Ministry of Higher Education and Scientific Research

UNIVERSITY YAHIA FARES OF MEDEA

Faculty of Science

Department of Mathematics and Computer Science



Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of Master in Computer Science

***Specialty*:** *Systems Engineering and Web Technology*

TOPIC:

Multi-Label Classification of Social Networks' Users Based on Services

- **Proposed and Directed by:**
  - MR. BOUJEBBOUR Karim.
- **Prepared and Presented by:**
  - MR. AMROUCHE Mohamed Islem.
  - MR. CHOUTRI Adel Djalil.

Acknowledgments

In the name of Allah, the most Merciful, the Most Compassionate all praise be to Allah, The lord of the worlds; and prayers and peace be upon Mohamed his servant and messenger.

First of all, we must acknowledge and thank the Almighty Allah before and after everything for his help and patience in order to accomplish this work.

We would love to express our deep gratitude and thanks to our dear supervisor Mr. Boudjebbour Karim, Professor at the Computer Science department at the university of Yahia Fares of Medea for his guidance, patience, collaboration, generosity and constructive suggestions for this dissertation.

We also thank all the members of our Families for their moral, physical efforts and advices that helped in the realization of this project.

We extend our deepest gratitude to all our teachers of the university of Yahia Fares, especially those at the Computer Science department, for the trainings they delivered during the last five years.

Special thanks to our jury members for devoting a time reading and examining our dissertation and their comments are warmly welcomed.

We also thank MRS. Taibouni Amal. & MR. HADDAD Islem for their enormous help and their fruitful pieces of advice.

Finally, we would like to thank all those who have contributed in one way or another to the realisation of this work.

Thanks to All of You

DEDICATION

I thank the Almighty God for granting me

The chance to take this step in my life

I dedicate this modest work, to the two people who are dearest to my heart.

To whom I owe my existence

My father and my mother, you were always by my side to support me

and encouraged me to fight without ever stopping halfway, may God help and

you.

To my aunt Amal for her considerable help and fruitful Tips

To Ms. Benyahia Zineb & Mr. Haddad Islem for their aid and support.

To my Grandmothers YAMINA & ZAHIA

To my parents and my brother Walid & my sister Ines

To all the CHOUTRI & TAIBOUNI family

To my binomial Islem Amrouche

To my dear friends

To all the people who love me

And all those who have helped me by any means

ADEL DJALIL CHOUTRI

DEDICATION

I thank the Almighty God for granting me

The chance to take this step in my life

I dedicate this modest work, to the two people who are dearest to my heart.

To whom I owe my existence

My father and my mother, you were always by my side to support me

and encouraged me to fight without ever stopping halfway, may God help and

you.

To Ms. Benyahia Zineb and Mr. Haddad Islem for their aid and support.

To my brothers Billel, Yacine & my sister Hadjer

To my sister's husband Dr.Atchi Yahia for his considerable support.

To all the Amrouche family

To my binomial Adel Choutri and

To my dear friends

To all the people who love me

And all those who have helped me by any means.

MOHAMED ISLEM AMROUCHE

- **Abstract:**

Keywords: *Multi-label classification, web service, social network, user profile, machine learning.*

Most classification problems assign a single class to each example or instance. However, there exist other classification tasks where each instance can be associated with one or more classes. The latter represents a task known as Multi-label classification. Our work consists in using this notion in the recommender systems field. For this purpose, we have designed a mechanism that extracts the interests of users based on the web services they have accessed to, in order to use that in a multi-label classification architecture. Once the classification model is built, It allows to predict web services that interest new users.

Our study was conducted on several datasets of different sizes concerning the number of labels and instances, where the instances represent users' profiles and the labels are the web services that interest these users.

We applied several multi-label classification techniques and compared the obtained results using a variety of evaluation measures, in order to extract the factors that would allow us to obtain better results, as well as the best technique for our case study.

- **تلخيص :**

الكلمات الرئيسية: التصنيف متعدد العلامات ، خدمة الويب ، الشبكة الاجتماعية ، تعريف المستخدم ، التعلم الآلي

معظم مشاكل التصنيف تتم عن طريق إرفاق علامة واحدة لكل عينة. ومع ذلك ، توجد مهام تصنيف أخرى حيث يمكن ربط كل عينة بعلامة أو أكثر، يعرف هذا بالتصنيف متعدد العلامات. يتمثل عملنا في استخدام هذه الفكرة في مجال أنظمة التوصية. لهذا قمنا بتصميم آلية تستخلص اهتمامات المستخدمين بناءً على خدمات الويب التي قاموا بالولوج إليهم من قبل ، من أجل استخدام هذه المعلومات لتصميم بنية تصنيف متعددة العلامات. بمجرد بناء نموذج التصنيف ، يمكن للنموذج تنبؤ خدمات ويب تهم مستخدمين جد.

أجريت دراستنا على العديد من مجموعات البيانات ذات الأحجام المختلفة فيما يتعلق بعدد العينات والعلامات ، حيث تعتبر العينات كملفات تعريف المستخدمين، والعلامات كخدمات ويب تهم هؤلاء المستخدمين.

قمنا بتطبيق العديد من تقنيات التصنيف متعدد العلامات وقارنا النتائج التي تم الحصول عليها باستخدام مجموعة متنوعة من مقاييس التقييم، من أجل استخراج العوامل التي من شأنها أن تسمح لنا بالحصول على نتائج أفضل ، وكذلك أفضل تقنية لدراسة الحالة الخاصة بنا.

- **Resumé:**

Mots clés : *Classification multi-label, service web, réseau social, profil utilisateur, apprentissage automatique*

La plupart des problèmes de classification attribuent une seule classe à chaque exemple ou instance. Cependant, il existe une autre forme de classification où chaque instance peut être associée à une ou plusieurs classes. Cette dernière est connue sous le nom de classification multi-label. Notre travail consiste à utiliser cette notion dans le domaine des systèmes de recommandation. Pour cela, nous avons conçu un mécanisme qui extrait les intérêts des utilisateurs en fonction des services web auxquels ils ont accédé, afin de les utiliser dans une architecture de classification multi-label. Une fois le modèle de classification construit, il permettra de prédire les services web qui seront susceptible d'intéresser de nouveaux utilisateurs.

Notre étude a été menée sur plusieurs ensembles de données avec des tailles différentes concernant le nombre de classes (labels) et d'instances, où les instances représentent les profils des utilisateurs et les labels les services web qui intéressent ces utilisateurs.

Nous avons appliqué plusieurs techniques de classification multi-labels et comparé les résultats obtenus à l'aide de diverses mesures d'évaluation, afin d'extraire les facteurs qui nous permettraient d'obtenir de meilleurs résultats, ainsi que la meilleure technique pour notre étude de cas.

- **RS:** Recommender systems
- **CV:** Cross-validation
- **MLC:** Multilabel classification
- **MLD:** Multi-label dataset
- **Knn:** k-nearest neighbors
- **MLkNN:** Multi-labe k-nearest neighbors
- **NN:** Neural network
- **DNN:** Deep neural network
- **DT:** Decision tree
- **RF:** Random forest
- **BR:** Binary relevance
- **CC:** Classifier chains
- **COM:** Component object model
- **DCOM:** Distributed component object model
- **LAN:** Local area network
- **RPC:** Remote procedure call
- **XRD:** External data representation
- **XML:** Extensible markup language
- **SOAP:** Simple object access protocol
- **Rest:** Representational state transfer
- **HTTP:** Hypertext transfer protocol
- **HTTPS:** Hypertext transfer protocol secure
- **SOA:** Service-oriented architecture
- **JSON:** JavaScript object notation
- **WSDL:** Web services description language
- **URI:** Uniform resource identifier
- **IDDI:** Universal description, discovery, and integration
- **W3C:** World wide web consortium

- **EDI:** Electronic data interchange
- **FTP:** File transfer protocol
- **SMTP:** Simple mail transfer protocol
- **POP:** Post office protocol
- **B2c:** Business to consumer
- **B2B:** Business to business
- **OVA:** One vs all
- **OVO:** One vs one
- **ID3:** Iterative dichotomiser 3
- **i.e:** stands for id est, which means "that is."
- **ANN:** Artificial neural network
- **ReLU:** rectified linear unit

Table of Content

Chapitrer I: Web Services & Recommender Systems 3

1. Introduction: **3**

2. History and birth of web services: **3**

3. Service-oriented Architecture (SOA): **4**

3.1. Definition: 4

3.2. (SOA) Importance: 4

3.3. Components of the SOA Architecture: 5

3.3.1. Service Provider: 5

3.3.2. Service Consumer: 5

3.3.3. Service Registry: 5

4. Service (Software Component): **6**

4.1. Types of Services: 6

4.1.1. According to the Structure: 6

4.1.2. According to the Communication (Service Call): 6

5. Web Services Technologies: **7**

6. Definition of web Services: **7**

6.1. Definition 1: 7

6.2. Definition 2: 8

6.3. Features: 8

6.4. The Web Services Model: 8

6.4.1.1. Service Provider: 9

6.4.1.2. Service Requestor (service consumer): 9

6.4.1.3. Broker (Service Registry): 10

6.5. Operations on Web Services: 10

7. Web services protocols: 11

7.1. XML: 12

7.2. XML-RPC: 12

7.3. SOAP: 12

7.4. REST: 13

7.5. WSDL: 13

7.6. UDDI: 14

8. Web service protocol stack: 14

8.1. Transport Service: 15

8.2. XML Messaging: 15

8.3. Service Description Protocol (WSDL): 15

8.4. Service Discovery Protocol (UDDI): 16

9. Web services contributions: 16

9.1. Interoperability: 16

9.2. Standardized protocol: 16

9.3. Low cost communication: 16

9.4. Application of web services: 17

10. Web Services and Recommender Systems: 17

11. Recommender Systems: 17

12. Quality of Recommender Systems: 18

12.1. Relevance: 19

12.2. Novelty: 19

12.3. Serendipity: 19

12.4. Diversity: 19

13. Purpose of Recommender Systems: 19

14. History of Recommender Systems: 20

15. Areas of Recommender Systems: 21

16. User Profile: 21

17. Recommendation Techniques: 23

17.1. Content based method: 23

17.2. Advantages of this Method: 24

17.2.1. Disadvantages of this Method: 24

17.3. Collaborative Method: 25

17.3.1. User-user Collaborative Filtering: 25

17.3.2. Item-Item Collaborative Filtering: 26

17.4. Hybrid method: 26

Chapter II: Multi-label Classification 27

1. Introduction: 27

1.1. Lazy learners: 27

1.2. Eager learners: 27

2. Types of Classification: 27

2.1. Single-Label Classification: 28

2.1.1. Binary Classification: 28

2.1.2. Multiclass Classification: 29

2.1.3. Some of the algorithms used to perform a single-label classification: 30

2.1.3.1. Decision Tree: 30

2.1.3.2 Naive Bayes: 33

2.1.3.3. Artificial Neural Networks: 35

2.1.3.4. k-Nearest Neighbor (KNN): 43

2.2. Multi-Label Classification: 43

2.2.1. Case Studies: 44

2.2.1.1. Audio Categorization: 44

2.2.1.2. Image Categorization: 44

2.2.1.3. Bioinformatics: 45

2.2.1.4. Text Categorization: 45

2.2.2. Techniques for Solving a Multi-Label Classification problems: 46

2.2.2.1. Problem Transformation methods: 47

Chapter III: Conceptual Study 55

1. Introduction: 55

2. Objectives: 55

3. Related work: 55

4. Methodology: 58

4.1. Preprocessing: 59

- Acquiring the dataset 59
- Importing all the crucial libraries: 60
- Importing the dataset: 60
- Dropping unnecessary features: 60
- Identifying and handling the missing values: 60
- Encoding the categorical data: 60

- Splitting the dataset:..... 61
  - Test size: 62
  - Train size:..... 62
  - Random state:..... 62
- Feature scaling: 62
- Feature selection: 63

4.2. Methods and Algorithms:..... 63

5. Solution overview:..... 64

5.1. User profile description:..... 64

5.1.1. First Category (User's information):..... 65

5.1.2. Second Category (User's preferences of the desired web services): 65

5.2. Dataset Construction:..... 66

5.3. Models:..... 66

5.3.1. Models based on the problem transformation technique: 67

5.3.1.1. Binary Relevance with Decision Tree Classifier model:..... 68

5.3.1.2. Binary Relevance based with Random Forest Classifier Model: 70

5.3.1.3 Binary Relevance based with Deep Neural Network model: 71

5.3.2. Problem Adaptation Technique Model:..... 73

5.3.3. The Classifier Chain technique model :..... 74

5.3.3.1. Classifier Chain with Decision Tree Classifier:..... 75

5.3.3.2. Classifier Chain with Random Forest classifier: 76

6. Evaluation methods..... 76

6.1. Hamming Loss: 77

6.2. Accuracy: 77

6.3. Precision and Recall: 77

6.4. F1 score: 78

6.5. Subset Accuracy: 78

7. Note: 79

8. Conclusion: 81

Chapter IV: Evaluation & Conclusion 84

1. Introduction: 84

2. Development Tools: 84

2.1. Hardware used: 84

2.1.1. Computer: 84

2.2. Operating system: 84

2.3. Programming Languages: 85

2.3.1. Python: 85

Python Packages used: 85

2.4. Development environment used: 89

2.4.1. Anaconda: 89

2.4.2. Spyder: 89

2.4.3. Jupyter Notebook: 90

2.4.4. Sublime Text: 90

3. Models Evaluation: 90

3.1. Models evaluation results: 91

3.1.1. Binary Relevance-Random Forest: 91

3.1.2. Binary Relevance-Decision Tree: 93

3.1.3. Classifier Chain-Decision Tree: 94

3.1.4. Classifier Chain-Random Forest: 94

3.1.5. Multi-label k Nearest Neighbors: 95

3.1.6. Binary Relevance-Neural Network: 96

3.1.7. Models comparison with five-fold Cross-Validation: 97

3.2. Models Evaluation Discussion: 98

3.3. Evaluating with Cross-Validation: 104

4. Interface: 111

4.1. Django: 112

5. Conclusion: 115

LIST OF FIGURES:

FIGURE I-1: A GENERAL DIAGRAM OF AN SOA ARCHITECTURE. 6
FIGURE I-2: WEB SERVICE ARCHITECTURE (SOAP, WSDL, UDDI). [99] 9
FIGURE I-3: WEB SERVICE OPERATIONS. 10
FIGURE I-4: THE DIFFERENT PROTOCOLS OF WEB SERVICES. 11
FIGURE I-5: DIAGRAM SHOWING THE SOAP PROTOCOL MESSAGE STRUCTURE. 13
FIGURE I-6: WEB SERVICE PROTOCOL LAYER'S. 15
FIGURE I-7: GOALS OF RECOMMENDATION ENGINES. 18
FIGURE I-8: USER PROFILING. 21
FIGURE I-9: CONTENT BASED METHOD. 24
FIGURE I-10: USER-USER MODEL. 25

FIGURE I-11: ITEM-TO-ITEM MODEL. 26
FIGURE II-1: BINARY CLASSIFICATION PROBLEM. (HERRERA ET AL., 2016). 28
FIGURE II-2: MULTI-CLASS CLASSIFICATION PROBLEM. (HERRERA ET AL., 2016) 29
FIGURE II-3: AN EXAMPLE OF A CLASSIFICATION TREE TO PREDICT THE GENDER OF A HUMAN BASED ON THE HEIGHT END THE WEIGHT 29
FIGURE II-4: A FIGURE SHOWS HOW RANDOM FOREST TREES VOTE FOR A PREDICTION.(GANDHI, 2019). 33
FIGURE II-5: GAUSSIAN DISTRIBUTION (NORMAL DISTRIBUTION).(GANDHI, 2019) 34
FIGURE II-6: NEURAL NETWORK WITH HIDDEN LAYERS.(ERKER, N.D.) 35
FIGURE II-7: PERCEPTRON. (CHAUHAN, 2019) 36
FIGURE II-8: THRESH HOLD FUNCTION.(CHAUHAN, 2019) 37
FIGURE II-9:THE SIGMOID FUNCTION 37
FIGURE II-10: RELU FUNCTION.(SIVARAJKUMAR, 2019) 38
FIGURE II-11: FORWARD AND BACK PROPAGATION.(CHAUHAN, 2019) 40
FIGURE II-12: ARCHITECTURE OF DIFFERENT TYPES OF NEURAL. 41
FIGURE II-13: IMAGE SHOWING K-NEAREST NEIGHBORS (KNN) ALGORITHM, WITH A NUMBER AS A VALUE OF K. (ASIRI, 2018) 45

FIGURE II-14: MULTI-LABEL CLASSIFICATION PROBLEM 43

FIGURE II-15: THE DIFFERENCE BETWEEN BINARY CLASSIFICATION, MULTICLASS CLASSIFICATION. AND MULTI-LABEL CLASSIFICATION. YAHOO JAPAN.(*EXACT PASSIVE-AGGRESSIVE ALGORITHM FOR MULTICLASS CLASSIFICATION USING SUPPORT CLASS | REQUEST PDF*, N.D.) 45

FIGURE II-16: IMAGE SHOWING K-NEAREST NEIGHBORS (KNN) ALGORITHM, WITH A NUMBER AS A VALUE OF K. (ASIRI, 2018) 43

FIGURE II-17: BINARY RELEVANCE (BR) TRANSFORMATION DIAGRAM.(HERRERA ET AL., 2016) 48

FIGURE II-18: LABEL POWERSET (LP) TRANSFORMATION DIAGRAM.(HERRERA ET AL., 2016) 49

FIGURE III-1: A VIEW OF OUR METHODOLOGY. 58

FIGURE III-2: A VIEW OF OUR PRE-PROCESSING APPROACH. 59

FIGURE III-3: PROFILE'S DESCRIPTION. 64

FIGURE III-4: FEATURES WITH LABELS (CLASSES). 65

FIGURE III-5: TRANSFORMATION TECHNIQUE MODEL'S. 68

FIGURE III-6: BINARY RELEVANCE WITH DECISION TREE CLASSIFIER MODEL. 68

FIGURE III-7: DECISION TREE ALGORITHM. 69

FIGURE III-8: RANDOM FOREST ALGORITHM. 70

FIGURE III-9: BINARY RELEVANCE BASED WITH DEEP NEURAL NETWORK MODEL. 71

FIGURE III-10: NEURAL NETWORK ARCHITECTURE MODEL. 72

FIGURE III-11: ADAPTION TECHNIQUE MODEL. 73

FIGURE III-12: (MLKNN) PRINCIPLE EXAMPLE TO CLASSIFY A NEW INSTANCE. (SETTOUTI ET AL., 2019) 74

FIGURE III-13: CLASSIFIER CHAIN TECHNIQUE MODEL. 75

FIGURE III-14: CLASSIFIER CHAIN WITH DECISION TREE CLASSIFIER. 75

FIGURE III-15: CLASSIFIER CHAIN WITH RANDOM FOREST CLASSIFIER. 76

FIGURE III-16: STRUCTURE OF THE CONFUSION MATRIX WITH TP, FN, FP AND TN VALUES.(HORAK ET AL., 2017) 79

FIGURE III-17: 4-FOLD CROSS-VALIDATION. (SANJAY, 2018) 80

FIGURE IV-1: EVALUATING ALL THE MODELS AND DATASETS USED USING ACCURACY (THE HIGHER THE BETTER). 99

FIGURE IV-2: EVALUATING ALL THE MODELS AND DATASETS USED USING HAMMING LOSS (THE LOWER THE BETTER). 100

FIGURE IV-3: EVALUATING ALL THE MODELS AND DATASETS USED USING F1-SCORE MACRO (THE HIGHER THE BETTER) 101

FIGURE IV-4: EVALUATING ALL THE MODELS AND DATASETS USED USING F1-SCORE MICRO (THE HIGHER THE BETTER). 102

FIGURE IV-5: EVALUATING ALL THE MODELS AND DATASETS USED USING SUBSET ACCURACY (THE HIGHER THE BETTER). 103

FIGURE IV-6: COMPARING OUR MODELS WITH EACH OTHER USING THE EVALUATION METRICS CALCULATED. 104

FIGURE IV-7: COMPARING OUR MODELS WITH EACH OTHER USING THE EVALUATION METRICS CALCULATED THROUGH 5-FOLD CROSS-VALIDATION ON A 15 LABELS/100K SAMPLES DATASET..... 106

FIGURE IV-8: A PLOT ILLUSTRATING THE RUNNING TIME OF ALL DATASETS WITH ALL MODELS (WITHOUT K-FOLD CV) (THE LOWER THE BETTER). 108

FIGURE IV-9: A PLOT ILLUSTRATING THE RUNNING TIME OF THE DATASETS WITH 100000 SAMPLES WITH ALL MODELS WITH 5-FOLD CV (THE LOWER THE BETTER)..... 110

FIGURE IV-10: THE RUNNING DIFFERENCE BETWEEN MODELS WITH 10 AND 15 LABELS ON 100000 SAMPLES DATASETS.) 111

FIGURE IV-11: FIRST PART OF THE APPLICATION FORM USED FOR MAKING THE PREDICTIONS….113

FIGURE IV-12: SECOND PART OF THE APPLICATION FORM USED FOR MAKING THE PREDICTIONS……113

FIGURE IV-13: THIRD PART OF THE APPLICATION FORM USED FOR MAKING THE PREDICTIONS……114

FIGURE IV-14: THE PREDICATED WEB SERVICES WITH EVALUATION METRICS RUSULTS..……114

List of Tables:

TABLE IV-1: BINARY RELEVANCE RANDOM FOREST MODEL'S RESULTS 93
TABLE IV-2: BINARY RELEVENCE DECISION TREE MODEL'S RESULTS 94
TABLE IV-3: CLASSIFIER CHAIN DECISION TREE MODEL'S RESULTS 94
TABLE IV-4: CLASSIFIER CHAIN RANDOM FOREST MODEL'S RESULTS. 95
TABLE IV-5: MULTI-LABEL K NEAREST NEIGHBORS MODEL'S RESULTS. 96
TABLE IV-6: BINARY RELEVANCE NEURAL NETOWRK MODEL'S RESULTS 97
TABLE IV-7: MODELS COMPARISON WITH FIVE-FOLD CROSS-VALIDATION. 98

General Introduction

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

One important subset of AI is Machine learning, as the name suggests, it trains a machine how to learn, so basically modern solutions have used Machine Learning to structure huge data clusters using classification methods. And the latter means assigning every instance of data to a specific class “labeling them” using intelligent predicting algorithms. And this can literally be applied to anything and everything, from predicting tomorrow’s weather to diagnosing the possibility of a patient to have a certain disease.

In this work, we applied machine learning to predict what web services users on the web might be interest in, using their profiles, however in this case each instance (user) should be assigned to multiple web services (labels), which can’t be done using the traditional classification where each instance is given a single label, which is known as a single-label classification, so a different method has to be applied, but what can it be?

The solution for this is using the Multi-label classification instead of the Single-label classification.

Multi-label classification is an expansion of the classification problem, which involves the prediction of multiple labels simultaneously associated with a single instance, which helps in organizing and structuring data, it can be used in a tremendous amount of fields like protein function classification, music categorization and semantic scene classifications, it also can be used for Recommender Systems to bring relevant content to each user independently on the web.

Multi-label classification has a lot of different algorithms to approach it, in this project we outlined most of them, however we applied them only to a specific domain, which is to predict which web services particular users would most likely have an interest into as mentioned earlier, and this will certainly solve the confusion and the disruption users fall into when being around a lot of irrelevant web services.

In chapter one, we went through a brief description about web services and recommender systems, Then in chapter two we introduced classification presenting some algorithms that would solve this task, then we gave a detailed description about multi-label classification along with the methods and the techniques that are able to solve the MLB problems. Later in chapter three we conducted the conceptual aspect of our project, talking about the models and the methods we used to solve the task and to evaluate it.

Finally in chapter four we discussed and compared our models with each other using some of the well known and accurate evaluation metrics.

Web Services &

Recommender Systems



1. Introduction:

Recommender systems (also known as recommendation systems) are one of the very popular technological solutions in the recent years, and a market of billions of dollars, all due to their huge economic and commercial impact, its programs represent a wild card for modern marketing because of the rapid development of the web and its emergence in multiple fields.

Recommender systems have a direct impact on the multiple services on the internet which come in different forms such as e-commerce, e-bank, social networks, digital libraries, remote cinema...etc, however the attraction towards these services is difficult, that's why recommender systems are important, they direct the internet user considering the profile and behaviours towards products that could potentially interest of the user and satisfy his needs. According to statistics made in 2007 the Traffic of recommender systems alone represents 10 to 30% of internet traffic [1].

2. History and birth of web services:

Web services were born due to the effort of multiple organizations sharing a common interest in developing an (electronic marketplace) that allows the creation of a simple and universal means of communication and data sending between different interlocutors without worrying about how to send a transaction between third parties and how to interpret their different data coming from different organizations. Their goal is to eliminate the isolation of their computer systems with others.

The story of web services begins with the communication between two processes that exchanged information with each other using pipe and shared memory and a COM (Component Object Model). It was extended to LAN networks where two machines could exchange information using DCOM, RPC and SOCKETS but the communication was impossible when the systems were different between the interlocutors because each manufacturer had its own representation of these processes, and this opened the door to XRD (External Data Representation) to solve this problem of incompatibility on which they could specify common standards for the representation of data.

[1] A report entitled Personalized product recommendations: Predicting shoppers needs, emarket from https://twinklemagazine.nl/2009/05/EMarketer_Etailers_worstelen_met_aanbevelingen/

With the advent of the internet and the evolution of the web, the XDR format (which was binary) has been developed to a textual format called (XML) and a new communication protocol has also arrived between applications on the internet called (SOAP) which contributed to the birth of web services.

3. Service-oriented Architecture (SOA):

3.1. Definition:

SOA (Service-Oriented Architecture) is one of the latest trends in software. Service oriented architectures do not represent a technology but rather a method for structuring applications under a service-based approach. (TMEGLIT, 2007)

SOA architecture is based on an architectural style that defines an interaction model between three primary parties: the service provider, who publishes a service description and provides the implementation for the service, a service consumer, who can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service. The service broker provides and maintains the service registry, although nowadays public registries are not in vogue. (Arsanjani & Corporation, 2005)

3.2. (SOA) Importance:

Rather than favoring an application architecture based on technical constraints, the SOA architecture proposes to divide the components of an application into reusable business services (Bey, 2018) and thus the service oriented architecture, defines the way to make software components reusable via service interfaces, its interfaces use common means of communication (standards) in such a way that they can be quickly integrated into a new application without the need for extensive integration each time, each service in a SOA has the code and data integrations necessary to perform a function or a complete and discrete data manipulation.

Service Oriented Architecture represents an important phase in the evolution of application development and integration in recent years. Prior to the emergence of SOA in the 1990s. Developers had to create deep integration and a complex point-to-point connection between applications that using data or services that are hosted in other systems. They had to recreate a new connection each time a software application needed a service. The SOA architecture eliminates

the need to recreate deep integration when an application manipulates external information or services.

SOA-Architecture features the following proprieties:

- Flexibility.
- Interoperability: the ability of two or more systems to interact and work together without technological constraints.
- Reliability.
- Security.
- Reusability: the ability of a system to invoke an existing service that has already been run as well as other services.

3.3. Components of the SOA Architecture:

The SOA Architecture consists mainly of three components:

3.3.1. Service Provider:

The service provider is the source element of a service. It is identified by a contract that defines what the proposed service does and how to access it. And the latter is used by consumers to access the service. (Bey, 2018)

3.3.2. Service Consumer:

The service consumer is the entity requesting a service. To do this it uses the registry to find out what services are available to meet his needs. Once the service has been located, the service consumer extracts the corresponding interface contract from the registry. (Bey, 2018)

3.3.3. Service Registry:

Represents a service registry or directory where the interface contract for each service is recorded along with information used to locate the service. The requester uses the service registry to search for the appropriate service and extract its requirements. (Bey, 2018)

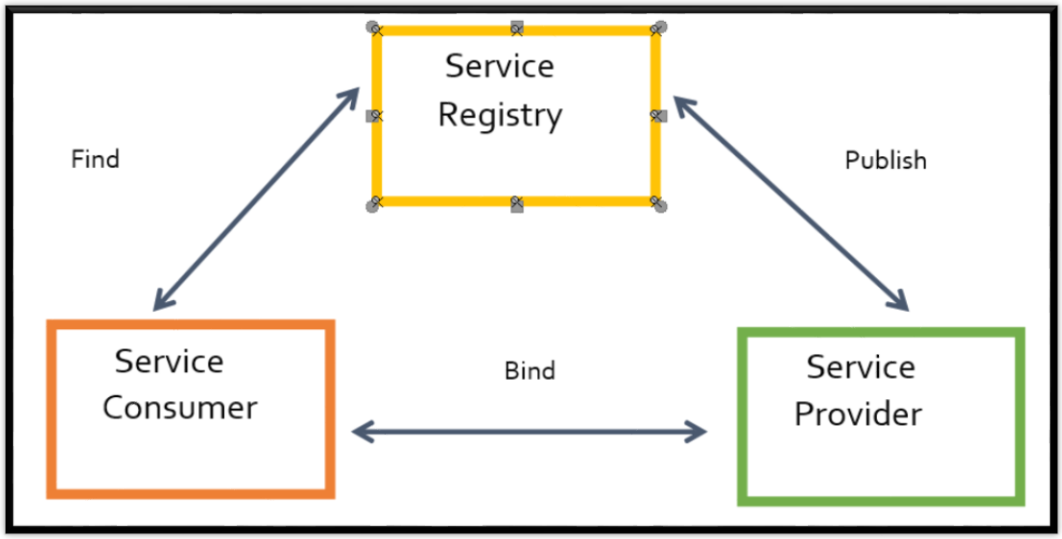


Figure I-1: A General Diagram of An SOA Architecture.

4. Service (Software Component):

It is the focal point of the service-oriented architecture. It represents a logical level of access to the processing, as a service represents a well-defined function or set of functions. It is a component that does not depend on any external service. It is divided into operations that the service can perform, it exposes an interface that defines the offered process in a form of an input message and another of response.(Papazoglou & Van Den Heuvel, 2007)

4.1. Types of Services:

There are several types of services:

4.1.1. According to the Structure:

- Business service: issued from the book of specifications.
- Architecture service: issued from the application itself (usually implemented by one or more business services).

4.1.2. According to the Communication (Service Call):

- **Synchronous Communication:** The calling party requests a service and waits for the service to complete. Only when it receives the result of the service it continues with its

work. A timeout may be defined, so that if the service does not finish within the defined period the call is assumed to have failed and the caller continues.

- **Asynchronous Communication:** The calling party initiates a service call but does not wait for the result. The caller immediately continues with its work without caring for the result. [2]

5. Web Services Technologies:

Web services make interactions between programs. They allow companies to reduce the cost of e-commerce, to deploy solutions more quickly and to open up new opportunities.

The key to reaching this new horizon is a common model for program-to-program communication.

Web services make it possible to integrate applications faster, easier and cheaper than ever before. Integration takes place at a higher level of the protocol stack, based on messages that focus more on the semantics of services and less on the semantics of network protocols, allowing flexible integration of business functions. These features are ideal for connecting web-based enterprises, both between and within enterprises. They provide a unifying programming model for application integration inside and outside the enterprise. (Gottschalk et al., 2002)

6. Definition of web Services:

There are several definitions to define a web service:

6.1. Definition 1:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a format that can be processed by a machine (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP/REST messages, typically conveyed using HTTP with an XML only serialization for SOAP or with XML, JSON, TEXT... serialization for REST, all in conjunction with other Web-related standards.

[2] definition of Synchronous and asynchronous communication from: https://docs.plm.automation.siemens.com

6.2. Definition 2:

A Web service is an interface that describes a collection of operations that are accessible through standardized XML messaging. A Web service is described using a standard, formal XML notion (XML/JSON/TEXT. for REST), called its service description. It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location. The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written. This allows and encourages Web Services-based applications to be loosely coupled, component-oriented, cross-technology implemented. Web Services fulfil a specific task or a set of tasks. They can be used alone or with other Web Services to carry out a complex aggregation or a business transaction.(Gottschalk et al., 2002)

6.3. Features:

A web service is characterised by:

- URI that defines it.
- Its links can be described in an XML format.
- Its definition can be discovered by other web services.
- It can interact directly with other web services through the XML language using internet protocols (usually HTTP and HTTPS).
- A web service must be secured.
- A web service must be reliable in its implementation.

