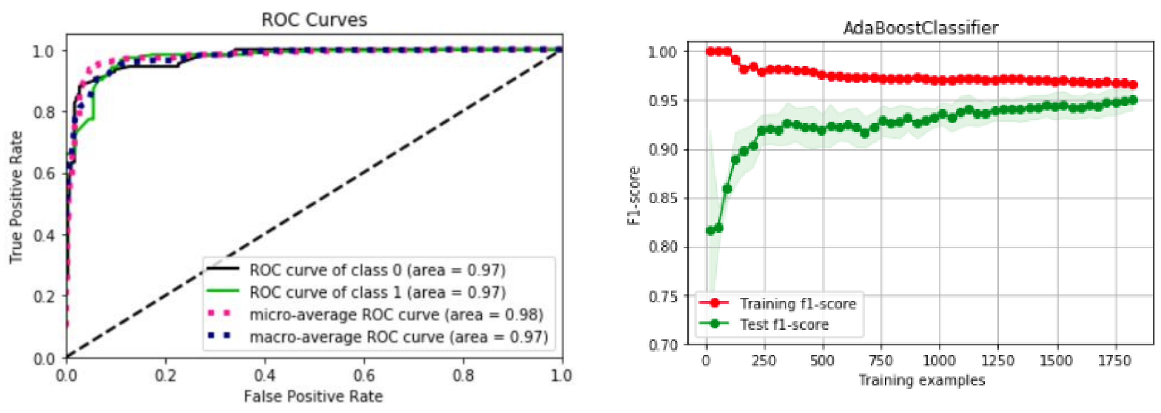


Image 13. Learning and ROC curve of the Ada Boost Classifier.

Random Forest Classifier

	precision	recall	f1-score	support
0	0.96	0.93	0.95	56
1	0.98	0.99	0.99	244
avg / total	0.97	0.96	0.98	300

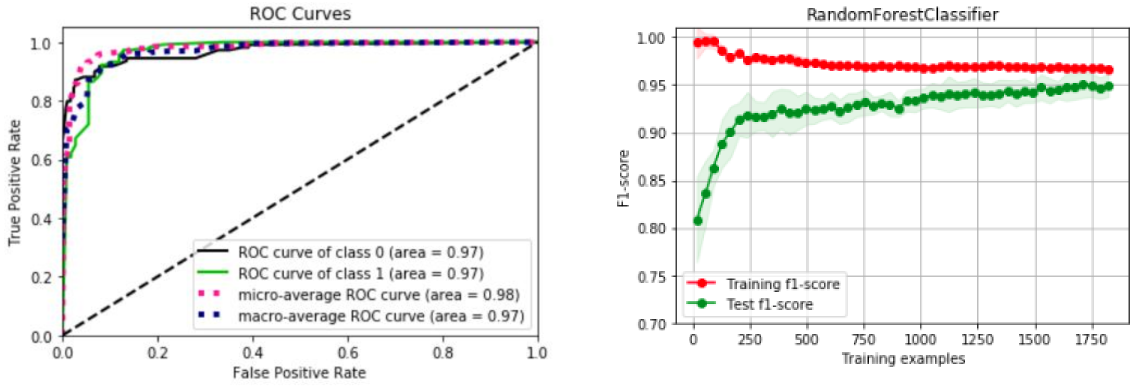


Image 14. Learning and ROC curve of the Random Forest Classifier.

