



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

---

# **Machine learning based warning system for failed procurement classification doc- uments**

Master's thesis in Data Science and Artificial Intelligence

Anastasios Tzinieris

---

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2022



MASTER'S THESIS 2022

Machine learning based warning system for failed  
procurement classification documents

Anastasios Tzinieris



UNIVERSITY OF  
GOTHENBURG

---



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2022

Machine learning based warning system for failed procurement classification documents

Anastasios Tzinieris

© Anastasios Tzinieris, 2022.

Supervisor: Aila Särkkä, Department of Mathematical Sciences

Advisor: Jonathan Liljegren, Tendium AB

Examiner: Umberto Picchini, Department of Mathematical Sciences

Master's Thesis 2022

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Typeset in L<sup>A</sup>T<sub>E</sub>X

Gothenburg, Sweden 2022

# Machine learning based warning system for failed procurement classification documents

Anastasios Tzinieris

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

## Abstract

Warning systems in the Machine Learning field of study, is a tool that generates a warning based on a model's prediction results. This thesis's study topic is to create such system to identify possible problematic procurement classification documents. Given a database of a company, a dataset was created for which a feature analysis was made to investigate which properties of a document can cause an either classification or formatting error. The challenging part of the research was the feature engineering since each feature had to be preprocessed differently based on the importance of the information contained.

Moreover, different supervised machine learning methods were implemented and hyperparameter tuned, using an algorithm called Grid Search. After the evaluation and comparison of the models, XGBoost Classifier was found to be the most successful both in terms of performance and computational time achieving 90,5% accuracy. However, by gathering more data, especially containing formatting errors, it is anticipated that the performance of the warning system using the XGBoost will be improved.

Keywords: Warning system, supervised learning, machine learning, feature engineering, XGBoost Classifier.



# Acknowledgements

First of all I would like to express my sincerest gratitude to my supervisor at Chalmers University of Technology, Aila Särkkä whose guidance and constant feedback throughout the duration of the thesis have been incredible. Secondly i would like to thank my advisor in Tendium AB, Jonathan Liljegren who helped me gather the data needed for the research and gave his input in any challenge i had. Last but not least i would like to thank my examiner Umberto Picchini for his collaboration and examination of the thesis.

Anastasios Tzinieris, Gothenburg, June 2022





# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Aim . . . . .	2
1.3 Limitations . . . . .	3
1.4 Outline of the thesis . . . . .	4
<b>2 Theory</b>	<b>5</b>
2.1 Machine learning . . . . .	5
2.1.1 Supervised learning . . . . .	5
2.1.2 Overfitting - Underfitting . . . . .	6
2.1.3 K-fold cross validation . . . . .	7
2.1.4 Decision Trees . . . . .	7
2.1.5 Random Forest . . . . .	9
2.1.6 Gradient Boosting . . . . .	9
2.1.7 Logistic Regression . . . . .	9
2.1.8 Support Vector Machines . . . . .	10
2.1.9 XGBoost . . . . .	10
2.1.10 Neural Networks . . . . .	11
2.2 Evaluation Metrics . . . . .	12
2.2.1 Confusion Matrix . . . . .	12
2.2.2 Accuracy . . . . .	13
2.2.3 Precision . . . . .	13
2.2.4 Recall . . . . .	13
2.2.5 F beta - score . . . . .	13
2.3 Libraries . . . . .	14
2.3.1 Pandas . . . . .	14
2.3.2 Scikit-learn . . . . .	15
2.3.3 Tensorflow . . . . .	15
<b>3 Data</b>	<b>17</b>
3.1 Data gathering . . . . .	17
3.2 Data description . . . . .	18
3.3 Target label of dataset . . . . .	19

3.3.1	Analysis of annotation data . . . . .	19
3.3.2	Creation of Target label . . . . .	20
<b>4</b>	<b>Feature Engineering</b>	<b>21</b>
4.1	Feature Analysis . . . . .	21
4.1.1	Type of document . . . . .	21
4.1.2	Document producer . . . . .	25
4.1.3	CPV Code . . . . .	27
<b>5</b>	<b>Modeling</b>	<b>29</b>
5.1	Data Preparation . . . . .	29
5.1.1	Document language . . . . .	29
5.1.2	Type of document . . . . .	30
5.1.3	Classification Label . . . . .	30
5.1.4	Document Producer . . . . .	30
5.1.5	CPV Code . . . . .	31
5.2	Model implementation . . . . .	32
5.2.1	Support Vector Machines . . . . .	32
5.2.2	Logistic Regression . . . . .	32
5.2.3	Random Forest . . . . .	33
5.2.4	XBGooost . . . . .	33
5.2.5	Neural Network . . . . .	33
<b>6</b>	<b>Results</b>	<b>35</b>
6.1	Important labels . . . . .	35
6.2	Results . . . . .	35
<b>7</b>	<b>Conclusion</b>	<b>39</b>
7.1	Discussion . . . . .	39
7.2	Limitations . . . . .	39
7.3	Future Work . . . . .	40
7.4	Conclusion . . . . .	41
<b>A</b>	<b>Appendix 1</b>	<b>I</b>
<b>B</b>	<b>Appendix 2</b>	<b>III</b>
B.1	Source code for data preparation . . . . .	III
B.2	Confusion matrices . . . . .	IV

# List of Figures

1.1	Structure of warning system. . . . .	3
2.1	Visual difference between Classification and Regression problems . . .	6
2.2	Examples of underfitting and overfitting in comparison to the optimal scenario. . . . .	7
2.3	Illustration of k-fold cross validation. . . . .	7
2.4	Example of a decision tree. . . . .	8
2.5	Flow of Random Forest Classifier. . . . .	8
2.6	Fit of a Logistic function and visualization of the threshold value. . .	10
2.7	Support vector machine hyperplane selection. . . . .	10
2.8	Neural Network architecture. The network has one input layer, three hidden layers and one output layer. . . . .	11
2.9	Example of Dataframe and its properties. . . . .	14
3.1	Example of a DataFrame containing relevant columns of the dataset.	18
4.1	Pie chart of correct and incorrect classification across the different types of documents. . . . .	22
4.2	Distribution of confidence scores intervals across all the incorrect clas- sifications. . . . .	23
4.3	Distribution of confidence scores intervals across all the incorrect clas- sifications for each different file type. . . . .	24
4.4	Distribution of confidence scores intervals across all the correct clas- sifications for each different file type. . . . .	25
4.5	Ratio of PDF files with and without a producer. . . . .	26
4.6	Top 10 producers with the highest percentage of failure. . . . .	26
4.7	Percentage of failure for the top 20 most common CPV codes. . . . .	27
5.1	Transformation of documentLanguage into categorical features. . . . .	29
5.2	Transformation of filepath into categorical features. . . . .	30
5.3	Transformation of a sample of labels in classificationLabelId into cat- egorical features. . . . .	30
5.4	Transformation of a sample of documentProducer into categorical features. The blue color indicates a converter which was previously seen by the model and the red color an updated converter. . . . .	31
A.1	Distribution of confidence scores intervals across all the correct clas- sifications. . . . .	I

B.1	Confusion matrix of the Random Forest Classifier. . . . .	IV
B.2	Confusion matrix of the Logistic Regression. . . . .	IV
B.3	Confusion matrix of the Support Vector Machines. . . . .	V
B.4	Confusion matrix of the XGBoost Classifier. . . . .	V
B.5	Confusion matrix of the Base Neural Network. . . . .	V
B.6	Confusion matrix of the Advanced Neural Network. . . . .	VI

# List of Tables

2.1	Sample of Confusion matrix containing TP,TN,FP,FN values. . . . .	13
3.1	Count of rows in each dataset. . . . .	18
3.2	Count of correct and incorrect rows of the dataset . . . . .	19
3.3	Number of rows with at least one error reported. . . . .	20
3.4	Count of errors across correct and incorrect classifications. . . . .	20
3.5	Distribution of the rows for the target label needChecking. . . . .	20
4.1	Number of datapoints for each type of document (PDF, DOCX, DOC, XLSX) . . . . .	21
4.2	Number of incorrectly classified rows with respect to the confidence scores. . . . .	22
4.3	Number of incorrectly classified rows with respect to confidence scores for each different file type. . . . .	23
4.4	Number of correctly classified rows with respect to the scores for each different file type. . . . .	24
6.1	Performance of the Machine learning models without Grid Search . .	36
6.2	performance of the Machine learning models on important labels without Grid Search . . . . .	36
6.3	Performance of Machine learning models with Grid Search . . . . .	37
6.4	Perfomance of Machine learning models on important labels with Grid Search . . . . .	37
6.5	Machine learning models performances summary for all metrics both with Grid Search and without . . . . .	37
A.1	Number of correctly classified rows with respect to confidence scores.	I
A.2	Count of incorrect rows for different PDF producers. . . . .	II



# 1

## Introduction

This thesis covers the study of feature selection and application of machine learning (ML) methods to produce a warning system for possible errors in a classification of procurements. Procurement, by definition, is the process of finding and agreeing to terms, and acquiring goods, services, or works from an external source, often via a tendering or competitive bidding process. Today, the public procurement market is not as technologically advanced as other domains in society [1]. The thesis was done at a company located in Stockholm called Tendium that has as its goal to make the public procurement process automated by using machine learning methods.

### 1.1 Background

Tendium delivers automatically curated data to its customers. This information is related to a certain flow, among which, several classification models are used to predict the output along with pre-processing and post-processing steps to prepare the input for training and deliver the output with the appropriate formatting. Since the level of automation is not absolute and dependent on predictions, there is a need in some cases to manually review and potentially correct some of the data before delivering them to the customers. This process is done by an annotation team whose responsibility is to go over each output of the classification flow for every document in a procurement and annotate accordingly the predictions.

Verifying a delivery results in an increase both in terms of time and cost. Since the annotation is very important for every company working with machine learning methods, it must be used constructively and avoid spending time verifying classifications. For this reason, the thought of a warning system was created which will notify the users if the output is potentially wrong on either the classification or the formatting stage. Machine learning is commonly used to create such systems and thus, it was selected for this research.

### 1.2 Aim

This research focuses on extracting features and using machine learning methods to create a warning system which will help the users to distinguish which documents need checking and which do not. As a result, a serious amount of time will be saved by this process which will then help Tendium to automate this part of their process. Every procurement consists of multiple documents and each document is then split to sections (paragraphs) which are being classified depending on their context. Thus, the warning system should first evaluate the content of each document and then provide a final positive or negative output to the user if the procurement needs to be checked or not.

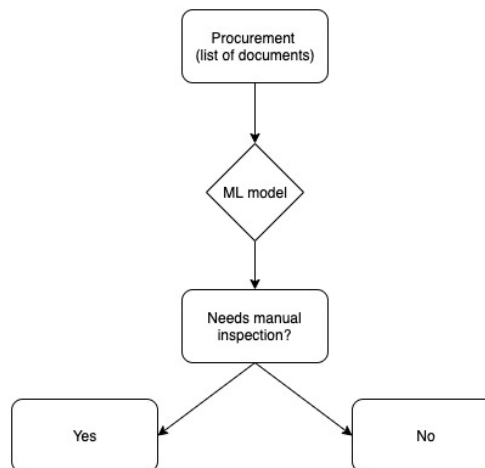
Since the output will be binary, there are multiple machine learning methods that can tackle binary classification problems. The methods that will be used and evaluated are :

- Random Forrest
- Logistic Regression
- Support Vector Machines
- XGBoost Classifier
- Neural Networks

since they are often successful at solving binary classification problems with multiple features. The theoretical background needed for these ML methods will be thoroughly described in Chapter 2.

The input of the methods will be the properties of each document, extracted during the flow of parsing and classification. After analyzing and selecting the appropriate features, the methods above will be tested and compared and the model with the higher performance metric will be selected to be used as the active model of the system. The output of this system will indicate the procurement needs manual inspection by the annotation team. The complete functionality of the warning system can be shown below.





**Figure 1.1:** Structure of warning system.

## 1.3 Limitations

Real data are always a challenge when it comes to machine learning. Firstly, there is no guarantee that the data will be enough to have an unbiased result since it's difficult to avoid having imbalance datasets (datasets that do not have the same amount of data for every different class). This suggests that the selection of the dataset and research around it should be the best one possible to achieve the desirable results. Secondly, the workflow of gathering data changed over time as new methods were used for classification and a new annotation for parser errors (reporting of wrong formatting of the output) was introduced after December 2021. As a result, the dataset used in this project might not include all the data available.