6.4. The Web Services Model:

The web services model is based on the SOA architecture and therefore the latter consists of three main elements:

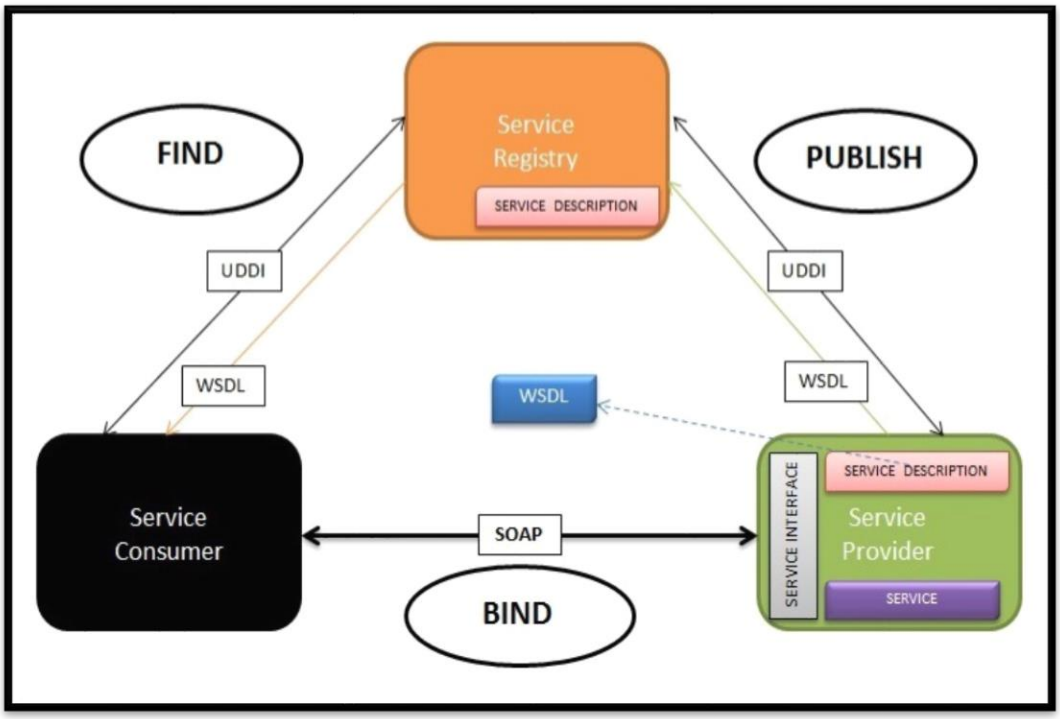


Figure I-2: Web Service Architecture (SOAP, WSDL, UDDI).[3]

6.4.1.1. Service Provider:

The provider creates the web service and makes it available to the clients' applications who want to use it. So, it is the source element of a service. It is identified by a contract that defines what the proposed service does and how to access it. It is used by requesters to access the service. (Fella, 2018)

6.4.1.2. Service Requestor (service consumer):

A requestor is nothing but the client application that needs to contact a web service. The client application can be a .NET, JAVA, or any other language-based application which looks for some sort of functionality via a web service. To do this it uses the broker to find out what services are

[3] Illustration of the web service Architecture and operations from https://dzone.com/articles/soap-web-services-using-cxfjibx-jax-ws

available to meet his needs. Once the service has been located, the requester retrieves the corresponding interface contract from the broker. (Fella, 2018)

6.4.1.3. Broker (Service Registry):

The broker is nothing but the application which provides access to the UDDI. The UDDI enables the client application to locate the web service. This component has been added in order to reduce the gap between the customer service and the concerned web service (service provider) It is defined as a searchable register of service descriptions where a service register.

Providers publish their service descriptions. Service requesters find services and obtain binding information (in service descriptions) for services under development for static linking or during execution for dynamic linking. (Gottschalk et al., 2002)

6.5. Operations on Web Services:

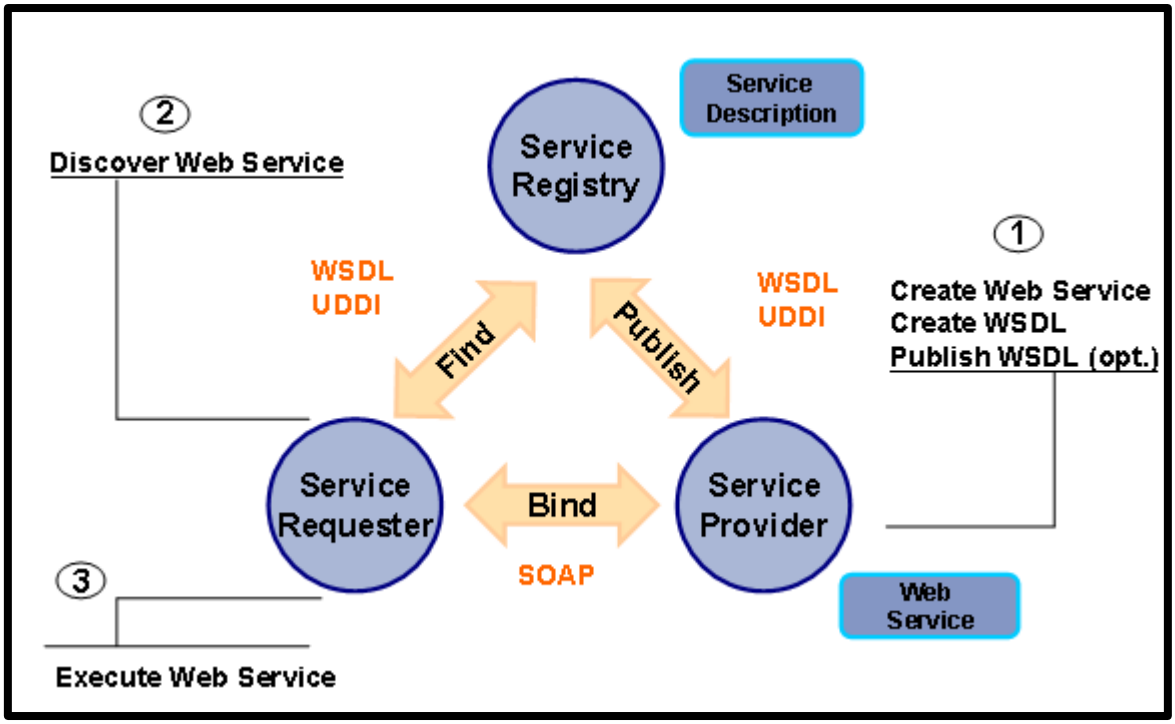


Figure I-3: Web Service operations.[4]

[4] Web Service Architecture from https://thecustomizewindows.com/2013/08/web-services-what-are-web-services/

The operations that can be executed in the web service architecture are the following:

Publish: A provider informs the broker (service registry) about the existence of the web service by using the broker's publish interface to make the service accessible to clients.

Find: This operation allows the search; the service requester retrieves a service description directly with the Find operation or queries the service registry for the type of service required. The Find operation can occur in two different phases of the service requester's lifecycle:

- At design time: to retrieve the service interface description for the program development.
- At runtime which allows to retrieve the link and the location of the service description for the invocation.

Bind: With the information it gained from the broker (service registry) about the web service, the requestor is able to bind, or invoke, the web service.

7. Web services protocols:

WSDL, SOAP, UDDI constitute the set of key technologies of web services, they deliver communication between the different services.

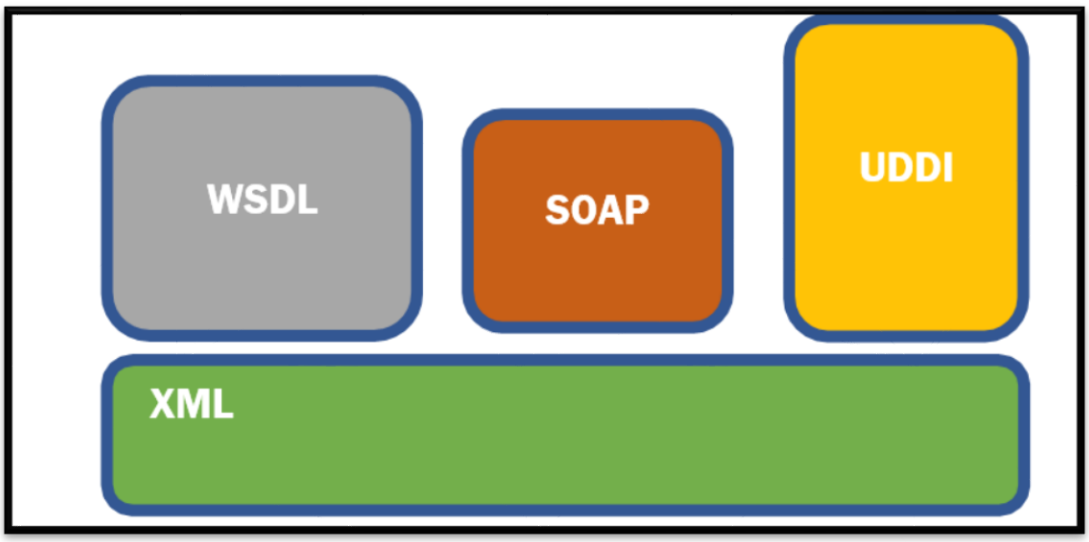


Figure I-4: The Different Protocols of Web Services.

7.1. XML:

(Extensible Markup Language) it is an extensible markup language that was developed by the XML Working Group under the auspices of the W3C. The XML 1.0 specification is recognized as a recommendation by the W3C, making it a recognized language. XML is a standard that serves as a basis for creating specialized markup languages; it is a "meta language". It is general enough that XML-based languages, also called XML dialects, can be used to describe all kinds of data and text. It is therefore partly a data format. Its purpose is, to transfer data and their structures, in a exchange in computer system and before. Allowing the coding of any type of data, from EDI (Electronic Data Interchange) to the most complex documents, its potential is to become the universal and multilingual standard for information exchange[5]

7.2. XML-RPC:

At the most basic level, XML-RPC allows to make function calls over networks. XML-RPC is a combination of RPC architecture with XML and HTTP technologies. XML-RPC allows computers to easily share resources over a network. This means you can give users direct access to the information they need to process to read and reuse systems already built in new contexts, or mix and match programs so everyone can focus on what they do best.(st.loren, 2001)

7.3. SOAP:

SOAP abbreviation of (Simple Object Access Protocol), it is a communication protocol created by the W3C that allows the exchange of information between web services in decentralized and distributed environments, its purpose is to provide scalability, neutrality and platform independence in communication between services. SOAP allows packet transfer in XML format using HTTP, FTP, SMTP and POP protocols.[6]

[5] Wikipedia Definition xml from:
https://fr.wikipedia.org/wiki/Extensible_Markup_Language
[6] Wikipedia Definition of Soap:
https://en.wikipedia.org/wiki/SOAP

7.4. REST:

Is a Software Architectural Style that defines a set of constraints to be used for creating Web services.

Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet.

RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations.[7] Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations

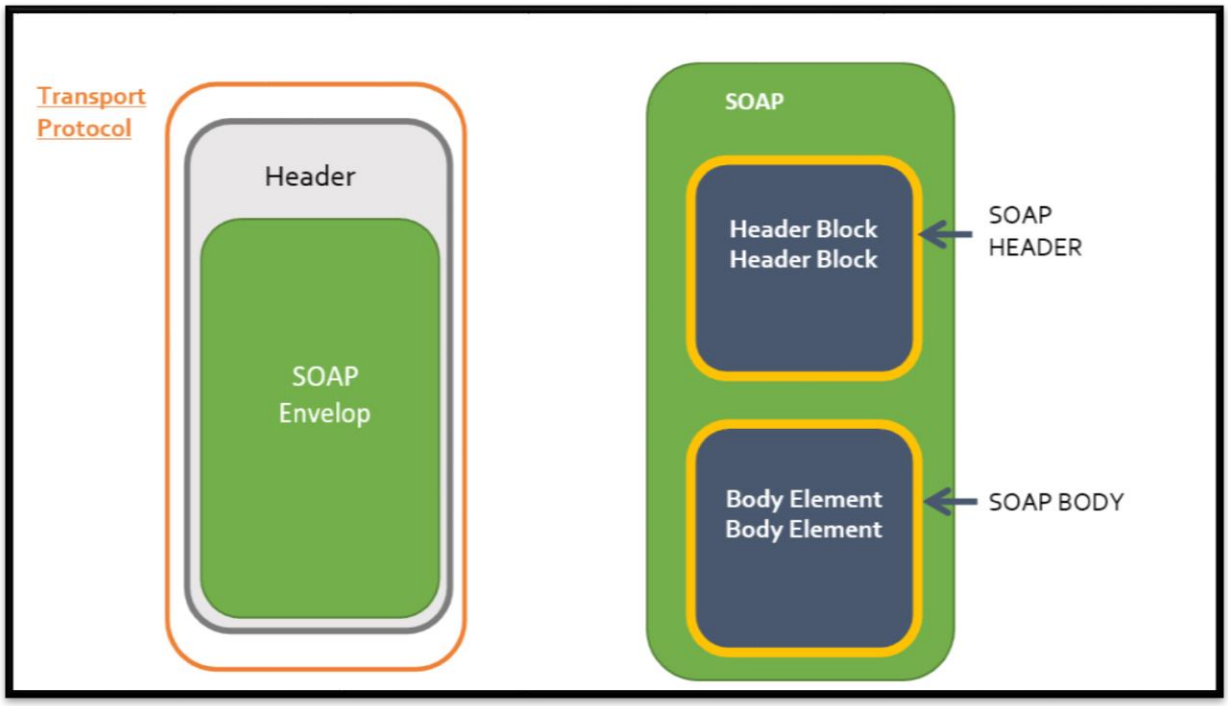


Figure I-5: Diagram showing The SOAP Protocol Message Structure.

7.5. WSDL:

WSDL is the abbreviation of (Web Services Description Language) is defined as a language for describing web services in XML format it is based on SOAP and HTTP mechanisms. Operating on document- and procedure-oriented messages. Operations and messages are described in an

[7] Wikepdia definition of Rest protocol from https://en.wikipedia.org/wiki/Representational_state_transfer

abstract way, then linked to a concrete network point to allow the description of termination points and their related messages regardless of the message formats or the network protocol used.

7.6. UDDI:

UDDI stands of (Universal Description, Discovery, and Integration) constitutes a standard for web services directories where providers publish data about their activities, the different services they offer and the technical details of each service (WSDL). Thus, UDDI has been designed to reduce the gap between client applications and web services. The UDDI specification provides an API allowing client services to be able to consult and extract data from a provider. UDDI is governed by a specific W3C standard.

Each UDDI register stores 3 types of information:

- Supplier data (white pages).
- The public data of each service (green pages).
- Data about the supplier's business activity (yellow pages).[8]

8. Web service protocol stack:

In order to perform the operations on the web services described previously: bind, find and publish, it is necessary to have a web service stack that encompasses them, and the latter is made up of four layers that are :

[8] Wikipedia definition of uddi from http://xml.coverpages.org/uddi.html

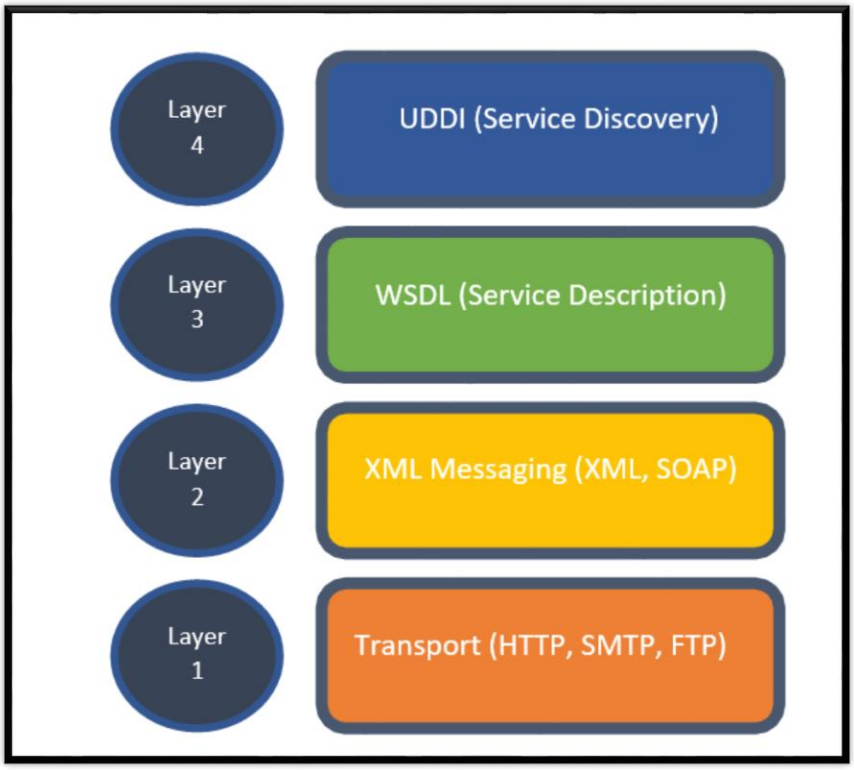


Figure I-6: Web service Protocol Layer's**.**

8.1. Transport Service:

This layer represents the Protocol for transferring messages between applications. The most commonly used protocol is HTTP (Hyper Text Transport Protocol), SMTP (Simple Mail Transfer Protocol) can also be used, as well as FTP (File Transfer Protocol).

8.2. XML Messaging:

This layer allows the encoding of data in XML language so that messages can be understood at each end of a network connection. It includes protocols such as SOAP and XML-RPC.

8.3. Service Description Protocol (WSDL):

The service description layer is a stack of description documents. It is manipulated by the WSDL, which defines the interface and interaction mechanics of the services. An additional description is required to specify the activity, the context, the service qualities and the service-to-service relationships. The WSDL document can be supplemented by other service description

documents to describe these higher-level aspects of the Web service. For example, the business context[9].

8.4. Service Discovery Protocol (UDDI):

Centralizes services in a common registry so that network Web services can publish their location and description which makes it easy to find out which services are available on the network. The (UDDI) was intended for this purpose, but it has not been widely adopted. (Dai & Steele, 2005).

9. Web services contributions:

Expose an existing function on a network: A web service is a portion of code that can be invoked remotely using the HTTP protocol. Thus web services allow the exposure of their functionality on a network and other applications can use its features (Papazoglou & Van Den Heuvel, 2007).

9.1. Interoperability:

Web services inherited this feature from their architectures based on the SOA architecture described above, allowing applications that may be technologically heterogeneous to communicate without worrying about how to do so.

9.2. Standardized protocol:

Web services are based on industry standardized communication protocols that manifest themselves on its different layers of communication namely (transport service, XML messaging, service description and service discovery). This offers companies a wide range of choices and reduced communication costs.

9.3. Low cost communication:

This is possible because web services use the SOAP via the HTTP protocol which allows the implementation of these services (web services) via internet which can be very economical for a company instead of using proprietary solutions such as EDI.

[9] Definition of WSDL from W3C website
https://www.w3.org/TR/wsdl.html

9.4. Application of web services:

The services have multiple applications nowadays in different areas of B2B and B2C. such as e-commerce, e-governance, e-banking, etc.

- **B2c (business to consumer):** Represents all applications dedicated to general public.
- **B2b (business to business):** Represents business-to-business applications.

10. Web Services and Recommender Systems:

During the last few years the use of the web has exploded considerably causing the emergence of many giant web services for AMAZON, NETFLIX, EBAY and many others, a lot of e-commerce sites use recommender systems in order to suggest to users the right content that corresponds precisely to their preferences and interests in order to keep them attached to their services for long time possible.

Recommender systems are crucial for some companies and are the source of a large percentage of revenues as well as an important differentiator factor from their competitors (Rocca, n.d.). In October 2006 Netflix announced a price of one Million dollars for the developer who is able to make their recommendation systems more accurate to 10%. This asserts that the stakes of recommendation systems that have become significant not only from an economic point of view for commercial systems but also more generally from a user satisfaction point of view, as the user is the heart of such systems. Understanding and anticipating the user's needs and expectations are essential to obtain satisfaction. (Lemdani, 2016)

Thus, today's recommendation systems are a must in the daily user's online journey on the web for all organizations.

11. Recommender Systems:

Recommender systems focus on the problem of finding the most appropriate items (products, services) for particular users (businesses, or individuals) (*Recommender System Application Developments | Decision Support Systems*, n.d.)

From this context Recommendation systems can be defined as algorithms aiming at suggesting relevant items to users (items being movies to watch, texts to read, products to buy or others depending on the sector of activity) by predicting users' interest in these items (*Recommender*

System Application Developments | Decision Support Systems, n.d.). based on related information about items, users and interactions between items and users

From another point of view, these systems make it possible to minimize users' choices by presenting them with suggestions of articles (items) that are most likely within their interest provoking enthusiasm and purchase behavior.

From a technical perspective the most general objective of the development of recommendation systems was to minimize data overload by retrieving the most important information and services from a large amount of data via specialized services.

12. Quality of Recommender Systems:

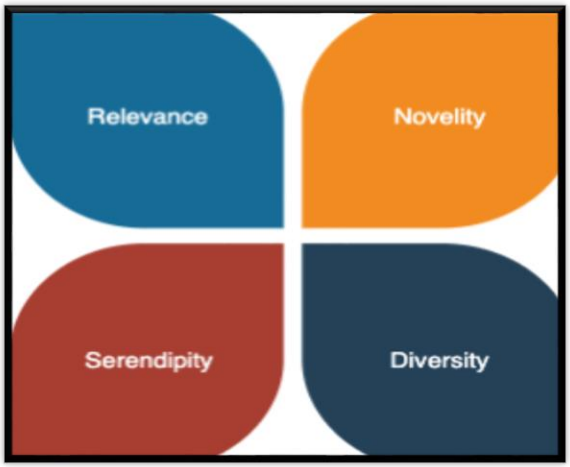


Figure I-7: Goals of Recommendation Engines. [9]

The main objective of recommender systems is to increase the sales figures of their products and services and to learn from their customers in order to improve sales. Charu C Aggarwal, in his book "Recommender Systems", summarized the desired goals of recommendation engines in four points [10]: (Aggarwal, 2016)

[10] Illustration Describing Qualites of Recommender Systems from https://towardsdatascience.com/the-remarkable-world-of-recommender-systems-bff4b9cbe6a7

12.1. Relevance:

Relevance is a key element in the success of a recommendation engine because users are likely to consume items that are likely to be of interest to them, although relevance is the main objective of a recommendation system, however, it represents just a part of its objectives. (Aggarwal, 2016)

12.2. Novelty:

In addition to relevance, novelty is another essential factor. Recommended items will make more sense if the user has never seen or consumed them before. (Aggarwal, 2016)

12.3. Serendipity:

Sometimes recommending unexpected items to customers can increase sales. Except, according to the author, hazard differs from novelty:

"If a new Indian restaurant opens in a neighborhoud, then recommending that restaurant to a user who normally eats Indian food is a novelty but not necessarily a happy coincidence. On the other hand, when the same user is recommended Ethiopian food and does not know that he or she might like it, the recommendation is fortuitous". (Aggarwal, 2016)

12.4. Diversity:

It is equally important to increase the diversity of articles in the recommendations. Simply recommending items that are similar is not very helpful.

13. Purpose of Recommender Systems:

The existence of recommender systems works to achieve the following objectives(Aggarwal, 2016):

- Generate recommendations especially in b to c architectures.
- Reduced effort for users to search for interesting items and bring items that they would not have thought of.
- Increase users' session time.
- Eliminate unnecessary and cumbersome data load.
- Increase sales of products and services.

- A market that is currently boosting billions of dollars.

Note: the goal of recommender systems is to reach the hooked state which means recommendations at the same time close and diversified for more user retention in the application.

14. History of Recommender Systems:

Recommender systems emerged in the 1990s. Their purpose at the time was to address the problems of information overload on the web and e-services.

All start with the very recommender system named Information Lens System (Malone et al., 1987) can be considered the ancestor of these systems.

Several hypotheses of inventions that could improve the results of item filtering have been proposed, including(Goldberg et al., 1992), (Resnick et al., 1994), (Shardanand & Maes, 1995), among others.

(Goldberg et al., 1992) project on the theory of improving the outcome of item and email filtering by proposing a collaborative filtering system called TapesTry. This research consisted in taking user queries on the tapestry system and their ratings on documents. In order to filter the users who present the same queries. The recommendation is based on searching for information based on the content of documents but also on users' annotations and judgments.

Then (Shardanand & Maes, 1995), (Resnick et al., 1994) proposes an approach very similar to the first one where they used the group-lens system which is based like the tapestry system on the ratings of the documents except that these ratings were active or the user had to take the step of rating the document.

On the other hand, the Ringo system of (Shardanand & Maes, 1995) unlike the other two systems focused on music and was based on a collaborative recommendation system and a manual annotation system. The latter focused on the social proximity between individuals to make recommendations, which could be music, albums and artists.

In order to make a recommendation to a user, the recommendation was calculated and inferred from a similarity calculation with other users.

As time went by and with the emergence and expansion, of the internet several recommendation systems were developed and new technologies were discovered in this field as well as the range of applications (e.g. Netflix (movies), Amazon (online shopping)).

Finally, the collection of user and item information for all the systems developed to date is a key step for the proper functioning of these systems.

15. Areas of Recommender Systems:

Recommender systems cover several areas, like:

- Advertisements on websites.
- Search engines (which pages are presented)
- Purchases offered in e-commerce.

Exemple.: movies (Netflix), e-commerce (AMAZON, EBAY), music (iTunes store), tourism (Trivago).

- Landing patches (social networks).

16. User Profile:

The success of a Recommendation engine depends on its ability to find the best top recommendations for people so it makes sense to focus on finding things people will love. (Pandey, 2019)



Figure I-8: User Profiling.[11]

[11] Picture that illustrate User profiling task from https://www.pngwing.com/en/free-png-bfmti

The user is the main element of a recommender system because the recommendation articles provided by its systems are offered to the user, who could potentially consume them, however, they must be of interest to the user, which is the crucial objective of its systems. In order to do this, they need to collect and extract data from the user that will be useful for the recommendation. This information will be defined beforehand on what is called user model. (Felfernig et al., 2019)

The user model can be defined as "a set of knowledge about the user explicitly or implicitly represented, used to improve his/her interaction with the machine" according to.[12]

A user template is created to present a personalized content to each user. This representation is essential for the production of the personalization effect, the absence of this template will make the (web) pages homogeneous for any individual. This model will create a schema of data to be collected in order to build user profiles. Its profiles represent an instantiation of the user model.

A user profile contains information about a structured user according to a user model, this information is first collected directly or indirectly from users, then processed and interpreted before obtaining the user profile which will then be used by recommender systems.

However, a user profile can be composed of different types of information, like:

1. Personal information of the user: date of birth, address, gender, age, nationality ....
2. A user preference model that represents a description of the types of items (product, service) that may be of interest to the user.
3. The probability that the user is interested in this article. For the sake of efficiency, can be used to retrieve the n items most likely to interest the user.
4. A history of the user's interactions with the system. This may include storing the items that a user has viewed with other information about the user's interaction, (for example, whether the user purchased the item or a note he or she gave to the item). A user's interaction history can be used for a number of purposes, including the following:
    - The system can simply display recently visited items to make it easier for the user to return to these items.
    - The system can filter from a recommender system an item that the user has.

[12] From the conference article intitule ‘An Evaluation of Adapted Hypermedia Techniques Using Static User Modelling’ From
http://wwwis.win.tue.nl/ah98/Hothi/Hothi.html

- This representation can also provide data for algorithms for learning user models.

17. Recommendation Techniques:

Some of the best-known recommendation techniques are:

17.1. Content based method:

Content-based approaches are the oldest approaches of recommender systems, they consist in recommending items to a user that correspond to his/her content where the term 'content-based' represents all the items appreciated by the user before, as well as the history of the user. This method requires a description of the items as well as a knowledge of the users' preferences (user profile).

That is why content-based approaches use additional information about users and/or elements. If we consider the example of a film recommender system, the additional information can be, for example, age, gender, job or other personal information of the users as well as the category of the watched films like main actors, duration or other characteristics for the films and objects

The idea of content-based methods is to try to build a model, based on the available "functionalities", that explains the observed user-item interactions. (Rocca, n.d.)

The calculation of the recommendation in this model is done by calculating the cosine similarity between the user profile which is represented by vector A and the characteristics of an item which are represented by vector B

$$\mathbf{cos}(\Theta) = \frac{\boldsymbol{A.B}}{||\boldsymbol{A}||\boldsymbol{B}||} = \sum \frac{\boldsymbol{AiBi}}{\sqrt{\sum \boldsymbol{A_i^2}}\sqrt{\sum \boldsymbol{B_i^2}}} \quad \textbf{(I-1)}$$

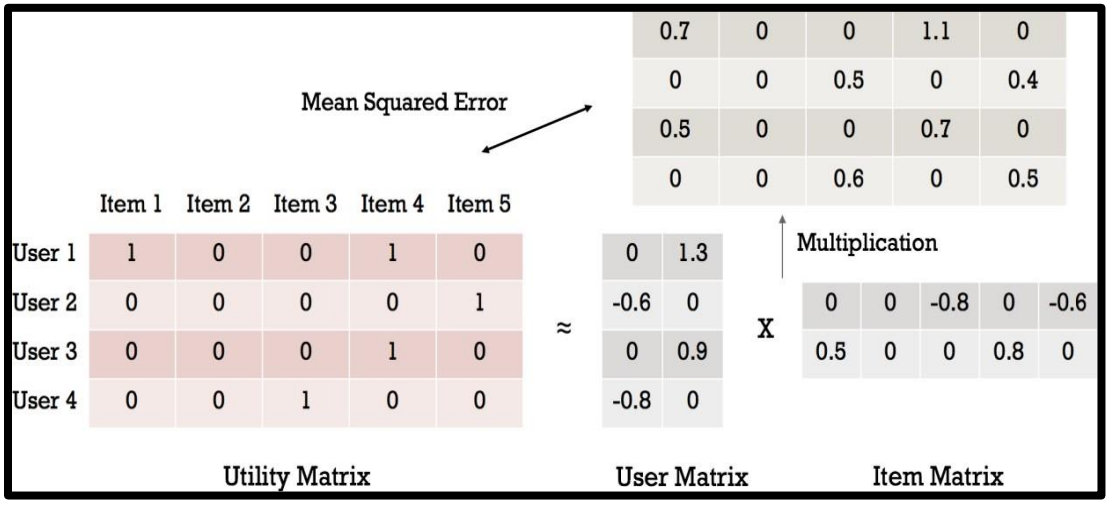


Figure I-9: Content Based Method. [11]

The cosine similarity results are arranged in a descending order so that the item at the top and the closest is the one intended by the user. [13]

17.2. Advantages of this Method:

- The user will have articles that really interest him.
- The user is likely to be satisfied with the quality of the recommendation.

17.2.1. Disadvantages of this Method:

- The user will not get new recommendations (which have no relation to his/her content).
- Business cannot expand if the user does not try other products.
- The cosine matrix has to be computed each time the data changes in the user vector or item, which is a huge traffic load.

[13] content based recommendation (illustration and explanation) from https://towardsdatascience.com/essentials-of-recommendation-engines-content-based-and-collaborative-filtering-31521c964922

17.3. Collaborative Method:

Collaborative methods for recommender systems are based solely on past recorded interactions between users and items to produce new recommendations.

These interactions are stored in the "user-item interaction matrix". (ROCCA, 2019) There are mainly two methods in order to calculate the recommendation with collaborative methods these methods are:

17.3.1. User-user Collaborative Filtering:

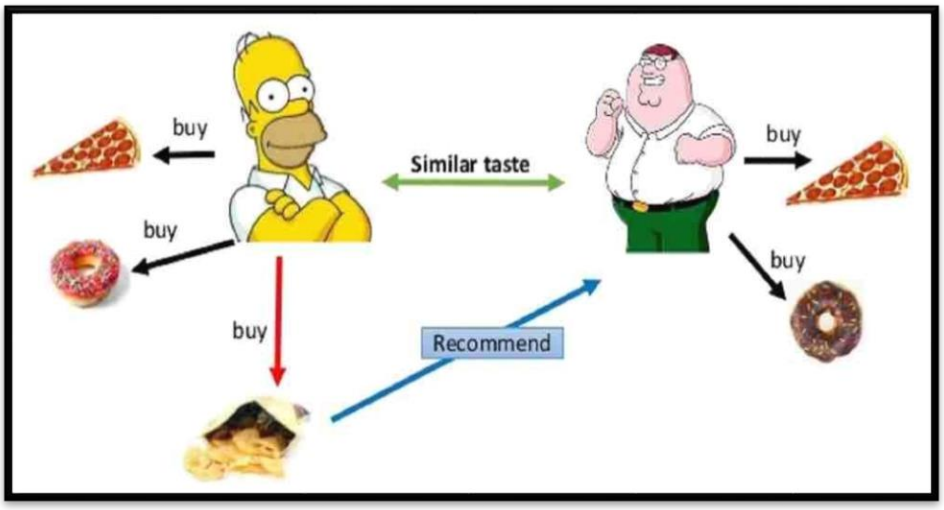


Figure I-10: User-User Model.[14]

This method includes a vector of all items purchased or used by a user and a score assigned to each item that represents the user's satisfaction with that item. A similarity is computed between the different pairs of users with the cosine formula on which we obtain a matrix n*n. For each user we choose the item recommended to the user having the most important similarity. (Lemdani, 2016)

[14] Illustration for User-User collaborative filtering from https://towardsdatascience.com/tagged/mean-squared-error

17.3.2. Item-Item Collaborative Filtering:

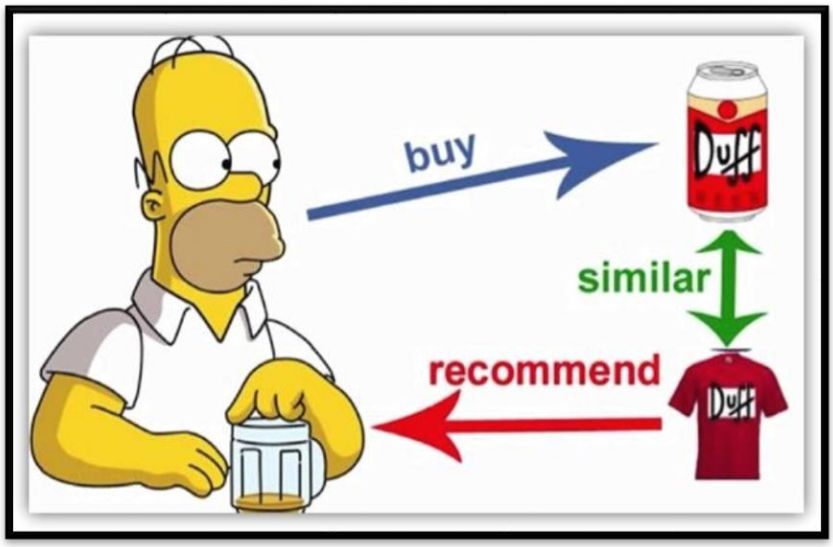


Figure I-11: Item-To-Item Model.[15]

It is based on the same principle of the first method except that this time similarities are calculated between items by taking into account the items selected by the user, so a matrix of size m*m being m the number of items, the items with the greatest similarities will be those selected at the recommendation .

