

**A LEARNING-BASED DEMAND
CLASSIFICATION SERVICE WITH USING
XGBOOST IN INSTITUTIONAL AREA**

**A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of**

MASTER OF SCIENCE

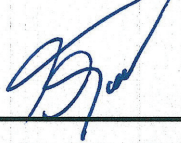
in Computer Engineering

**by
Çağrı GÜRAKIN**

**July 2019
İZMİR**

We approve the thesis of **Çağrı GÜRAKIN**

Examining Committee Members:



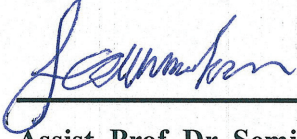
Assoc. Prof. Dr. Tolga AYAV

Department of Computer Engineering, İzmir Institute of Technology



Assist. Prof. Dr. Işıl ÖZ

Department of Computer Engineering, İzmir Institute of Technology



Assist. Prof. Dr. Semih UTKU

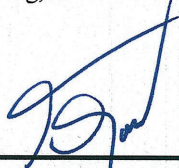
Department of Computer Engineering, Dokuz Eylül University

16 July 2019



Assoc. Prof. Dr. Tolga AYAV

Supervisor, Department of Computer
Engineering, İzmir Institute of Technology



Assoc. Prof. Dr. Tolga AYAV

Head of the Department of Computer
Engineering in İzmir Institute of Technology

Prof. Dr. Aysun SOFUOĞLU

Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank my advisor Assoc. Prof. Dr. Tolga Ayav for his ultimate patience and guidance throughout my thesis process. His encouragement and suggestions enabled me to complete my thesis.

I must express my gratitude to my mother Mukaddes Seçkin for providing me unfailing support throughout my years of study and through the process of writing this thesis. A special thanks to my wife Şeliha Gürakın, for her continuous support throughout my research period.

I thank my colleagues in Yaşar Bilgi (Astron) for their support on my research, it would become really difficult to accomplish my thesis without their understanding.

ABSTRACT

A LEARNING-BASED DEMAND CLASSIFICATION SERVICE WITH USING XGBOOST IN INSTUTIONAL AREA

This study, purposes to explain the development stages and methodology of data classification service that has a text-based adaptable programming interface. One of the successful classification algorithms, XGBoost, was preferred in the study. The dataset that is used in the study obtained by 'Digital Business Tracking Application' of a name anonymized company. The dataset is tested by using different classification algorithms and detailed performance evaluation was conducted. As a result, highest accuracy rate is obtained with 'Data Classification Service' which was developed by using XGBoost algorithm.

Keywords: Supervised-learning, multinomial classification, XGBoost, text-classification, natural language processing.

ÖZET

KURUMSAL ALANDA XGBOOST İLE ÖĞRENME TABANLI TALEP SINIFLANDIRMA SERVİSİ

Bu çalışma, metin-tabanlı, veriler için uyarlanabilir bir programlama arayüzüne sahip, sınıflandırma servisi geliştirme aşamalarını ve çalışmada takip edilen metodolojiyi konu alır. Çalışmada, başarılı sınıflandırma algoritmalarından biri olan XGBoost tercih edilmiştir. Çalışmada kullandığımız veri kümesi, bilgilerini anonimleştirdiğimiz bir şirketin ‘Dijital İş Takip Uygulaması’ aracılığı ile elde edilmiştir. Veri kümesi farklı sınıflandırma algoritmaları ile de test edilmiş ve performans değerlendirmeleri yapılmıştır. Çalışmalarımızda en yüksek doğruluk oranı XGBoost algoritması ile geliştirdiğimiz veri sınıflandırma servisi ile elde edilmiştir.