In addition, feature engineering is the most challenging part given that there is a lot of data available, in comparison to what is needed at the end. For instance, there is a possibility of including data from the annotation which will affect the outcome of the ML model and may lead to biased results. Another example is that there might be irrelevant features available which will increase time of prediction and might not be of any use to the model or in the worst case scenario even result in worse performance, since there might be broad and unconnected to the specific problem of interest.

Last but not least, the human error might affect the results since the data have been annotated by a team. Specifically, data points similar to each other might have been annotated differently by separate members of the team. Those data points, given that they contain valuable information, will confuse the model and may not be taken into account even though they should.

### 1.4 Outline of the thesis

This thesis is divided into 7 chapters. Chapter 1 is the Introduction, followed by Chapter 2 which covers the theoretical background needed for this work including description of the tools and the ML methods used during the course of the thesis. Chapter 3 includes a thorough description of the data and the process of acquiring and preparing them for the feature engineering which is the subject of Chapter 4. The construction and evaluation of each ML method is presented in Chapter 5. The comparison of the different models and visualization of their results is covered in Chapter 6 with the final conclusions and limitations of the research being listed in Chapter 7.

# 2

## Theory

The following chapter consists of the theoretical background of all the different tools and libraries along with the machine learning models used in this research.

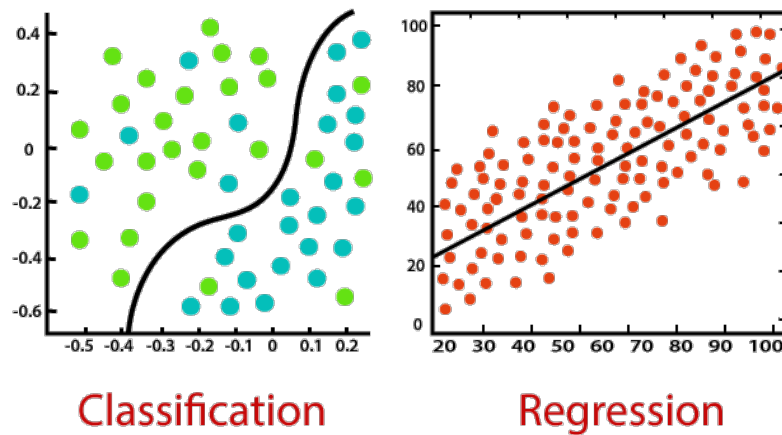
### 2.1 Machine learning

#### 2.1.1 Supervised learning

*Supervised learning*, also known as supervised machine learning, is an approach of creating a machine learning model in which the data of the problem have been already annotated [2]. In other words, each datapoint has a specific label indicating what the classification output should be. The most common example is the classification of spam emails where the dataset consists of texts(emails) and each text has a label specifying if the text is spam or not. The goal of a supervised learning model is to use the training data (by the assumption that the data are correctly annotated) and try to yield a desired output for unknown datapoints.

Supervised learning can be separated into two sub-problems:

- **Classification** : The process in which a function must be found to map an input (x) to a discrete output (y). An example for classification problems is email classification into spam or not spam.
- **Regression** : The process in which a function must be found to map an input (x) to a continuous output variable (y). Regression problem can be used for example in weather forecasting.

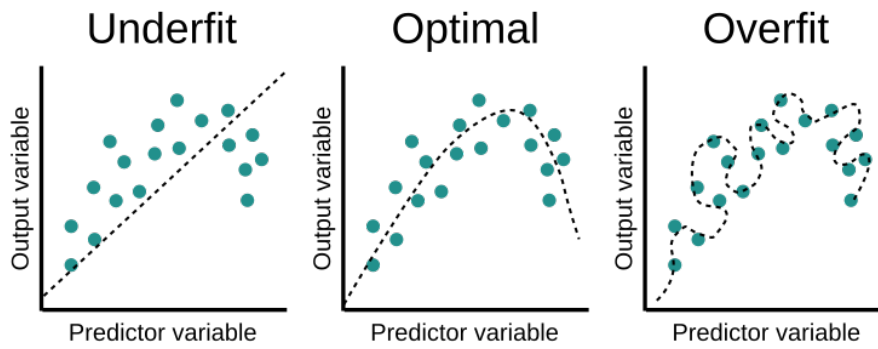


**Figure 2.1:** Visual difference between Classification and Regression problems

### 2.1.2 Overfitting - Underfitting

When implementing machine learning models, the main goal is to achieve the best performance for the model. However, in the machine learning field it is of great importance that a model will be evaluated before using it on real problems. When working with real data, obviously, the chances of managing to achieve the perfect score are very low. Especially in text classification problems, usually texts can differ a lot from each other and thus, it is impossible for the model to have seen the exact same example in the past.

The trade off that is important for these problems is the bias-variance trade off or in a more familiar term underfitting and overfitting of the model [3]. When a model has high bias and low variance or a model is *underfitting* (first graph of Figure 2.2), it means that the mapping function has not the necessary properties and input variables for being able to predict correctly newly seen data. On the other hand, when talking about *overfitting* (third graph of Figure 2.2), it means that the model has high variance and low bias which means that it has seen so many data points over and over again, at a point which it manages to know everything about the training set. However, the generalization error is too high, meaning that it cannot predict correctly new data. The reason behind this might be that there are many important features and the model now is too sensitive to outliers (datapoints that differ significantly from the training set). Overfitting can be seen when the training error is low but the testing error is high [4].



**Figure 2.2:** Examples of underfitting and overfitting in comparison to the optimal scenario.

### 2.1.3 K-fold cross validation

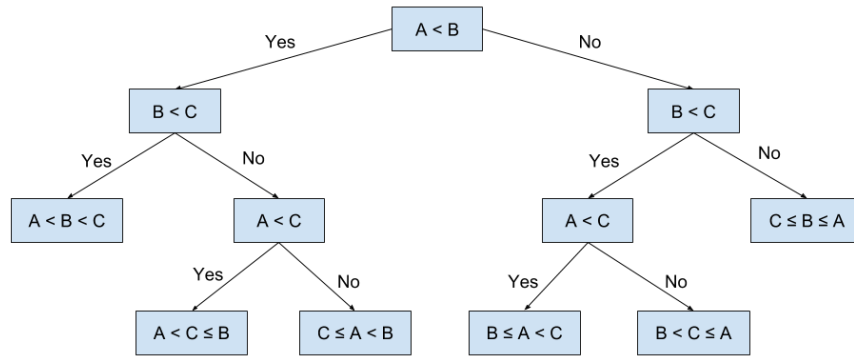
A regular process for creating and evaluating machine learning methods is that usually the dataset is split into training and testing set (70% and 30% respectively) [5]. To evaluate the performance of the model, the so called  $k$ -fold cross validation, where the complete data set is divided and processed  $k$  times or folds, can be used. Each time, a new test set is created and evaluated upon, leaving the remaining dataset as a training set. In other words, for  $k$  folds there will be exactly  $k$  test sets completely different to each other and when put together they give the complete dataset (See Figure 2.3). The final evaluation score would be the average score for all  $k$  folds.



**Figure 2.3:** Illustration of  $k$ -fold cross validation.

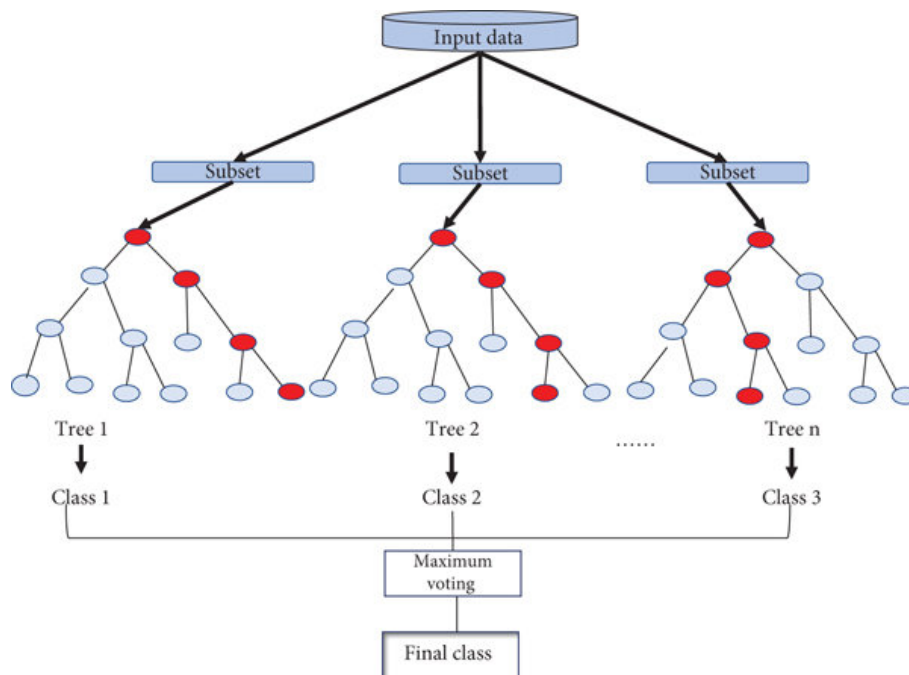
### 2.1.4 Decision Trees

For classification problems, a common approach is to use **Decision trees**. These are sequential models which represent a sequence of tests (questions) where each test compares a numerical attribute against a specified, by the model, threshold [6].



**Figure 2.4:** Example of a decision tree.

The tree is divided into non leaf nodes which are questions for a specific feature and leaf nodes which give the classification output based on the answers. The word decision tree derives from the hierarchy of the questions, representing the tree and the class of a datapoint is the first leaf node found for the sequence of tests. The order of the questions is extremely important and thus, the features (questions) should be categorised in a way that the first question should be the important for the prediction. A visualization of such a system is shown in Figure 2.4)



**Figure 2.5:** Flow of Random Forest Classifier.

### 2.1.5 Random Forest

A **random forest** (RF) classifier as the name implies is a classification method with consists of multiple decision trees constructing a forest (Figure 2.5). The number of decision trees operates as ensemble learning which in ML terms is the way of combining different models (in this case Decision trees) to achieve better predictive performance [7]. The concept behind Random forest Classifier is populating the model with many randomly formed decision trees and from the output of all those trees, picking the class that was predicted the most among them. The technique of combining all the models to create a new improved model is called *bagging*. The chance of overfitting is reduced because of the decision trees are randomly assembled and each decision tree has been trained on a different subset of the training dataset [8].

### 2.1.6 Gradient Boosting

The term **gradient boosting** is derived from the concept of "boosting" or enhancing a single weak model by merging it with a number of additional weak models to form a collectively strong model. Gradient boosting is a boosting modification in which the process of building weak models is described as a gradient descent method over an objective function. Gradient boosting is a supervised learning algorithm which targets the outcomes for the next model in an attempt to reduce errors in prediction. Targeted outcomes for each instance are determined by the gradient of the error with respect to the prediction (hence the name gradient boosting).

Since Gradient boosting is a supervised learning method, it takes a series of labeled training examples as input and constructs a model that attempts to accurately predict the label of each training example based on some additional non-label information about the example (known as features of the instance). The goal is to create an accurate model capable of automatically labeling future data with unknown labels.

### 2.1.7 Logistic Regression

One of the most popular methods used for supervised binary classification problems is **logistic regression**. It is used to forecast the categorical dependent variable from a group of independent factors and yields a probability between 0 and 1 which, given a specific threshold, results in the final predicted class. Its name derives from the logistic function in the form of sigmoid function used for mapping the predicted values to probabilities (Figure 2.6).

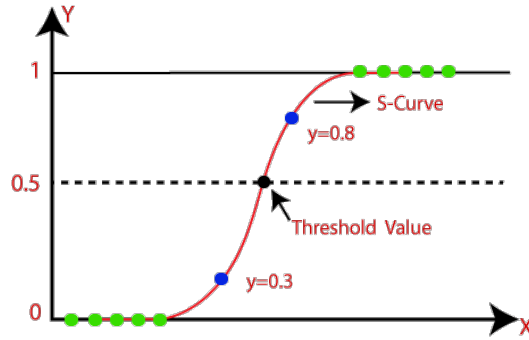
$$f(x) = \frac{1}{1 + e^{-x}}$$

, where  $x$  is the value needed to be transformed to probability between 0 and 1.