17.4. Hybrid method:

These approaches are a combination of collaborative approaches and content-based methods in order to benefit from the advantages of both methods and at the same time overcome the disadvantages of both (*Hybrid Web Recommender Systems | SpringerLink*, n.d.). For example, a cold start at the item level of collaborative approaches can be beneficial. the advantages of content-based approaches. (Lemdani, 2016)

[15] Image that illustrate the Item-to Item recommendation from https://towardsdatascience.com/build-your-own-recommender-system-within-5-minutes-30dd40388fbf

Multi-Label Classification



1. Introduction:

As years are passing, technology is getting involved in every tiny aspect in our lives, causing the available data to increase exponentially, hence there is a pressing need to organize it, one of the modern solutions is classification, this latter is the process of predicting the class of any given data. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function (F) from input variables (X) to discrete output variables (Y). (Asiri, 2018)

Classification belongs to the category of supervised learning where a function maps an input to an output based on example input-output pairs. Classification can be used in many fields such as credit approval, medical diagnosis, target marketing etc. (Asiri, 2018)

There are two types of learners in classification, which are lazy learners and eager learners.

1.1. Lazy learners:

Lazy learners simply store the training data and wait until a testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting.

Ex. k-nearest neighbor, Case-based reasoning (Asiri, 2018)

1.2. Eager learners:

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time for train and less time to predict.

Ex. Decision Tree, Naive Bayes, Artificial Neural Networks (Asiri, 2018)

2. Types of Classification:

Classification falls into two major categories: Single-label Classification and Multi-label Classification.

2.1. Single-Label Classification:

Single-label classification is concerned with learning from a set of examples that are associated with a single label l from a set of disjoint labels L, |L| > 1.

Single label classifications have two types of classification: Binary classification and Multiclass classification

- If |L|=2, then the learning task is called binary classification.
- If |L|>2, then it is called multi-class classification.[16]

2.1.1. Binary Classification:

This is an easiest classification problem we can face. The instances in a binary dataset have only one output attribute, and it can take only two different values. These are usually known as positive and negative, but can also be interpreted as (true /false) ,(1 and 0) any other combination of two values. A classic example of this task is spam filtering (Figure II-1), in which the classifier learns from the messages' content which ones can be considered as spam.(Herrera et al., 2016)

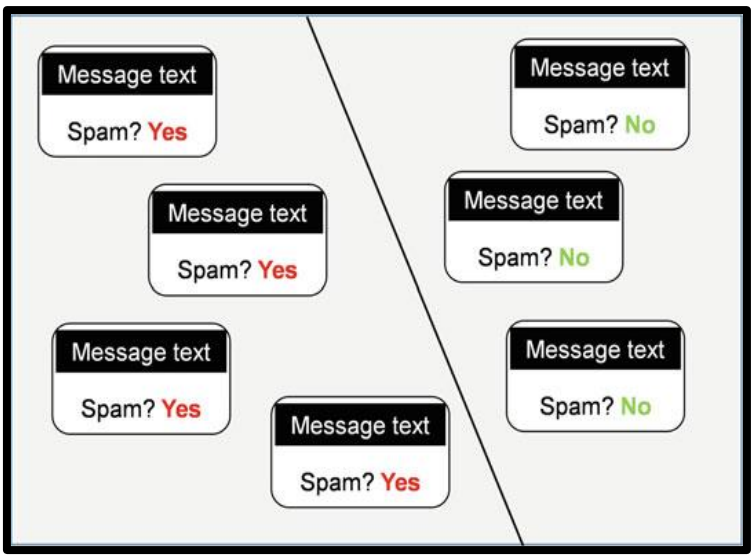


Figure II-1: Binary Classification Problem. (Herrera et al., 2016).

[16] Single label classification form https://www.igi-global.com/article/multi-label-classification/

Figure II-1 Spam filtering is a typical binary classification problem. The classifier learns from the messages text and predicts if it is spam or not. The problem could be not linear separable, as in this simple example [Herrera]

2.1.2. Multiclass Classification:

A multiclass dataset also has only one output attribute, like in the binary datasets, but it can hold any from a certain set of predefined values. The meaning of each value, and the value itself, would be specific to each application. The set of classes will be finite and discrete, on the contrary the task would not be classification but regression. Class values could have an order relationship or not. One of the best-known multiclass classification examples is iris species identification (Figure II-2)**.** From four characteristics of the flower, the petal and sepal lengths and widths, the classifier learns how to classify new instances into the corresponding family. Many multiclass classification algorithms rely on binarization , a method that iteratively trains a binary classifier for each class against the others, following a One-vs-All (OVA) approach, or for each pair of classes, using a One-vs-One (OVO) way.

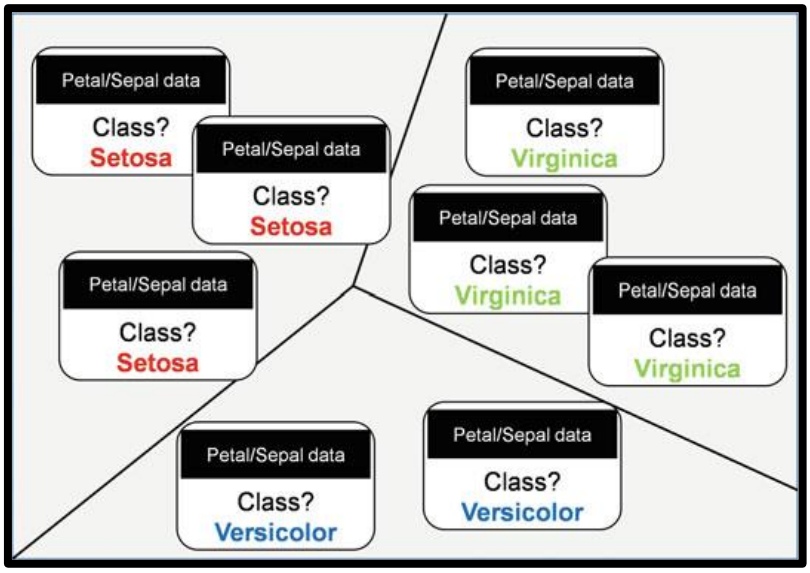


Figure II-2: Multi-class Classification Problem. (Herrera et al., 2016)

From (Figure II-2) Iris species categorization is a classical multiclass classification problem. The classifier learns from the length and width of petal and sepal, predicting which of the three species the flower belongs to. (Herrera et al., 2016)

2.1.3. Some of the algorithms used to perform a single-label classification:

There are many classification algorithms available now, but it is not possible to conclude which one is the best. It depends on the application and nature of available data set. For example, if the classes are linearly separable, the linear classifiers like Logistic regression, Fisher's linear discriminant can outperform sophisticated models and vice versa.

2.1.3.1. Decision Tree:

Decision Trees are a type of Supervised Learning Algorithms (meaning that they were given labeled data to train on). The training data is continuously split into two more sub-nodes according to a certain parameter. The tree can be explained by two things, leaves and decision nodes. The decision nodes are where the data is split. The leaves are the decisions or the final outcomes. You can think of a decision tree in programming terms as a tree that has a bunch of (If Statements) for each node until you get to a leaf node (The final Outcome). [17]

. Types of Decision Tree:

There are two types of Decision Tree:

- **Categorical Variable Decision Tree:** Decision Tree which has categorical target variable is known as a categorical variable decision tree.
- **Continuous Variable Decision Tree:** Decision Tree which has continuous target variable is known as a Continuous Variable Decision Tree.(Rishabah, 2017)

[17] Decision tree explanation from Medium, link https://medium.com/

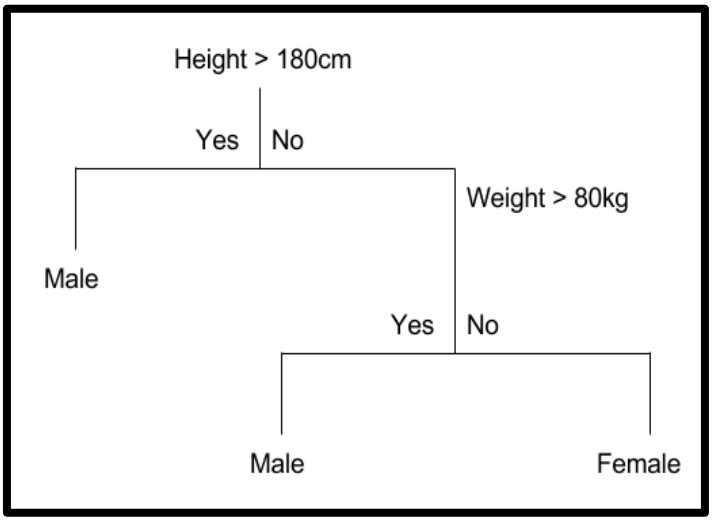


Figure II-3: An example of a classification tree to predict the gender of a human based on the height end the weight. (Chakur, 2019)

. Algorithm used in Decision Trees:

There are many algorithms used in Decision Trees such as (ID3, GINI Index, Chi-Square, Reduction in Variance).

In our project we discussed ID3 only, as it is the most convenient and suitable for our work, as it provided us the best results.

. ID3:

The core algorithm for building decision trees is called (ID3). Developed by J. R. Quinlan, this algorithm employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree.

. Entropy:

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided then it has entropy of one. (Rishabah, 2017)

$$Entropy = -\sum \mathrm{p(x)}\ )\ \log \mathrm{p(x)} \quad \textbf{(II-1)}$$

- P(x) is a fraction of examples in a given class

. **Information gain:**

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches). (Rishabah, 2017)

$$\text{gain}(S, A) = \text{Entropy}(S) - \sum \frac{|s_v|}{s} \quad \textbf{(II-2)}$$

where A is the attribute being considered, S is the set of training examples being considered, and Sv is the subset of S with value v for attribute A

. **The Random Forest Classifier:**

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see Figure II-4).

The fundamental concept behind random forest is simple but a powerful one, the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

- A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.(Gandhi, 2019)

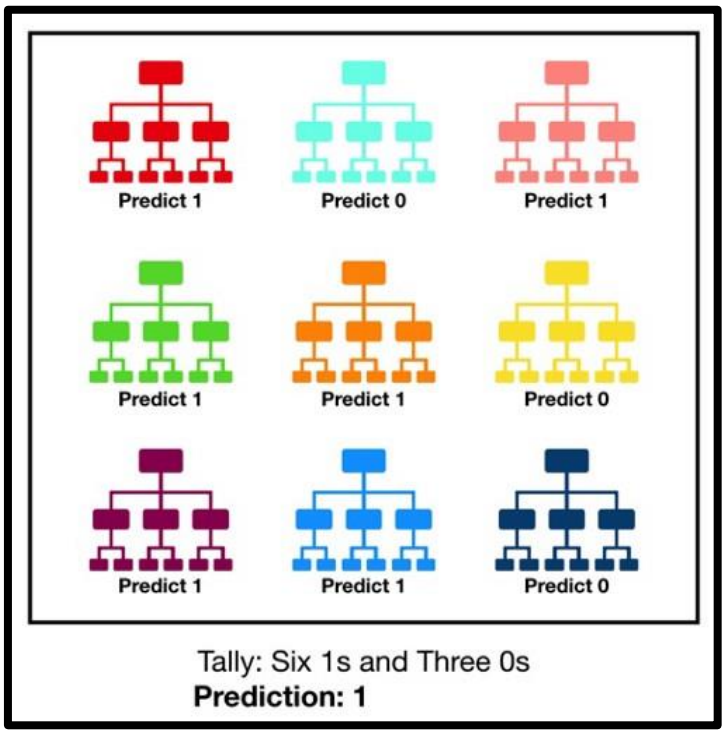


Figure II-3: A figure shows how Random Forest trees vote for a prediction.(Gandhi, 2019).

2.1.3.2 Naive Bayes:

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)} \quad \textbf{(II-3)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence (the feature) and A is the hypothesis (the label). The assumption made here is that the predictors/features are independent. Means that the presence of one particular feature does not affect the other. Hence it is called naive.

. Types of Naive Bayes Classifier:

. Multinomial Naive Bayes:

This is mostly used for document classification problem, whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

. **Bernoulli Naive Bayes:**

This is similar to the Multinomial Naive Bayes, but the predictors are Boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

. **Gaussian Naive Bayes:**

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

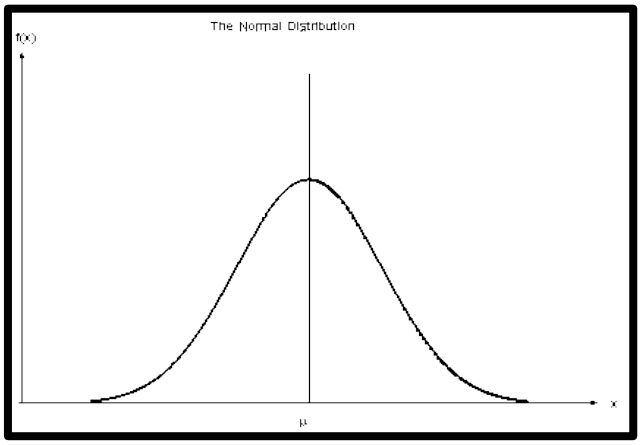


Figure II-4: Gaussian Distribution (Normal Distribution).(Gandhi, 2019)

Since the way, the values are present in the dataset changes, the formula for conditional probability changes to:

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_i)^2}{2\sigma_y^2}\right) \quad \textbf{(II-4)}$$

Naive Bayes algorithms are mostly used in sentiment analysis, spam filtering, recommendation systems etc. They are fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent. In most of the real life cases, the predictors are dependent, this hinders the performance of the classifier.(Gandhi, 2019)

2.1.3.3. Artificial Neural Networks:

Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the biological nervous system such as brain process information. It is composed of large number of highly interconnected processing elements (Neurons) working in unison to solve a specific problem.(Chauhan, 2019)

Neural Networks are a Network of multiple layers of neurons consisting of nodes which are used for classification and prediction of data provided, some data as input to the Network. There is an input layer, one or many hidden layers and an output layer. All the layers have nodes and each node has a weight which is considered while processing information from one layer to the next layer. (Sharma et al., 2020)

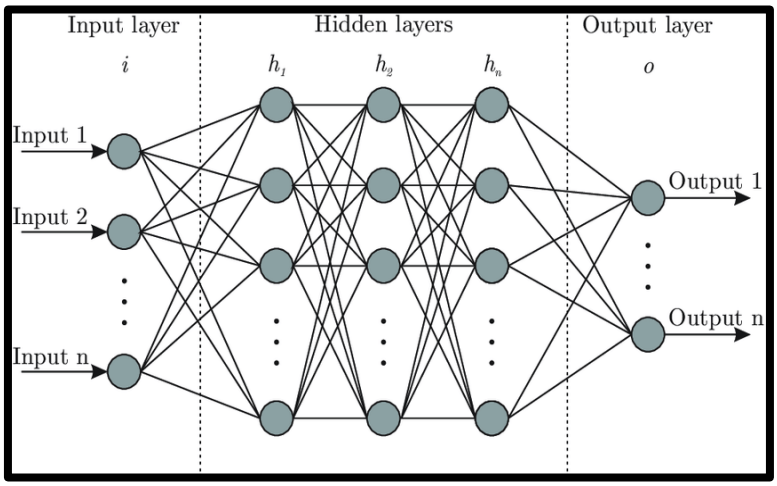


Figure II-5: Neural network with hidden layers.(Erker, n.d.)

. Neurons:

Biological Neurons (also called nerve cells) or simply neurons are the fundamental units of the brain and nervous system, the cells responsible for receiving sensory input from the external world via dendrites, process it and gives the output through Axons.

The ANN Neuron is inspired by the biological neuron and it is called a perceptron. A single layer neural network is called a Perceptron. It gives a single output.

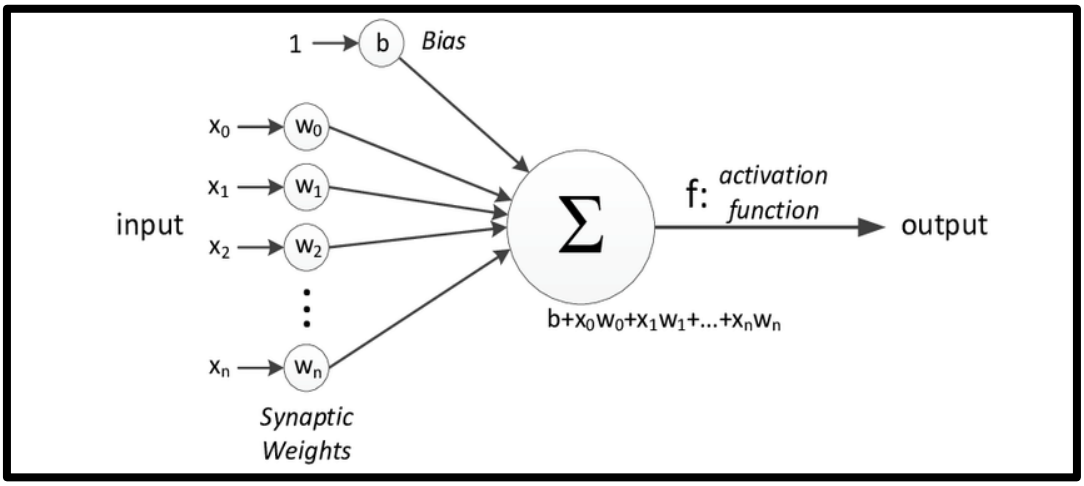


Figure II-6: Perceptron. (Chauhan, 2019)

In (Figure II-7), for one single observation, `x0, x1, x2, x3...x(n)` represents various inputs (independent variables) to the network. Each of these inputs is multiplied by a connection weight or synapse. The weights are represented as w0, w1, w2, w3…. w(n). Weight shows the strength of a particular node.(Chauhan, 2019)

- b is a bias value. A bias value allows you to shift the activation function up or down.

In the simplest case, these products are summed, fed to a transfer function (Activation function) to generate a result, and this result is sent as output.

Mathematically:

$$\mathbf{x1}.\mathbf{w1} + \mathbf{x2}.\mathbf{w2} + \mathbf{x3}.\mathbf{w3} \ldots\ldots \mathbf{xn}.\mathbf{wn} = \sum \mathbf{xi}.\mathbf{wi} \quad \mathbf{(II\text{-}5)}$$

Now activation function is applied $(\sum xi.wi)$. (Chauhan, 2019)

. Activation Function:

Net inputs are the most important units in the structure of a neural network which are processed and changed into an output result known as unit's activation by applying function called the activation function or threshold function or transfer function which is a scalar to scalar transformation. To enable a limited amplitude of the output of a neuron and enabling it in a limited range is known as squashing functions. A squashing function squashes the amplitude of output signal into a finite value. (Sharma et al., 2020)

. Types of Activation functions:

. Thresh Hold function:

A Binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.

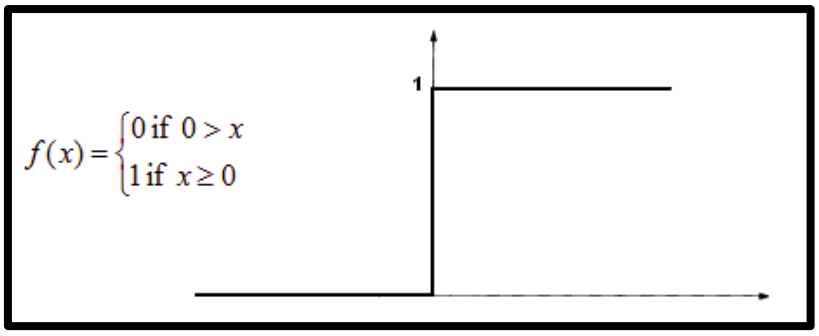


Figure II-7: Thresh Hold Function.(Chauhan, 2019)

. Sigmoid or Logistic Activation Function:

The Sigmoid Function curve looks like a S-shape (Figure II-8)

The main reason why we use sigmoid function is because it exists between 0 and 1. Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points. The function is monotonic, but function's derivative is not. (Sharma et al., 2020)

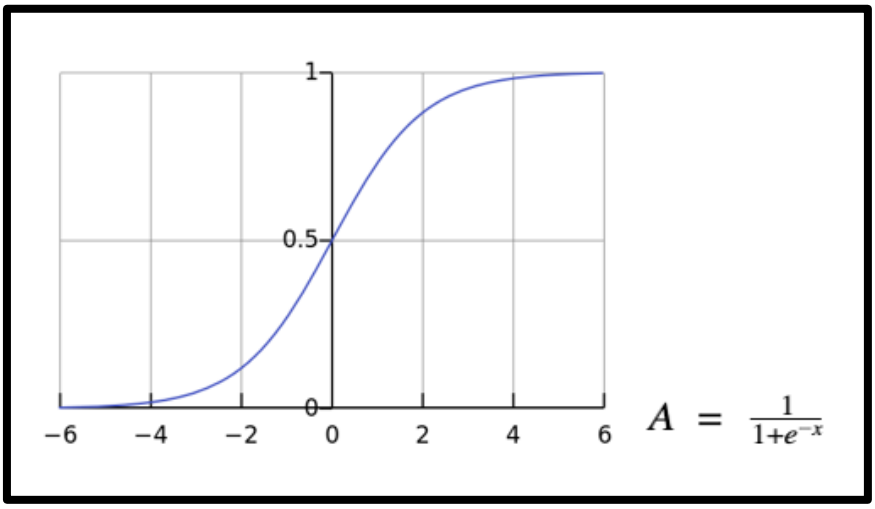


Figure II-9: Sigmoid Function. (sharma, 2017)

. Rectified Linear Unit (ReLU) Activation Function:

The (ReLU) is the most used activation function in the world right now. Since it is used in almost all the convolutional neural networks or deep learning.

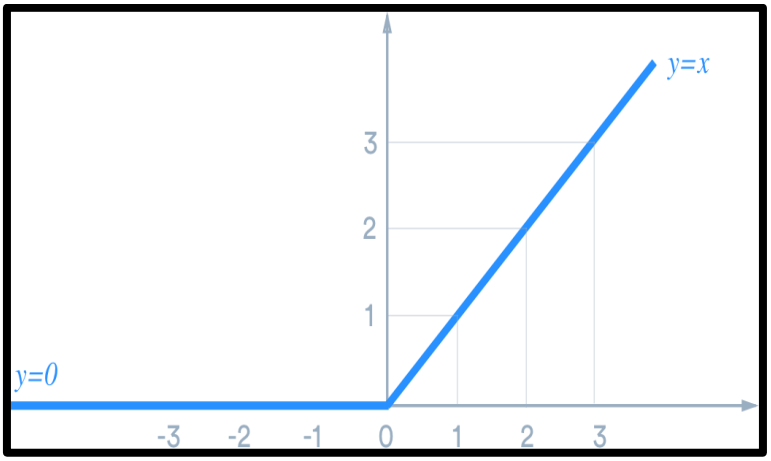


Figure II-9: ReLU Function.(sivarajkumar, 2019)

- The ReLU is half rectified (from bottom). f(y) is zero when y is less than zero and f(y) is equal to y when y is above or equal to zero. Range: [0, +∞ [
- The function and its derivative both are monotonic… (sharma, 2017)
- There still exist other activation functions like Tanh, Exponential Linear Unit, Swish, SoftMax. The idea is they all work in different circumstances depending on the type of work.

. The way Neural Networks learn:

Looking at an analogy may be useful in understanding the mechanisms of a neural network. Learning in a neural network is closely related to how we learn in our regular lives and activities.

We perform an action and are either accepted or corrected by a trainer or coach to understand how to get better at a certain task. Similarly, neural networks require a trainer in order to describe what should have been produced as a response to the input. Based on the difference between the actual value and the predicted value, an error value also called Cost Function is computed and sent back through the system.

. Cost Function: One half of the squared difference between actual and output value:

For each layer of the network, the cost function is analysed and used to adjust the threshold and weights for the next input. Our aim is to minimize the cost function. The lower the cost function, the closer the actual value to the predicted value. In this way, the error keeps becoming marginally lesser in each run as the network learns how to analyse values.

We feed the resulting data back through the entire neural network. The weighted synapses connecting input variables to the neuron are the only thing we have control over.

As long as there exists a disparity between the actual value and the predicted value, we need to adjust those weights. Once we adjust them a little and run the neural network again, A new cost function will be produced, hopefully, smaller than the last.

We need to repeat this process until we scrub the cost function down to as small as possible.

The procedure described above is known as Back-propagation and is applied continuously through a network until the error value is kept at a minimum.(Chauhan, 2019)

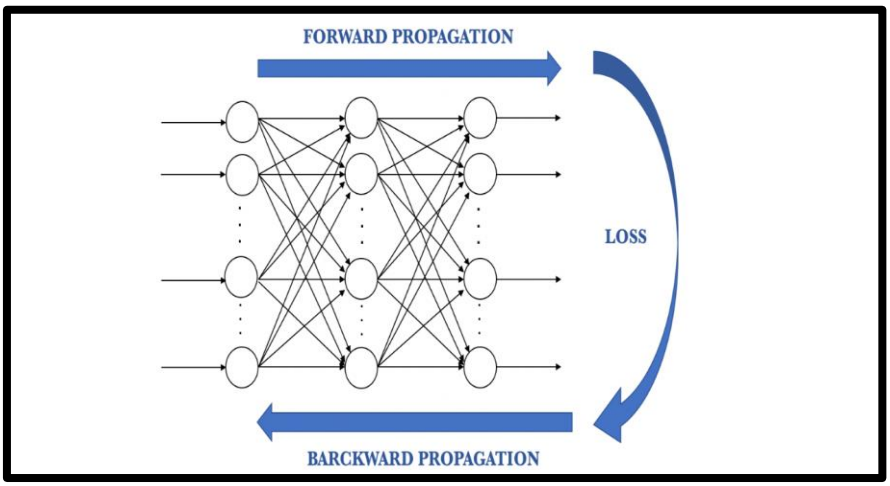


Figure II-10: Forward and Back propagation.(Chauhan, 2019)

. Types of Neural Networks:

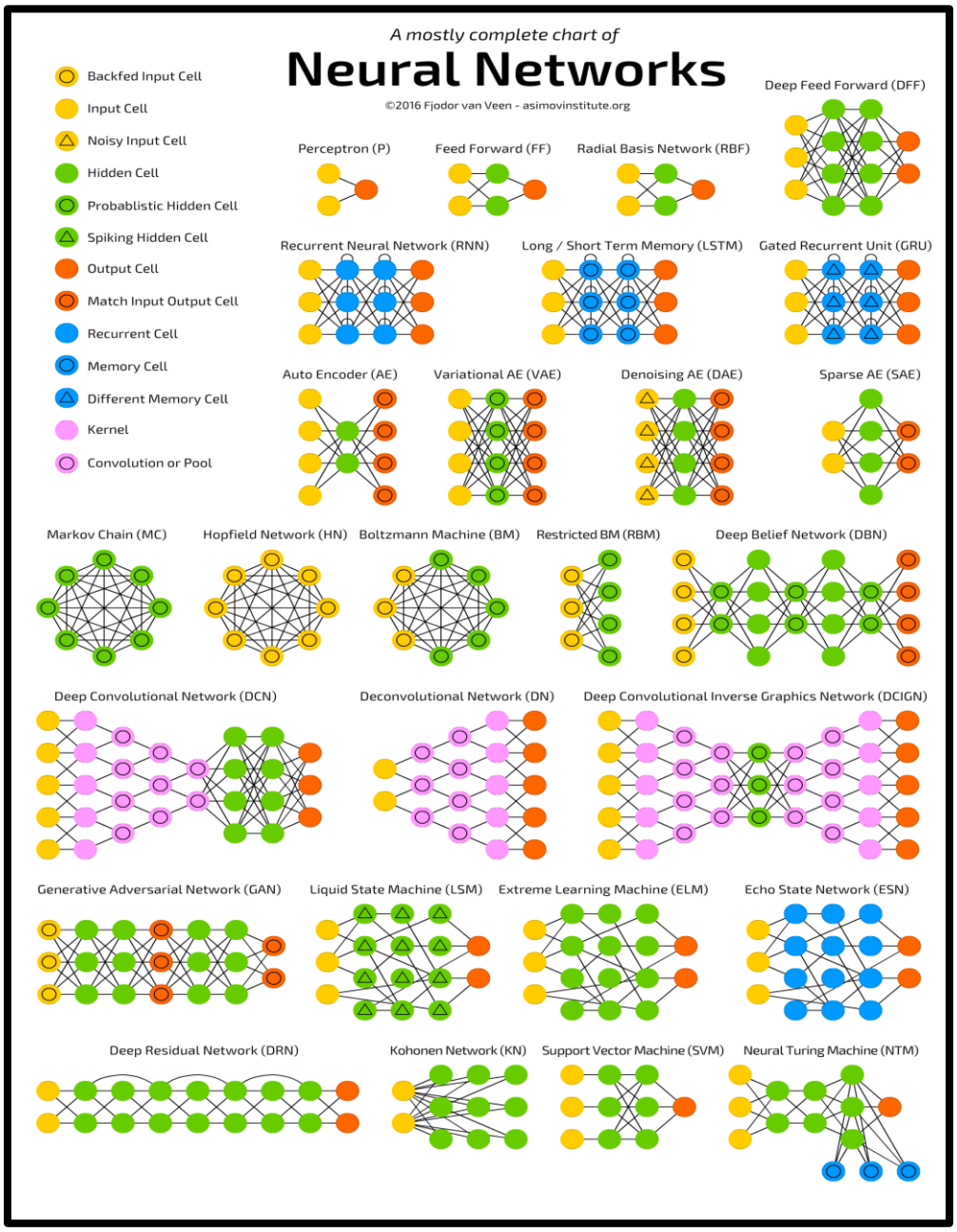


Figure II-11: Architecture of different types of Neural.[18]

[18] Illustration of different architectures of Neural Network from (asimovinstitute.org) https://www.asimovinstitute.org/author/fjodorvanveen/

. **Perceptron:**

. It is the simplest and oldest model of Neuron, It takes some inputs, sums them up, applies activation function and passes them to output layer.

. **Feed forward Neural Network (FFNN):**

FFNN follows the following rules:

- all nodes are fully connected.
- activation flows from input layer to output, without back loops.
- there is one layer between input and output (one hidden layer).
- In most cases this type of networks is trained using Backpropagation method. FFNN is usually used for Regression and classification.

. **Neural Networks (DFF):**

DFF opened pandora box of deep learning in early 90s. These are just FF NNs, but with more than one hidden layer.

. **Recurrent Neural Networks (RNN):**

Introduce different type of (cells — Recurrent) cells. The first network of this type was so called Jordan network, when each of hidden cell received its own output with fixed (delay — one) or more iterations. Apart from that, it was like common FFNN.

RNN is usually used for Time Series analysis.

. **Deep Convolutional Networks (DCN):**

Deep Convolutional Networks (DCN) or Convolutional Neural Networks (CNN) nowadays are stars of artificial neural networks. They feature convolution cells (or pooling layers) and kernels, each serving a different purpose. Convolution kernels actually process input data, and pooling layers simplify it (mostly using non-linear functions, like max), reducing unnecessary features. CNN is used for Computer Vision like image recognition. (Tech, 2017)

There still many other types of Neural Networks shown in (Figure II-13) choosing the right one highly depends on the type of problem we are trying to solve.

All the previously mentioned algorithms (Decision Tree, Naïve Bayes and Artificial Neural Networks) are eager learners since they train a model in advance to generalize the training data and use it for prediction later.

2.1.3.4. k-Nearest Neighbor (KNN):

k-Nearest Neighbor (KNN) is a lazy learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyses the closest k number of instances saved (nearest neighbors) and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors. (Asiri,)

To find the closest neighbors to an instance we calculate the distance between it and all other instances in the dataset. There are multiple methods to calculate the distance, like Euclidian distance and Manhattan distance.

The Euclidean distance can be calculated with the following equation:

$$(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)2} \quad \textbf{(II-6)}$$

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors.