Extra Trees Classifier

	precision	recall	f1-score	support
0	0.93	0.93	0.93	56
1	0.98	0.98	0.98	244
avg / total	0.96	0.95	0.95	300

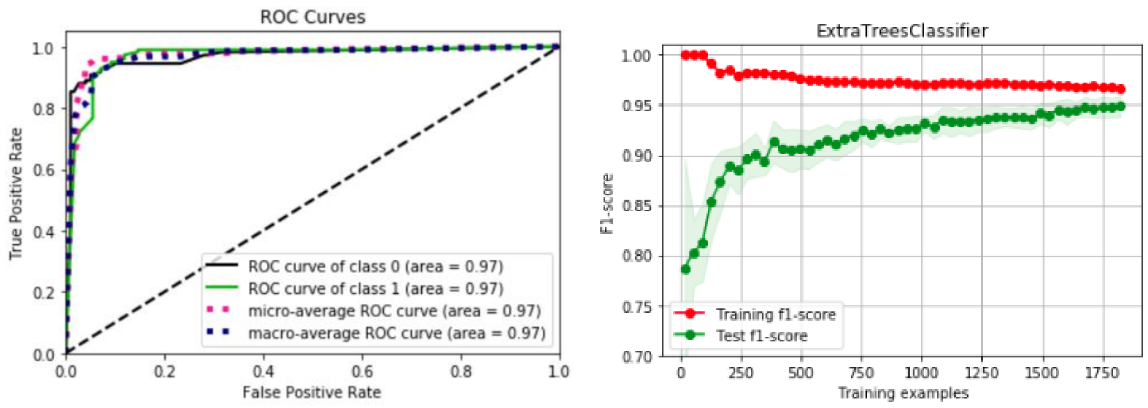


Image 15. Learning and ROC curve of the Extra Trees Classifier.

Decision Trees Classifier

	precision	recall	f1-score	support
0	0.96	0.93	0.95	56

1	0.98	0.99	0.99	244
avg / total	0.96	0.98	0.98	300

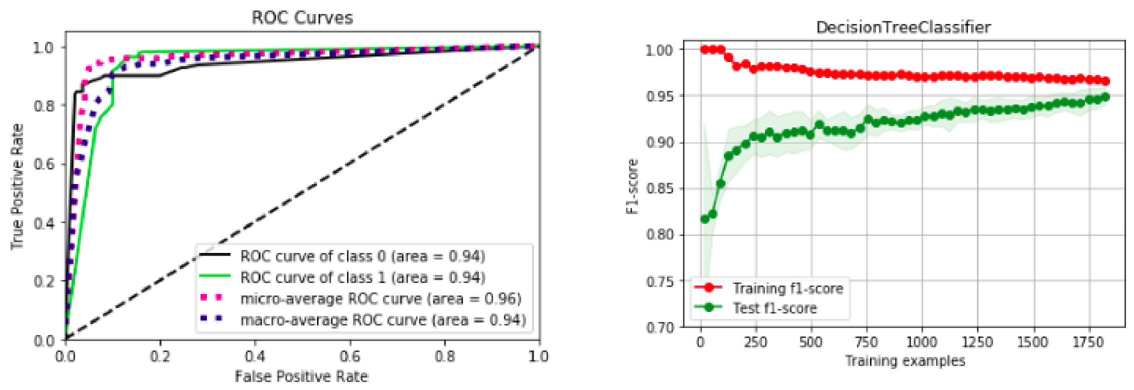


Image 16. Learning and ROC curve of the Dicision Trees Classifier.

SVM

	precision	recall	f1-score	support
0	0.78	0.89	0.83	56
1	0.97	0.94	0.96	244
avg / total	0.94	0.93	0.94	300