One can include linear inside the sigmoid function which will result in

$$f(x) = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

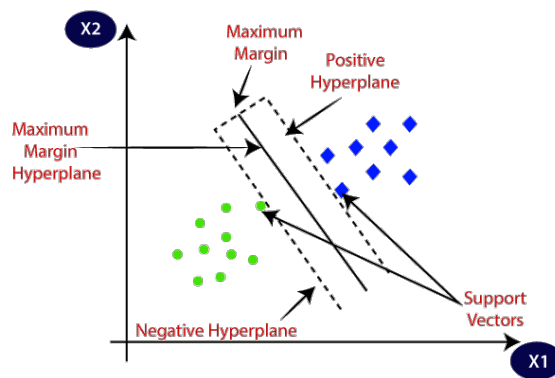
, where  $b_0$  is the bias or intercept term and  $b_1$  is the coefficient for the single input value  $x$ .  $f(x)$  is then the predicted output.



**Figure 2.6:** Fit of a Logistic function and visualization of the threshold value.

### 2.1.8 Support Vector Machines

Another example of supervised learning that will be used in this research is **Support Vector Machines** (SVM). The goal is to find a hyperplane between the datapoints to distinguish the two different cases in a binary classification problem. The hyperplane is created in a  $N$ -space where  $N$  is the total number of features available. The algorithm will try to create the hyperspace with as few features as possible and use additional features until an acceptable hyperplane is created. In the case that there are more than one desirable hyperplanes, it chooses the one with the higher margin between the two closest points which are called *support vectors* [9]. In this way the algorithm can select and classify unknown datapoints with a higher confidence. An illustration of SVM is shown below in the case of having only two features and thus in the 2D space.



**Figure 2.7:** Support vector machine hyperplane selection.

### 2.1.9 XGBoost

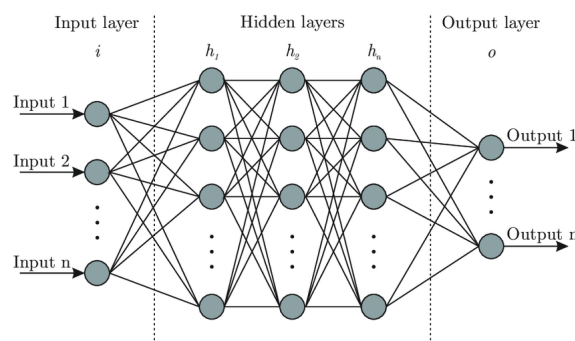
Extreme Gradient Boosting also known as **XGBoost** is a scalable and gradient-boosted decision tree (GBDT) machine learning library. It was selected in this project, since it is one of the most commonly used machine learning methods in



online competitions like Kaggle with extremely high performance. It combines properties mentioned before like ensemble learning, gradient boosting and decision trees. It is an implementation of gradient boosting with the difference that it is designed for maximizing the performance of the model and minimizing the computational speed. The biggest difference between gradient boosting and XGBoost Classifier is that, in the latter the decision trees are built in parallel instead of sequentially [10].

### 2.1.10 Neural Networks

An important subset of machine learning is deep learning which has been evolved over the years mainly because of the technological advances and the possibility of running computationally heavy models easily. Deep learning refers to the existence of **neural networks** also known as artificial neural networks (ANNs) which are series of algorithms whose goal is to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates with neurons signaling each other based on different mathematical formulas. Neural networks consist of node layers, one input layer, one or more hidden layers and one output layer representing the prediction output (see Figure 2.8). Each neuron (node) connects to another and has a specific weight and threshold. The neuron is activated if the output of each node is above a given threshold, resulting in the data to be sent to the next layer of the network.



**Figure 2.8:** Neural Network architecture. The network has one input layer, three hidden layers and one output layer.

Although there are numerous neural networks developed throughout the years, we will here focus only on the simple architecture of feedforward neural networks, or multi-layer perceptrons (MLPs) since more complicated NNs need both computational power and take more time in comparison to simple ANNs. Every layer applies certain mathematical operations on the inputs, and produces an output. It takes a vector of real values inputs, performs a linear combination of each attribute with the corresponding weight assigned to each of them which then is summed into a single value and passed through an activation function. Although there are a few activation functions available, the focus of this thesis will be on sigmoid (as mentioned before) and rectified linear activation function (ReLU) [11]. This function returns 0 if it receives any negative input, but for any positive value  $x$ , it returns that value back. Thus it gives an output that has a range from 0 to infinity. It is commonly

used because of its ability to converge faster without having negative values like in the case of tanh. The formulas for sigmoid and ReLu are shown below.

- **Sigmoid :**

$$f(x) = \frac{1}{1 + e^{-g(x)}}$$

- **ReLu :**

$$f(x) = x^+ = \max(0, x)$$

,where  $g(x)$  is a function (e.g. linear) and  $x$  is the input to a neuron

## 2.2 Evaluation Metrics

As mentioned above, one of the most challenging parts of machine learning is the evaluation of the model and avoidance of both underfitting and overfitting. To do that, there are some metrics which can be used to measure the performance and target the correct output. However before describing those terms and formulas, there are some valuable metrics needed regarding the classification. During a classification prediction , there are four types of outputs that could occur.

- **True positives :** Actual classification of datapoint is positive and predicted classification is positive.
- **True negatives :** Actual classification of datapoint is negative and predicted classification is negative.
- **False positives :** Actual classification of datapoint is positive and predicted classification is negative.
- **False negative :** Actual classification of datapoint is negative and predicted classification is positive.

### 2.2.1 Confusion Matrix

A confusion matrix is a table containing all the information mentioned above and has the following structure.

**Table 2.1:** Sample of Confusion matrix containing TP,TN,FP,FN values.

		Prediction outcome		total
		p	n	
actual value	p'	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
total		P	N	

### 2.2.2 Accuracy

Accuracy is defined as the percentage of correct predictions for the total datapoints of the test data. It can be calculated by dividing the number of correct predictions by the number of total predictions.

$$Accuracy = \frac{\text{correct predictions}}{\text{total predictions}}$$

### 2.2.3 Precision

Positive predictive value also known as precision is the fraction of true positives among all the instances that are predicted to belong to this class

$$Precision = \frac{\text{true positives}}{(\text{true positives}) + (\text{false positives})}$$

### 2.2.4 Recall

Recall (also known as sensitivity) is the fraction of true positives among all the instances that actually belong to this class

$$Recall = \frac{\text{true positives}}{(\text{true positives}) + (\text{false negatives})}$$

### 2.2.5 F beta - score

Although accuracy, precision and recall are very useful metrics for a model to be evaluated, there is often the need of taking into account both the precision and the recall like in this research.  $F_\beta$  score takes into consideration both the recall and the precision giving a weight (beta) that can be adjusted depending on the requirements

of the problem [12].

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

The values of beta can be:

- **beta > 1** : Recall is considered more important than precision.
- **beta < 1** : Precision is considered more important than recall.
- **beta = 1** : There is an optimal trade-off between the precision and the recall (also known as F1-score).

Since the problem did not require a certain focus to either of them, the F1-score was used to verify that the precision and recall were equally weighted.

## 2.3 Libraries

### 2.3.1 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis [13]. It is commonly used for data analysis because of its ability to easily manipulate tabular data in a structure called **Dataframe**. There are several ways of importing data through different file types such as comma-separated values (CSV), Microsoft Excel, Json or SQL database tables. With the use of Dataframe, pandas allows fast, in terms of computational speed, changes and manipulations such as selection, merging, and reshaping along with data cleaning features.

The diagram illustrates a Pandas DataFrame with the following structure and annotations:

	0	1	2	3	4	5
	homeTeamName	awayTeamName	startTime	duration_minutes	attendance	dayNight
0	Cubs	Reds	2016-07-05 18:20:00 UTC	188	41310	D
1	Indians	Astros	2016-09-08 16:10:00 UTC	194	15275	D
2	Padres	Giants	2016-09-25 20:40:00 UTC	185	28456	D
3	Diamondbacks	Brewers	2016-08-07 20:10:00 UTC	211	24021	D
4	Giants	Dodgers	2016-04-09 20:05:00 UTC	202	41224	D
5	Blue Jays	Indians	2016-07-02 17:07:00 UTC	199	46197	D
6	Reds	Pirates	2016-04-09 17:10:00 UTC	180	22799	D
7	Cubs	Mariners	2016-07-30 18:20:00 UTC	157	41401	D
8	Rockies	Phillies	2016-07-10 20:10:00 UTC	191	32113	D
9	Yankees	Blue Jays	2016-05-26 20:05:00 UTC	154	38391	D

Annotations in the diagram:

- SERIES**: Points to the 'awayTeamName' column.
- DATA**: Points to the 'startTime' column.
- INDICES**: Points to the row index column (0-9).
- LABELS**: Points to the column headers.
- AXIS**: Points to the vertical axis (rows) and horizontal axis (columns).

**Figure 2.9:** Example of Dataframe and its properties.

### 2.3.2 Scikit-learn

The Scikit-learn library (commonly known as sklearn) is an open source learning library that supports supervised and unsupervised learning. It also provides various tools for model training, data preprocessing, model selection and evaluation. Many of the ML methods, used for this research such as Random Forest, Support Vector Machines and Logistic Regression were implemented with the use of sklearn. In addition, the library has the ability of using efficiently structures such as pandas Dataframes.

### 2.3.3 Tensorflow

TensorFlow is an open source library, created by Google, for numerical computation and machine learning. It uses Python to create applications such as neural networks while the execution of those applications are made in C++. Developers can create graph-structures which describe how the input data are passed through a number of processing steps. Each step, also known as node indicates a mathematical operation and each edge between the nodes is a multidimensional data array which is referred to as tensor. Tensorflow framework is used in this project for creating and evaluating the neural network.



# 3

## Data

This research were made on real data provided by Tendium AB. Since the process evolves information about procurements, the data provides are paragraphs extracted by text documents. Each paragraph is then classified depending on the context of the text. All the information about the documents and the classification flow are located in a database. The following chapter covers the process of data gathering along with the structure and description of the dataset.

### 3.1 Data gathering

When working on real problems such as this thesis, the data must be collected or extracted by a database. In this case, the data were gathered by Tendium's database consisting of information for every different section of a document of each procurement. This information about a section in a dataset will be either referred to as a row or datapoint in the rest of the research. However, the challenging part of this process was the selection of the data that would help to solve the problem because the databases consist of tables containing metadata and information that in hand were irrelevant.

The tool used for this process was pgAdmin in which a server of the company's database was saved. The programming language that this interface uses for manipulating tabular data is *postgreSQL* (also known as postgres) which is one of the most commonly used languages in database management systems. With the use of this tool, three different tables containing information about the documents resulting in an over of 1.600.000 datapoints table. Nevertheless, this table is the complete database containing both annotated and non-annotated data which in this case would be impossible to include since supervised learning methods requires a target label. Including only the annotated data returned about 125.000 rows which were saved as a excel file to be used in the python environment.

The last part of the process was taking into account any changes of the company's process regarding annotations and classifications. Specifically, a new process of annotating parser errors, which were formatting issues during the post-processing step of the main flow, was introduced in November 2021. The initial assumption was that this new field, that was contained in the table extracted, could be of use in predicting texts with possible formatting issues since there will be information provided by the annotators and thus, a target label to be used in the training of

Dataset description	Number of rows
Complete dataset	1.674.324
Annotated dataset	125.409
Final dataset	28.790

**Table 3.1:** Count of rows in each dataset.

a model. Moreover, the classification model used in the main flow was updated on October 22nd, 2021. To be sure that both arguments were satisfied, the final dataset gathered was from December 1st, 2021, consisting of 28.790 rows. The following table describes the number of rows for each step of the process.