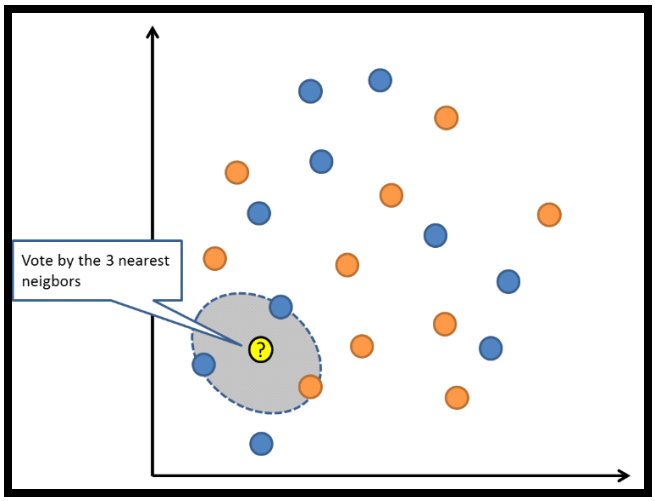


Figure II-13: Image showing k-nearest neighbors (KNN) algorithm, with a number 3 as a value of K**.** (Asiri, 2018)

2.2. Multi-Label Classification:

Unlike the two previous classification models, in multilabel classification [label 1, label 2, …label n] each one of the data instances has associated a vector of outputs, instead of only one value. The

length of this vector is fixed according to the number of different labels in the dataset. Each element of the vector will be a binary value, indicating if the corresponding label is relevant to the sample or not. Several labels can be active at once. Each distinct combination of labels is known as label set. Figure II-II-12 depicts one of the classical multilabel applications, image labeling. The dataset has four labels in total and each picture can be assigned any of them, or even all at once if there were a picture in which the four concepts appear. (Herrera et al., 2016)

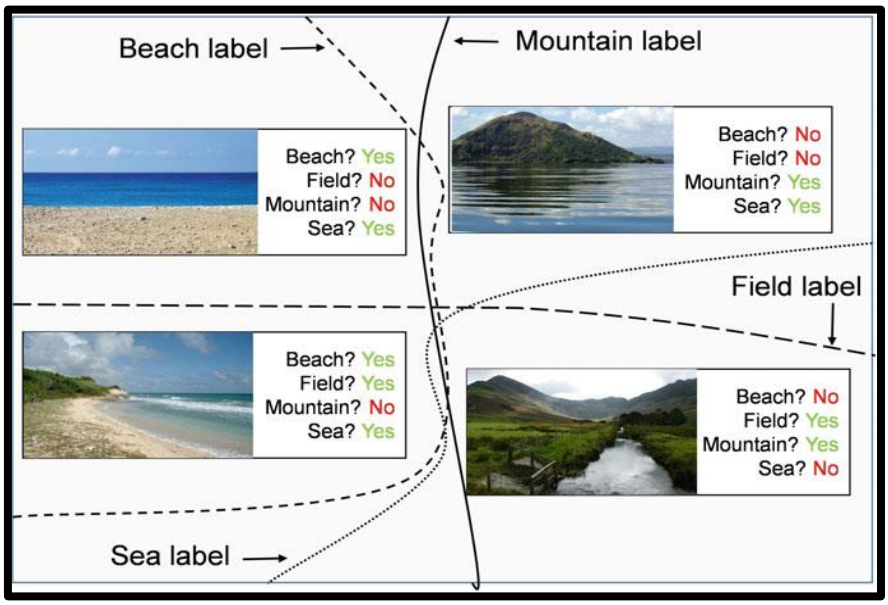


Figure II-14 : Multi-label Classification Problem. (Herrera et al., 2016)

2.2.1. Case Studies:

2.2.1.1. Audio Categorization:

Songs can be classified into different genres provoking different emotions and moods like "relaxing-calm", or "sad-lonely" etc. Music may evoke more than one different emotion at the same time, that's why Single-label classification and regression cannot model this multiplicity, however, multi-label classification can solve this problem where a piece of music may simultaneously belong to more than one class.

2.2.1.2. Image Categorization:

Image semantic understanding is now typically formulated as a multi-label problem. In this setting, each image may be simultaneously categorized into more than one set of predefined

categories. An important difference between multi-class classification and multi-label classification is that classes in multi-class classification are assumed to be mutually exclusive whereas in multi-label classification are normally interdependent from one another. (Cabral et al., 2011)

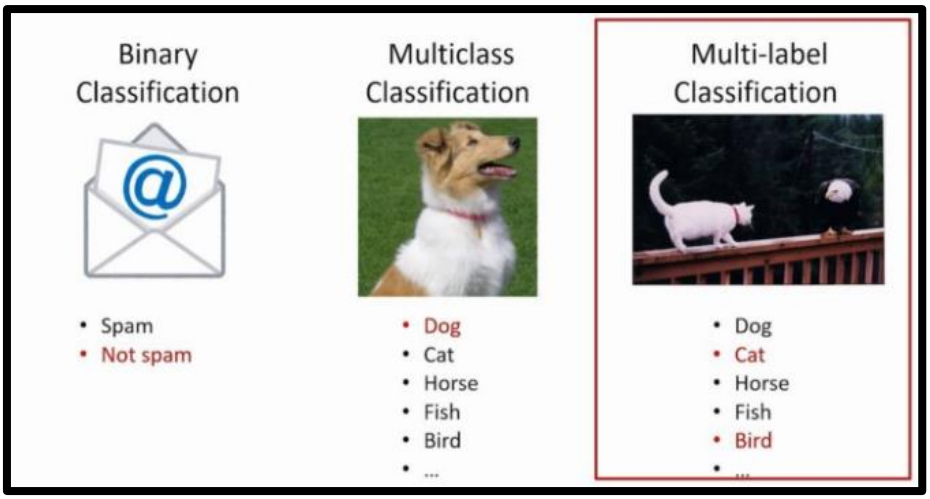


Figure II-13: The difference between Binary classification, Multiclass Classification. and Multi-label classification. Yahoo Japan.(*Exact Passive-Aggressive Algorithm for Multiclass Classification Using Support Class | Request PDF*, n.d.)

2.2.1.3. Bioinformatics:

Multi-label type of bioinformatics data widely exists in clinical text data, gene data, protein data, and so on. For example, a patient suffering from cough and fever should be associated with both two disease "labels" in the clinical records. Formally, let X=R*d* denote the *d* dimensional feature space and Y={0,1}*q* denote the *q*− dimensional label space, where each example in multi-label bioinformatics data can be denoted as (xi,Yi)(xi∈X,Yi⊆Y) . Due to the rapidly expanding quantity of multi-label bioinformatics data resources, techniques based on multi-label learning demonstrate the superiority in mining useful information for this type of data. (Guo et al., 2019)

2.2.1.4. Text Categorization:

Text classification has traditionally been one of the most popular problems in information retrieval, natural language processing and machine learning. In the simplest case, the task of text classification is as follows: A set of training documents T={X1,X2,...Xm}, each labelled with a

class value from a set of k distinct labels, from the set{1,2,..k}, is used to learn a classification model, that captures the relationship between features in the documents and their labels. Subsequently, for a test document with unknown label, the model is used to predict a label. The problem of text classification finds wide appeal in various domains for tasks such as news selection and grouping, Document organization in digital libraries, websites, social feeds, etc.,(Herrera et al., 2016) Email classification including spam filtering. A variation of this problem in which each document can belong to any number of classes (labels) is referred to as a multi-label text classification problem. (Nayak et al., 2015) where each text can belong to multiple labels, like an email who can fall into different categories at the same time (spam, advertisement, fraud…etc.), or some news that can be identified as (novelty, technology, entertainment...etc.)

2.2.2. Techniques for Solving a Multi-Label Classification problems:

Basically, there are three methods to solve a multi-label classification problem, namely:

- **Problem Transformation methods:** divide the multi-label problem into one or more conventional single-label problems.
- **Problem Adaptation methods:** generalize single-label algorithms to cope with multi-labeled data directly.
- **Ensemble methods:** incorporate the merits of two previous approaches

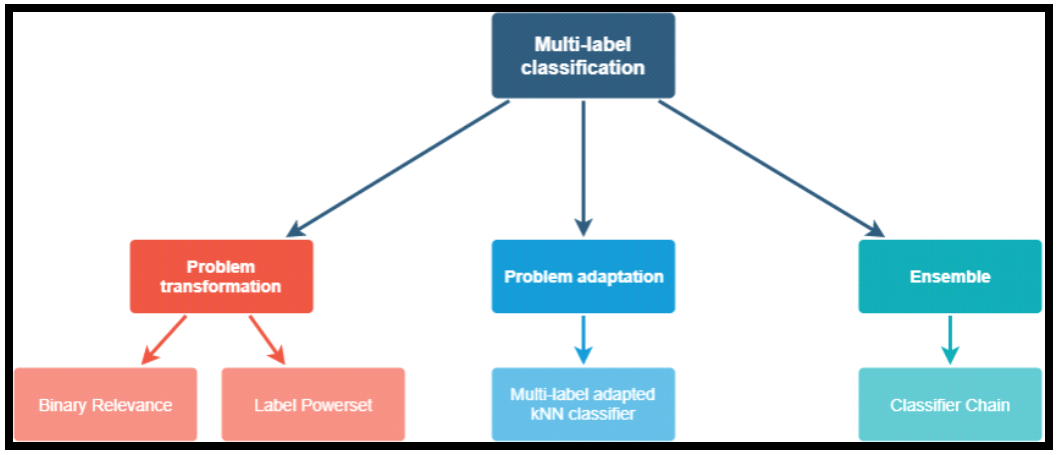


Figure II-16: Image showing the different methods of multi-label classification.(Deustschman, 2019)

2.2.2.1. Problem Transformation methods:

Problem transformation method which is the transformation-based multi-label classification algorithms aim to convert the original dataset into one or more simpler datasets that can be delivered to traditional classification algorithms. In a certain way, these methods act as a pre-processing phase, producing new datasets from the original ones, combined with a post-prediction step in charge of uniting the individual votes from each classifier into a joined prediction [Herrera]. in other words, the problem transformation maps the multi-label learning task into one or more single-label learning tasks.

After dividing the Multi-label learning task into one or multiple single-label learning tasks, some classifiers should carry out the training for those single-label learning tasks. The classifiers can be the algorithms which are used for the single-label classification like Naive Bayes, Decision Trees, Random Forest classifier, Artificial Neural Network…etc.

This method can be carried out in two different ways:

- Binary Relevance.
- Label Powerset.

. **Binary Relevance (BR):**

The baseline approach, called the binary relevance method extents to independently training one binary classifier for each label. In this method, a multi-label problem is converted into |L|

number of binary SL classification problems where L is a set of labels. Each of the binary classifiers votes separately to get the final result. Since the labels are predicted independently of each other, label dependencies are not taken into account. (Pushpa & Karpagavalli, 2017)

BR has linear computational complexity with respect to the number of labels and can easily be parallelized.

For example, let us consider a case as shown in (Figure II-14). We have our dataset where X is the independent feature and Y's are the target variables. MLD stands for Multi-label dataset, BID stands for Binary Dataset

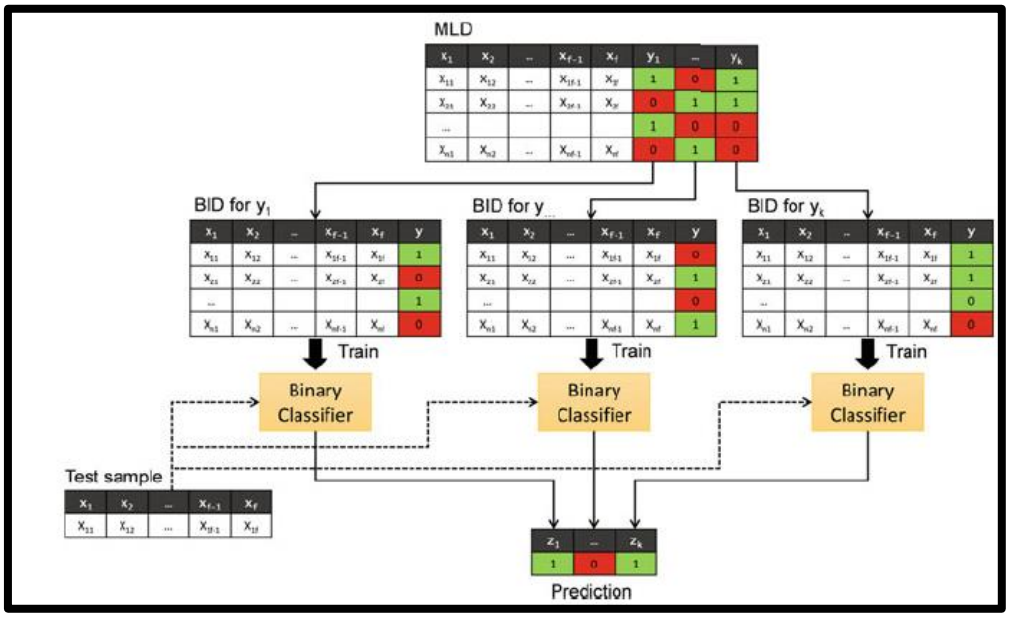


Figure II-15: Binary Relevance (BR) Transformation Diagram.(Herrera et al., 2016)

The original MLD (top) generates as many BIDs as labels there are, using each one to train a binary classifier. When a new test sample arrives, it is given to each individual classifier, joining their predictions to obtain the final label set. (Herrera et al., 2016)

The binary classifications are done using the single-label classification algorithms, like the ones explained previously.

Advantages:

The complexity is linear which means which is relatively small, allowing a small computational time in comparison to other methods, and less resources consumption when using computers to solve it, also it's so simple.

Disadvantages:

Ignoring correlation between labels by treating each label independently.

. Label Powerset (LP):

The most natural approach is the label powerset method which generates a new class for every combination of labels and then solves the problem using multiclass classification approaches. (Pushpa & Karpagavalli, 2017)

So, label powerset transforms the multi-label problem into a single multi-class problem as shown in figure LP:

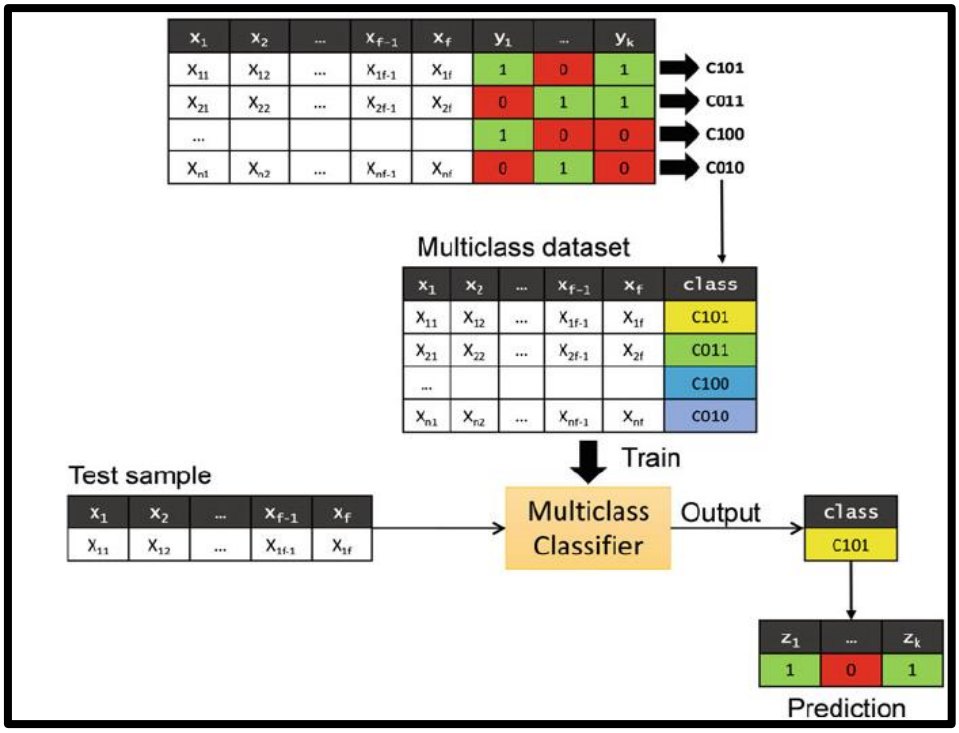


Figure II-16: Label Powerset (LP) Transformation Diagram.(Herrera et al., 2016)

The label sets in the original MLD are translated to unique class identifiers and then used to train a multiclass classifier. After processing a new test sample, the output is back translated to obtain the label predictions. (Herrera et al., 2016)

So, Label Powerset has given a unique class to every possible label combination that is present in the training set. And then a single-label classification algorithm is applied to the multi-class table obtained with the label powerset method.

. Advantages:

This method takes label correlation into account which is really important for more reliable results.

. Disadvantages:

This method has a high computational complexity which would result in a bigger computational time, and more consumption of resources when using computers to solve it. This method also can lead to imbalanced dataset with multiple classes associated with few examples.

. Problem Adaptation (Adapted Algorithm):

Adapted algorithm, as the name suggests, is about adapting the algorithm to directly perform multi-label classification, rather than transforming the problem into different subsets of problems.

Unlike the problem transformation-based methods, those following the adaptation approach aim to prepare existing classification algorithms to make them able to manage instances with several outputs. The changes that must be introduced in the algorithms can be quite simple or really difficult, depending on the nature of the original method and also the way the existence of several labels is going to be considered. There are proposals of multi-label classifiers based on trees, neural networks, vector support machines, instance-based learning techniques, and probabilistic methods. (Herrera et al., 2016) In this work we discussed the instance-based multi-label classifier known us multi-label K Nearest Neighbor (MLkNN).

. Multi-label k-Nearest Neighbors (MLkNN):

Multi-label k-Nearest Neighbors (MLkNN) (Zhang & Zhou, 2007) was the first lazy approach proposed specifically for multi-label classification. This is also a binary relevance approach which considers each label independently as a binary classification problem. Instead of a standard k-NN method, however, MLkNN uses the maximum a-posteriori (MAP) (Salton et al., 2016) approach

combined with k-NN.(Torgo et al., n.d.) It starts by building a limited model that consists of two pieces of information:

- The a priori probabilities for each label. These are simply the number of times each label appears in the Multi-label dataset divided by the total number of instances. A smoothing factor is applied to avoid multiplying by zero.
- The conditional probabilities for each label, computed as the proportion of instances with the considered label whose *k*-nearest neighbors, also have the same label.

These probabilities are independently computed for each label, facing the task as a collection of individual binary problems. Therefore, the potential dependencies among labels are fully dismissed by this algorithm. After this limited training process, the classifier is able to predict the labels for new instances. When a new sample arrives, it goes through the following steps:

1. First, the *k*-nearest neighbors of the given sample are obtained. By default, the (L−norm) (Euclidean distance) is used to measure the similarity between the reference instance and the samples in the Multi-label dataset.
2. Then, the presence of each label in the neighbors is used as evidence to compute maximum a posteriori (MAP) probability from the conditional ones obtained before.
3. Lastly, the label set of the new sample is generated from the MAP probabilities. The probability itself is provided as a confidence level for each label, thus making possible to also generate a label ranking.

It has been shown that MLkNN performed better than several algorithms.

Advantages:

It takes the correlations between labels into account which helps in more reliable results.

Disadvantages:

When using this method with the binary relevance strategy, the ratio estimation may not be accurate, also data distributions for some labels are imbalanced.

. Ensemble Methods:

. Classifier Chains (CC):

The classifier chains (CC) method implements the idea of using partial conditioning together with the true label information. CC selects an order on the set of labels $\{z_1, z_3, z_3, .., z_n\}$,which can be formally written as a bijective function (permutation):

$$\tau : \{1, \ldots, m\} \longrightarrow \{1, \ldots, m\} \quad \textbf{II-7)}$$

- Labels will be chained along this order τ:

$$\tau : \{1, \ldots, m\} \longrightarrow \{1, \ldots, m\}. \quad \textbf{(II-8)}$$

However, for this work the permutation shall be τ = id (only for simplicity reasons). The labels therefore follow the order $\{Z1{-}{>}\ z2{-}{>}\ z3..Zn\}$ In a similar fashion to the binary relevance (BR) method, CC trains m binary classifiers C_k , which are responsible for predicting their corresponding label, the $z_k, k = 1, \ldots, m$ classifiers are C_k of the form:

$$C_k : \mathcal{X} \times \{0,1\}^{k-1} \longrightarrow \{0,1\}, \quad \textbf{(II-9)}$$

Where $\{0,1\}^0 := \emptyset$. For a classifier C_k the feature space is augmented by the true label information of all previous labels $z1, z2, \ldots, zk{-}1$ hence, the training data of C_k consists of all observations $\left(\left(\mathbf{x}^{(i)}, y_1^{(i)}, y_2^{(i)}, \ldots, y_{k-1}^{(i)}\right), y_k^{(i)}\right),\ i = 1, \ldots, n$, with the target $y_k^{(i)}$.

In the example from the above, this would look like:

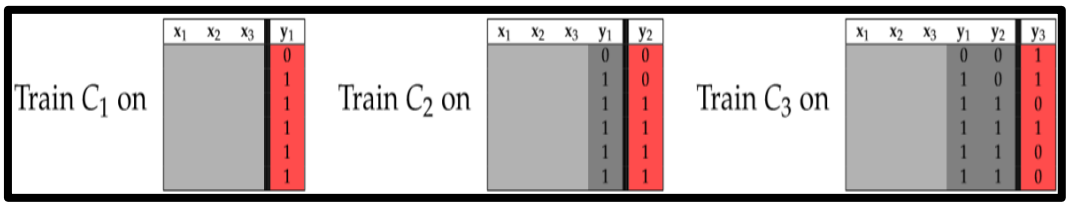


Figure II-19 : Classifier Chain Diagram.[19]

[19] exemple of classifier chain function from http://www.analyticsvidhya.com/ link https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/

At prediction time, when an unlabeled observation is $\mathbf{x}^{(l)}$ labeled, a prediction is $\left(\hat{y}_1^{(l)}, \ldots, \hat{y}_m^{(l)}\right)$ obtained by successively predicting the labels along the chaining order:

$$\begin{aligned}
\hat{y}_1^{(l)} &= C_1\left(\mathbf{x}^{(l)}\right) \\
\hat{y}_2^{(l)} &= C_2\left(\mathbf{x}^{(l)}, \hat{y}_1^{(l)}\right) \\
&\vdots \\
\hat{y}_m^{(l)} &= C_m\left(\mathbf{x}^{(l)}, \hat{y}_1^{(l)}, \hat{y}_2^{(l)}, \ldots, \hat{y}_{m-1}^{(l)}\right)
\end{aligned} \quad \textbf{II-10)}$$

Just like in the problem transformation technique, the classifiers in the CC can be the algorithms that are used for single-label classification like, Naive Bayes, Decision Trees, Random Forest classifier, Artificial Neural Network…etc. [12]

Several factors have an impact on the performance of the Classifier Chains:

• The length of the chain. A high number (k−1) of preceding classifiers in the chain comes with a high potential level of feature noise for the classifier C_k. One may assume that the probability of a mistake will increase with the level of feature noise in the input space. Then the probability of a mistake will be reinforced along the chain, due to the recursive structure of CC.

• The order of the chain. Some labels may be more difficult to predict than others. The order of a chain can therefore be important for the performance. It can be advantageous to put simple to predict labels in the beginning and harder to predict labels more towards the end of the chain. Some heuristics for finding an optimal chain ordering have been proposed in (Read et al., 2014). However, these methods are not subject of this article.

• The dependency among labels. For an improvement of performance through chaining, there should be a dependence among labels, CC cannot gain in case of label independence. However, CC is also only likely to lose if the binary classifiers C_k cannot ignore the added features $\mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$. (Probst et al., 2017)

Advantages:

This method takes label correlation into consideration, and it has an acceptable computational complexity.

Disadvantages:

Accuracy heavily depends on the order, so it must be chosen very carefully, but the real issue is for (N labels) there are (N! possible orders), so choosing the order gets harder when the number of labels increases. (Deustschman, 2019)

Conceptual Study

1. Introduction:

The Multi-label classification is a multi-purpose tool that can be used in a wide range of domains, from recommender systems to medical diagnosis, thus it can be approached through different methods and algorithms, choosing the right method is crucial for a coherent and accurate work, the choice depends on what the study is for, and what type of data you actually have.

2. Objectives:

When browsing on the web, facing ads or recommendations about web services that we are not interested into is irritating and annoying so creating relevance between the users' interests and the web services offered is almost mandatory, some websites create this relevance based on constant trainings about the users' behaviours, recording every click, every swipe and every search the user makes, however this takes time to bring an accurate relevance as it's based on constant trainings about the user's behaviours, so offering the right web services to specific users before their first interaction can be challenging, but thanks to Multi-label classification that can actually do that based on the user's profile that he/she filled when signing up to their accounts.

The training can be made with a dataset that contains features and labels, Labels here are the web services, so when a new user fills the features when signing up, the algorithm can predict the users' interests to offer relevant services .

And that is what we did in this work, we used the Multi-label classification for recommending web services to users based on their profiles using some algorithms to train the dataset, which contains profiles with the corresponding web services. The goal is to give the new users relevant web services that interest them, thus we needed achieve a high level of accuracy, precision and good evaluation results.

3. Related work:

Many multi-label classification-based researches were carried out to solve problems in different domain. A review of few researches works are presented in this section:

A study was made by some scholars in AI in 2013 (de Oliveira et al., 2013), Offering a recommender system based on the Multi-label classification for educational purposes based on student profiles.

The author attempted to assess the students' performance with the objective of improving the learning process. They developed a recommender system that suggests classes of activities based on the learners' needs. The authors introduced the concept of "student profile" which captures the student performance in a particular programming activity. The task of recommending activities was considered as a multi-label classification task aiming to map classes to each instance of student's profile resulting in student data instances being mapped to multidimensional profiles that are, in turn, mapped to classes of programming activities.

The system also has the ability to make recommendations of classes of activities to new learners which are based on existing and similar profiles. the system is based on the MLkNN algorithm and was able to achieve results matching human recommendations most of the time.

Another paper considers the task of emotion recognition from music as a multi-label classification task because a piece of music may have more than one emotion at the same time. the research work proposes the Binary Relevance (BR) based Least Squares Twin Support Vector Machine (LSTSVM) multi-label classifier for emotion recognition from music. The performance of the proposed classifier is compared with the eight existing multi-label learning methods using fourteen evaluation measures in order to evaluate it from different point of views. The experimental result suggests that the proposed multi-label classifier-based emotion recognition system is more efficient and gives satisfactory outcomes over the other existing multi-label classification approaches. (de Oliveira et al., 2013)

Another study suggests categorizing images into multiple classes simultaneously, Using the Multi-label image classification which focuses on the problem that each image can have one or multiple labels. It is known that manually labelling images is time consuming and expensive. In order to reduce the human effort of labelling images, especially multi-label images, they proposed a multi-label SVM active learning method. they also proposed two selection strategies: Max Loss strategy and Mean Max Loss strategy. Experimental results on both artificial data and real-world images demonstrated the advantage of proposed method. (Li et al., 2004)

The authors proposed Hypotheses-CNN-Pooling (HCP) for multi-label classification of where large single label image datasets are trained by CNN. An arbitrary number of object segment hypotheses are taken as the inputs, then a shared CNN is connected with each hypothesis, and finally the CNN output results from different hypotheses are aggregated with max pooling to produce the ultimate multi-label predictions. (Wei et al., 2015)

In (Cerri et al., 2009) the author compared Methods for Multilabel Classification of Proteins Using Machine Learning Techniques, Protein function classification is a typical example of multi-label classification, since a protein may have more than one function. They used both algorithm independent approach (Random k-Label (RAKEL) method) and algorithm dependent approach (ML-kNN) with evaluation metrics accuracy, precision, recall, F-measure and sub-accuracy. They used yeast and protein datasets and analysed the performance of the One-Against-All, Label-Powerset, Cross-Training, ML-kNN and RAKEL.

A book demonstrating multiple methods about the multi-label classification techniques like Problem Transformation, Problem Adaptation and Ensemble of classifier chains. (Herrera et al., 2016)

Methods like Binary relevance, Label Powerset, MLkNN algorithm and more are all explained, evaluated and compared with each other.

4. Methodology:

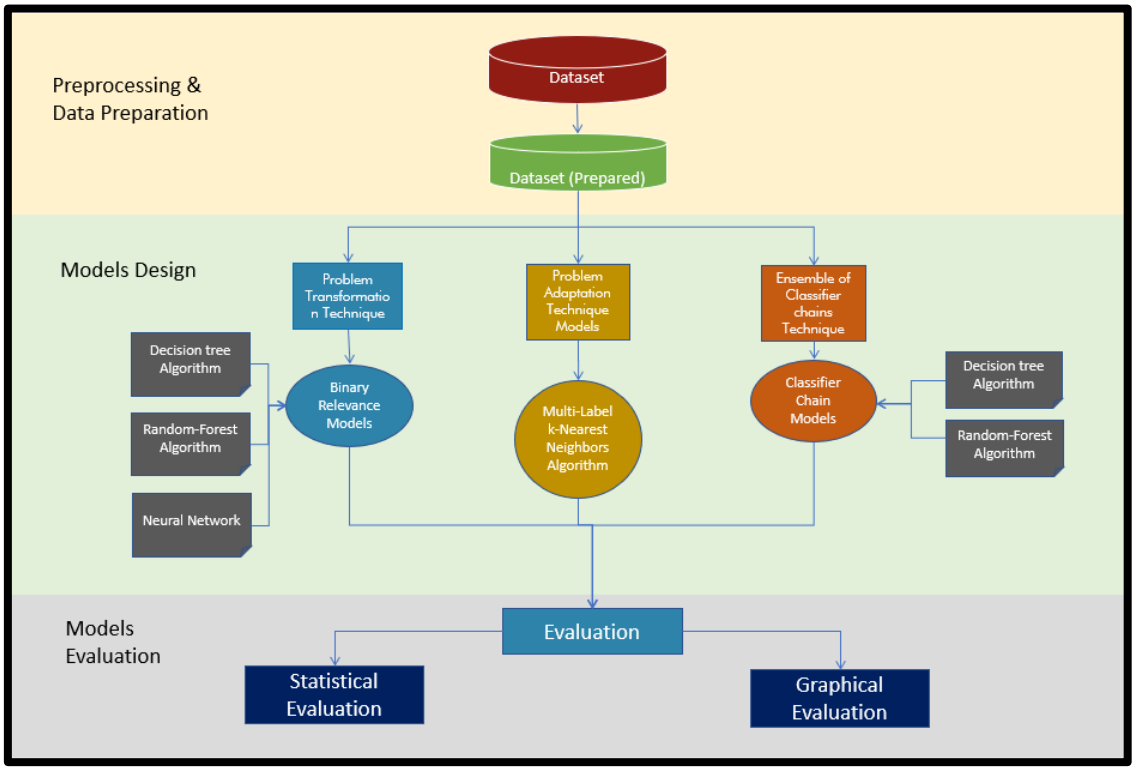


Figure III-1: A view of our Methodology.

A dataset is a collection of data. In other words, a dataset corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the dataset in question. In Machine Learning projects, we need a training dataset**.** It is the actual dataset used to train the model for performing various actions.

Machine learning depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible.

We conducted our study on a couple of Multi-label datasets, all of them have the same number of features which is 18 and they are the same features for every dataset. We have profiles of users as instances and the labels as web services.

The number of instances is from 30000 to 100000 the number varies from a dataset to another. For the labels, some datasets have 10 and others have 15 labels. The reason why we used different

datasets with different numbers is to see how much those numbers contribute to the results; we discuss those results in the discussion chapter.

4.1. Pre-processing:

Some of the information gathered from data sources can be inconsistent and/or irrelevant. Differences in scales, noise presence, and other anomalies have to be adequately addressed through cleaning methods by removing unimportant data, only the useful and truly relevant information is selected for further steps. In addition, depending on its nature, several pre-processing methods for data reduction can be applied to data. All of them aim to prepare the data in order to ease the learning conducted in the next phase. (Herrera et al., 2016)

We performed pre-processing into a couple of steps:

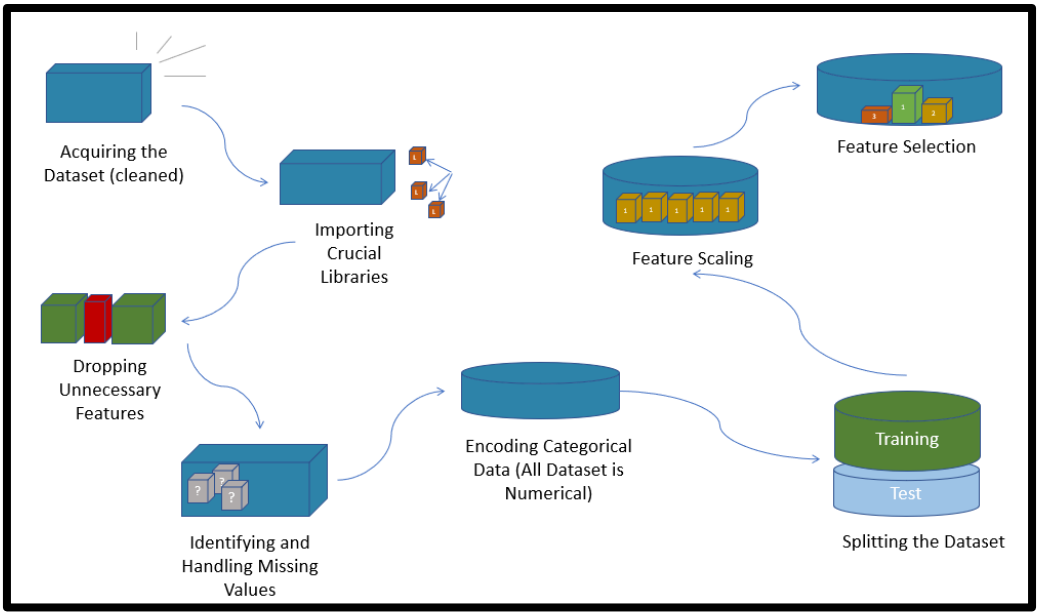


Figure III-2: A view of our pre-processing approach.

- **Acquiring the dataset**

The datasets we used in our study were generated by an application of a work done in 2017.

(Boudjema,2017)

- **Importing all the crucial libraries:**

  Here, we imported all the crucial and the necessary libraries who are responsible for data pre-processing such as NumPy and Pandas which will be explained in the discussion chapter

- **Importing the dataset:**

  In this step, we imported the dataset to the code with the corresponding directory

- **Dropping unnecessary features:**

  Some features have no correlation with the other features and play no role in predicting the labels such as the ID and the Name features, keeping them would affect the results negatively and would increase the running time, so dropping them is crucial, in our datasets we dropped the ID and the Name.