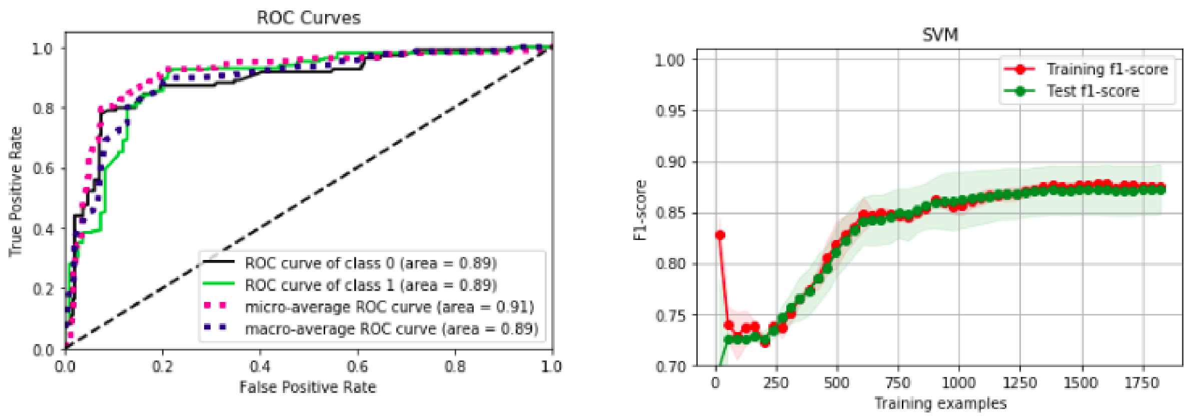


Image 17. Learning and ROC curve of the SVM.

K Neighbors Classifier

	precision	recall	f1-score	support

0	0.89	0.86	0.87	56
1	0.97	0.98	0.98	244
avg / total	0.95	0.95	0.95	300

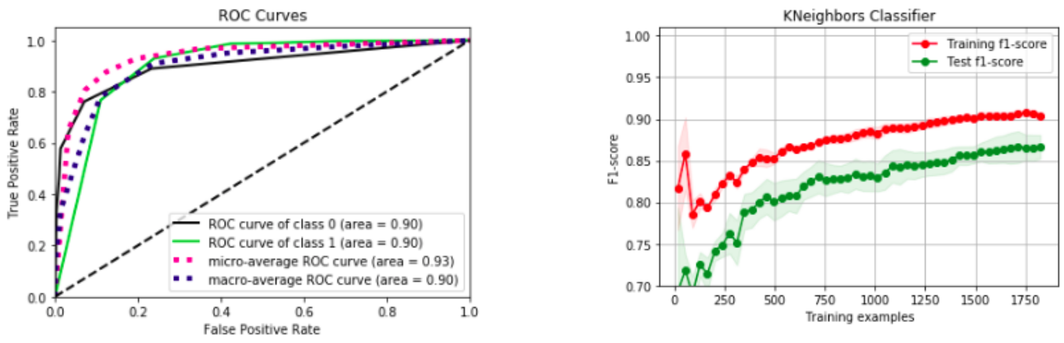


Image 18. Learning and ROC curve of the K Neighbors Classifier.

Bagging Classifier

	precision	recall	f1-score	support
0	0.96	0.89	0.93	56
1	0.98	0.99	0.98	244
avg / total	0.97	0.97	0.97	300

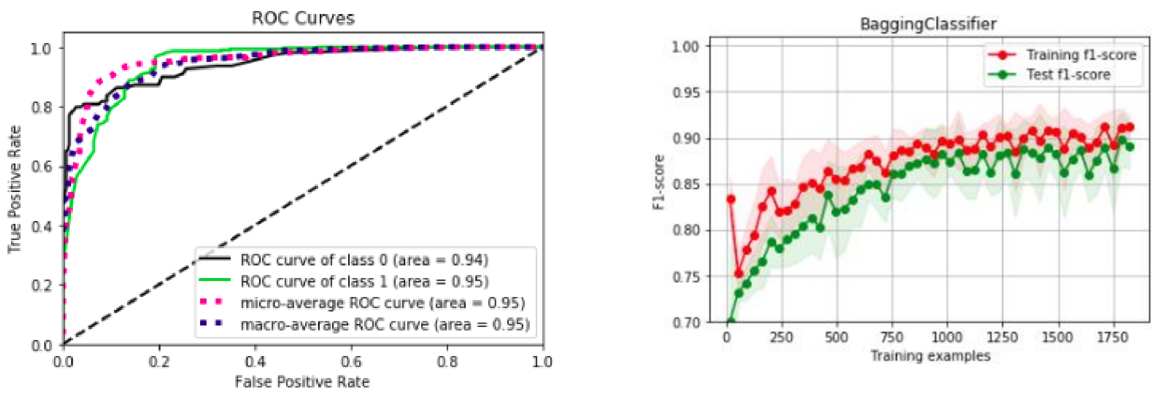


Image 19. Learning and ROC curve of the Bagging Classifier.