## 3.2 Data description

Before doing the feature engineering, each feature must be thoroughly described since it was important for the correct selection of the data. Some of the features contained in the dataset will not be described in this chapter since they were not used but will be included in Appendix A. The dataset containing features that were inspected in this research is located below in form of a DataFrame as was described in 2.3.1 .

procurementId	filePath	parserErrors	documentLanguage	documentProducer	classificationConfidenceScore	correct	labelBoxesUpdatedAt	classificationLabelId
61dd3c14d54236062de8dbf	document_files/61dd3c14d54236062de8dbf/tender...	[]	sv	Adobe PDF Library 15.0	0.547595	True	2022-03-16 09:20:19.372	C66
600a8caa60fdee161fbd833	document_files/600a8caa60fdee161fbd833/tender...	[]	NaN	NaN	0.999997	False	2022-03-01 15:11:20.874	C65
601a5dd801e1545637a78c64	document_files/601a5dd801e1545637a78c64/tender...	[]	NaN	NaN	0.318250	False	2022-03-15 11:16:47.637	C67
600697459730d71a933a026a	document_files/600697459730d71a933a026a/tender...	[]	NaN	NaN	1.000000	True	2022-03-08 10:13:47.326	C70
60069e27abc104cf701a966	document_files/60069e27abc104cf701a966/tender...	[]	NaN	NaN	0.468022	False	2022-03-14 14:58:12.119	C65
60069e28abc104cf701a96d	document_files/60069e28abc104cf701a96d/tender...	[]	NaN	NaN	0.999999	False	2022-03-14 14:59:31.073	C65
6007ebb8d7a5eb5cce467a03	document_files/6007ebb8d7a5eb5cce467a03/tender...	[]	NaN	NaN	0.986554	False	2022-03-14 14:59:08.780	C65
6007ebb8d7a5eb5cce467a03	document_files/6007ebb8d7a5eb5cce467a03/tender...	[]	NaN	NaN	0.999973	False	2022-03-14 14:59:42.895	C65
6007ebb8d7a5eb5cce467a03	document_files/6007ebb8d7a5eb5cce467a03/tender...	[]	NaN	NaN	0.996068	True	2022-03-08 10:19:06.269	C70
6007f4d23225a7e625b76be	document_files/6007f4d23225a7e625b76be/tender...	[]	NaN	NaN	0.437418	False	2022-03-14 15:01:43.596	C65
6007f18423225a7e625b6e54	document_files/6007f18423225a7e625b6e54/tender...	[]	NaN	NaN	0.610894	True	2022-03-14 15:02:02.215	C65
6007ebcbb49561d381cef5cf	document_files/6007ebcbb49561d381cef5cf/tender...	[]	NaN	NaN	0.602689	False	2022-03-08 10:20:35.194	C70
6007ebcbb49561d381cef5cf	document_files/6007ebcbb49561d381cef5cf/tender...	[]	NaN	NaN	0.241385	False	2022-03-08 10:20:43.591	C70
601a603d9617ab76288dca03	document_files/601a603d9617ab76288dca03/tender...	[]	NaN	NaN	0.162316	False	2022-03-15 11:10:15.854	C67
601a603cf86a7409f4b22b0f	document_files/601a603cf86a7409f4b22b0f/tender...	[]	NaN	NaN	0.986648	True	2022-03-14 13:28:50.518	C67
601bac5cf86a7409f4b22b0f	document_files/601bac5cf86a7409f4b22b0f/tender...	[]	NaN	NaN	0.466102	False	2022-03-15 11:28:34.041	C67
600fd2a15b4cb70648535b53	document_files/600fd2a15b4cb70648535b53/tender...	[]	NaN	NaN	0.454923	True	2022-03-08 10:29:58.278	C70
602248a4584e1236163187da	document_files/602248a4584e1236163187da/tender...	[]	NaN	NaN	0.401434	True	2022-03-08 11:27:18.718	C69
602a3057584e12361631f4a	document_files/602a3057584e12361631f4a/tender...	[]	NaN	NaN	0.019875	False	2021-12-13 14:35:19.977	C8
602a3057584e12361631f4a	document_files/602a3057584e12361631f4a/tender...	[]	NaN	NaN	0.002908	False	2021-12-13 14:35:17.688	C96
6059a55093519968ab8b4057	document_files/6059a55093519968ab8b4057/tender...	[]	NaN	NaN	0.333186	False	2022-02-04 15:53:43.518	C68
6017ba1b09a302b27548d5	document_files/6017ba1b09a302b27548d5/tender...	[]	NaN	NaN	0.260517	False	2022-02-04 15:50:57.728	C68
5fc137c6c8e2771da8b47ce	document_files/5fc137c6c8e2771da8b47ce/tender...	[]	NaN	NaN	0.999990	True	2022-01-19 12:56:23.866	C23
5fc137c6c8e2771da8b47ce	document_files/5fc137c6c8e2771da8b47ce/tender...	[]	NaN	NaN	0.139522	False	2022-01-19 12:58:46.298	C120
5fc137c6c8e2771da8b47ce	document_files/5fc137c6c8e2771da8b47ce/tender...	[]	NaN	NaN	0.088132	False	2022-01-19 12:59:02.434	C120
5fc137c6c8e2771da8b47ce	document_files/5fc137c6c8e2771da8b47ce/tender...	[]	NaN	NaN	0.435201	False	2022-01-19 13:01:26.700	C126
6033678bc773973df1191f1	document_files/6033678bc773973df1191f1/tender...	[]	NaN	NaN	0.531235	False	2022-02-04 15:02:46.389	C68
60545f697cc1869121e42a8	document_files/60545f697cc1869121e42a8/tender...	[]	NaN	NaN	0.055394	False	2022-03-15 11:08:25.111	C69
603ca2f5684802a28fae962	document_files/603ca2f5684802a28fae962/tender...	["Incorrectly labeled headline"]	NaN	NaN	0.999969	True	2022-02-18 13:07:25.078	C69
60618eeab57b76d6d164fc	document_files/60618eeab57b76d6d164fc/tender...	["Incorrectly labeled headline"]	NaN	NaN	0.996034	True	2022-02-18 13:08:04.783	C69

**Figure 3.1:** Example of a DataFrame containing relevant columns of the dataset.

Description of each feature in the DataFrame:

- **procurementId** : A unique id code indicating the procurement that the document is included in. Since every document may have more than one section, multiple rows will have the same procurement id.
- **filePath** : The filepath of each document in the database. The file types that can exists are XLSX (Microsoft Excel), DOC (Microsoft Word), DOC (Microsoft word with Open XML format), PDF (Portable Document Format).
- **parserErrors** : A list of strings containing any possible errors found during the parsing of the document or the post-processing of the output.



- **documentLanguage** : A string indicating the language code of the document.
- **documentProducer** : In the case of PDF documents, the name of the application (string) used to transform the document to PDF.
- **classificationConfidenceScore** : The prediction score of the classifier used to classify each section. Since it is a probability the values are between 0 and 1.
- **correct** : The annotation given for each classification. if the classification was correct then this field will be True, otherwise False.
- **LabelBoxesUpdatedAt** : A timestamp suggesting the date of a change that was made in a specific row.
- **classificationLabelId** : The predicted label of the classifier. Tendium's flow uses a multi-label text classifier to classify each text into one or more labels depending on the context of the paragraph. The total amount of labels is 145 in the form of "CX", where X is a number between 1 and 145. However, there are some labels that are more valuable for the company and thus, prioritised. Those labels are described more in Chapter 6 during the evaluation.

In addition to this dataset, one JSON file was provided by the company, containing a mapping between each procurement and its CPV codes. These codes are 8-digit numbers describing an industry. The JSON file provides the information about the industry importance of each procurement.

### 3.3 Target label of dataset

Since the goal of the project was to create a warning system for identifying possible errors in either the classification step or the formatting of the text, it was necessary to create another column to represent the target label of the dataset. The information to do that was available on two possible columns, the **correct** and the **parserErrors** columns. The former returns the annotation for the classification step and the latter the indication of an error in the formatting.

#### 3.3.1 Analysis of annotation data

The data gathered represent all the annotated texts in terms of the classification. In other words, all 28790 rows contain information (either True or False) in the column "**correct**". However, it was important to verify that the data were not imbalanced which means that there were enough data for both True and False annotations. The results of this analysis (see Table 3.2) fulfilled the requirements although the number of correct classifications was more than the incorrect ones.

Classification	Datapoints
Correct	18718
Incorrect	10072

**Table 3.2:** Count of correct and incorrect rows of the dataset

The second part of this analysis was to check the company's newly added feature of report possible errors made during the parsing of the text. Moreover, the user, along with the annotation of classification, had the option to report in a text field one or more errors if existed. In terms of data structure, this field contains either an empty list, indicating that the text had no errors or a list containing one or more string explaining the problem found such as "incorrectly headline", "missing information" etc.. Unfortunately, the data regarding parser errors were not as many as we first thought with only 282 errors reported in a period of five months (Table 3.3).

Parser Errors	Datapoints
Exist	282
Not exist	28508

**Table 3.3:** Number of rows with at least one error reported.

Last but not least, both the features were included to provide information about how many of the errors were correct classifications or not. The reason behind this test was to check if both the features were necessary to be included in the target label, otherwise the parser errors were already covered by the classification annotation. The results in Table 3.4 revealed that the errors were balanced between correct and incorrect classifications. Despite the low number of errors which probably would not affect the results of the warning system, it was an additional information which was included in the creation of the target label.

Classification	Errors
Correct	154
Incorrect	128

**Table 3.4:** Count of errors across correct and incorrect classifications.

#### 3.3.2 Creation of Target label

After the analysis above, a new column called **needsChecking** was created which, in the case of either an error in the classification or the format is reported, was annotated as zero (0), otherwise it was annotated as one (1). The distribution shown in the Table 3.5 indicates that there is still an imbalance in the dataset between the rows that need to be checked and those that do not. Nevertheless, the amount of data for each class was enough to proceed to the feature extraction.

needsChecking	Number of rows
No ( 0 )	18.564
Yes ( 1 )	10.226

**Table 3.5:** Distribution of the rows for the target label needChecking.

# 4

## Feature Engineering

Once the data were acquired, the most important part of the thesis was to select and analyze all the important features also known as feature engineering. The following chapter covers the analysis of features, selection and preprocessing of the dataset before the implementation of the ML models.

### 4.1 Feature Analysis

The annotation of the target label, as mentioned in the previous chapter, was made considering the output of the classification and the report of possible parser errors. Thus, during the feature engineering the focus was on inspecting possible features available with respect to these two parameters.

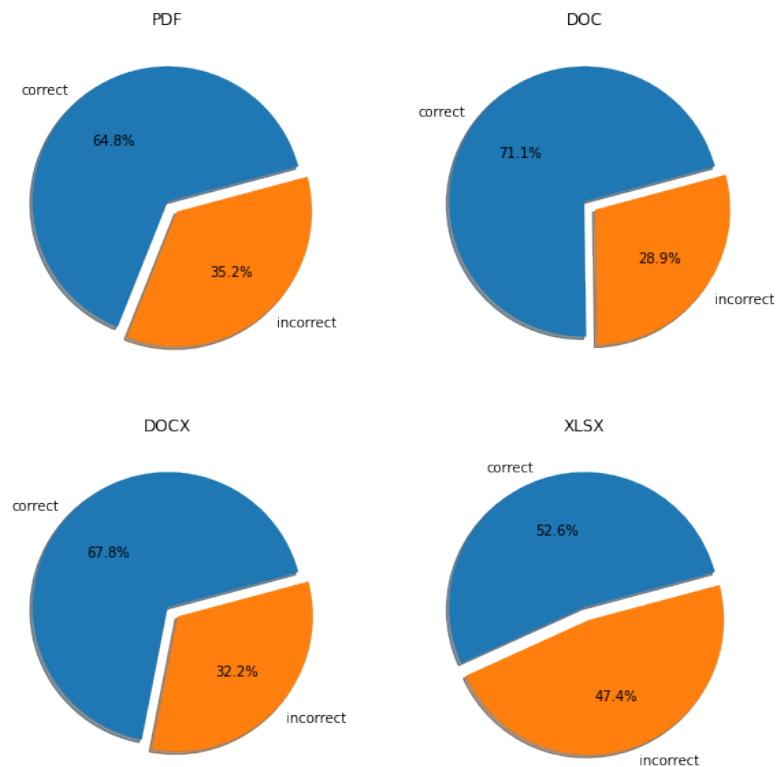
#### 4.1.1 Type of document

The initial thought was to investigate the different types of files handled by the company and observe how many of those contain wrong classifications. It is known that PDF, DOCX and DOC documents are handled by the same classifier. On the other hand, XLSX documents classified by another ML model which from previous evaluations is less effective in the classification. In addition, since procurement documents usually are in the form of either PDF or DOCX, the number of such documents was larger than XLSX and DOC documents (See Table 4.1 ).