- **Identifying and handling the missing values:**

  Missing values can be present in a dataset for disparate reasons. They can exist due to a failure in previous data transcription steps, or simply because someone has left an empty field. Pre-processing can consist in removing the instances with missing values or following some data imputation scheme, usually replacing the missing value by an estimation model. (McKinney, 2013)

  Fortunately for us, our datasets were generated with no missing values.

- **Encoding the categorical data:**

  Categorical data refers to the information that has specific categories within the dataset

  In our datasets we have: Sex, Educational level, Contact and Religion.

  - Machine Learning models are primarily based on mathematical equations. Thus, you can intuitively understand that keeping the categorical data in the equation will cause certain issues since you would only need numbers in the equations.
  - So, for the sex feature we replaced “Male” with 1, and “Female” with two
  - For the educational level we replaced the grades with numbers, ex: primary with 1, secondary with 2.

- For Contact, Religion we used the One-Hot encoding method which encodes the values as a binary vector array, in other words **One-Hot Encoding** simply creates one column for every possible value and put a 1 in the appropriate column. We also used this method for the description taking every word of each text as a feature.
- The contact feature represents the email address of the user, it comes in the form of (userX@serviceY.abc) where X represents the id of the user, serviceY represents the service, and abc represents the domain name (com, org and dz), we have split this feature into two features, one feature is serviceY and the other is the domain name, and we applied the One-Hot Encoding method for each feature of those.
- The Address feature was not categorical, it was a numerical data given in the form of geographical coordinates (longitude # altitude), what we did was splitting it into two features, longitude and altitude.
- The languages feature contains multiple languages, each language takes a value of only zero or one, in other words, each language plays the role of a feature, however we did not do this with one hot encoding, the dataset came with this format by default. For simplicity reasons we refer to them as one feature when we say we have 18 features, but in reality, they are treated as multiple features by the code. Same thing for interests, it is actually composed of three features, interests in economy, interests in culture, interests in sports.

- **Splitting the dataset:**

When we work on a model and want to train it, we obviously have a dataset. But after training, we have to test the model on some test dataset. For this, we will use a dataset which is different from the training set we used earlier. But it might not always be possible to have so much data during the development phase.

In such cases, the obvious solution is to split the dataset we have into two sets, one for training and the other for testing; and we do this before we start training our model, and that is exactly what we did.

There are a few parameters that we need to understand before we use the class:

- **Test size:** This parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if we pass 0.5 as the value, the dataset will be split 50% as the test dataset. If we are specifying this parameter, we can ignore the next parameter.
- **Train size:** We have to specify this parameter only if we are not specifying the test-size. This is the same as test-size, but instead we tell the class what percent of the dataset we want to split as the training set.
- **Random state:** Here we pass an integer, which will act as the seed for the random number generator during the split. Or we can also pass an instance of the random state class, which will become the number generator. If we do not pass anything, the Random state instance will be used randomly. (Srinidhi, 2018)

In our work we have split our dataset in an 80-20 ratio which is the most common practice in data science

- **Feature scaling:**

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one. (Roy, 2020)

In our datasets, the age and the educational level columns for example, do not have the same scale, the age is taken with days so a 23 years old guy would be 8395 days, however the educational level has values of 1, 2, 3 …..10 . In such a scenario, if you compute any two values from the age and educational level columns, the age values will dominate the educational level values and deliver incorrect results. Thus, we had to remove this issue by performing feature scaling for Machine Learning.

The most common techniques of feature scaling are Normalization and Standardization.

In our datasets we used the normalization method which is called Min-Max scaling which scales all values in a fixed range between 0 and 1 using the following equation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \textbf{(III-1)}$$

where x' is the normalized value and x is the original value.

max(x) and min(x) are the maximum and the minimum values of the feature respectively.

It can be easily seen that when x=min, then x'=0, and When x=max, then x'=1.

So the entire range of values of X from min to max are mapped to the range 0 to 1.(Saini, 2019)

- **Feature selection:**

Feature Selection is the process that automatically or manually selects those features which contribute most to the prediction variable or output in which we are interested in.(Shaikh, 2018)

Having irrelevant features in our data can decrease the accuracy of the models and make our model learn based on irrelevant features. So, Feature Selection is one of the core concepts in machine learning which can impact the performance of the model positively.

In our work we used Univariate feature selection which works by selecting the best features based on univariate statistical tests. We compare each feature to the target variable, to see whether there is any statistically significant relationship between them. It is also called analysis of variance (ANOVA). When we analyse the relationship between one feature and the target variable, we ignore the other features. That is why it is called 'univariate'. Each feature has its test score. (Liang, 2019) Finally we select the K highest scoring features to use for training the model, in our work we transformed the values obtained with the sigmoid function, to restrict the range between 0 and 1, then we applied a threshold, if the value is less than 0.5, we do not take the feature, otherwise we take it, however all the features had values higher than 0.5 because the dataset was generated with a mechanism that makes all the features relevant.

4.2. Methods and Algorithms:

We have used the three available techniques to perform a Multi-label classification:

1. Problem transformation: We used both Binary Relevance and Label Powerset methods using Decision tree, Random Forest, and Neural Network algorithms.

2. Problem adaptation: We used the Multi-label K-Nearest Neighbor (MLKNN) and Support Vector Machine (SVM).
3. Ensemble: we used Classifier Chain (CC).

We will get into this in detail in the models' section.

5. Solution overview:

5.1. User profile description:

The datasets we used in our study were generated by an application of a work done in 2017 (Boudjema,2017), the work's goal was discovering clusters of communities based on the users profiles and the web services they have interacted with, the profiles are characterized by a number of attributes, however we selected only the attributes that work well for our classification. The selected attributes are illustrated in figure III-3:

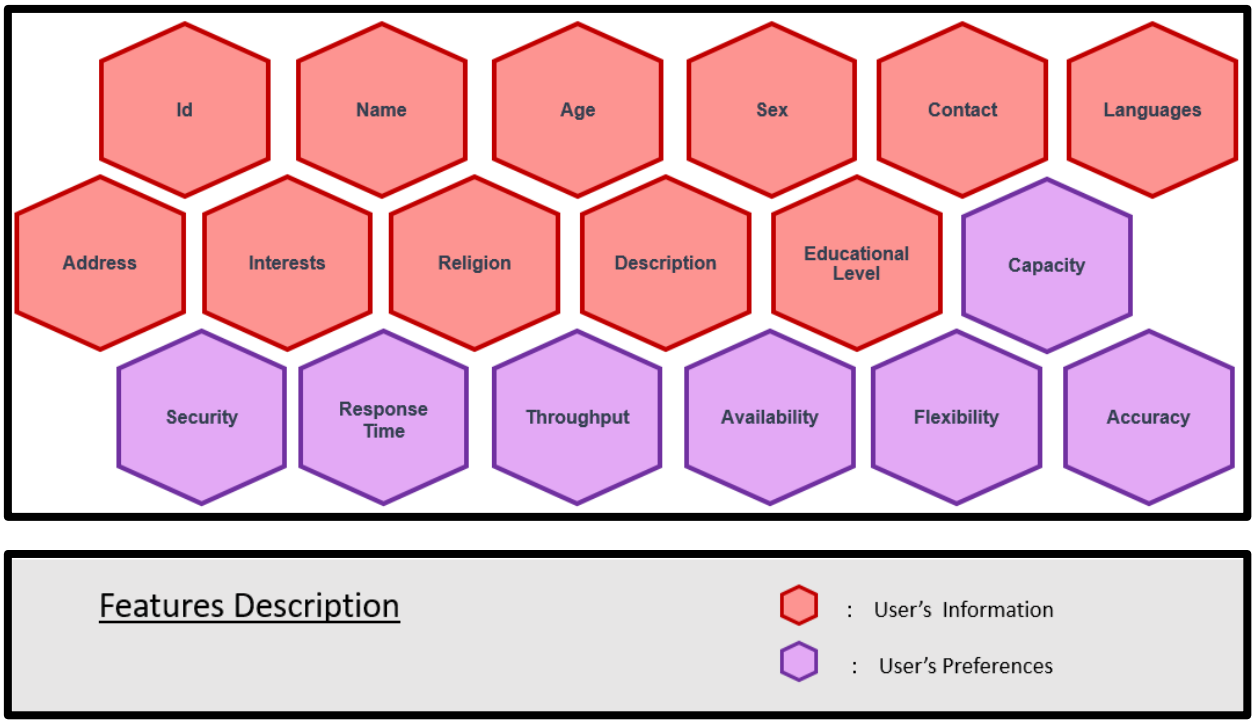


Figure III-3: Profile's Description.

:

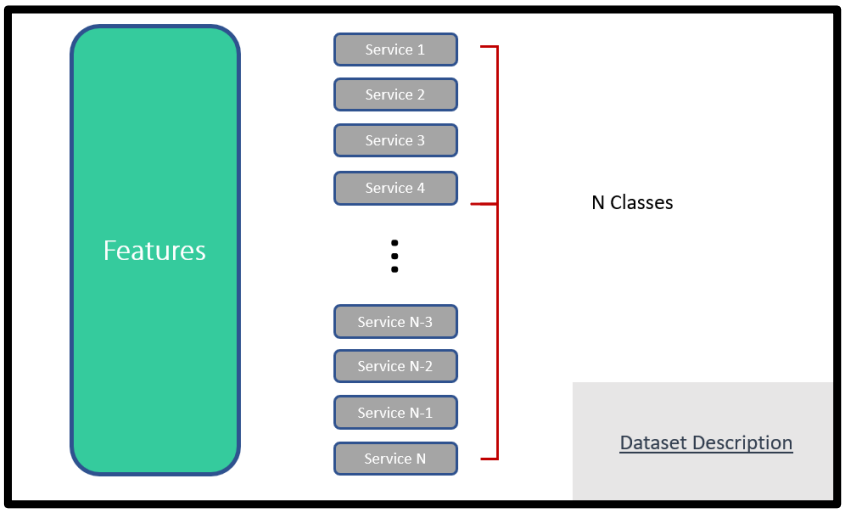


Figure III-4: Features with labels (classes).

We distinguish two main categories in the elements that represent the profile, each category has its own attributes:

5.1.1. First Category (User's information):

Its attributes are:

- **Identity:** Id of the user.
- **Name:** The name of the user.
- **Age:** The Age of the user.
- **Sex:** The gender of the user.
- **Contact:** The email of the user.
- **Educational Level:** The user's educational grade.
- **Languages:** list of languages that the user knows.
- **Interests:** It is composed of three interests, economy, culture and sports
- **Description:** A brief description explaining what type of services the user wants.
- **Address:** The user's location.
- **Religion:** The user's religion or affiliation

5.1.2. Second Category (User's preferences of the desired web services):

Its attributes are:

- **Capacity:** It is the maximum number of simultaneous requests that the system supports with guaranteed performance.
- **Response time:** It is the speed of execution of a query or the time taken to send a query and receive a response.
- **Throughput:** It is the ability of a service to perform its required functions under specified conditions over a given period of time.
- **Availability:** It is the probability that the system is active.
- **Flexibility:** It represents the rate for a service to be performed correctly in the case of ineligible, incomplete or inconsistent data.
- **Accuracy:** It represents the error rate produced by the service.
- **Security:** It represents the level of safety on a scale of 1 to 10, with 10 being the maximum safety. This is the level of security of the desired web service.

The user would specify all these attributes according to his/her needs and how he/she wants the desired web services to act.

5.2. Dataset Construction:

The construction of the dataset containing the users and the web services they have accessed to, is done in four processes:

- Profile creation process.
- Process for calculating similarity between users and web services.
- Processes for generating and filtering user access to web services.
- Community generation process.

All these processes are described in (Boujema,2017)

At the end of these processes, a file is generated containing the users' attributes as well as binary type columns specifying whether each user has access to the existing services that represent the assignment classes.

5.3. Models:

A machine learning model is a file that has been trained to recognize certain types of patterns.

We train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once we have trained the model, we can use it to reason over data that it has not seen before and make predictions about those data. (de Oliveira et al., 2013)

In our work we have tested a lot of models that were based on the Multi-label classification methods and algorithms which we mentioned previously, so we will go step by step explaining every model we have used.

5.3.1. Models based on the problem transformation technique:

The problem transformation which basically maps the multi-label learning task into one or more single-label learning tasks (explained in the 2nd chapter), can be achieved through some methods like Binary Relevance and Label Powerset, we will start by talking about our models which were based on binary relevance:

We will start by talking about the models which are based on the Binary Relevance (BR) Method

BR works by decomposing the multi-label learning task into a number of independent binary learning tasks (one per class label) (explained in the 2nd chapter), however to train the independent binary learning tasks we would need an algorithm that can do this. For BR we used Random Forest, Decision Tree and Deep Neural Network, so 3 models were created using the BR method:

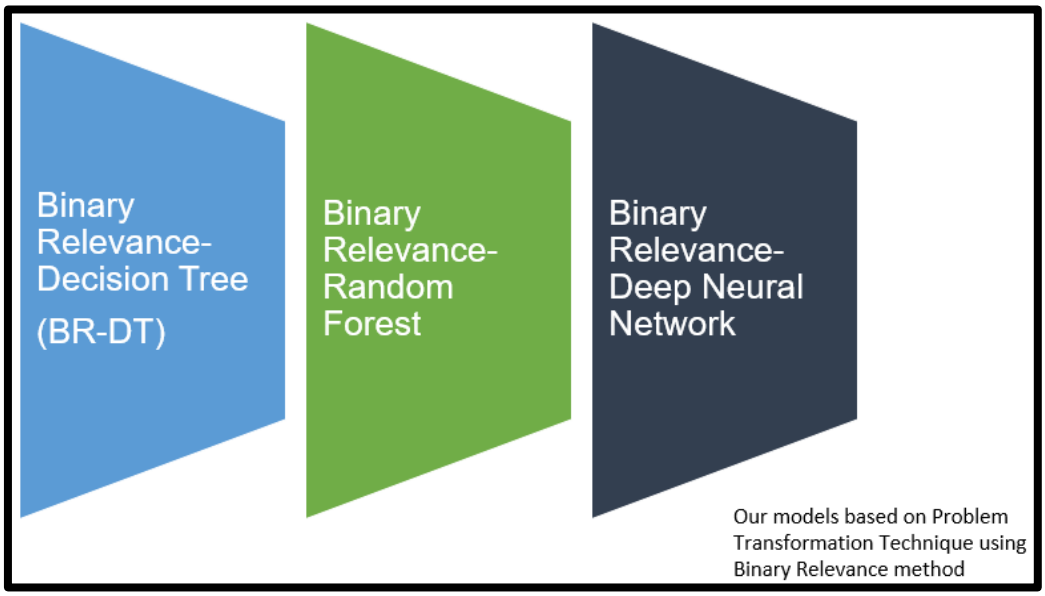


Figure III-5: Transformation technique models.

5.3.1.1. Binary Relevance with Decision Tree Classifier model:

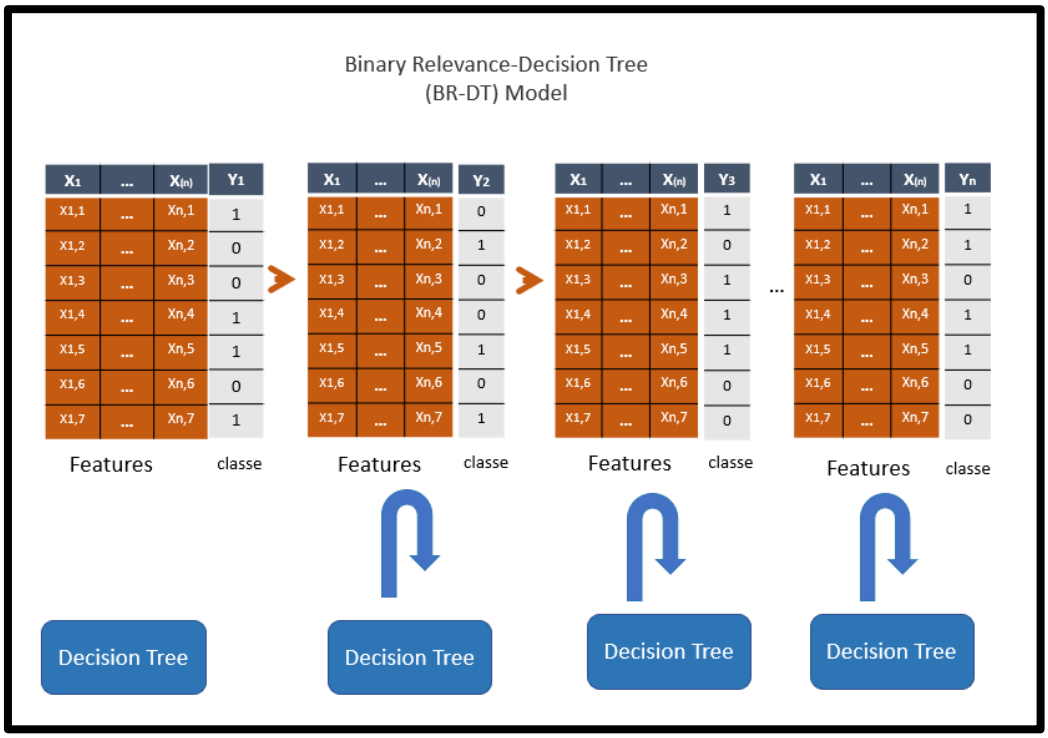


Figure III-6: Binary Relevance with Decision Tree Classifier model.

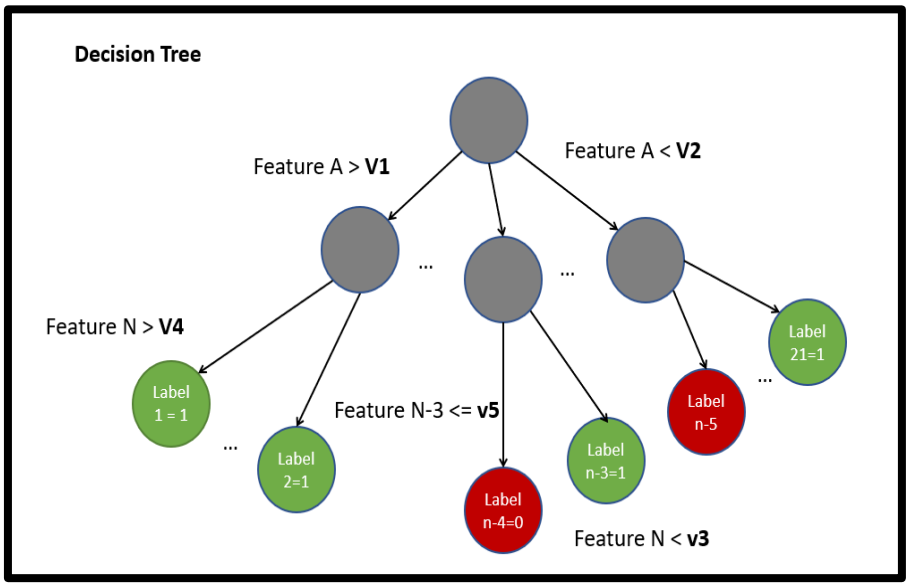


Figure III-7: Decision Tree algorithm.

For a Decision Tree Classifier to make decisions, it uses some algorithms like: ID3, Gini Index, Chi-Square, Reduction in Variance.

In our model we used ID3 which is the core algorithm for building decision trees**.** Developed by J. R. Quinlan, this algorithm employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree. (Rishabah, 2017) (This method is well explained in chapter 2)

5.3.1.2. Binary Relevance based with Random Forest Classifier Model:

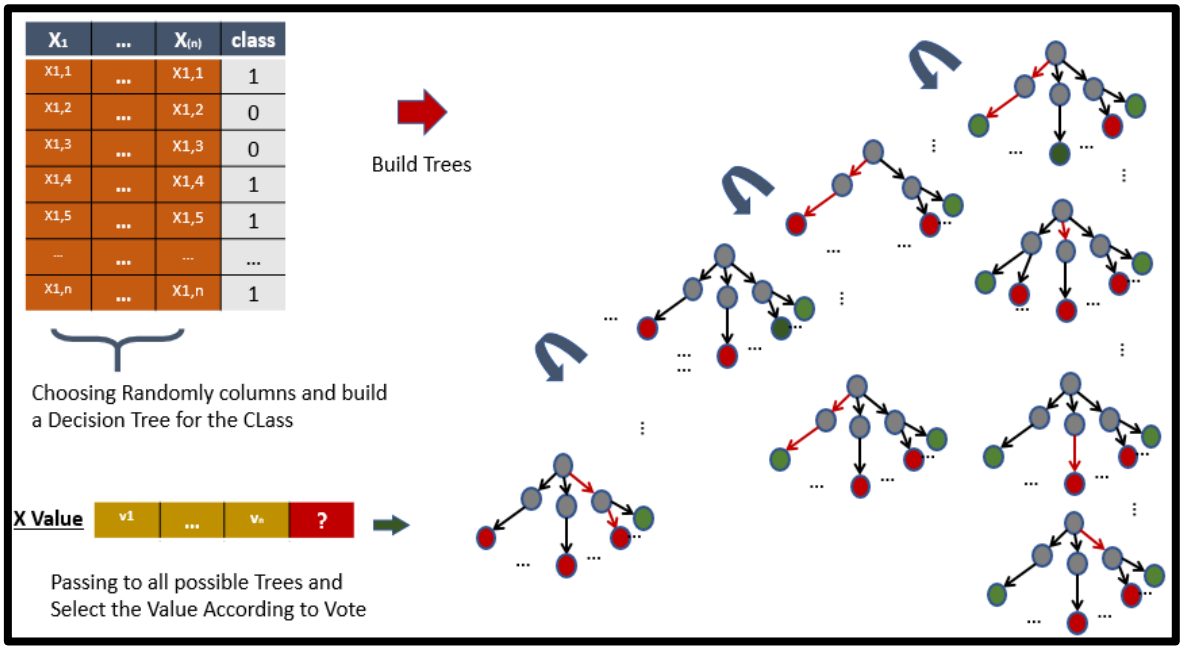


Figure III-8: Random Forest algorithm.

Random Forest Classifier uses multiple Decision Trees and then those trees would predict and vote for what the label is. For the trees to make decision they used some algorithms like ID3 and GINI to make decisions (as explained in the BR-DT method) In our model we have used ID3 because that what brought us the best results.

The way Random Forest Classifier works is well explained in chapter 2.

5.3.1.3 Binary Relevance based with Deep Neural Network model:

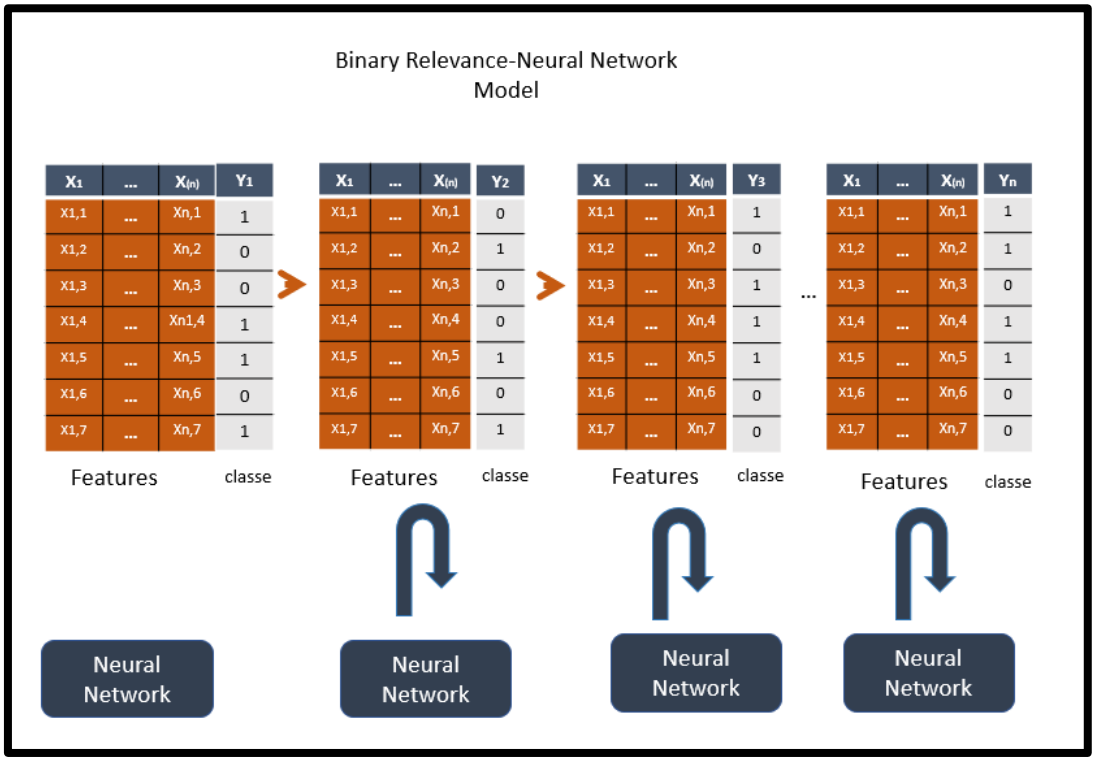


Figure III-9: Binary Relevance based with Deep Neural Network model.

We have used feed forward multilayer perceptron with 2 hidden layers, the reason we used 2 hidden layers is because it gave us the best results after testing different numbers of hidden layers.

In the 10 labels dataset, the input layer consists of 69 neurons which is the total number of features after pre-processing, the first hidden layer consists of 50 neurons and the second hidden layer consists of 30 neurons .

In the 15 labels dataset, the number of neurons in the hidden layers is slightly different, with 51 and 33 neurons for the first and the second layer respectively.

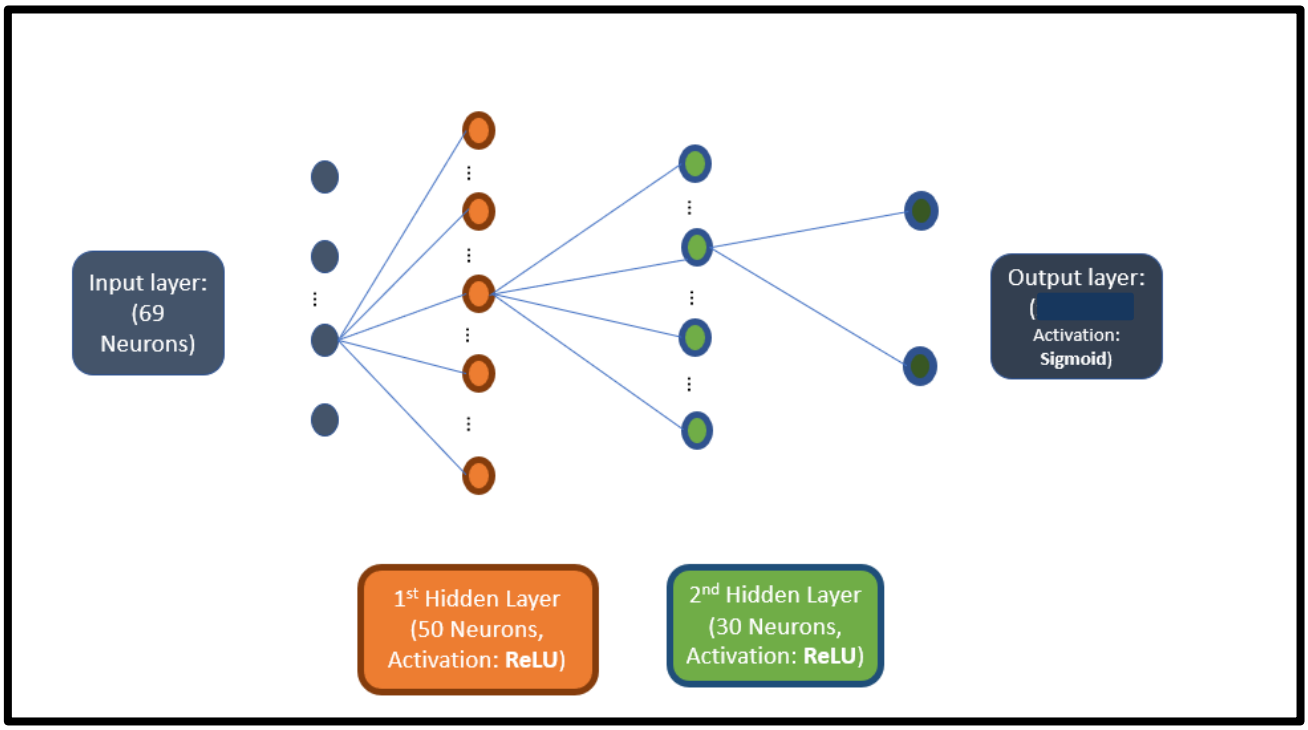


Figure III-10: Neural Network Architecture model.

For the input and hidden layers we used the Rectified Linear Unit (ReLU) activation function, The ReLU is the most used activation function in the world right now since it is used in almost all the convolutional neural networks or deep learning (sharma, 2017).

For the output layer we used the Sigmoid activation function, the main reason why we used the Sigmoid function is because it exists between (0 to 1) and our outputs (target variables) are zeros and ones.

We have used an epoch of 50, One Epoch is when an entire dataset is passed forward and backward through the neural network only once, passing the entire dataset through a neural network is not enough. we need to pass the full dataset multiple times to the same neural network. A large number of epochs can cause overfitting, so we had to find the best number that maximizes the results without overfitting and that number was 50.

We used a batch size of 64, batch size is the total number of training examples present in a single batch. We can not pass the entire dataset into the neural net at once. So, we divide the dataset

into number of batches or sets or parts (Sharma et al., 2020) , after each batch the neural network updated by updating the weights and the biases, so a small batch size means more updates which results in a better model, however it can cause overfitting, but we kept an eye on it, so number 64 gave us the best results without overfitting.

Another point worth mentioning is that we didn't use feature selection in our Deep Neural Network model because DNN adjusts the weights of the features and the biases by itself, also we can't remove any feature because all features contribute positively in the prediction process because the datasets were well generated to make all the features relevant as explained in (Boudjema,2017).

5.3.2. Problem Adaptation Technique Model:



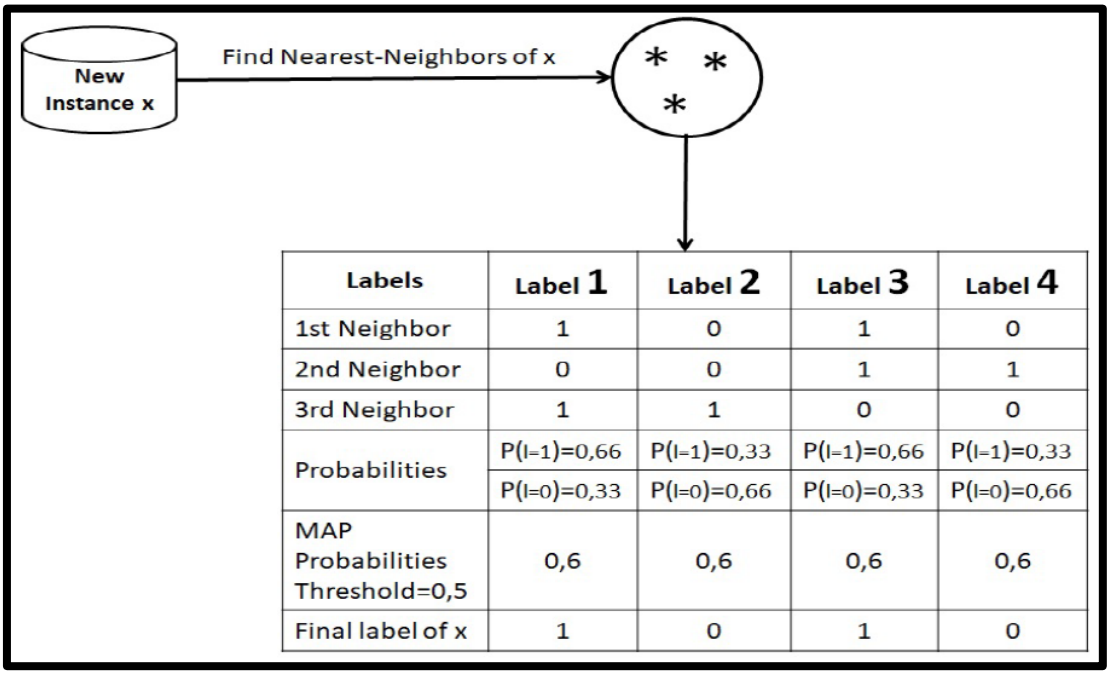
Figure III-11: Adaption technique model.

For this technique we used the Multi-label K Nearest Neighbors (MLkNN) to build our model.

First, the MLKNN searches for K-Nearest- Neighbors by measuring the similarity between the new instance and all the examples of the dataset using the Euclidean distance, we used some tests to tune the parameters for optimal results like the K value (number of nearest neighbors), for this purpose we used a grid search function which goes through multiple numbers of K from a specific range we choose, and then it gives the results of each value of K so that we can select the K that offers best results, by doing this we selected. The number 19 as a value of K.

Our model selects the nearest neighbors using the Euclidian distance, same way it does with traditional single label KNN. (KNN is well explained in chapter 2)

After that the algorithm calculates the probabilities of each label of the nearest neighbors to predict the labels of the new instance, the procedure is as shown in (figure III-14).