Several other tests were implemented such as creating larger amount of augmented data or increasing the length of left out test set. The performance of the classifiers was almost the same with the already represented once.

6. Discussion and Limitations

Trying to combine all the theoretical background, gained through the lectures and personal study, in order to solve real world problems, it is always fascinating.

In this work, we realized how important is the data study and we learned how to deal with problems that dataset may contains (e.g. imbalance). Studying and understanding the dataset in a major key on creating an efficient model. Moreover, a deep understanding of what information your data contain will guide the way to solve problems that will appear. Indeed, we needed to use augmented data in order to have a model with better performance. If we did not have a good understanding of our data, we will not be able to produce the augmented data in a proper and efficient way. As a result, instead of eliminating the imbalance problem, it could be transformed in overfeeding or misevaluation problem.

Dealing with images was also one very constructive and creative point of this work. There are several well train algorithms for face recognition, thus we employed one of them. Although, before using the already created model, there is an image pre-processing step which contain the image downloading, image filtering (e.g. there is face or not) and the image size converting (e.g. using padding). The process of face recognition (including the pre-processing) demands good hardware equipment in order not to consume a lot of time.

Entity resolution problem was indicated in two main points in the thesis, one as candidates matching and one like face recognition matching. On the first, we had to deal with individuals' profiles and match them by using different statistic techniques implemented in python. On the second, we had a wide range of individual's images that needed to be matched by using the face recognition process. Indeed, the images of a candidate (which could be 5 images) should be matched together with the images of another candidate (which could have 8 images) and have a numerical result.

The main limitations that we faced were the semi-structured, imbalanced and small dataset we had. The lack of having the same elements for all the candidates produces many missing values (remove features). For instance, some of the candidates did not have educational or the demographics elements. Moreover, many of the URL links that were provided did not work, as a result we could not have access to the information.

It is worth mentioning that the model and the feature evaluation are very crucial for every type of machine learning implementation. There is a wide range of ways to evaluate the performance of the model and the features, although every time this should be adopted on the problem and the dataset. For instance, our dataset was imbalance if we got in consideration only the precision, recall and f1 scare without having a look on confusion matrix or REC probably we would have wrong perspective of the model performance. Thus, is essential the model and feature evaluation with different methods.

7. Conclusion and future work

Entity resolution of semi-structured or unstructured data is one of the biggest challenges that scientific community deals with the last decade. This work aims to demonstrate the potential difficulties of performing entity resolution and present an alternative way of solving the entity resolution problem of a semi-structured imbalanced dataset. Indeed, the floating search methods show a great promise of being useful in semi-structured or unstructured databases. Our approach combines machine learning and deep learning techniques in order to enhance and contribute on solving the entity resolution problem. Moreover, plethora of python libraries such as Fuzzywuzzy, Difflib, Urltools and Face recognition, have been employed, individually or in combination, to achieve work's needs.

The experiments present the performance of a different prospective solution of a real-world entity resolution problem. Matching candidates base on their profile characteristics and images based on the face depicted on them. In this sense, producing a dataset with entity matched features in order to develop a classification model. Moreover, problems such as imbalanced and semi-structured dataset have been surpassed by using augmented data or statistical techniques.

Future work will be devoted to enhance the matching and prediction analysis, considering other parameters (e.g., skills, age and more images) , extending the experiments for larger dataset and develop other parts of the approach.

References

[1] Whang, S. E., Marmaros, D., & Garcia-Molina, H. (2013). Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering*, *25*(5), 1111-1124.

[2] Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, *194*, 28-61.

[3] Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI magazine*, *26*(1), 83.

[4] Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, *69*(2), 197-210.