Type of Document	Datapoints
PDF	25192
DOCX	3655
DOC	522
XLSX	344

**Table 4.1:** Number of datapoints for each type of document (PDF, DOCX, DOC, XLSX)

Moreover, for each filetype a check was made to find the percentage of correct and incorrect classifications. In Figure 4.1, a pie chart is showing, in which the blue color represents the correct classifications and the orange color the incorrect classifications. It can be seen that XLSX documents are being falsely classified more frequently than PDF and DOCX which had similar performances (64.8% and 67.8% success rate respectively).



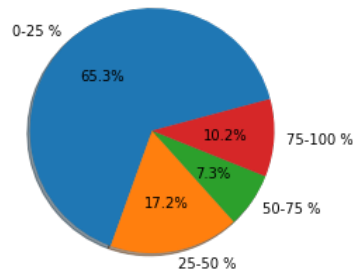
**Figure 4.1:** Pie chart of correct and incorrect classification across the different types of documents.

Once this first observation was made, it was important to check also the confidence scores intervals for all documents and separately for file type. The confidence score, as described in Chapter 3, is a number between 0 and 1 indicating the prediction of the classifier for each row. The reason of this test was to check if the prediction is correlated to the annotation. In other words, investigate if the classifier was unable to predict a certain row with confidence and thus, the classification was wrong. The percentages were divided into four intervals along with a threshold to indicate the rows which, although they had high confidence score, they were wrong. Table 4.2 shows the number of falsely classified rows for different confidence score intervals while Figure 4.2 reveals the percentage of those intervals towards all the incorrect instances.

Confidence Score	Datapoints
0-25 %	6592
25-50 %	1733
50 - 75 %	732
75 - 100 %	1034
>95%	569

**Table 4.2:** Number of incorrectly classified rows with respect to the confidence scores.

The results above indicated that in 82.5% of the cases, the model did not predict a



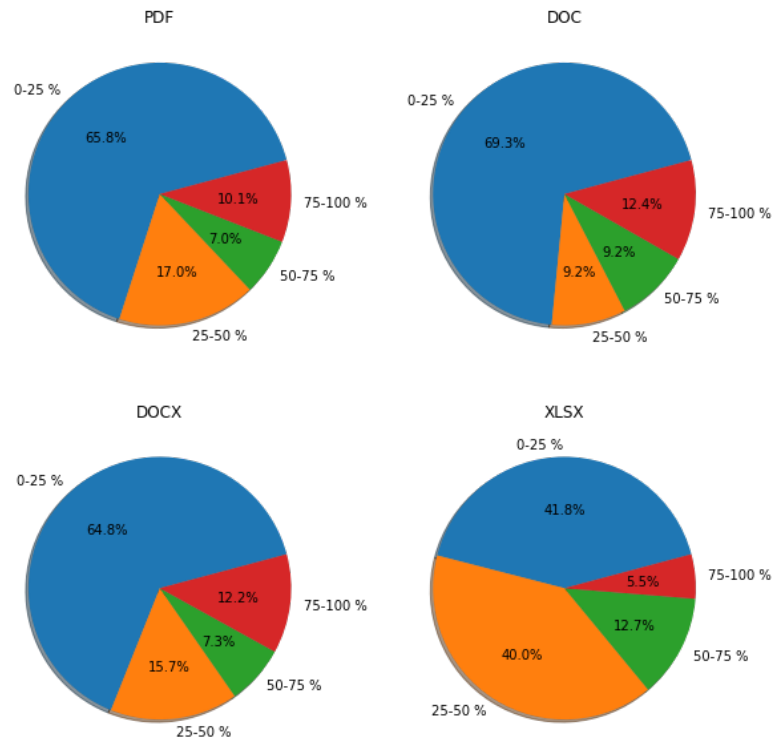
**Figure 4.2:** Distribution of confidence scores intervals across all the incorrect classifications.

label with certainty (below 50%) and the result was expected since low prediction scores result in wrong classifications. However, there was also a need to investigate each type of file separately to verify that this result follows the same pattern for all four different type of file. The outcome of this test (Table 4.3 and Figure 4.3) proves right this assumption. Although for XLSX documents the interval 0-25 % was not as high as the other file types, in the case of 0-50% the percentage is similar across all files (80%).

Confidence Score	Datapoints			
	PDF	DOCX	DOC	XLSX
0-25 %	5836	568	106	69
25-50 %	1509	138	14	66
50 - 75 %	223	64	14	21
75 - 100 %	895	107	19	9
>95%	488	11	62	5

**Table 4.3:** Number of incorrectly classified rows with respect to confidence scores for each different file type.

One interesting observation from the analysis of incorrect classifications was that for PDF, DOCX, DOC documents, which are using the same classifier, the amount of misses with above 75% prediction score was higher than the one between 50 - 75%. This initialized the idea to find how many correct classifications exist. The desired output was that predictions over a certain threshold (i.e 95%) should most likely succeed in the classification. The same process was followed, first for all types of documents and then for each type separately. The results shown below are specifically for the case of each file type separately. Nevertheless, the overall results for the correct classifications can be found in Table A.1 and Figure A.1 in Appendix A.

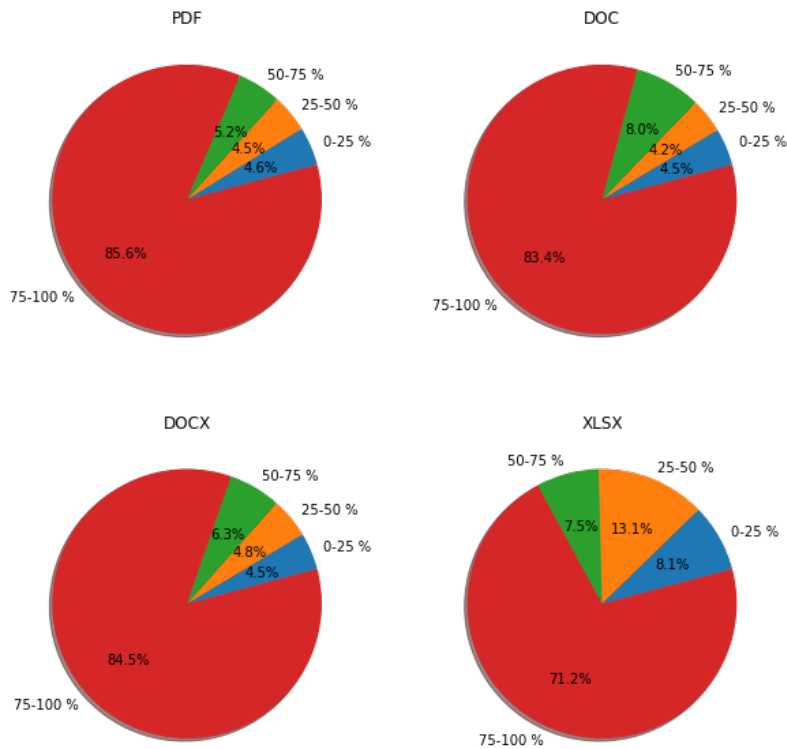


**Figure 4.3:** Distribution of confidence scores intervals across all the incorrect classifications for each different file type.

In comparison to the first investigation (incorrect predictions), the correct predictions seemed to follow a better pattern since 85% for PDF and DOCX documents have above 75% prediction score (See Figure 4.4). Moreover, in Table 4.4 can be observed that, although the majority of the datapoints are above 75%, the rows above 95% percent are similar. In other words, in most of the cases the model is over 95% confident for a prediction which is also correct. In addition, considering the two investigations, the assumption that a specific file type may affect a possible error in classification was valid along with the fact that the confidence score was highly correlated with a possible failure. Thus, both of the features were included as data for the training and testing of the model.

Confidence Score	Datapoints			
	PDF	DOCX	DOC	XLSX
0-25 %	700	74	15	13
25-50 %	689	80	14	21
50 - 75 %	791	104	27	12
75 - 100 %	12974	1404	281	114
>95%	11467	1207	237	89

**Table 4.4:** Number of correctly classified rows with respect to the scores for each different file type.



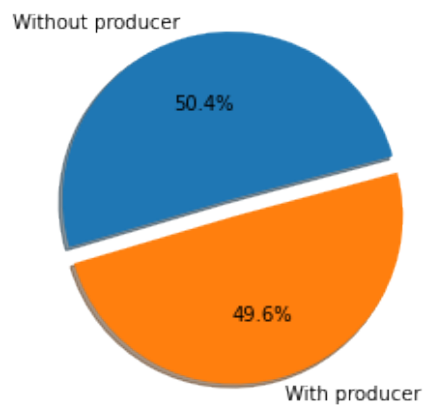
**Figure 4.4:** Distribution of confidence scores intervals across all the correct classifications for each different file type.

### 4.1.2 Document producer

To produce a PDF document, someone must use a certain application (i.e Adobe library) and convert a text document such as DOCX. The information of this application is stored in the company's data since November. However, there are cases where the extraction of the producer of the PDF document fails thus, not all PDF documents contain this additional feature. The dataset which will be used contains this extra information on around 50% of the rows with PDF as filetype (See Table Figure 4.5).

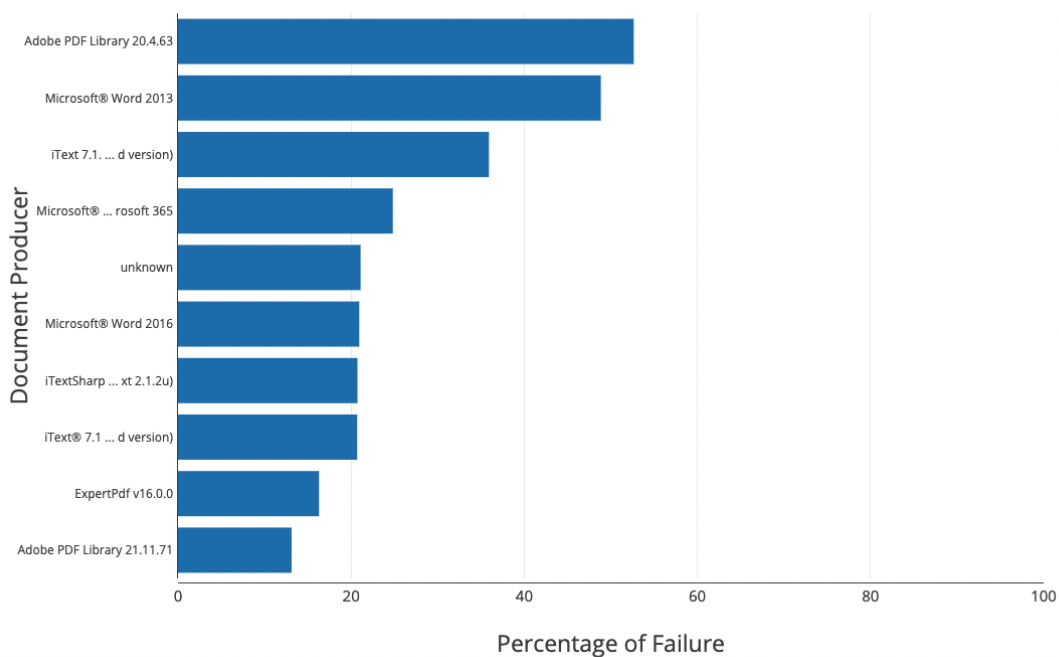
When it comes to applications, they frequently get updated. Because the information, stored in the dataset, contains the full version of application, it was suspected that multiple documents will have the same converter but a different versions. In addition, the main goal of the analysis was to check if there are specific programs, with their respective versions, which frequently fail the main flow.

The process of this inspection was to search only the texts with either error in the classification or the formatting and select the most common producers of those rows. After acquiring the count of rows for each (common) producer, we calculated the ratio of those rows with respect to the total number of incorrect rows which have the same producer. The percentages of failure for the top 10 common producers is shown in Figure 4.6. For clarification of the complete names ,the top 20 most



**Figure 4.5:** Ratio of PDF files with and without a producer.

common producers with their respective number of rows can be found in Table A.2.