Labels	Label 1	Label 2	Label 3	Label 4
1st Neighbor	1	0	1	0
2nd Neighbor	0	0	1	1
3rd Neighbor	1	1	0	0
Probabilities	P(l=1)=0,66	P(l=1)=0,33	P(l=1)=0,66	P(l=1)=0,33
	P(l=0)=0,33	P(l=0)=0,66	P(l=0)=0,33	P(l=0)=0,66
MAP Probabilities Threshold=0,5	0,6	0,6	0,6	0,6
Final label of x	1	0	1	0

Figure III-12: (MLkNN) principle example to classify a new instance. (Settouti et al., 2019)

5.3.3. The Classifier Chain technique model :

The classifier chain method uses a set of binary classifiers linked in a chain, C0, C1, . . . , Cn is constructed, where a classifier Ci uses the predictions of all the classifier Cj, where j <i, and it considers the previous labels as features. So it combines the computational efficiency of the Binary Relevance method while still being able to take the label dependencies into account for classification. (this method is explained in Chapter 2).

In our work we used a single classifier for each model, because according to our tests, that what gave us the best results.

The models we used using the classifier chain technique are shown in (figure III-15):

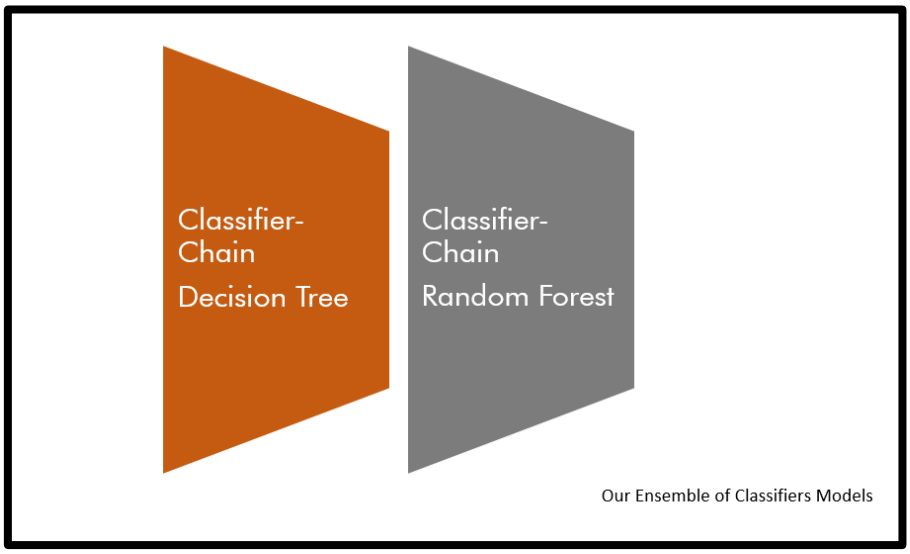


Figure III-13: Classifier Chain technique model.

5.3.3.1. Classifier Chain with Decision Tree Classifier:

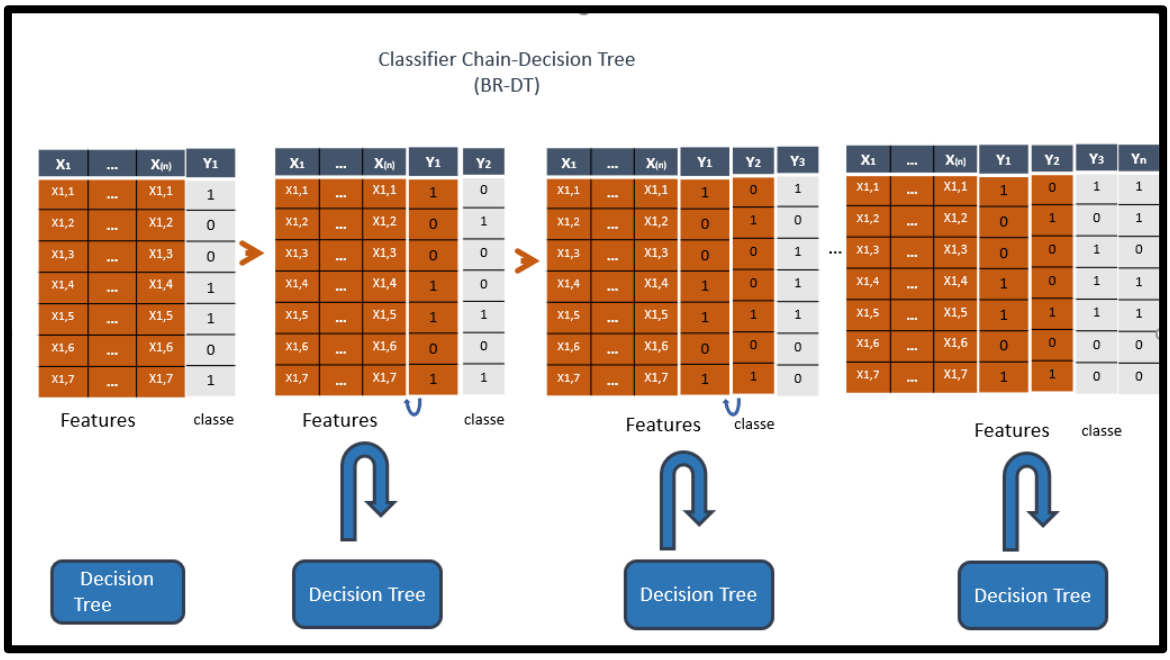


Figure III-14: Classifier Chain with Decision Tree classifier.

Same like our Binary Relevance with Decision Tree Classifier model, we have used ID3 for the trees to make decisions, because that what gave us the best results.

5.3.3.2. Classifier Chain with Random Forest classifier:

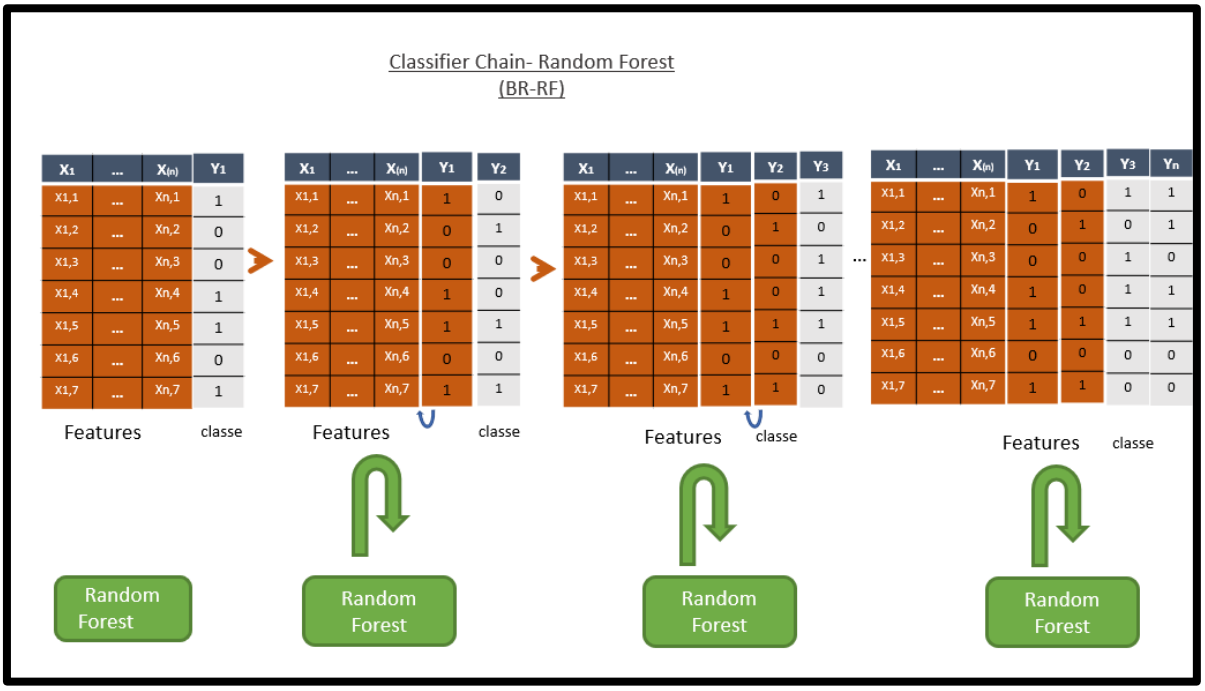


Figure III-15: Classifier Chain with Random Forest classifier.

Here we also used the ID3 algorithm for the predictions.

The order of the labels in the chain affects the results, that is why there is a method that suggests constructing many chains with all the orders possible, and then we make a voting ensemble of classifier chains by averaging the binary predictions of the chains and apply a threshold of 0.5.

However, we didn't apply it in our model, we chose one single chain with the default order, because trying all orders means trying k factorial possibilities where k is the number of labels, our datasets have 10 and 15 labels which means trying 3628800 for 10 labels and 1.30*10^12 for 15 labels possibilities, which would take a huge running time, hence it's not feasible for us. That is why we used the default order, but even with this choice, it still delivers great results.

6. Evaluation methods:

In order to compare these different techniques and algorithms, we must apply some evaluation metrics to test the efficiency of each classification model, there for we will introduce some of the

very common and widely used classification evaluation metrics that can be used for multi-label classification.

6.1. Hamming Loss:

Hamming loss is probably the most commonly used performance metric in Muli-label classification. This is not surprising, as it is easy to calculate. The Δ operator returns the symmetric difference (also known as exclusive or ⊕) between *Yi*, the real labelset of the ith instance, and *Zi*, the predicted one. The |*r*| operator counts the number of 1 s in this difference, in other words the number of miss predictions. The total number of mistakes in the *n* instances is aggregated and then normalized taking into account the number of labels and number of instances.

N: The number of data instances in the multi-label dataset

k: the total number of labels.

Yi: The real labelset of the ith instance.

Zi: The predicted labelset of the ith instance.

| | counts the number of 1s inside this operator.

$$HammingLoss = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n}|Y_i \Delta Z_i| \quad \textbf{III-2)}$$

6.2. Accuracy:

In the multi-label field, Accuracy is defined as the proportion between the number of correctly predicted labels and the total number of active labels, in both real label set and the predicted one. The measure is computed by each instance and then averaged, as all example-based metrics.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \quad \textbf{(III-3)}$$

6.3. Precision and Recall:

Precision is considered one of the more intuitive metrics to assess multi-label predictive performance. It is calculated as the proportion between the number of labels correctly predicted and the total number of predicted labels. Thus, it can be interpreted as the percentage of predicted

labels which are truly relevant for the instance. This metric is usually used in conjunction with Recall that returns the percentage of labels correctly predicted among all truly relevant labels. That is, the ratio of true labels is given as output by the classifier.

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}$$

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|} \quad \textbf{III-4)}$$

6.4. F1 score:

F1 score, also known as balanced F-score or F1-measure

The F1 score can be interpreted as a weighted average of the precision and recall, This way a weighted measure of how many relevant labels are predicted and how many of the predicted labels are relevant is obtained. An F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \textbf{III-5)}$$

6.5. Subset Accuracy:

This is possibly the strictest evaluation metric. It is also known as the exact match, since full labelsets, the predicted and the real one, are compared for full equality, it considers a correct prediction for an instance if all the labels of that instance are predicted correctly, otherwise it'll return 0. The larger is the labelset, the lower the likelihood that the classifier produces exactly the correct output. Therefore, for datasets with large sets of labels obtaining low Subsect Accuracy is something usual.

$$SubsetAccuracy = \frac{1}{n}\sum_{i=1}^{n}[\![Y_i = Z_i]\!]$$

(III-6) (Herrera et al., 2016)

7. Note:

The previous metrics are example-based metrics which are firstly evaluated by each instance and then averaged according to the number of instances considered. Therefore, the same weight is assigned to every instance in the final score, whether they contain frequent or rare labels.

There are other metrics called label-based metrics known as macro-averaging and micro-averaging. Any of the metrics obtained from a binary partition of labels, such as Precision, Recall or F1 score, can also be computed using these strategies.

In the macro-averaging approach, the metric is evaluated once per label, using the accumulated counters for it, and then the mean is obtained dividing by the number of labels. This way the same weight is assigned to each label, whether it is very frequent or very rare.

On the contrary, the micro-averaging strategy first adds the counters for all labels and then computes the metric only once. Since the predictions where rare labels appear are combined with that made for the most frequent ones, the former are usually diluted among the latter. Therefore, the contribution of each label to the final measure is not the same. (Herrera et al., 2016)

For better understanding, let's take precision for example using TP, FN, FP and TN for a better demonstration:

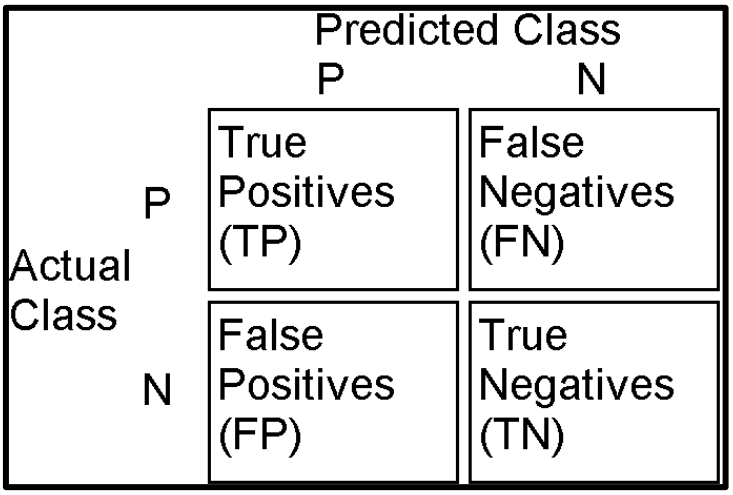


Figure III-16: Structure of the confusion matrix with TP, FN, FP and TN values.(Horak et al., 2017)

An example-based precision would be:

$$\textbf{Precision}=\frac{TP}{TP+FP} \quad \textbf{(III-7)}$$

-Macro averaging precision would be:

$$\textbf{Precision Macro}=\frac{PRECISION1+\cdots+PRECISIONk}{k}\textbf{, where k is the number of labels (III-8)}$$

-Micro averaging precision would be:

$$\textbf{Precision Micro}=\frac{TP1+\cdots+TPk}{TP1+\cdots+TPk+FP1+\cdots+FPk} \quad \textbf{(III-9)}$$

If there are labels with more instances than others and we want to bias our metric towards the most populated ones, it is better to choose micro averaging.

However, if there are labels with more instances than others and we want to bias our metric toward the least populated ones (or at least we do not want to bias toward the most populated ones), here it is better to use macro averaging.

K-Fold Cross Validation:

K-Fold technique is a popular evaluation technique, it generally results in a less biased model because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data. This method follows the below steps. (Sanjay, 2018)

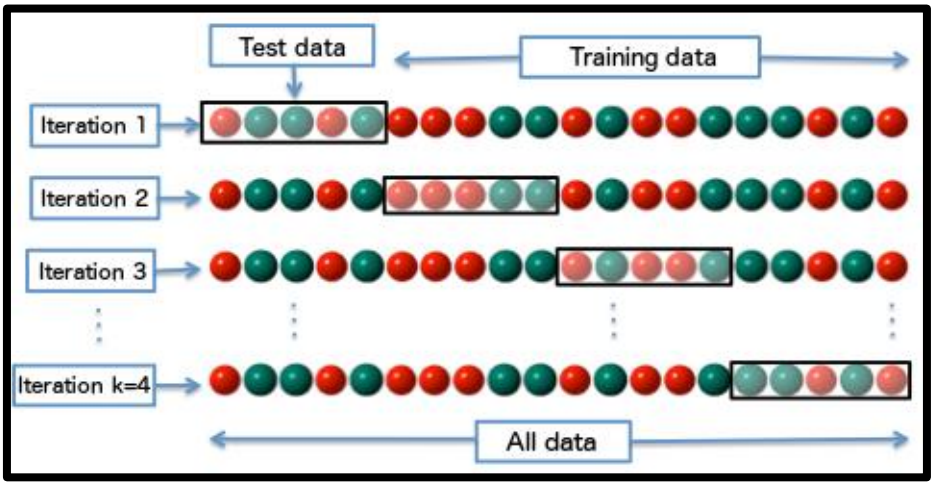


Figure III-17: 4-fold Cross-Validation. (Sanjay, 2018)

8. Conclusion:

In this chapter we addressed the overall mechanism of the application we introduced in this work, along with the goal we tried to achieve. We went through it step by step from acquiring and generating the datasets to how we could transform it to a suitable and usable data by the machine learning algorithms.

We explained how the datasets were constructed and what are the elements that compose the datasets, we also talked about the techniques, the methods, the algorithms and the models used for our work, however the choice of the model would depend on the results of the evaluation metrics of each one, we will talk about this in detail going through some comparisons in the next chapter.

1. Split the entire data randomly into K folds (value of K should not be too small or too high, ideally we choose 5 to 10 depending on the data size), in our work, we chose 5 as a value of k.
2. Then fit the model using the K-1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.
3. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model. (Sanjay, 2018)

We used 5-fold Cross-Validation in our work to evaluate our models, with all of the following metrics: Accuracy, Hamming Loss, Macro precision, Micro precision, Macro recall, Micro recall, Macro F1-score, Micro F1-socre, Subset accuracy (Exact match).

Our evaluations are explained in detail in chapter 4.

Evaluation & Discussion

1. Introduction:

In this chapter we present the efficiency of our work, mentioning all the tools we have used, both hardware and software, along with the programming language and the libraries, and of course we will tackle the results of each model proposed in the previous chapter with the different datasets, discussing and analyzing the results with numbers and plots to come up with a reliable conclusion.

2. Development Tools:

2.1. Hardware used:

2.1.1. Computer:

- Processor: Intel(R) Core (TM) **i5-7200U** CPU @ **2.40 Ghz 2.70Ghz**
- Random Access Memory (RAM): **8GB**
- Graphical Processing Unit (GPU): **NVIDIA 920MX** with **2GB** of adapter RAM

2.2. Operating system:

The computer use Microsoft Windows 10 as the operating system used for the development environment.

Microsoft Windows, commonly referred to as Windows, is a group of several proprietary graphical operating system families, all of which are developed and marketed by Microsoft. Microsoft introduced an operating environment named Windows on November 20, 1985, as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer (PC) market with over 90% market share, overtaking Mac OS, which had been introduced in 1984.

On September 30, 2014, Microsoft announced Windows 10 as the successor to Windows 8.1. It was released on July 29, 2015 and addresses shortcomings in the user interface first introduced with Windows 8. Changes on PC include the return of the Start Menu, a virtual desktop system,

and the ability to run Windows Store apps within windows on the desktop rather than in full-screen mode. [20]

2.3. Programming Languages:

We used Python as a programming language under Anaconda distribution.

2.3.1. Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse [21]. It can also be extended to make system calls to almost all operating systems and to run code written in C or C++. Due to its ubiquity and ability to run on nearly every system architecture, Python is a universal language found in a variety of different applications.[22] The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python Packages used:

. NumPy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

[20] Wikipedia Microsoft office definition from https://en.wikipedia.org/wiki/Microsoft_Windows#Windows_10
[21] Definition of Python from https://www.python.org/doc/essays/blurb/
[22] Python from (OpenSource.com) https://opensource.com/resources/python

At the core of the NumPy package, is the ndarray object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.

The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. [23]

. Pandas:

Pandas is an open source Python library for highly specialized data analysis. Currently it is the reference point that all professionals using the Python language need to study and analyse data sets for statistical purposes of analysis and decision making. This library has been designed and developed primarily by Wes McKinney starting in 2008; later, in 2012, Sien Chang, one of his colleagues, was added to the development. Together they set up one of the most used libraries in the Python community. Pandas arises from the need to have a specific library for analysis of the data which provides, in the simplest possible way, all the instruments for the processing of data, data extraction, and data manipulation. This Python package is designed on the basis of the NumPy

[23] Definition of Numpy from (numpy .org) https://numpy.org/doc/stable/user/whatisnumpy.html

library. This choice, we can say, was critical to the success and the rapid spread of pandas. In fact, this choice not only makes this library compatible with most of the other modules, but also takes advantage of the high quality of performance in calculating of the NumPy module.(Pushpa & Karpagavalli, 2017)

. Matplotlib:

Matplotlib is a Python library specializing in the development of two-dimensional charts (including 3D charts); in recent years, it has been widespread in scientific and engineering circles.

Among all the features that have made it the most used tool in the graphical representation of data, there are a few that stand out:

- extreme simplicity in its use
- gradual development and interactive data visualization
- greater control over graphic elements
- export in many formats such as PNG, PDF, SVG, and EPS.

Moreover, since Matplotlib is a Python library, it allows you to exploit the full potential of other libraries available to any developer which implements with this language. In fact, with regard to data analysis, Matplotlib normally cooperates with a set of other libraries such as NumPy and pandas, but many other libraries can be integrated without any problem. (Nelli, 2015)

. Sickit-learn:

Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data pre-processing, model selection and evaluation, and many other utilities.

Scikit-learn provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to a particular type of data.

We have used Sickit-learn to import our algorithms like Decision Tree and Random Forest. We also used it to evaluate the models with cross validation and evaluation metrics (Accuracy, Hamming Loss, F1-score, Precision, Recall and Subset accuracy).

In the Sickit-learn package there is a module that we used called joblib which provides utilities for pipelining Python jobs, it was responsible for saving our models so we don't have to repeat the training every time we run the application, once the model is saved, we can load it later while making the prediction.

. Scikit-multilearn:

Scikit-multilearn follows the idea of building a multi-label classification library on top of an existing classification solution. In the case of Python, the obvious choice for a base library is the scipy stack with scikit-learn. The primary goal of scikit-multilearn is to provide an efficient Python implementation of as many multi-label classification algorithms as possible, both to the open source community and commercial users of the Python data science stack of scikit/scipy.

The scikit-multilearn library extends the multi-label classification offering of scikit-learn by:

- extending the family of classifiers available to the Python community with more advanced algorithm adaptation and problem transformation (label space division) approaches.
- implementing multi-label embedding-based classification methods (Szymański & Kajdanowicz, 2017)

We have used Sickit-multilearn to import the techniques and the methods used for multi-label classification like: Problem adaptation technique using the binary relevance method, Problem transformation technique using MLkNN and the technique of ensemble using the Classifier Chain method.

. Keras:

Keras is a library that provides highly powerful and abstract building blocks to build deep learning networks. The building blocks Keras provides are built using Theano (which is used for manipulating and evaluating mathematical expressions) as well as TensorFlow (which is an alternative to Theano for building computational graphs, automatically deriving gradients, etc.). Keras supports both CPU and GPU computation and is a great tool for quickly prototyping ideas (Ketkar, 2017)

Keras has the following key features:

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine. [24]
- We used Keras for building our deep neural network model that is based on binary relevance

2.4. Development environment used:

2.4.1. Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large scale data pre-processing , predictive analytics..etc.),that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and 'Travis Oliphant' in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, which are both not free.[25]

2.4.2. Spyder:

Spyder is a powerful scientific open source cross-platform development environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of

[24] Kearas Definition from (Kears.studio.com) link
https://keras.rstudio.com/
[25] Difinition of anaconda from Wikipedia link
https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)

a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.[26]

2.4.3. Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more. [27]

2.4.4. Sublime Text:

Sublime Text editor is a sophisticated cross-platform source code editor which is widely used among developers. It includes wide features such as Syntax Highlight, Auto Indentation, File Type Recognition, Sidebar, Macros, Plug-in and Packages that make it easy for working with code base. .[28]

3. Models Evaluation:

In our work we have used 8 datasets to train our models, the datasets have the same features, however they differ in the number of samples (instances) and labels.

4 of them have 10 labels with 30k, 50k, 75k, 100k samples datasets (where k means one thousand)

And the other 4 have 15 labels, with 30k, 50k, 75k, 100k samples as well.

We trained all of our models with all of the 8 datasets, and evaluated them with the following evaluation metrics: Accuracy, Hamming Loss, Precision Micro, Precision Macro, Recall Micro,

[26] Spyder Environment definition and features from https://www.spyder-ide.org/
[27] Jupyter Definition and Features from https://jupyter.org/
[28] Sublime text environment definition & features from https://www.sublimetext.com/

Recall Macro, F1 score Micro, F1 score macro, Subset accuracy (Exact Match) and finally we evaluated the running time of each model with each dataset.

3.1. Models evaluation results:

The following tables show the obtained results

3.1.1. Binary Relevance-Random Forest:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
30000 users	80.1	96.44	2.49	97.90	98.14	98.02	98.18	98.25	98,22	415.08
50000 users	83.12	97.02	2.081	98.28	98.39	98.33	98.51	98.52	98.52	886.03
75000 users	85.33	97.36	1.8067	98.51	98.61	98.56	98.68	98.72	98.71	1548.54
100000 users	86,59	97,70	1.583	98.68	98.76	98.73	98.83	98.90	98.87	2039.19
Dataset with 15 labels										
30000 users	72.66	95.99	2.59	97.94	97.95	97.93	98.08	98.02	98.05	731.95
50000 users	75.48	96.42	2.29	98.12	98.22	98.17	98.25	98.30	98.27	1447.44
75000 users	77.1	96.68	2.10	98.23	98.41	98.32	98.35	98.47	98.41	2322.09
100000 users	80.27	97.15	1.79	98.50	98.61	98.55	98.61	98.68	98.65	2093.19

Table 1: Binary Relevance Random Forest model's Results

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
30000 users	80.1	96.44	2.49	97.90	98.14	98.02	98.18	98.25	98,22	415.08

50000 users	83.12	97.02	2.081	98.28	98.39	98.33	98.51	98.52	98.52	886.03
75000 users	85.33	97.36	1.8067	98.51	98.61	98.56	98.68	98.72	98.71	1548.54
100000 users	86,59	97,70	1.583	98.68	98.76	98.73	98.83	98.90	98.87	2039.19
Dataset with 15 labels										
30000 users	72.66	95.99	2.59	97.94	97.95	97.93	98.08	98.02	98.05	731.95
50000 users	75.48	96.42	2.29	98.12	98.22	98.17	98.25	98.30	98.27	1447.44
75000 users	77.1	96.68	2.10	98.23	98.41	98.32	98.35	98.47	98.41	2322.09
100000 users	80.27	97.15	1.79	98.50	98.61	98.55	98.61	98.68	98.65	2093.19

Table IV-2: Binary Relevance Random Forest model's Results

3.1.2. Binary Relevance-Decision Tree:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
30000 users	91.93	98.56	0.94	99.21	99.28	99.25	99.29	99.35	99.32	7.50
50000 users	94.2	98.97	0.66	99.49	99.44	99.47	99.55	99.51	99.53	16.14
75000 users	95.43	99.20	0.52	99.56	99.62	99.58	99.50	99.65	99.63	20.15
100000 users	95.92	99.29	0.46	99.62	99.65	99.63	99.65	99.68	99.67	27.09
Dataset with 15 labels										
30000 users	88.58	98.33	1.01	99.19	99.18	99.18	99.24	99.24	99.24	11.45
50000 users	91.51	98.80	0.73	99.43	99.39	99.41	99.47	99.43	99.45	20.27

75000 users	92.71	98.96	0.62	99.48	99.52	99.50	99.51	99.56	99.53	33.37
100000 users	93.94	99.14	0.51	99.59	99.60	99.59	99.62	99.62	99.62	44.96

Table IV-3: Binary Relevance Decision Tree model's results

3.1.3. Classifier Chain-Decision Tree:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
30000 users	92.06	98.53	0.955	99.19	99.31	99.24	99.26	99.37	99.32	6.55
50000 users	94.33	98.95	0.681	99.40	99.50	99.45	99.47	99.56	99.51	10.25
75000 users	94.96	99.06	0.606	99.53	99.50	99.51	99.58	99.55	99.57	15.89
100000 users	95.725	99.20	0.509	99.59	99.61	99.60	99.63	99.64	99.64	21.67
Dataset with 15 labels										
30000 users	88.47	98.22	1.05	99.21	99.07	99.14	99.27	99.14	99.20	10.96
50000 users	91.6	98.75	0.76	99.36	99.42	99.39	99.40	99.45	99.42	17.65
75000 users	93.2	98.91	0.64	99.46	99.50	99.49	99.50	99.54	99.52	331.56
100000 users	94.16	99.07	0.56	99.55	99.55	99.55	99.58	99.58	99.58	43.128

Table IV-4: Classifier Chain Decision Tree model's results

3.1.4. Classifier Chain-Random Forest:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score	Recall (%) (micro)	Precision (%) (micro)	F1-score	Duration (seconds)

						(%) (macro)			(%) (micro)	
Dataset with 10 labels										
30000 users	77.33	95.82	2.94	97.47	97.85	97.65	97.79	97.98	97.86	221.09
50000 users	81.35	96.68	2.30	97.98	98.37	98.17	98.23	98.47	98.35	470.23
75000 users	83.31	96.98	2.09	98.23	98.42	98.32	98.45	98.55	98.50	824.44
100000 users	85.14	97.34	1.85	98.45	98.61	98.53	98.63	98.73	98.68	1222.98
Dataset with 15 labels										
30000 users	68.99	95.19	3.1502	97.44	97.5	97.45	97.60	97.64	97.62	492.03
50000 users	72.91	95.80	2.742	97.80	97.8	97.82	97.94	97.94	97,94	1085.44
75000 users	75,32	96,36	2.3507	98.07	98.1	98.12	98.17	98.26	98.22	1867.45
100000 users	77.39	96.71	2.121	98.15	98.48	98.32	98.24	98.55	98.40	2715.13

Table IV-5: Classifier Chain Random Forest model's results.

3.1.5. Multi-label k Nearest Neighbors:

	Subset accuracy (%)	**Accuracy (%)**	**Hamming loss (%)**	**Recall (%) (macro)**	**Precision (%) (macro)**	**F1-score (%) (macro)**	**Recall (%) (micro)**	**Precision (%) (micro)**	**F1-score (%) (micro)**	**Duration (seconds)**
Dataset with 10 labels										
30000 users	33.33	83.14	12.82	87.33	91.97	89.47	88.83	92.47	90.61	275,82

50000 users	35.46	83.87	12.30	88.06	91.97	89.88	89.47	92.62	91.02	856,80
75000 users	35.59	84.10	12.11	87.84	92.48	90.00	89.26	93.10	91.14	1471.15
100000 users	36.61	84.35	11.92	87.95	92.69	90.17	89.34	93.30	91.28	2342.94
Dataset with 15 labels										
30000 users	23.92	81.02	13.73	86.35	91.26	88.63	87.29	91.59	89.39	178.21
50000 users	25.29	81.69	13.23	87.03	91.37	89.05	88	91.67	89.80	668.57
75000 users	26.15	82.15	12.86	86.96	92.08	89.35	87.93	92.33	90.07	1315.75
100000 users	27.28	82.53	12.57	87.23	92.25	89.57	88.20	92.49	90.29	2229.81

Table IV-6: Multi-label K Nearest Neighbors model's results.

3.1.6. Binary Relevance-Neural Network:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
30000 users	74.2	95.39	3.27	97.11	97.66	97.38	97.41	97.86	97.64	85.22
50000 users	78.08	96.25	2.67	97.43	98.29	97.86	97.67	98.50	98.08	137.56

75000 users	80.86	96.68	2.35	97.77	98.49	98.09	97.96	98.65	98.30	212.297
100000 users	83.55	97.19	1.99	98.45	98.34	98.39	98.61	98.53	98.57	181.92
Dataset with 15 labels										
30000 users	63.85	94.45	3.67	96.75	97.36	97.04	96.90	97.53	97.21	135.42
50000 users	68.55	95.57	2.91	97.11	98.19	97.65	97.273	98.31	97.80	209.24
75000 users	74.06	96.33	2.38	97.56	98.59	98.07	97.77	98.66	98.19	324.16
100000 users	76.8	96.64	2.24	98.30	98.09	98.19	98.44	98.19	98.31	397.58

Table 7: Binary Relevance Neural Network model's results

3.1.7. Models comparison with five-fold Cross-Validation:

	Subset accuracy (%)	Accuracy (%)	Hamming loss (%)	Recall (%) (macro)	Precision (%) (macro)	F1-score (%) (macro)	Recall (%) (micro)	Precision (%) (micro)	F1-score (%) (micro)	Duration (seconds)
Dataset with 10 labels										
BR_dc	95.42	99.2	0.52	99.6	99.6	99.5	99.6	99.6	97	241.42
BR_RF	86.57	97.64	1.62	98.68	98.73	98.70	98.85	98.83	98.84	5133.83
BR_NN	83.49	97.12	1.99	98.12	98.66	98.39	98.80	98.35	98.57	969.78
CC_DC	95.8	99.2	0.50	99.59	99.61	99.60	99.6	99.64	99.63	109.10

CC_RF	85.73	97.48	1.75	98.48	98.73	98.60	98.85	98.65	98.75	6063.25
MLKNN	84.24	12.04	87.92	93.18	90.1	89.32	92.57	91.21	5565.14	84.24
Dataset with 15 labels										
BR_dc	93.2	99.02	0.58	99.52	99.54	99.53	99.56	99.55	99.56	401.81
BR_RF	85.43	97.51	1.71	98.47	98.81	98.64	98.90	98.65	98.77	6899.87
BR_NN	77.38	96.83	2.08	98.30	98.33	98.31	98.43	98.44	98.43	1504.18
CC_DC	94.1	99.1	0.57	99.5	99.6	99.6	99.6	99.7	99.5	177.63
CC_RF	78.76	96.87	2.02	98.25	98.52	98.38	98.60	98.35	98.47	10513.37
MLKNN	26.57	82.28	12.82	87.01	92.14	89.39	87.98	92.38	90.13	5730.15

Table 8: Models Comparison with five-fold Cross-Validation.