[5] On, B. W., Koudas, N., Lee, D., & Srivastava, D. (2007, April). Group linkage. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (pp. 496-505). IEEE.

[6] Kalashnikov, D. V., & Mehrotra, S. (2006). Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (TODS)*, *31*(2), 716-767.

[7] Cohen, W., Ravikumar, P., & Fienberg, S. (2003, August). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation* (Vol. 3, pp. 73-78).

[8] Hernández, M. A., & Stolfo, S. J. (1995, June). The merge/purge problem for large databases. In *ACM Sigmod Record* (Vol. 24, No. 2, pp. 127-138). ACM.

[9] Jin, L., Li, C., & Mehrotra, S. (2003, March). Efficient record linkage in large data sets. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on* (pp. 137-146). IEEE.

[10] De Vries, T., Ke, H., Chawla, S., & Christen, P. (2009, November). Robust record linkage blocking using suffix arrays. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 305-314). ACM.

[11] Gravano, L., Jagadish, H. V., Ipeirotis, P. G., Srivastava, D., Koudas, N., & Muthukrishnan, S. (2003). Approximate String Joins in a Database (Almost) for Free--Erratum.

[12] McCallum, A., Nigam, K., & Ungar, L. H. (2000, August). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 169-178). ACM.

[13] Bilenko, M., Kamath, B., & Mooney, R. J. (2006, December). Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on* (pp. 87-96). IEEE.

[14] Michelson, M., & Knoblock, C. A. (2006, July). Learning blocking schemes for record linkage. In *AAAI* (pp. 440-445).

[15] Whang, S. E., Menestrina, D., Koutrika, G., Theobald, M., & Garcia-Molina, H. (2009, June). Entity resolution with iterative blocking. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (pp. 219-232). ACM.

[16] Thor, A., & Rahm, E. (2007, January). MOMA-A Mapping-based Object Matching System. In *CIDR* (pp. 247-258).

[17] Benjelloun, O., Garcia-Molina, H., Kawai, H., Larson, T. E., Menestrina, D., Su, Q., ... & Widom, J. (2006). *Generic entity resolution in the serf project*. Stanford InfoLab.

[18] Bilenko, M., & Mooney, R. J. (2003, August). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 39-48). ACM.

[19] Ho, T. K., Hull, J. J., & Srihari, S. N. (1994). Decision combination in multiple classifier systems. *IEEE transactions on pattern analysis and machine intelligence*, *16*(1), 66-75.

[20] Christen, P. (2008, August). Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1065-1068). ACM.

[21] Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, *69*(2), 197-210.

[22] Kumar, S., & Hebert, M. (2005, October). A hierarchical field framework for unified context-based classification. In *null* (pp. 1284-1291). IEEE.

[23] Wang, H., Fan, W., Yu, P. S., & Han, J. (2003, August). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 226-235). AcM.

[24] Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, *9*(3), 293-300.

[25] Qu, Y., Adam, B. L., Yasui, Y., Ward, M. D., Cazares, L. H., Schellhammer, P. F., ... & Wright, G. L. (2002). Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. *Clinical chemistry*, *48*(10), 1835-1843.

[26] Quinlan, J. R. (1996, August). Bagging, boosting, and C4. 5. In *AAAI/IAAI, Vol. 1* (pp. 725-730).

[27] Horton, P., & Nakai, K. (1997, June). Better Prediction of Protein Cellular Localization Sites with the it k Nearest Neighbors Classifier. In *Ismb* (Vol. 5, pp. 147-152).

[28] Papadakis, G., Tsekouras, L., Thanos, E., Giannakopoulos, G., Palpanas, T., & Koubarakis, M. (2017, May). JedAI: The Force behind Entity Resolution. In *European Semantic Web Conference* (pp. 161-166). Springer, Cham.

[29] https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

[30] https://scikit-learn.org/stable/modules/generated/sklearn.tree.
DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[31] https://xgboost.readthedocs.io/en/latest/tutorials/dart.html