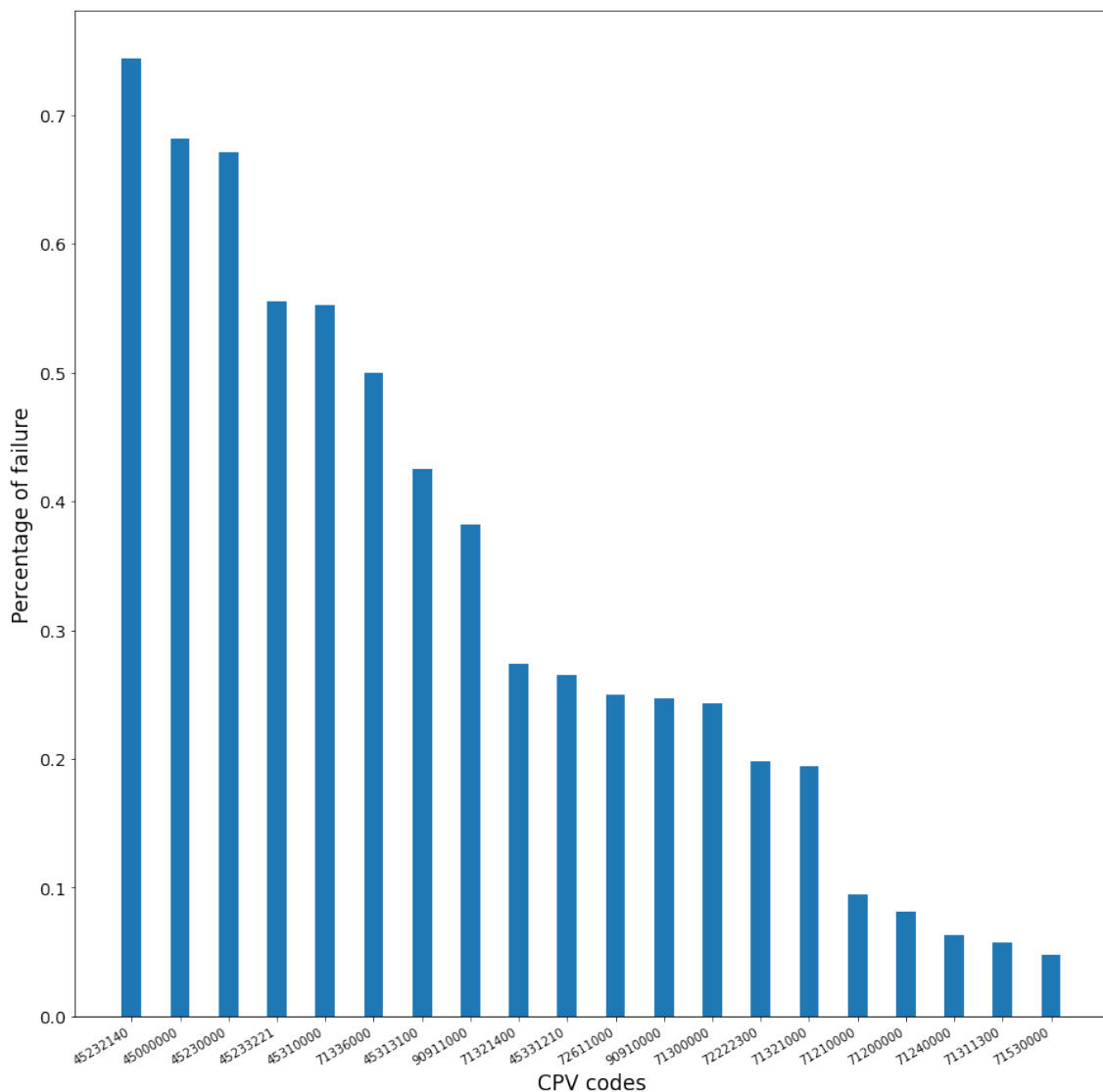
**Figure 4.6:** Top 10 producers with the highest percentage of failure.

Given the results above, it was certain that there were different versions of the same program and most importantly in the common producers. Grouping the versions together was impossible since the difference in the percentages of failure were extremely high. However, keeping both the information about the program and the version was important and thus, selected as a feature in the research’s dataset.



### 4.1.3 CPV Code

The last feature that was added was the CPV Code. Having another file with CPV codes for each different procurement, it was possible to include the information in the dataset to be further analyzed with respect to errors in either classification or formatting of the text. The same process was followed as with the most common producers. For each CPV code, the number of incorrect rows were counted and resulted in another plot showing the percentage of failure for the top 20 most common CPV codes (Figure 4.7) . By ocular inspecting the results, the failure did not exceed the 8% on any of the CPV codes. Despite this, the CPV code was selected as a feature because it would provide extra information which in this research was important since the features available were insufficient.



**Figure 4.7:** Percentage of failure for the top 20 most common CPV codes.



# 5

## Modeling

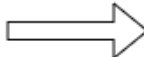
After identifying the most important features with the previous chapter's analysis, the next step of the process was the modeling. In this chapter the data preparation and the different ML methods used along with their architectures will be discussed. For each model there will be a thorough analysis of the different simulations made and the different parameters that were manipulated. The results from the evaluation are shown in the next chapter along with the comparison of all the different models.

### 5.1 Data Preparation

As it was mentioned in the previous chapter, the features selected contained both strings (i.e the producer of the document ) and integers (i.e the confidence score). However, when inputting a dataset into a model, the data should be structured in a way that all the values are normalized values and, specifically for this thesis, are between zero (0) and one (1). For this reason each feature was transformed to fit the model requirements.

#### 5.1.1 Document language

Since the data gathered were annotated, the language of the documents was Swedish because the company does not handle English documents at the time that this thesis was written. Despite that, there were multiple rows that instead of 'sv' as a language they had "language not found". The reason behind this was that the same method for extracting the producer of the document was used to extract the language and it did not always succeed depending on the format of the PDF. To be able to handle the two possible outputs, two columns replaced the one called "documentLanguage" and instead of a string they could get either zero or one, where one indicated the language of the document (Figure 5.1). This transformation was made by a function in pandas library called **get\_dummies** (See B.1).



documentLanguage
sv
not_found

documentLanguage_sv	documentLanguage_not_found
1	0
0	1

**Figure 5.1:** Transformation of documentLanguage into categorical features.

### 5.1.2 Type of document

The same process was followed for the type of document. First, the name of the path was removed since it was irrelevant for the research and only the extension of the path (pdf,xlsx,docx,doc) was kept. The resulted column was then divided into four different columns, one for each type of document. The transformed columns contained also the "filepath\_" in their header names because it was important to separate each feature in order not to have any conflicts with other features that may contain the same name (Figure 5.2).

filepath		filepath_pdf	filepath_xlsx	filepath_doc	filepath_docx
path.pdf	→	1	0	0	0
path.xlsx		0	1	0	0
path.doc		0	0	1	0
path.docx		0	0	0	1

**Figure 5.2:** Transformation of filepath into categorical features.

### 5.1.3 Classification Label

The feature regarding the predicted label during the classification flow is the **classificationLabelId**. Each row in the dataset has been predicted as one or more labels depending on the context of the text. In comparison to the first two features, the classificationLabelId may have more than one label and thus, after transforming the column to binarized features, there might be multiple columns activated (Shown in green color in Figure 5.3). The total number of labels and additional columns that were created was 145.

classificationLabelId		classificationLabelId_C67	classificationLabelId_C95	classificationLabelId_C5	classificationLabelId_C69	classificationLabelId_C105
C67	→	1	0	0	0	0
C95		0	1	0	0	0
C5,C69		0	0	1	1	0
C105		0	0	0	0	1

**Figure 5.3:** Transformation of a sample of labels in classificationLabelId into categorical features.

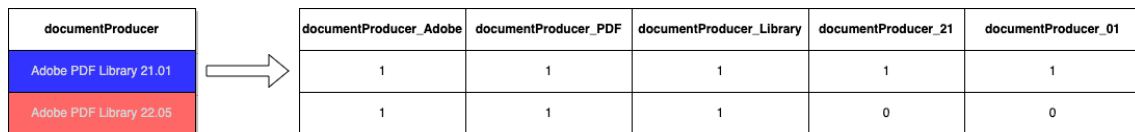
### 5.1.4 Document Producer

To convert the **documentProducer** of PDF files to categorical features, three different approaches were possible. The first one was to create categorical features based on only the name of the producer of the document (converter) and excluding

the version. However, after the analysis in the previous chapter, there were important differences between versions regarding the performance and possible error in the formatting.

The second approach was to include the version and separate the columns based on the complete name of the converter. One possible flaw of this approach was the case where future documents have been converted with an updated version. Moreover, the model will extract the document producer but it will not find the feature mapped to this producer and the information will be lost.

The approach that was selected was to divide each converter to tokens where each token represents a word or a number. In result, the categorical features will be all the tokens for every converter. In that way, the model will be able to use part of the information even though the version might be different. To transformation was done by sklearn-library's **CountVectorizer()** to transform the features to categorical and then add a prefix to each feature to specify the type of the feature in order to avoid confusion between the numbers of the versions (See B.1) . To illustrate this approach, Figure 5.4 contains a sample of features created. The last row of the graph shows a version that the model hasn't been trained on and the information that would be extracted in this case.



documentProducer	documentProducer_Adobe	documentProducer_PDF	documentProducer_Library	documentProducer_21	documentProducer_01
Adobe PDF Library 21.01	1	1	1	1	1
Adobe PDF Library 22.05	1	1	1	0	0

**Figure 5.4:** Transformation of a sample of documentProducer into categorical features. The blue color indicates a converter which was previously seen by the model and the red color an updated converter.

### 5.1.5 CPV Code

The last feature needed to be converted before proceeding to the implementation of the models was the **cpvCode**. Similarly to the classification label, each row of the dataset had multiple cpvCodes. To convert these codes into features, the sklearn's **Multibinarizer()** was used resulting in additional 2150 categorical features. The drawback of this feature was that the size of data was increased which affected the overall computational speed.

Concatenating all the different categorical features extracted returned 2677 columns in the final preprocessed and transformed dataset.

## 5.2 Model implementation

Although the models tested in this research had different architectures, the evaluation was performed in the same way and followed the same process. As mentioned in Chapter 2, a k-fold cross validation was used for every model with  $k=5$  which means that the dataset was divided five times into different training and testing sets. The results of the k-fold cross validation were combined and returned the average F1-score which was the main metric of evaluating the models. In addition, depending on the performance of some models, a method was used to further investigate and possible increase the performance.

The method used is a function which belongs to sklearn library called **GridSearch()** which takes as an input, a model and a list of parameters to be evaluated. Each model architecture has multiple parameters to be considered and it is challenging to select the optimal values for each parameter. GridSearch trains and tests the same model with different combinations of these parameters and returns the optimal values of those parameters. However, it was not used for every model since in the case of neural networks, the computational time for doing such search was increased drastically compared to other models.

### 5.2.1 Support Vector Machines

To implement the model using the Support Vector Machines Algorithm, the sklearn was used which provides a method for creating such algorithm. The most important parameter of the SVM is the *kernel* which is the way of separating the features within a dataset. There are four different kernels:

- *linear*: A linear kernel which is a dot product of any two given observations
- *poly*: A polynomial kernel is a more generalized form of the linear kernel and can distinguish curved or nonlinear input space.
- *rbf*: A Radial basis function kernel that can map an input space in infinite dimensional space.
- *sigmoid* : A sigmoid kernel which is similar to the sigmoid activate function in neural networks.

During the testing, the *linear* kernel was used for its two characteristics:

- It is mostly used when there are a large number of features in a data set.
- It is faster compared to the other kernels.

### 5.2.2 Logistic Regression

One of the most commonly used methods in binary classification problem is the Logistic Regression. Sklearn provides the function *LogisticRegression()* which contains multiple parameters. The *optimization algorithm* (solver) which is the algorithm for optimizing the performance of the model depending on the features and the *penalty*

which is the regularization (either L1 or L2) corresponded to the solver. To further investigate the model, all the different solvers were tested in the GridSearch and the default values of the model (solver='lbfgs' and penalty = 'l2') were returned and thus used for the final evaluation and comparison. The solver *lbfgs* stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno and it approximates the second derivative matrix updates with gradient evaluations.

### 5.2.3 Random Forest

An ensemble model used next in this research was Random Forest Classifier. In addition to the first two models, the RF is also a built in function of sklearn. The main difference is the amount of parameters that need to be checked and can affect the performance. the two parameters that were manipulated were:

- **n\_estimators** : The number of trees in the model (default = 100)
- **max\_depth**: The maximum depth of the tree. If the value is "None" all the nodes are expanded until all leaves have been discovered.

To find the optimal parameters, a grid search was made with a list of possible value that can be found in B. The resulted values were 100 n\_estimators and "None" as the max\_depth of the tree.