3.2. Models Evaluation Discussion:

In this section, we will discuss and compare the models with each other using the evaluation metrics mentioned previously with all of the eight different datasets:

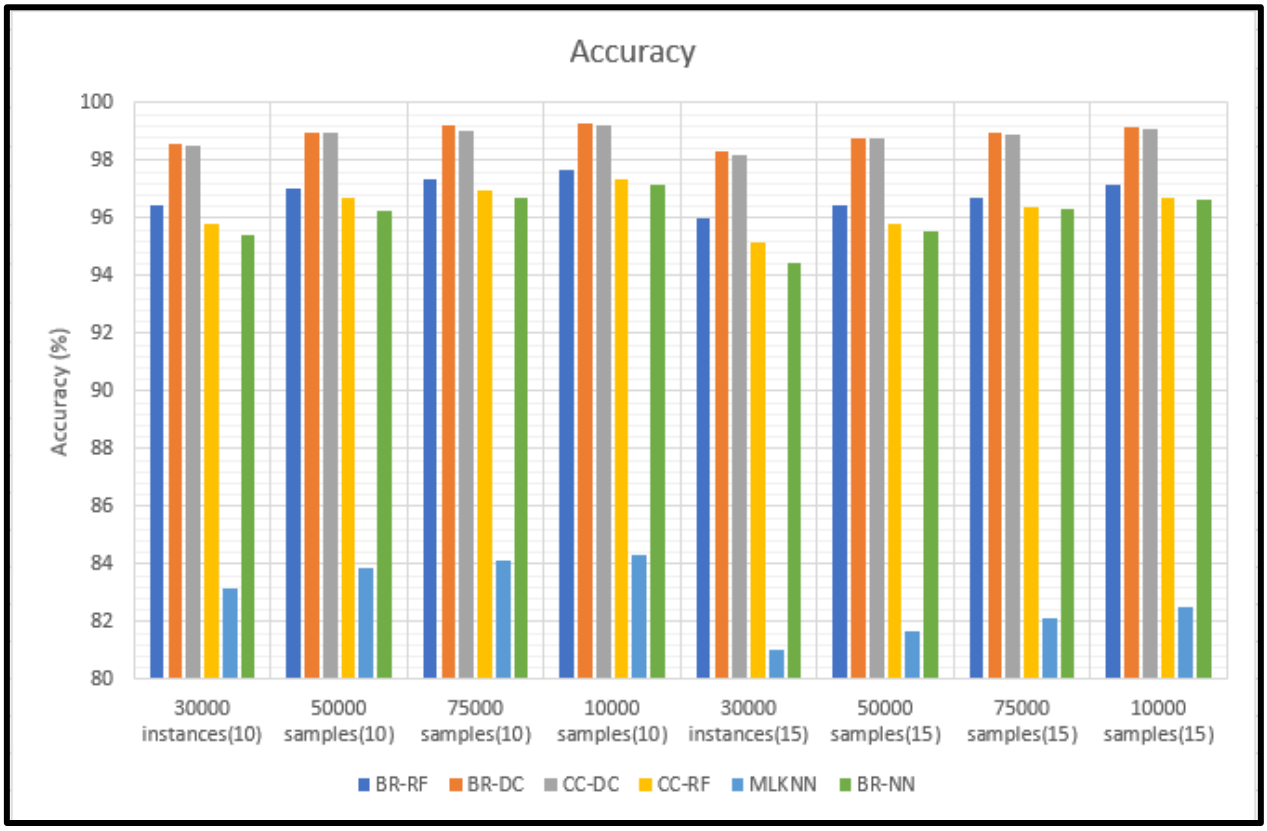


Figure IV-1: Evaluating all the models and datasets used using accuracy (The higher the better).

- BR-DC was the best model in terms of accuracy (from 98.33% to 99.29% depending on the dataset), slightly ahead of CC-DC, and they are followed by BR-RF then CC-RF then BR-NN and finally in the last place we find MLkNN (from 81.02% to 84.35%). This applies to all the datasets.
- When the number of samples increases, the model trains better, hence, we can clearly see that there are some improvements when increasing in the numbers of samples.
- The 10 labels datasets performed slightly better than the 15 labels ones.

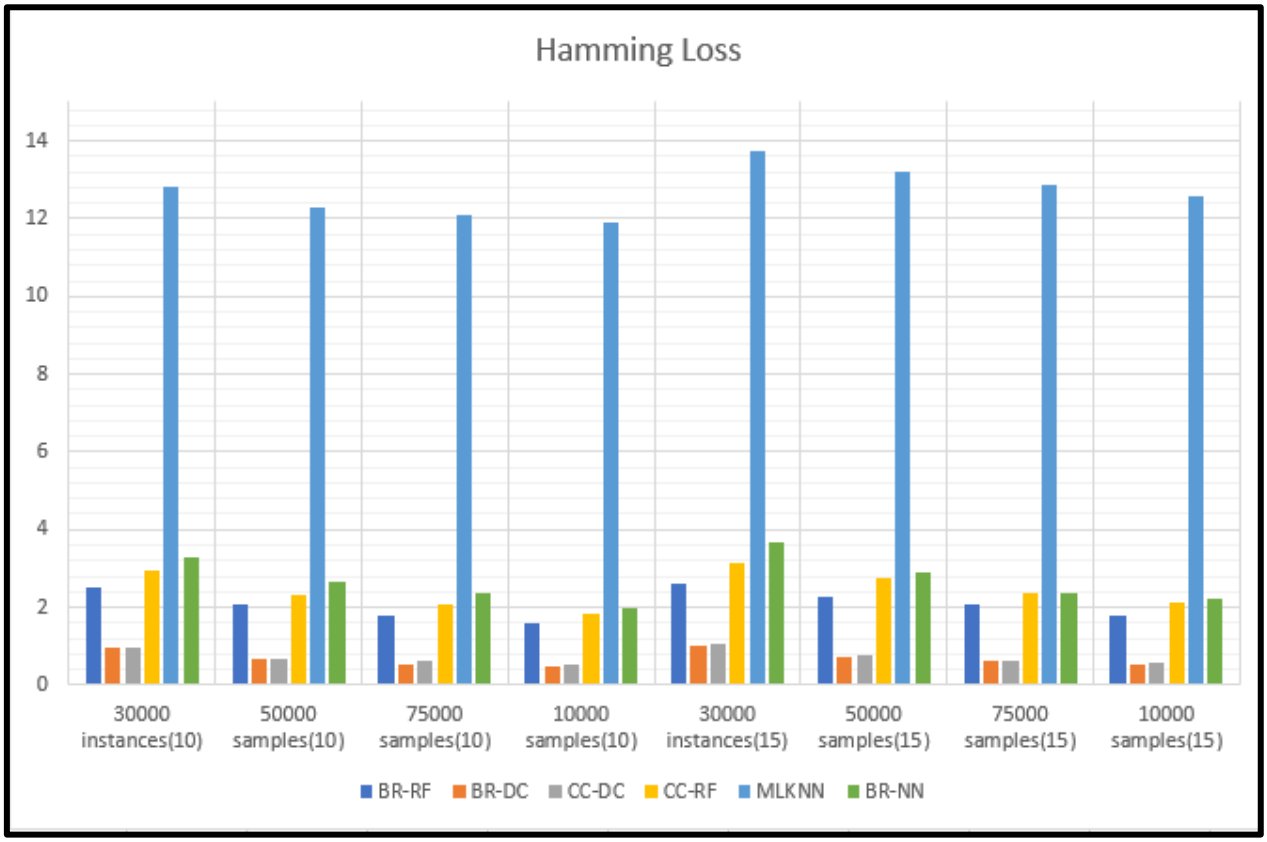


Figure IV-2: Evaluating all the models and datasets used using Hamming Loss (the lower the better).

From (figure IV-2) we can extract three main pieces of information:

- BR-DC was the best model in terms of Hamming Loss (from 0.46% to 1.01% depending on the dataset), slightly ahead of CC-DC, and they are followed by BR-RF then CC-RF then BR-NN and finally in the last place we find MLkNN (from 11.92% to 13.73%). This applies to all the datasets.
- When the number of samples increases, the model trains better, hence, we can clearly see that there are some improvements when increasing in the numbers of samples.
- The 10 labels datasets performed slightly better than the 15 labels ones.

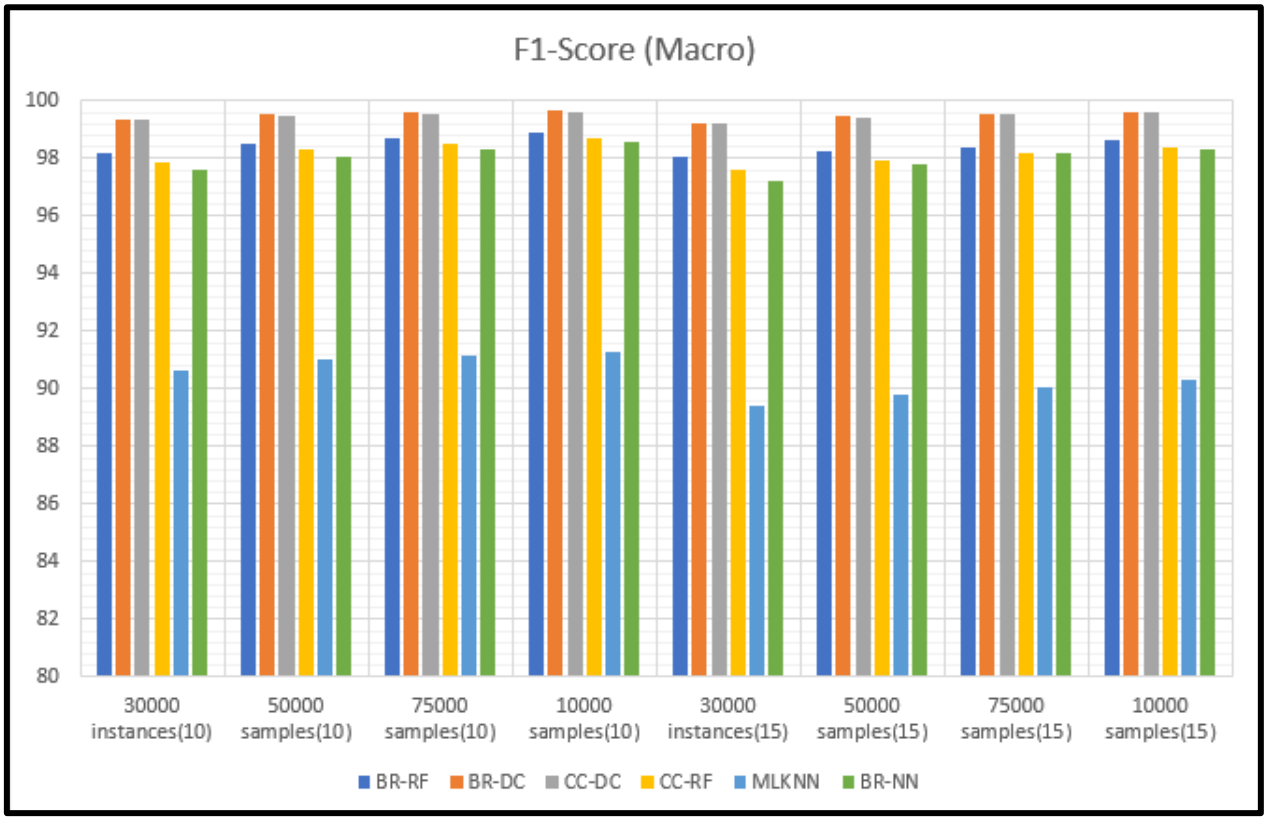


Figure IV-3: Evaluating all the models and datasets used using F1-score Macro (the higher the better)

From (figure IV-3) we can extract three main pieces of information:

- BR-DC was the best model in terms of F1-score Macro (from 99.18% to 99.63% depending on the dataset), slightly ahead of CC-DC, and they are followed by BR-RF then CC-RF then BR-NN and finally in the last place we find MLkNN (from 88.63% to 90.17%). This applies to all the datasets
- When the number of samples increases, the model trains better, hence, we can clearly see that there are some improvements when increasing in the numbers of samples.
- The 10 labels datasets performed slightly better than the 15 labels ones.

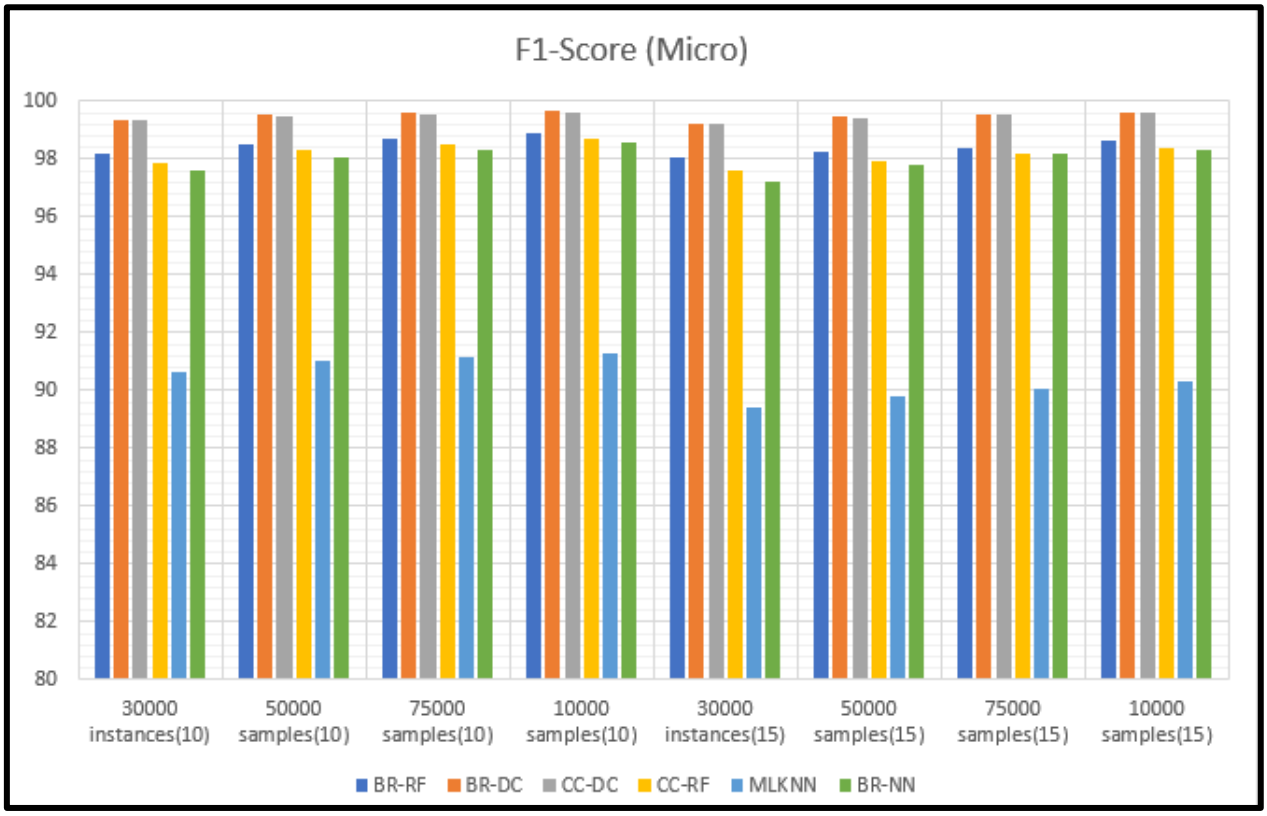


Figure IV-4: Evaluating all the models and datasets used using F1-score Micro (the higher the better).

According to (figure IV-4), the extracted pieces of information are the same extracted from (figure IV-3) (F1-score Macro), only that the results here are slightly better in the micro averaging, the slight improvements are applied to all the models.

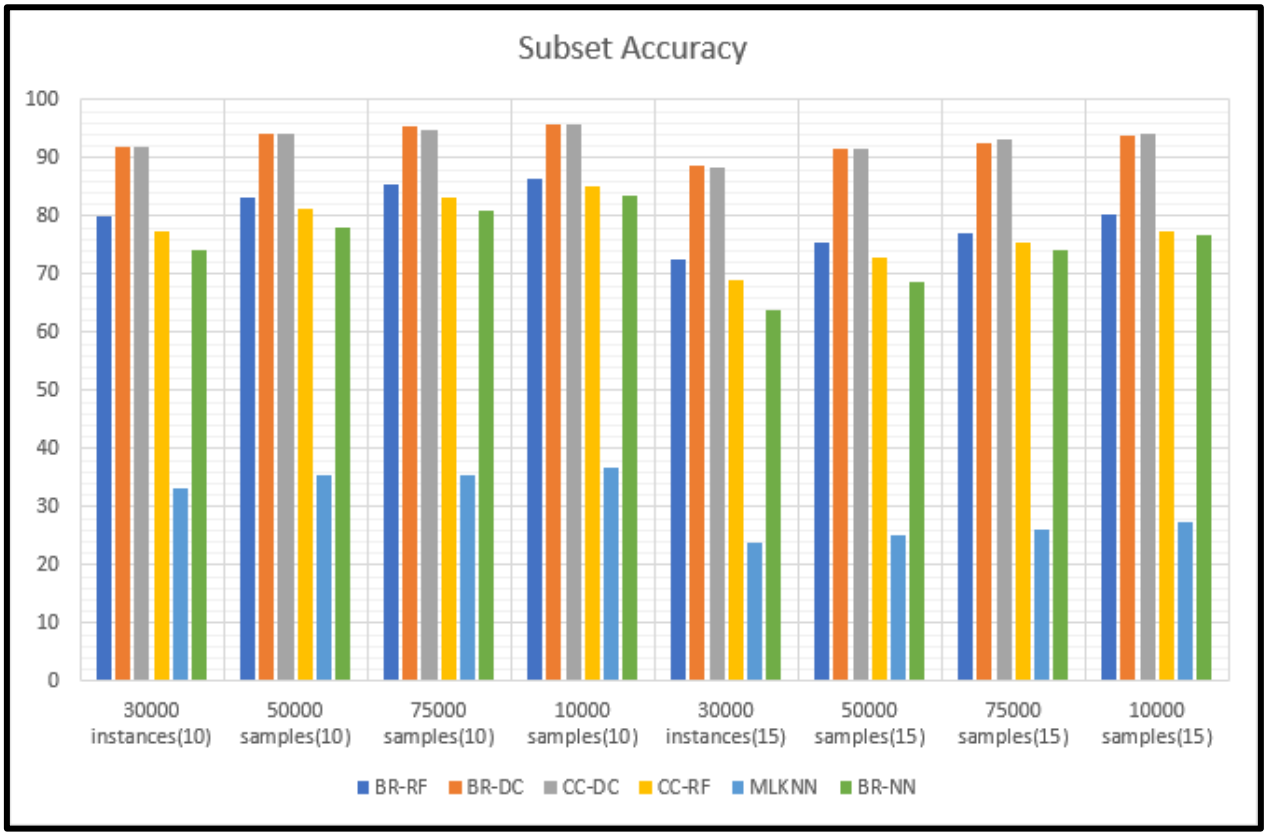


Figure IV-5: Evaluating all the models and datasets used using Subset accuracy (the higher the better).

From (figure IV-4), the main pieces of information extracted are:

- The best model in terms of subset accuracy was sometimes taken BR-DC and sometimes by BR-CC, always the difference between those two models is incredibly small. In the 10 labels datasets, BR-CC performed slightly better in both 30k and 50k datasets, however, BR-DC performed slightly better in the 75k and 100k datasets.

  In the 15 labels datasets BR-DC was only slightly better in the 30k samples dataset. For the 50k, 75k and 100k samples datasets BR-CC was slightly better.

  The results of BR-DC were in the range of 88.58% to 95.92% depending on the dataset, and the results of CC-DC were in the range of 88.47 % to 95.725%....These two models were always the best in the subset accuracy for all the datasets tested, followed by BR-RF then CC-RF then BR-NN and finally in the last place we find MLkNN (from 23.92% to 36.61%). This applies to all the datasets.

- When the number of samples increases, the model trains better, hence, we can clearly see that there are some improvements when increasing in the numbers of samples.
- The 10 labels datasets performed slightly better than the 15 labels ones.

We didn't plot the Macro and Micro Precision as well as the Macro and Micro Recall as we did with the previous plots because F1-score is actually the weighted average of both precision and recall, so both are taken into consideration when evaluating with F1-score. However, we will mention them in the next plots of Cross-Validation evaluations.

3.3. Evaluating with Cross-Validation:

In this section we will compare all the models we created using all the metrics mentioned in this chapter using 5-fold Cross Validation, however, we will only apply this to the 100k samples datasets of both 10 and 15 labels.

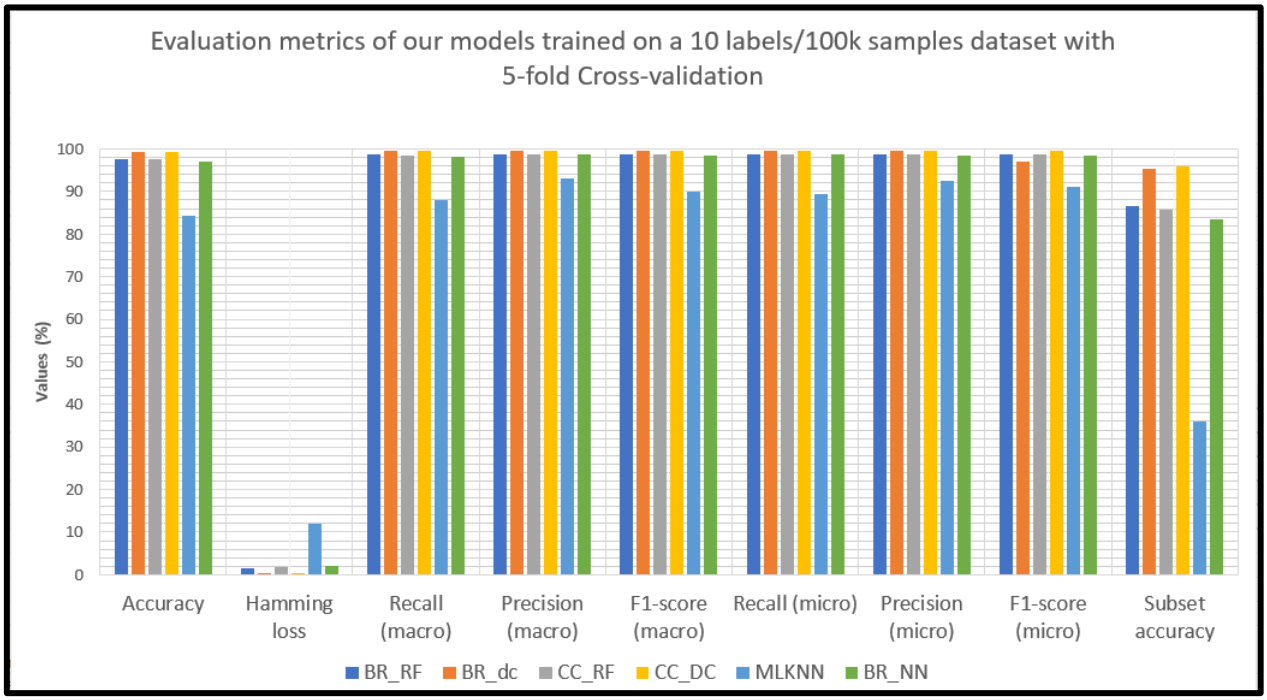


Figure IV-6: Comparing our models with each other using the evaluation metrics calculated through 5-fold Cross-Validation on a 10 labels/100k samples dataset.

From (figure IV-5), we extracted some important pieces of information:

- Applying 5-fold Cross-Validation to the metrics gave us almost identical results to those without 5-fold CV, only a slight difference more or less in the fractional parts of the values obtained showcased in percentages, which decreases the chances that the models are overfitting.
- In Accuracy BR-DC and CC-DC obtained the best results with 99.2% and 99.2% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 97.64%, 97.48% and 97.12% respectively, and finally MLkNN with an accuracy of 84.24%.
- In Hamming Loss CC-DC obtained the best results with 0.50%, slightly ahead of BR-DC with 0.52%, followed by BR-RF, CC-RF and BR-NN with 1.62%, 1.75% and 1.99% respectively, and finally MLkNN with a Hamming Loss of 12.04%.
- In Recall Macro BR-DC and CC-DC obtained the best results with 99.6% and 99.59% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.68%, 98.48% and 98.12% respectively, and finally MLkNN with a Recall Macro of 87.92%.
- In Precision Macro CC-DC and BR-DC obtained the best results with 99.61% and 99.6% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.73%, 98.73% and 98.66% respectively, and finally MLkNN with a Precision Macro of 93.18%.
- In F1-score Macro CC-DC and BR-DC obtained the best results with 99.6% and 99.5% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.7%, 98.6% and 98.39% respectively, and finally MLkNN with F1-score Macro of 90.1%.
- In Recall Micro CC-DC and BR-DC obtained the best results with 99.6% and 99.6% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.85%, 98.85% and 98.80% respectively, and finally MLkNN with a Recall Micro of 89.32%.
- In Precision Micro CC-DC and BR-DC obtained the best results with 99.64% and 99.6% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.83%, 98.65% and 98.35% respectively, and finally MLkNN with a Precision Micro of 92.57%.
- In F1-score Micro, CC-DC obtained the best results with 99.63%, slightly ahead of BR-RF with 98.84%, followed by CC-RF, BR-NN and BR-DC with 98.75%, 98.57% and 97% respectively, and finally MLkNN with a F1-Score Micro of 91.21%.

- In subset accuracy, CC-DC obtained the best results with 95.8%, slightly ahead of BR-DC with 95.42%, followed by BR-RF, CC-RF and BR-NN with 86.57%, 85.73% and 83.49% respectively, and finally MLkNN with a subset accuracy of 35.95%.

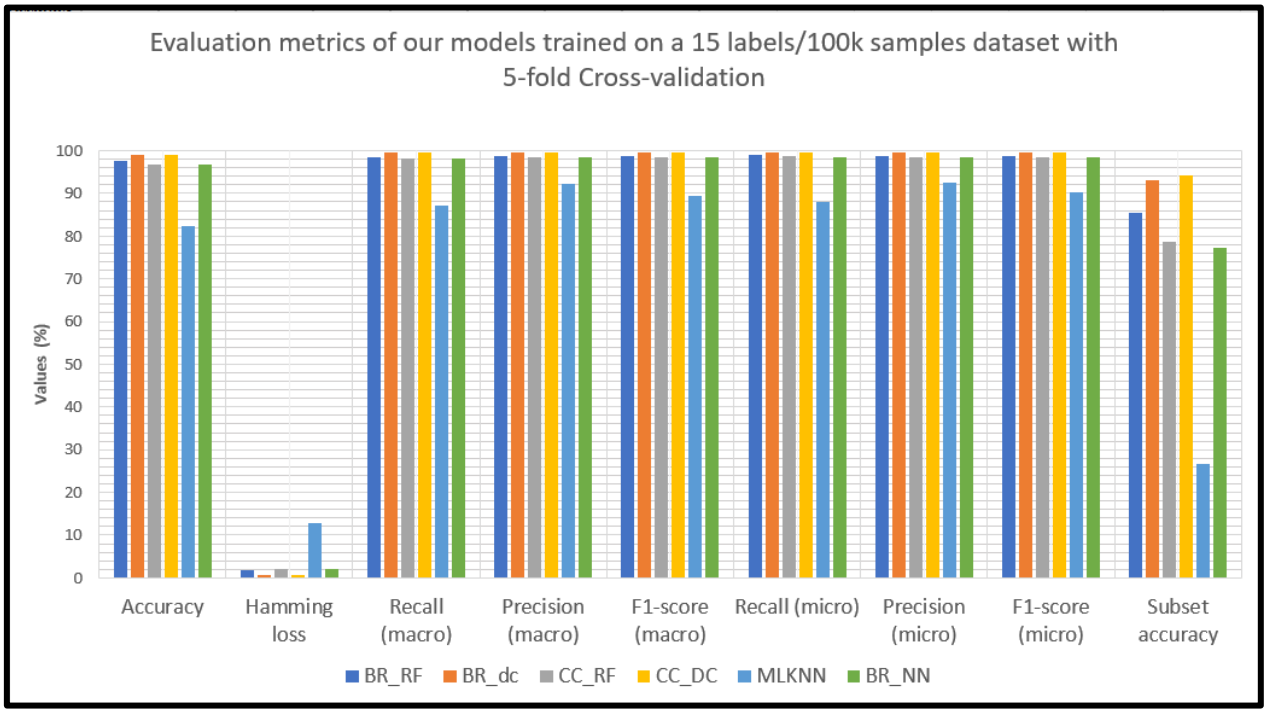


Figure IV-7: Comparing our models with each other using the evaluation metrics calculated through 5-fold Cross-Validation on a 15 labels/100k samples dataset.

From (figure IV-5), we extracted some important pieces of information:

Same as with 10 labels, applying 5-fold Cross-Validation to the metrics with a dataset of 15 labels gave us almost identical results to those without 5-fold CV, only a slight difference more or less in the fractional parts of the values obtained showcased in percentages, which decreases the chances that the models are overfitting.

- In Accuracy CC-DC and BR-DC obtained the best results with 99.1% and 99.02% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 97.51%, 96.87% and 96.83% respectively, and finally MLkNN with an accuracy of 82.28%.

- In Hamming Loss CC-DC and BR-DC obtained the best results with 0.57% and 0.58% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 1.71%, 2.02% and 2.08% respectively, and finally MLkNN with a Recall Macro of 12.82%.
- In Recall Macro BR-DC and CC-DC obtained the best results with 99.52% and 99.5% respectively, slightly ahead of BR-RF, BR-NN and CC-RF with 98.47%, 98.30% and 98.25% respectively, and finally MLkNN with a Recall Macro of 87.01%.
- In Precision Macro CC-DC and BR-DC obtained the best results with 99.6% and 99.54% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.81%, 98.52% and 98.33% respectively, and finally MLkNN with a Precision Macro of 92.14%.
- In F1-score Macro CC-DC and BR-DC obtained the best results with 99.6% and 99.53% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.64%, 98.38% and 98.31% respectively, and finally MLkNN with F1-score Macro of 89.39%.
- In Recall Micro CC-DC and BR-DC obtained the best results with 99.6% and 99,56% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.90%, 98.60% and 98.43% respectively, and finally MLkNN with a Precision Micro of 87.98%.
- In Precision Micro, CC-DC obtained the best results with 99.7%, slightly ahead of BR-DC with 99.55%, followed by BR-RF, BR-NN and CC-RF with 98.65%, 98.44% and 98.35% respectively, and finally MLkNN with a Precision Micro of 92.38%.
- In F1-score Micro BR-DC and CC-DC obtained the best results with 99.56% and 99.5% respectively, slightly ahead of BR-RF, CC-RF and BR-NN with 98.77%, 98.47% and 98.43% respectively, and finally MLkNN with a F1-score Micro of 90.13%.
- In subset accuracy, CC-DC obtained the best results with 94.1%, slightly ahead of BR-DC with 93.2%, followed by BR-RF, CC-RF and BR-NN with 85.43%, 78.76% and 77.38% respectively, and finally MLkNN with a subset accuracy of 26.57%.

According to (figure IV-5) and (figure IV-6), there is an important note to consider, the number of labels didn't affect the results, 10 and 15 labels datasets showed almost same results with only a very tiny difference. Except for the subset accuracy where we can see a small difference. BR-DC, BR-RF and CC-DC were only reduced by 1% to 2% in subset accuracy, on the other hand BR-NN, CC-RF and MLkNN were reduced from 6 to 9%.

For the rest of the metrics, the 15 labels dataset experienced a very tiny reduction in the values obtained; most of the change only affected the fractional part.

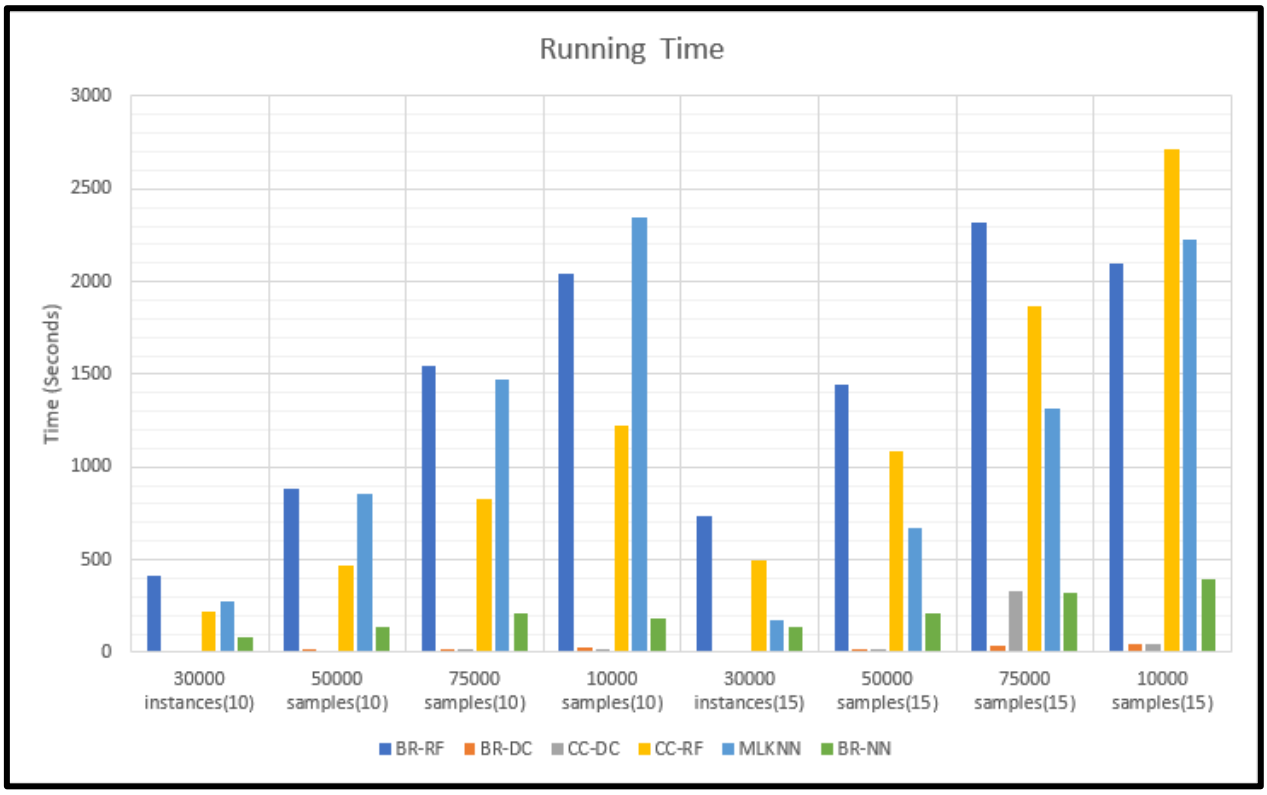


Figure IV-8: A plot illustrating the running time of all datasets with all models (without k-fold CV) (the lower the better).