Anahtar kelimeler: Gözetimli öğrenme, çok terimli sınıflandırma, XGBoost, metin sınıflandırma, doğal dil işleme.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF TERMS AND ABBREVIATIONS	xi
CHAPTER 1. INTRODUCTION	1
1.1. Motivation	1
1.2. Contributions.....	1
1.3. Organization of Thesis	2
CHAPTER 2. BACKGROUND.....	3
CHAPTER 3. IMPLEMENTED TECHNOLOGIES.....	5
3.1. XGBoost Classification Technique (State of the Art).....	5
3.2. Other Models and Scores for Classification.....	8
3.2.1 Random Forest.....	9
3.2.2. Naive Bayes Classifier	10
3.2.3. Decision Tables	11
3.2.4. Random Tree	12
3.2.5. ZeroR Classifier.....	13
3.2.6. OneR Classifier	14
3.2.7. Other Classification Results	15
3.3. Used Other Technologies in DCS (Environments, Packages and Libraries).....	15
3.3.1 Python Panda Data Frames.....	16
3.3.2. Scikit Learn Application Interface	16
3.3.3. Matplot Library	17
3.3.4. Numpy Package.....	18
3.3.5. Graphviz Application Interface	17
CHAPTER 4. PROPOSED SOLUTION TO THE PROBLEM.....	19
4.1. A Sample: Digital Business Tracking Application and Basics of DCS Implementation	19
4.1.1. A Digital Business Tracking Application.....	19
4.1.2 Demand flows in Digital Business Tracking Application	21
4.2. Basics of Data Classification Service (DCS).....	22
4.2.1 About the Data Source and Content	25
4.2.2. Pre-processing Methods Used in DCS	30

4.2.3. Demo of Data Classification Service	36
CHAPTER 5. EXPERIMENTAL RESULTS.....	38
5.1 Manual Measuring Accuracy Score for Data Classification Service.....	38
5.2. Auto Generated Scores for Data Classification Service	39
CHAPTER 6. FUTURE WORKS AND CONCLUSION	45
REFERENCES.....	48

LIST OF TABLES

<u>Table</u>	<u>Page</u>
Table 3.1. Comparison of major tree boosting systems.....	6
Table 3.2. XGBoost algorithm scores in DBTA dataset by using department.....	8
Table 3.3. Random Forest scores in DBTA dataset by using department classes..	9
Table 3.4. Naïve Bayes Classifier scores in BTA dataset by using department. .	11
Table 3.5. Decision Table scores in BTA dataset by using department classes...	12
Table 3.6. Random Tree Algorithm scores in BTA dataset by using department.	13
Table 3.7. ZeroR Algorithm scores in BTA dataset by using department classes	14
Table 3.8. OneR Algorithm scores in BTA dataset by using department classes.	15
Table 4.1. Shows the average of cluster-based filters on department in DBTA dataset.....	34
Table 4.2. Shows the average of token-based filters on department in DBTA dataset.....	34

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 3.1. A Part of boosting tree that is used in DCS model.	5
Figure 3.2. Algorithms and Scores using 1000 data from DBTA Dataset with using Weka tools. (XGB score is observed with DCS).	15
Figure 3.3. A view from installation step of Graphviz with Anaconda.	18
Figure 4.1 Demand flow in A Digital Business Tracking Application.	21
Figure 4.2. A test-card that shows the model that accepts responsibility as a target..	23
Figure 4.3. An example prediction for specialist and department models with using the Data Classification Service.	24
Figure 4.4. An example of the anonymized words in the dataset. Replacement of the words are red marked.	25
Figure 4.5. The test system specification.	28
Figure 4.6. The graph shows count of demands on a department in the DBTA dataset. The yellow bar represents a department name called: 'User Support' in the natural set.	29
Figure 4.7. After stemming some of the words are deleted by the stop-words filter.....	31
Figure 4.8. The figure shows a short list for Turkish Stop-word list.	31
Figure 4.9. Pseudo code of a DCS function that removes a line which includes at least one corrupted words or a which has word count smaller than average token size. ('i' assumed as an incremental integer)	33
Figure 4.10. A graph that represents feature importance on DBTA Dataset.	35
Figure 4.11. Graphical user interface of the Data Classification Service demo.	36
Figure 5.1. Manually evaluated prediction results for department.	38