### 5.2.4 XGBoost

The most effective model in terms of computational speed and performance was the XGBoost Classifier. To implement this algorithm although, a new library called **xgboost** was installed, the functionality along with most of the parameters in the Random Forest Classifier were the same. The difference is that XGBoost also contains some additional parameters regarding the boosting phase of the algorithm. One of those parameters is called **learning rate** which determines the step size at each iteration while the algorithm optimizes toward its objective. A low learning rate makes the computation slower, and requires more rounds to achieve the same reduction in residual error as a model with a high learning rate. But it optimizes the chances to reach the best optimum. The learning rate along with the n\_estimators and max\_depth as in the case of RF were used in the Grid Search to provide the best possible set up given a list of values for all three parameters. The optimal set up provided by the GridSearch was:

- **learning\_rate** : 0.1
- **n\_estimators** : 500
- **max\_depth** : 5

### 5.2.5 Neural Network

Although in all the previous models, it was possible to do an effective Grid Search to approach an optimal set up, in neural networks is usually challenging to achieve that. Neural networks was dependent on the number of layers, number of neurons for each layers along with the rest of parameters such as learning rate or loss function. To have a unbiased way of comparing the performance between all the models, two neural networks were created.

The first one was a simple MLP neural network containing one hidden layer with 10 neurons. The activation function for the hidden layer was ReLu and as this was a binary classification problem, the sigmoid function was used as the activation function in the output layer. After designing the architecture of the network using the keras library *binary\_crossentropy* was used as the loss function and *adam* as the optimizer in the compiling step. Adam stands for Adaptive Moment Estimation and is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. The method is efficient when working with large problem involving a lot of data or parameters. The performance of the model during the training was calculated by monitoring the accuracy.

The second neural network had the same properties as the first one in terms of activation functions, loss function and optimization algorithm. The difference between them is the architecture which was more advanced. The second model contained three hidden layers with 512, 256, 128 neurons for each layer respectively. This change although in the next chapter will be proven better in terms of performance, it was the most computational expensive in terms of training speed.

To avoid overfitting, an Early Stopping function was added to both neural networks and were trained for 10 epochs each using a 5-fold cross validation. The use of early stopping was to monitor the loss during the training and if the loss had an increase for three straight epochs, the training would be stopped.



# 6

## Results

The next chapter is focusing on the results and a comparison between them. As mentioned before, all models followed a 5-fold cross validation and were evaluated on the accuracy, f1-score, recall and precision.

### 6.1 Important labels

During the data description in Chapter 3, it was brought up that the labels of the classification in the company's flow are divided into two categories, important and unimportant labels. Specifically, some labels are more common within a procurement than others and thus, makes them more important and worthy to be evaluated separately. The total number of labels is one hundred and forty five (145) but only thirty one (31) of them are more likely to be located in a procurement. In addition to the evaluation metrics that were used using the **sklearn** library, one metrics was created to calculate the accuracy based on only the important labels. The classifier in the main flow is a multi-label classifier which means that one or more labels can be classified for a certain text. To implement this metric, all the rows containing only labels which do not belong to the important set were excluded. The accuracy was then calculated by evaluating how many correct predictions were found for each model, divided by the total number of datapoints located in the test set. This accuracy will be referred to as  $Acc_i$  in the results in the next section.

$$Acc_i = \frac{\text{correct predictions of important labels}}{\text{total number of rows with important labels}}$$

### 6.2 Results

For every method covered in Chapter 5 there were two runs of evaluation. The first run was using the default values of each model that are provided by the built in functions in sklearn. The second run was made using the optimal values provided by the Grid Search. Specifically, every model apart from the neural network was given as an input in the Grid Search module. After running each method for every possible combination of the parameters specified, a set of optimal values for the input was returned. This input was used to run the cross validation after which, the average value for each metric was calculated. In addition the separate evaluation metric  $Acc_i$  was calculated regarding the important labels only.

An exception during the second test is the absence of the neural networks because they contain many parameters that can be optimized and many different values that each parameter can take. Thus, the architecture was already specified and tested using the parameters mentioned in Chapter 5.

The results for the 5-fold cross validation with the default values for each method can be shown in Table 6.1. The algorithms using decision trees ensembles like XGBoost and Random Forest performed better in comparison to simpler methods as Logistic Regression and Support Vector Machines and they had a slight improvement over the neural networks and specifically the case of the advanced, three hidden-layers neural network. In terms of the weighted ,between the precision and the recall, F1-score, the best model was XGBoost achieving 89,9% on the test set. Evaluating the accuracy for only the 31 important labels (See Table 6.2), the XGBoost Classifier outperformed every model achieving 91,2%.

Model	F1-score	Precision	Recall	Accuracy
Random Forest	89,5 %	89,5 %	89,6 %	89,6 %
Logistic Regression	87,1 %	87,1 %	87,2 %	87,1 %
Support Vector Machines	87 %	87 %	87,1 %	87,1 %
XGBoost	89,9 %	90 %	89,8 %	89,9 %
Base Neural Network	87,8 %	87, 9 %	87,7 %	87,8 %
Advanced Neural Network	89,5 %	89,6 %	89,4 %	89,5 %

**Table 6.1:** Performance of the Machine learning models without Grid Search

Model	Acc <sub>i</sub>
Random Forest	90,8 %
Logistic Regression	88,5 %
Support Vector Machines	88,8 %
XGBoost	91,2 %
Base Neural Network	88,9 %
Advanced Neural Network	89,9 %

**Table 6.2:** performance of the Machine learning models on important labels without Grid Search

In addition, running the 5-fold cross validation with the optimal parameters given during the Grid Search, verified that XGBoost was the most effective model for this problem. Each model improved less than XGBoost Classifier which had a significant 0.6% increase in the f1-score in comparison to the default values (Table 6.3). Moreover, XGBoost achieved 91,8% on the important labels which also increases 0,6% over the first run (See Table 6.4).

Model	F1-score	Precision	Recall	Accuracy
Random Forest	89,7 %	89,6 %	89,8 %	89,7 %
Logistic Regression	87,2 %	87,3 %	87,2 %	87, 2 %
Support Vector Machines	87,1 %	87,1 %	87,2 %	87, 2 %
XGBoost	90,5 %	90,6 %	90,5 %	90,5 %

**Table 6.3:** Performance of Machine learning models with Grid Search

Model	Acc <sub>i</sub>
Random Forest	91,1 %
Logistic Regression	89 %
Support Vector Machines	89,1 %
XGBoost	91,8 %

**Table 6.4:** Perfomance of Machine learning models on important labels with Grid Search

A summary with all the results for every model, including the evaluation metrics for both with and without grid search, can be shown in the table below. Based on the evaluation results, it was concluded that XGBoost was the most effective model both in computational speed and the performance in both precision and recall. Since the topic of the thesis was to be able to identify documents with possible errors in either the classification or in the formatting, recall was an important metric to look at, apart from f1 score. The reason was the trade off between the importance of missing problematic documents and reporting correct documents as problematic. To further investigate this, a confusion matrix was created for each model, showing the percentage of correct and incorrect predicted rows in comparison to the actual values. These plots (See Appendix B.2) indicated that all the models were more capable of miss-predicting problematic rows as correct rather than predicting correct rows as problematic. The confusion matrices were calculated given the optimal values from the Grid Search.

Model	Without Grid Search					With Grid Search				
	F1-score	Precision	Recall	Accuracy	Acc <sub>i</sub>	F1-score	Precision	Recall	Accuracy	Acc <sub>i</sub>
Random Forest	89,5 %	89,5 %	89,6 %	89,6%	90,1 %	89,6 %	89,6 %	89,7 %	89,7 %	91,1 %
Logistic Regression	87,1 %	87,1%	87,2 %	87,1 %	88, 7 %	87,2 %	87,2 %	87,3 %	87,2 %	89,2 %
Support Vector Machines	87 %	87 %	87,1 %	87,1 %	88,9 %	87,1 %	87,1 %	87,2 %	87,2 %	89,1 %
<b>XGBoost</b>	<b>89,9 %</b>	<b>90 %</b>	<b>89,8 %</b>	<b>89,9 %</b>	<b>90,7 %</b>	<b>90,5 %</b>	<b>90,5 %</b>	<b>90,6 %</b>	<b>90,6 %</b>	<b>91,8 %</b>
Base Neural Network	87,8 %	87, 9 %	87,7 %	87,8 %	88,9 %	-	-	-	-	-
Advanced Neural Network	89,5 %	89,6 %	89,4 %	89,5 %	89,9 %	-	-	-	-	-

**Table 6.5:** Machine learning models performances summary for all metrics both with Grid Search and without



# 7

## Conclusion

The final chapter covers the summary and discussion of the results along with the certain limitations which were noticed during the thesis. In addition, some suggestions were given regarding possible future work to improve the results.

### 7.1 Discussion

Given the data provided and the feature extraction and transformation steps added to the research, the most effective model in terms of f1-score was XGBoost as shown in the previous chapter. The best f1-score that was found within all the tests was 90,5%. The initial assumption was that all models will have difficulties in predicting problematic texts since the quality of the data and the possible features that could have been extracted were not enough. However, when it comes to classification problems, the result of XGBoost Classifier is promising and indication that the model can be used for this warning about problematic documents.

On the other hand, considering that each document has more than one paragraph (previously mentioned as row or datapoint of the dataset) and given that the model has 90,5% it might not be as effective in the document level since there is a high probability of missing one paragraph in a document. In addition, as mentioned in the previous chapter, even the highest evaluated model, had more false positives than negatives which makes the model sensitive in allowing problematic rows and predict them as correct.

Finally, since the data are increasing and the model will be trained frequently, XGBoost Classifier, as suggested by many researchers will be useful not only because of its performance results but also for its computational speed which was better in comparison to an advanced neural network or a Random Forrest algorithm with the same properties (number of trees and depth).

### 7.2 Limitations

During the research, two types of challenges were encountered. First, in Chapter 3 during the data gathering, it was mentioned that the data gathered were for a period of five months because of the new model, introduced in October 2021, and the formatting error reporting feature, introduced in November 2021. Although the total amount of data were enough for an initial testing and evaluation, there was a

need for more datapoints which would improve the results. In addition, despite of the annotations for the classification flow in company's process, the data regarding formatting issues were extremely insufficient (almost 1% of the total data) and the model would perform dramatically worse if implemented for predicting only rows that may contain formatting errors.

Second, in the process of feature engineering, the features selected were the ones that, initially thought, would have an impact in resulting in problematic classifications. However, there were more data available in different databases of the company that could potentially improve the results. Due to time limitation, not all features could be investigated and used in the best way possible to achieve better performance. Moreover, constructing a more advanced neural network with more layers and tuning its parameters was impossible due to both time limitation and computational power needed for these tasks.

### 7.3 Future Work

In a future research, it would be important to repeat the process and evaluate again the methods, given that the quality of the data would be better and the volume would be larger. In specific, when there will be enough data with formatting errors, a new analysis can be made to evaluate the model on only rows containing such errors and verify that the model can be useful in such errors as well.

Moreover, there can be an additional analysis on other databases or new metadata which could be added in the future and possibly extract more features that would help to identify whether a document to be problematic. For any new feature and for the existing ones would be worth to evaluate the results with different approaches of data preparation. For instance, in the document converter, it was mentioned that the approach followed was because of the version of the program as it was important to the label classification in the main flow. However, there might be other approaching of keeping the versioning and grouping of the features that would result an improvement of the results.

Last but not least, given that the time and hardware capabilities, it is important to further investigate different architectures of the model and especially in the neural network case. Although XGBoost classifier is commonly used in competition regarding binary classification, it is often neural networks that achieve the best performance. Thus, evaluating different architectures in terms of not only hidden layers and neuron but also different algorithms like Convolutional Neural Networks (CNN) or Long-Short Term Memory Neural Networks (LSTM) might result in an improvement of the performance. There are also additional techniques that prevent the model to overfit and improve its predictive capabilities such as Drop Out layers that can be inputted between the hidden layers whose functionality is to discard some training batches in order to reduce the risk of training on similar data. The reason why this was avoided in this thesis was because the data available were not enough and thus, such technique could not be used.

## 7.4 Conclusion

In the machine learning area, binary classification with supervised learning methods is a common problem which can be tackled via different methods. The goal of this research was to evaluate different ML algorithms than can be used for predicting problematic rows is a document based on either classification or formatting errors during the main flow of the company. During the data gathering, there was an uncertainty of succeeding in the research given the fact that the amount of data that were given seemed to be insufficient to perform such task. After analyzing the data, it was found that certain features resulted in problematic texts more frequently than others. A thorough investigation and preprocessing of those features was necessary before inputting them into the models in which each feature had to be transformed into categorical values. The approach for doing that was different for each feature with the producer of the document being the most challenging feature given that it was necessary to include the version of the program since it was found that different versions had differences in the percentage of failure.