Some important pieces of information extracted from (figure IV-7):

- In the 10 labels/30000 samples dataset, the best running time was obtained by CC-DC with 6.55 seconds slightly ahead of BR-DC with 7.50 seconds, followed by BR-NN, CC-RF and MLkNN with 52.22 seconds, 221.09 seconds and 275.82 seconds respectively, and in the last place BR-RF comes with 415.08 seconds.
- In the 10 labels/50000 samples dataset, the best running time was obtained by CC-DC with 10.25 seconds slightly ahead of BR-DC with 16.14 seconds, followed by BR-NN, CC-RF and MLkNN with 137.56 seconds, 470.23 seconds and 856.80 seconds respectively, and in the last place BR-RF comes with 886.03 seconds.

- In the 10 labels/75000 samples dataset, the best running time was obtained by CC-DC with 15.89 seconds slightly ahead of BR-DC with 20.15 seconds, followed by BR-NN, CC-RF and MLkNN with 212.297 seconds, 824,44 seconds and 1471.15 seconds respectively, and in the last place BR-RF comes with 1548.54 seconds.
- In the 10 labels/100000 samples dataset, the best running time was obtained by CC-DC with 21.67 seconds slightly ahead of BR-DC with 27.09 seconds, followed by BR-NN, CC-RF and BR-RF with 181,92 seconds, 1222.98 seconds and 2039.19 seconds respectively, and in the last place MLkNN comes with 2342.94 seconds.
- In the 15 labels/30000 samples dataset, the best running time was obtained by CC-DC with 10.96 seconds slightly ahead of BR-DC with 11.45 seconds, followed by BR-NN, MLkNN and CC-RF with 135.42 seconds, 178.21seconds and 492,03seconds respectively, and in the last place BR-RF comes with 731.95 seconds.
- In the 15 labels/50000 samples dataset, the best running time was obtained by CC-DC with 17.65 seconds slightly ahead of BR-DC with 20.27seconds, followed by BR-NN, MLkNN and CC-RF with 209.24 seconds, 668.57 seconds and 1085.44 seconds respectively, and in the last place BR-RF comes with 1447.44 seconds.
- In the 15 labels/75000 samples dataset, the best running time was obtained by BR-DC with 33.37 seconds then CC-DC with 331,56 seconds, followed by BR-NN, MLkNN and CC-RF with 324.16 seconds, 1315.75 seconds and 1867,45 seconds respectively, and in the last place BR-RF comes with 2322.09 seconds.
- In the 15 labels/100000 samples dataset, the best running time was obtained by CC-DC with 43.128 seconds slightly ahead of BR-DC with 44.96 seconds, followed by BR-NN, BR-RF and MLkNN with 397.58 seconds, 2093.19 seconds and 2229.81 seconds respectively, and in the last place CC-RF comes with 2715.13 seconds.

So as we can see CC-DC and BR-DC are definitely the best models by far in terms of running time, all due to decision tree that delivered excellent performance.

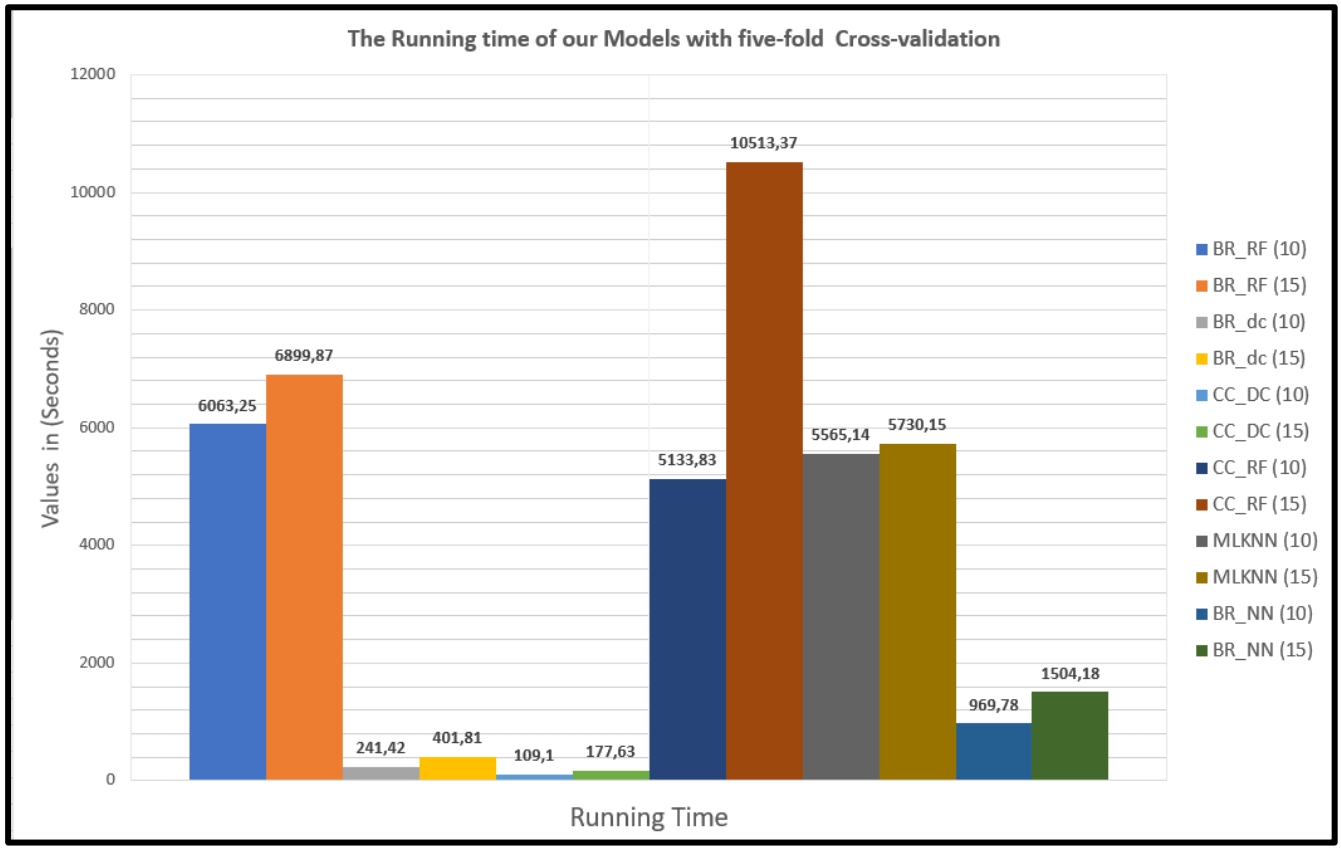


Figure IV-9: A plot illustrating the running time of the datasets with 100000 samples with all models with 5-fold CV (the lower the better).

From (figure IV-5), we can conclude two important pieces of information: the rank of the models in terms of running time and the effects of increasing the number labels on the running time.

The rank of running time with 5-fold CV for 100000 samples datasets with both 10 and 15 labels, from best to worst:

- 1st CC-DC for 10 labels with 109.1 seconds.
- 2nd CC-DC for 15 labels with 177.63 seconds.
- 3rd BR-DC for 10 labels with 241.42 seconds.
- 4th BR-DC for 15 labels with 401.81 seconds.
- 5th BR-NN for 10 labels with 969.78 seconds
- 6th BR-NN for 15 labels with 1054.18 seconds
- 7th BR-RF for 10 labels with 5133.83 seconds

- 8th MLkNN for 10 labels with 5565.14 seconds
- 9th MLkNN for 15 labels with 5730.15 seconds.
- 10th CC-RF for 10 labels with 6063.25 seconds
- 11th BR-RF for 15 labels with 6899.87 seconds
- 12th CC-RF for 15 labels 10513.37 seconds

CC-DC and BR-DC still have the best running time even after using 5-fold CV

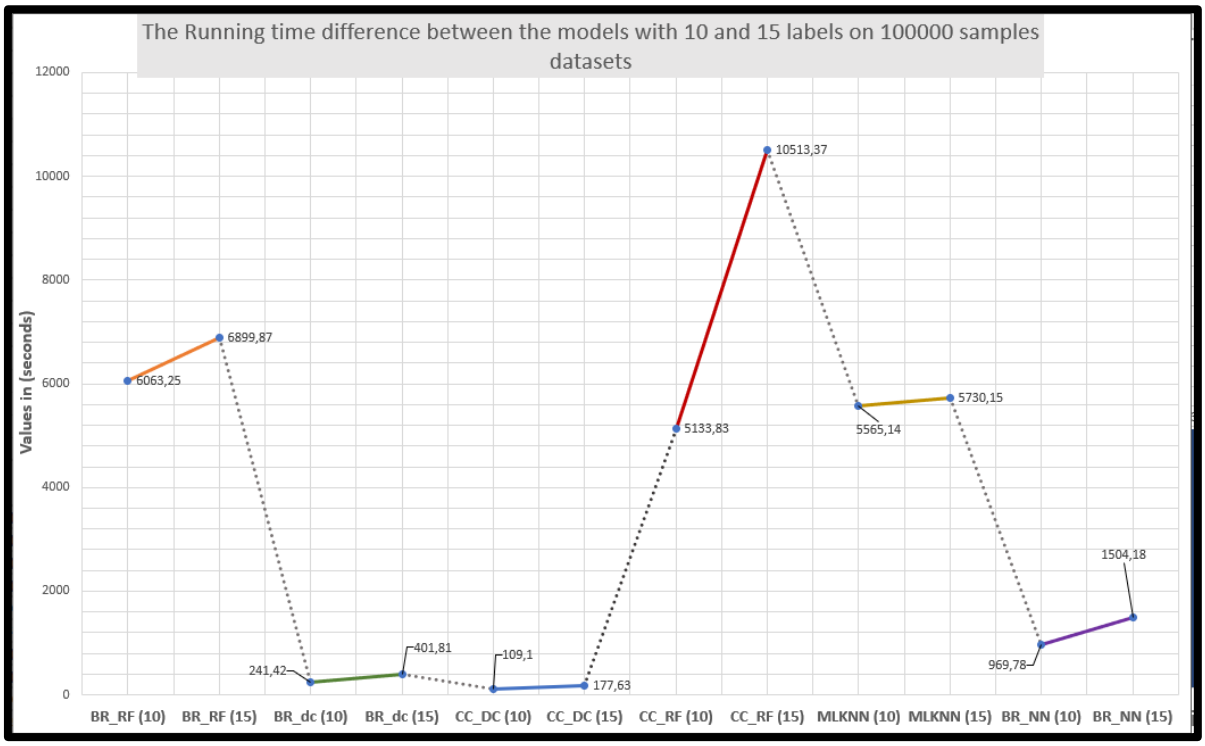


Figure IV-10: The Running difference between models with 10 and 15 labels on 100000 samples datasets.

The second conclusion from (figure IV-8), (figure IV-9) , (figure IV-10) is that increasing the number of labels results in an increase in the running time, some models were so affected like BR-DC, and others were not affected much like MLkNN. After examining the overall results of our models, we will go through a brief description of the interface we used to implement our models.

4. Interface:

We have used Django to create our application's interface designed as a website.

4.1. Django:

Django is a Python-based free and open-source web framework that follows the model-template-views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization established as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. [29]

[29] Django framework definition & features from Wikipedia link https://en.wikipedia.org/wiki/Django_(web_framework)

The application's graphical interface:

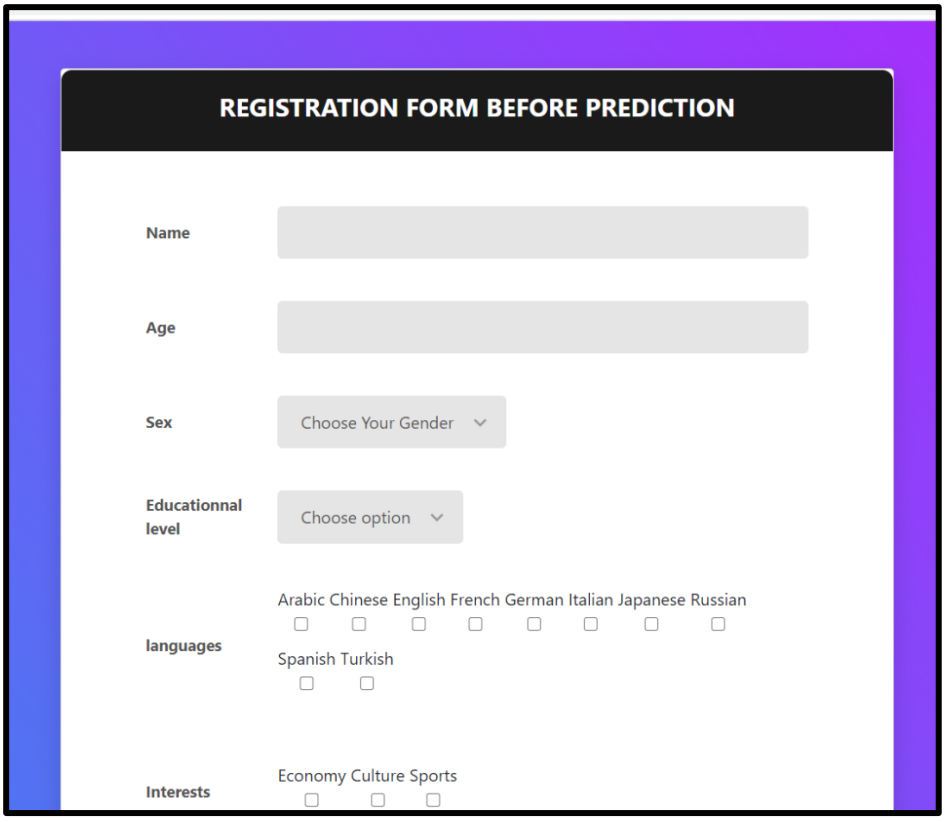


Figure IV-11: First part of the application form used for making the predictions

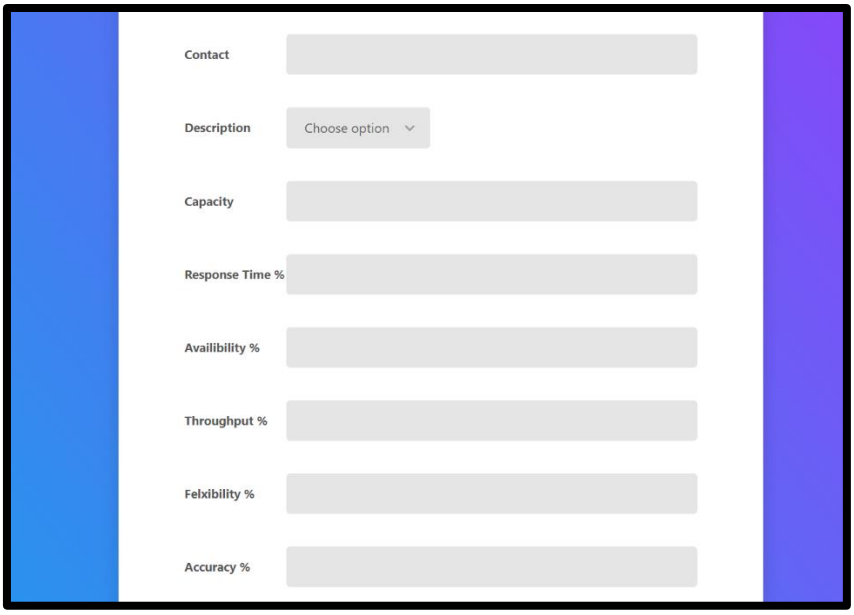


Figure IV-12: Second part of the application form used for making the predictions

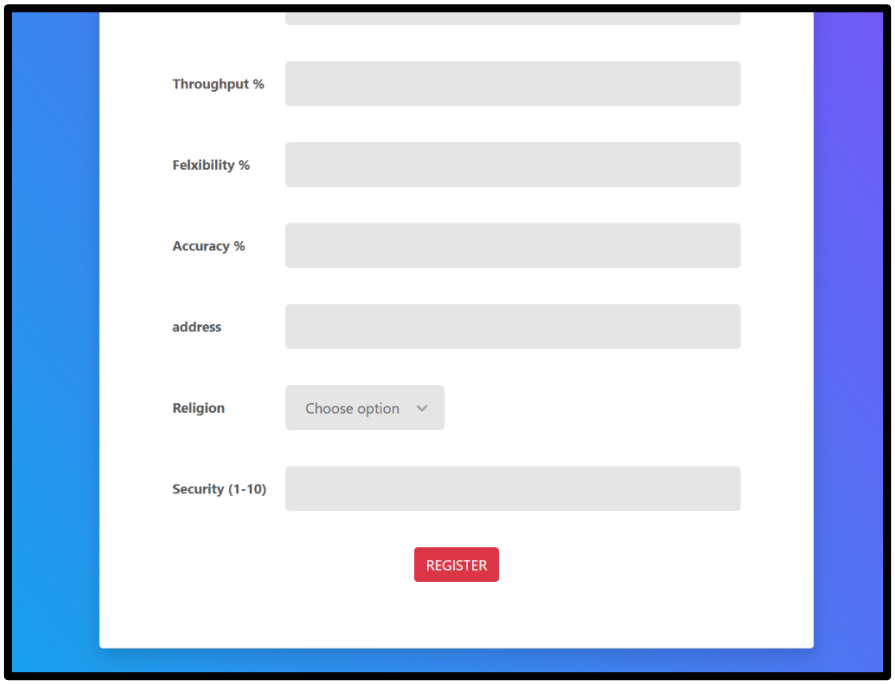


Figure IV-13: Third part of the application Form used for making the predictions

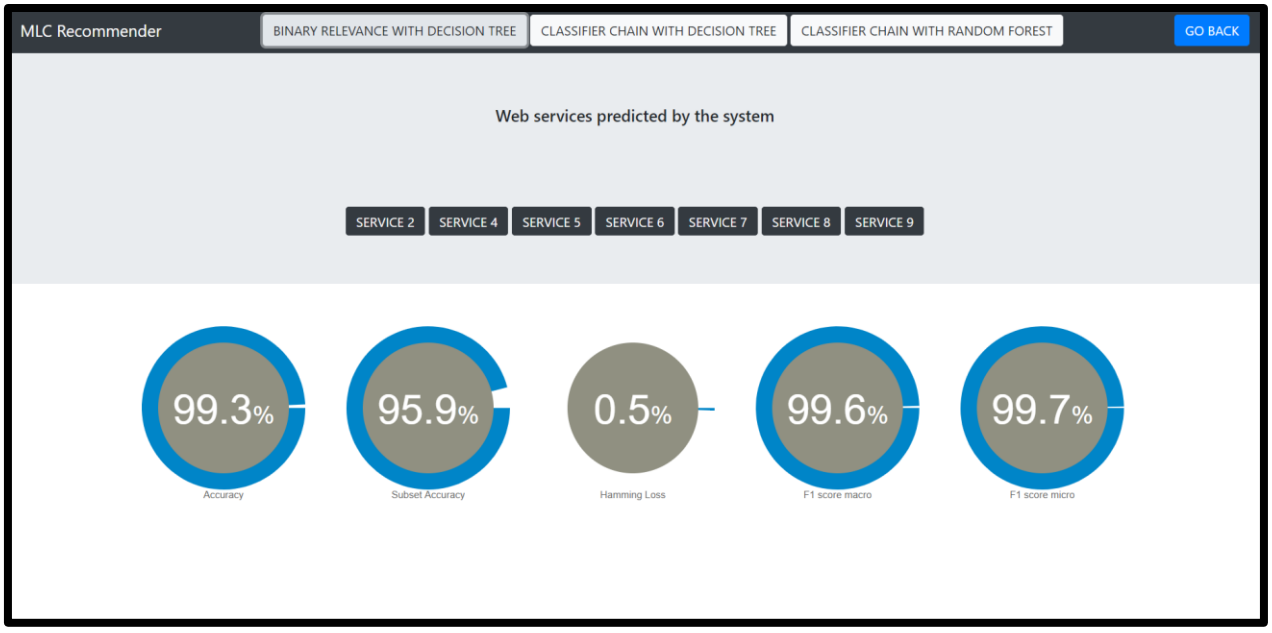


Figure IV-14: The predicated Web services with evaluation metrics measures

5. Conclusion:

In this chapter, we examined and evaluated the performance of our models using many evaluation metrics taking the running time in consideration.

Some models obtained excellent results, and others obtained decent results, so in general they were all good. CC-DC and BR-DC delivered best results in terms of both evaluation metrics and running time making them the most suitable models for solving our specific problem.

General Conclusion

Our work consists of applying a sophisticated classification task known as multi-label classification for recommending web services on the web. MLC can be approached through multiple techniques, methods and algorithms; we tried to cover the most suitable ones to our case who can deliver great results.

Our models were trained on multiple datasets who have profiles of users as samples, and web services as labels, the goal was to predict and suggest relevant services to new users based on their profiles. An MLC approach was a great solution as it can assign multiple classes to a single sample unlike single label classification tasks.

Our models were able to reach high efficiency in terms of performance being so accurate in making the predictions; however, there was a difference in performance between the models in terms of accurate predictions taking the running time in consideration.

The best models were Binary relevance with Decision Tree and Classifier Chains with Decision Tree, both obtained excellent results when assessed with evaluation metrics with a very small running time, making them the best for our case compared to other models we tested.

Learning from multi-label data was quite challenging but rather interesting, as it expands the use cases of classification, our approach concerns the field of recommender systems, using MLC for such a vast sophisticated field proves that it can be used for further contributions in the RS techniques, So our perspective for the future is to combine the well-known recommender systems techniques explained in chapter 1 with MLC for further improvements.

- **(Aggarwal, 2016)**

  Aggarwal, C. (2016). *Recommender Systems: The Textbook* (Springer International Publishing AG, Vol. 1).

- **(Arsanjani & Corporation, 2005)**

  Arsanjani, A., & Corporation, I. (2005). *Service-oriented modeling and architecture How to identify, specify, and realize services for your SOA*. 15.

- **(Asiri, 2018)**

  Asiri, s. (2018, June 11). *Machine Learning Classifiers.* Towards Data Science. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

- **(Bey, 2018)**

  Bey, F. (2018). *Composition des web services basée Temps* (thesis for Doctoral Degree) Université des Sciences et de la Technologie Houari Boumediène.

- **(Cabral et al.2011,)**

Cabral, R. S., Torre, F., Costeira, J. P., & Bernardino, A. (2011). Matrix completion for multi-label image classification. *Advances in Neural Information Processing Systems*, 190–198.

- **(Cerri et al., 2009)**

  Cerri, R., da Silva, R. R., & de Carvalho, A. C. (2009). *Comparing methods for multilabel classification of proteins using machine learning techniques. Brazilian Symposium on Bioinformatics*, 109–120.

- **(Chakur, 2019)**

  Chakur, A. (2019, June 6). *Decision Tree Classification (An introduction to Decision Tree Classifier)*. Towardsdatascience.

- **(Chauhan, 2019)**

Chauhan, N. S. (2019, October 3). *Introduction to Artificial Neural Networks(ANN)*. Https://Towardsdatascience.Com/. https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9

- **(Dai & Steele, 2005)**

  Dai, J., & Steele, R. (2005). UDDI access control. *Third International Conference on Information Technology and Applications (ICITA'05)*, *2*, 778–783 vol.2. https://doi.org/10.1109/ICITA.2005.291

- **(de Oliveira et al., 2013)**

  de Oliveira, M. G., Marques Ciarelli, P., & Oliveira, E. (2013). Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Systems with Applications*, *40*(16), 6641–6651. https://doi.org/10.1016/j.eswa.2013.06.011

- **(Deustschman, 2019)**

  Deustschman, Z. (2019, December 3). *Multi-Label Text Classification*. Towards Data Science. https://towardsdatascience.com/multi-label-text-classification-5c505fdedca8

- **(Exact Passive-Aggressive Algorithm for Multiclass Classification Using Support Class | Request PDF, n.d.)**

  Exact Passive-Aggressive Algorithm for Multiclass Classification Using Support Class | Request PDF. (n.d.). ResearchGate. Retrieved September 14, 2020, from https://www.researchgate.net/publication/220906661_Exact_Passive-Aggressive_Algorithm_for_Multiclass_Classification_Using_Support_Class

- **(Felfernig et al., 2019)**

  Felfernig, A., Polat-Erdeniz, S., Uran, C., Reiterer, S., Atas, M., Tran, T. N. T., Azzoni, P., Kiraly, C., & Dolui, K. (2019). An overview of recommender systems in the internet of things. *Journal of Intelligent Information Systems*, *52*(2), 285–309. https://doi.org/10.1007/s10844-018-0530-7

- **(Gandhi, 2019)**

Gandhi, R. (2019, June 12). *Naive Bayes Classifier*. Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2

- **(Goldberg et al., 1992)**

  Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70. https://doi.org/10.1145/138859.138867

- **(Gottschalk et al., 2002)**

  Gottschalk, K., Graham, S., Kreger, H., & Snell, J. (2002). Introduction to web services architecture. *IBM Systems Journal*, *41*(2), 170–177.

- **(Guo et al., 2019)**

  Guo, Y., Chung, F.-L., Li, G., & Zhang, L. (2019). Multi-Label Bioinformatics Data Classification With Ensemble Embedded Feature Selection. *IEEE Access*, *7*, 103863–103875. https://doi.org/10.1109/ACCESS.2019.2931035

- **(Herrera et al., 2016)**

  Herrera, F., Charte, F., Rivera, A. J., & Jesus, M. J. del. (2016). *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer International Publishing. https://doi.org/10.1007/978-3-319-41111-8

- **(Horak et al., 2017)**

  Horak, K., Klecka, J., Bostik, O., & Davidek, D. (2017). Classification of SURF image features by selected machine learning algorithms. *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, 636–641.

- **(Hybrid Web Recommender Systems | SpringerLink, n.d.)**

  Hybrid Web Recommender Systems | SpringerLink. (n.d.). Retrieved September 14, 2020, from https://link.springer.com/chapter/10.1007/978-3-540-72079-9_12

- **(Ketkar, 2017)**

  Ketkar, N. (2017). Introduction to Keras. In N. Ketkar (Ed.), *Deep Learning with Python: A Hands-on Introduction* (pp. 97–111). Apress. https://doi.org/10.1007/978-1-4842-2766-4_7

- (L emdani, 2016)

  L emdani, R. (2016). *Syst eme Hybride d'Adaptation dans les Syst emes de Recommandation* [Decertation]. Université paris Sackaly Central Suplelec.

- Li, X., Wang, L., & Sung, E. (2004). Multilabel SVM active learning for image classification. *2004 International Conference on Image Processing, 2004. ICIP'04.*, *4*, 2207–2210.

- **. (Liang, 2019)**

  Liang. (2019, August 7). *Feature selection using Python for classification problems*. Towards Data Science. https://towardsdatascience.com/feature-selection-using-python-for-classification-problem-b5f00a1c7028

- **(Malone et al., 1987)**

  Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987). Intelligent information-sharing systems. *Communications of the ACM*, *30*(5), 390–402. https://doi.org/10.1145/22899.22903

- **(McKinney, 2013)**

  McKinney, W. (2013). *Python for data analysis*. O'Reilly.

- **(Nayak et al., 2015)**

  Nayak, S., Ramesh, R., & Shah, S. (2015). *A Study of multilabel text classification and the effect of label hierarchy*.

- **(Nelli, 2015)**

  Nelli, F. (2015). Python Data Analytics: Data Analysis and Science Using Pandas. *Matplotlib, and the Python Programming Language*.

- **(Pandey, 2019)**

  Pandey, P. (2019, September 5). *The Remarkable world of Recommender Systems*. Towards Data Science. https://towardsdatascience.com/the-remarkable-world-of-recommender-systems-bff4b9cbe6a7

- **(Probst et al., 2017)**

Probst, P., Au, Q., Casalicchio, G., Stachl, C., & Bischl, B. (2017). Multilabel classification with R package mlr. *ArXiv Preprint ArXiv:1703.08991*.

- **(Papazoglou & Van Den Heuvel, 2007)**

  Papazoglou, M. P., & Van Den Heuvel, W.-J. (2007). Service oriented architectures: Approaches, technologies, and research issues. *The VLDB Journal*, *16*(3), 389–415.

- **(Pushpa & Karpagavalli, 2017)**

  Pushpa, M., & Karpagavalli, S. (2017). Multi-label Classification: Problem Transformation methods in Tamil Phoneme classification. *Procedia Computer Science*, *115*, 572–579. https://doi.org/10.1016/j.procs.2017.09.116

- **(*Recommender System Application Developments | Decision Support Systems*, n.d.)**

  Recommender system application developments | Decision Support Systems. (n.d.). Retrieved September 13, 2020, from https://dl.acm.org/doi/abs/10.1016/j.dss.2015.03.008

- **(Rishabah, 2017)**
- Rishabah, J. (2017, March 20). *Decision Tree. It begins here.* Medium.Com. https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134

- **(ROCCA, 2019)**

  Rocca, B. (n.d.). *Introduction to recommender systems | by Baptiste Rocca* | Towards Data Science. Retrieved September 13, 2020, from https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

- **(Roy, 2020)**

  Roy, B. (2020, April 6). *All about Feature Scaling*. Towards Data Science. Https://Towardsdatascience.Com/.

- **(Saini, 2019)**

  Saini, R. (2019, October 21). *Feature Scaling- Why it is required?* Medium.Com/. https://medium.com/@rahul77349/feature-scaling-why-it-is-required-8a93df1af310

- **(Salton et al., 2016)**

Salton, G., Ross, R., & Kelleher, J. (2016). Idiom token classification using sentential distributed semantics. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 194–204.

- **(Sanjay, 2018)**

  Sanjay, M. (2018, November 13). *Why and how to Cross Validate a Model?*. Towards Data Science https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f

- **(Settouti et al., 2019)**

  Settouti, N., Douibi, K., Bechar, M. E. A., Daho, M. E. H., & Saidi, M. (2019). Semi-Supervised learning with Collaborative Bagged Multi-label K-Nearest-Neighbors. *Open Computer Science*, *9*(1), 226–242.

- **(Shaikh, 2018)**

  Shaikh, R. (2018, 28). *Feature Selection Techniques in Machine Learning with Python*. Towards Data Science. https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

- **(Shardanand & Maes, 1995)**

  Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth." *CHI '95*. https://doi.org/10.1145/223904.223931

- (sharma, 2017)

  Sharma, A. (2017, March 30). *Understanding Activation Functions in Neural Networks*. Medium.Com. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

- **(Sharma et al., 2020)**

  Sharma, S., Sharma, S., Scholar, U., & Athaiya, A. (2020). *ACTIVATION FUNCTIONS IN NEURAL NETWORKS*. *4*(12), 7.

- **(sivarajkumar, 2019)**

sivarajkumar, sonish. (2019, 15). *ReLU — Most popular Activation Function for Deep Neural Networks*. Medium.Com. https://medium.com/@sonish.sivarajkumar/relu-most-popular-activation-function-for-deep-neural-networks-10160af37dda

- **(Srinidhi, 2018)**

  Srinidhi, S. (2018, July 30). *How to split your dataset to train and test datasets using SciKit Learn*. Medium.Com/. https://medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d

- **(st.loren, 2001)**

  st.loren, simon. (2001). *Programming Web Services with XML-RPC* (230th ed.). oreilly.

- **(Szymański & Kajdanowicz, 2017)**

  Szymański, P., & Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. *ArXiv Preprint ArXiv:1702.01460*.

- **(Tech, 2017)**

  Tech, A. (2017, August 4). *The mostly complete chart of Neural Networks, explained*. Towards Data Science. https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

- **(TMEGLIT, 2007)**

  TMEGLIT, N. (2007). *Un modèle de composition des services web semantique* (Thesis For doctoral degree TH4739.p). http://repository.usthb.dz/bitstream/handle/123456789/2439/TH4739.pdf?sequence=4&isAllowed=y

- **(Wei et al., 2015)**

  Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., Zhao, Y., & Yan, S. (2015). HCP: A flexible CNN framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(9), 1901–1907.