The ML algorithms selected during the research were the Logistic Regression, the Random Forest, the Support Vector Machine, the XGBoost Classifier and two neural networks with different complexities. To evaluate the methods, the F1 score, precision, recall and accuracy were used along with an additional metric which calculated the accuracy of only texts in which at least a label from a set of important labels (provided by the company) were classified during the main flow. Those metrics were calculated after a 5-fold cross validation performed for each model and using an algorithm for tuning the parameters called Grid Search for every model except the two neural networks

Given the time and resources that were given, XGBoost Classifier managed to achieve 90,5% F1 score and accuracy and 91,5% accuracy on the set of special labels and thus can be used for identifying problematic documents. Its advantage compared to the other methods investigated in this thesis was the computational speed in addition to high performance results. However, the results would improve if more data were available and more features for each document could be extracted and analyzed.





# Bibliography

- [1] Wikipedia contributors, “Procurement — Wikipedia, the free encyclopedia.” <https://en.wikipedia.org/w/index.php?title=Procurement&oldid=1086380929>, 2022. [Online; accessed 6-May-2022].
- [2] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” in *Machine learning techniques for multimedia*, pp. 21–49, Springer, 2008.
- [3] W. Koehrsen, “Overfitting vs. underfitting: A complete example,” *Towards Data Science*, 2018.
- [4] D. Bashir, G. D. Montañez, S. Sehra, P. S. Segura, and J. Lauw, “An information-theoretic perspective on overfitting and underfitting,” *CoRR*, vol. abs/2010.06076, 2020.
- [5] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, “The ‘k’ in k-fold cross validation,” in *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 441–446, i6doc. com publ, 2012.
- [6] S. B. Kotsiantis, “Decision trees: a recent overview,” *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013.
- [7] Wikipedia contributors, “Ensemble learning — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=Ensemble\\_learning&oldid=1067251577](https://en.wikipedia.org/w/index.php?title=Ensemble_learning&oldid=1067251577), 2022. [Online; accessed 11-May-2022].
- [8] M. Belgiu and L. Drăguț, “Random forest in remote sensing: A review of applications and future directions,” *ISPRS journal of photogrammetry and remote sensing*, vol. 114, pp. 24–31, 2016.
- [9] D. Meyer and F. T. Wien, “Support vector machines,” *R News*, vol. 1, no. 3, pp. 23–26, 2001.
- [10] T. Chen and C. Guestrin, “XGBoost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, aug 2016.
- [11] Wikipedia contributors, “Rectifier (neural networks) — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=Rectifier\\_\(neural\\_networks\)&oldid=1086744769](https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=1086744769), 2022. [Online; accessed 12-May-2022].
- [12] Wikipedia contributors, “F-score — Wikipedia, the free encyclopedia.” <https://en.wikipedia.org/w/index.php?title=F-score&oldid=1086969326>, 2022. [Online; accessed 4-June-2022].
- [13] Wikipedia contributors, “Pandas (software) — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=Pandas\\_\(software\)&oldid=1079541180](https://en.wikipedia.org/w/index.php?title=Pandas_(software)&oldid=1079541180), 2022. [Online; accessed 8-May-2022].

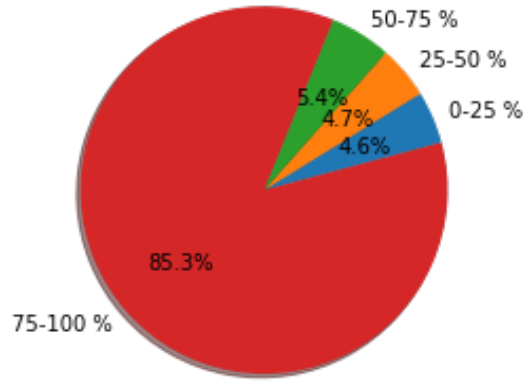


# A

## Appendix 1

Confidence Score	Datapoints
0-25%	804
25-50%	808
50-75%	937
75-100%	937
>95%	14806

**Table A.1:** Number of correctly classified rows with respect to confidence scores.



**Figure A.1:** Distribution of confidence scores intervals across all the correct classifications.

Name of Producer	Number of rows
iText® 7.1.16 ©2000-2021 iText Group NV (Mercell; licensed version); modified using iText® 7.1.16 ©2000-2021 iText Group NV (Mercell; licensed version)	885
unknown	448
Microsoft® Word 2016	208
ExpertPdf v16.0.0	188
Microsoft® Word for Microsoft 365	180
iText 7.1.16 2000-2021 iText Group NV (Mercell; licensed version); modified using iText 7.1.16 2000-2021 iText Group NV (Mercell; licensed version)	180
iTextSharp 4.1.2 (based on iText 2.1.2u)	129
Adobe PDF Library 20.4.63	119
Microsoft® Word 2013	85
Adobe PDF Library 21.11.71	34
Microsoft: Print To PDF	32
Microsoft Word 2016	29
Microsoft Word fr Microsoft 365	26
NONE	26
Adobe PDF Library 15.0	25
Skia/PDF m79	20
Microsoft® Word for Office 365	20
HiQPdf 10.17	15
Bluebeam Brewery 5.0	14
Adobe PDF Library 21.7.131	13

**Table A.2:** Count of incorrect rows for different PDF producers.

# B

## Appendix 2

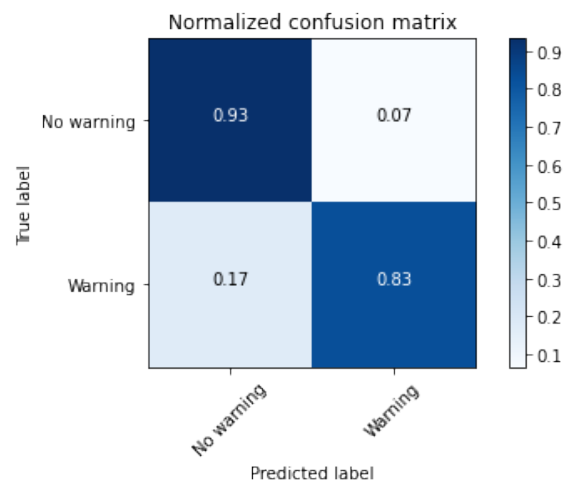
### B.1 Source code for data preparation

```
% Convert columns containing string to categorical columns
% Applied to filepath, classificationLabelId, documentLanguage
pandas.get_dummies(dataframe, columns = categorical_column)

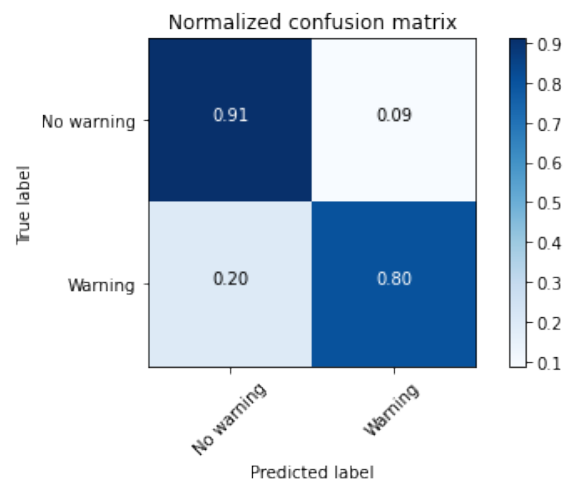
% Count vectorizer to transform non categorical features
% Applied to documentProducer
vectorizer = CountVectorizer()
transformed = vectorizer.fit_transform(df_column.tolist())
df_producer = pd.DataFrame(transformed.toarray(),
                           columns=vectorizer.get_feature_names()).add_prefix(prefix)

%MultiLabelBinarizer to transform columns containing multiple labels
%Applied to cpvCode
mlb = MultiLabelBinarizer()
res = pd.DataFrame(mlb.fit_transform(column),
                   columns=mlb.classes_,
                   index=cpv_codes.index).add_prefix(prefix)
```

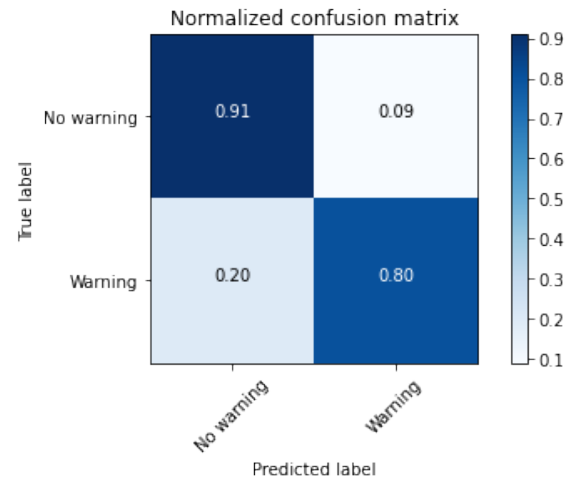
## B.2 Confusion matrices



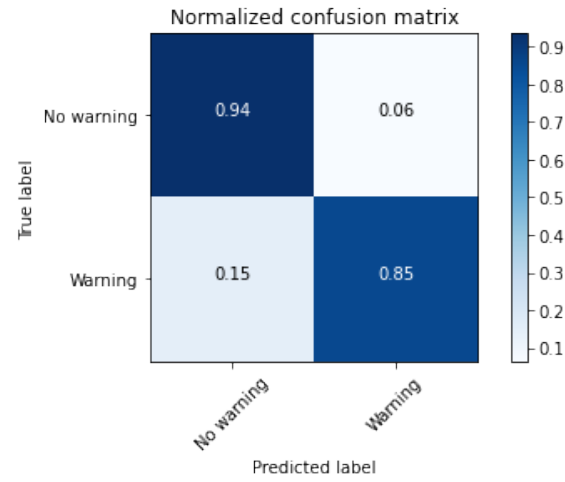
**Figure B.1:** Confusion matrix of the Random Forest Classifier.



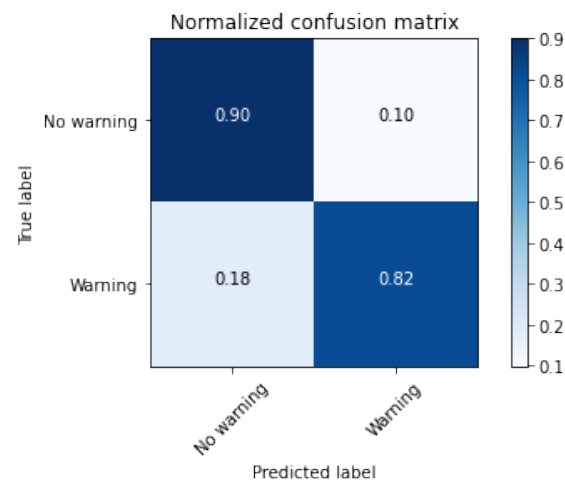
**Figure B.2:** Confusion matrix of the Logistic Regression.



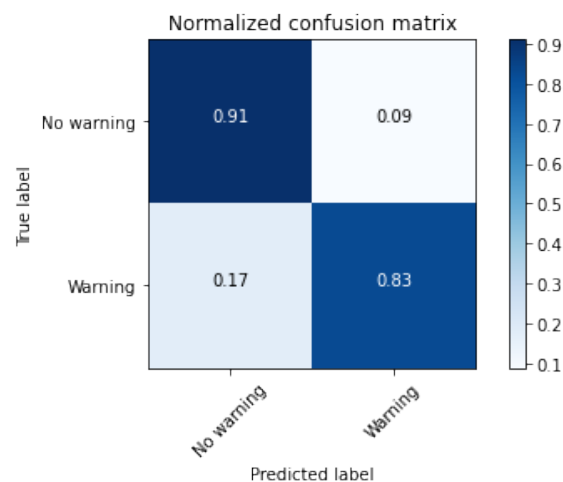
**Figure B.3:** Confusion matrix of the Support Vector Machines.



**Figure B.4:** Confusion matrix of the XGBoost Classifier.



**Figure B.5:** Confusion matrix of the Base Neural Network.



**Figure B.6:** Confusion matrix of the Advanced Neural Network.