Figure 5.2. Manually evaluated prediction results for specialist. 38

LIST OF TERMS AND ABBREVIATIONS

CSV	Comma Separated Values
DCS	Data Classification Service
BTA	Business Tracking Application
DBTA	Digital Business Tracking Application
GBM	Gradient Boosting Machines
GUI	Graphical User Interface
IT	Information Technologies
TF-IDF	Term Frequency – Inverse Document Frequency
VPN	Virtual Private Network
k-NN	K-nearest neighbor
LDAP	Lightweight Directory Access Protocol
ML	Machine Learning
NLP	Natural Language Processing
NB	Naive Bayes
XGB	Extreme Gradient Boosting
OTP	One-Time password
SAP	System Application Products
SVM	Structural Vector Machine
SLA	Service Level Agreement

XGBoost.....Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

1.1. Motivation

Software applications (Jira, HPQC, DBTA and etc.) which don't have specific rules for dispatching requests to related person, group or other target that is related with request may cause some problems such as time loss & late solution, waste of labor force and wrongly directed requests.

A name anonymized company integrated a manual demand dispatching process on their business tracking application. (refer to: 4.1.1. A Digital Business Tracking Application) In this real-life experiment, the company produced different kinds of precious data for the data scientists. The thesis closely correlated with the data collected on the experiment during 2018. In this time interval, stakeholders observed some malfunctions of the manual dispatching such as time loss before demands are dispatched, effort consuming in order to find correct department or specialist for demands.

Furthermore, manual dispatching caused many wrongly directed demands and around 2166 hours are spent annually for dispatching of approximately 65000 demands. (average time for dispatching a demand is stored in the DBTA as two minutes)

1.2. Contributions

Universal interface of 'Data Classification Service' provides modular and adaptable structure for different text classification needs in different platforms.

Basically DCS dispatches demands automatically, that comes from non-technical employee to the related specialist or department of an organization. Simple infrastructure of the service provides many advantages like 'time efficiency to solve demands', 'finding the right specialist or department for demands' and 'decreasing dispatching effort' in industrial use.

In manual dispatching demands may lose time on the dispatcher before dispatcher process the demand. In a corporate company this time loss can be critical. DCS will eliminate this service time.

The other problematic issue on manual dispatching is incorrect assignments of demands. The DCS offers high accuracy for finding right department at first try.

Another advantage of using the service is to reduce the need for staff to make dispatching. Each demand starts its journey with the additional dispatcher time when it manually dispatched. Basically its observed for one-year period; for a dispatching operation a dispatcher spend nearly two minutes in each demand. The Data Classification Service, reduces or eliminates those kind of the human-oriented dispatching issues.

1.3. Organization of Thesis

The thesis begins with a short introduction part that mentions the problem definition of the thesis and continues with the advantages of using the Data Classification Service. (Motivation and Contributions subtitles)

The second chapter is related with literature review (background). Briefly, some of the previous works about text classification and supervised learning methods are summarized in this chapter.

The third chapter is about implemented technologies in this study. The XGBoost algorithm and the other technologies that are used in the thesis are explained here. Also, some other popular algorithms are tested with Weka tool to provide a basis for the work done with XGBoost and DBTA dataset. (Other Models and Scores for Classification subtitle.)

The fourth chapter is about the solution that is proposed in the study. This part begins with ‘A Sample: Digital Business Tracking Application’ and explains the Data Classification Service in detail and gives some results that are succeeded with the service.

The last chapter is about future works and conclusion. The chapter explains what kind of improvements can be made on the DCS and what kind of applications can be built with the infrastructure and summarizes advantages and disadvantages of using the ‘Data Classification Service’ with the current perspective of this study.

CHAPTER 2

BACKGROUND

Schapire and Singer's (2000) study which is called as “BoosTexter: A Boosting-based System for Text Categorization”, two extensions of AdaBoost have been used; with the first extension, all correct labels was determined and with the second one, it is aimed to rank these correct labels so as to get highest ranks. Then, they presented a new text categorization method named as Boostexter and compared its results with the other four different text and speech categorization algorithms. In terms of several measures, they experimented that as a multilabel text categorization method BoosTexter’s performance was better than other methods which they studied.

Aly (2005) published a study named as “Survey on Multiclass Classification Methods” he explained and applied different classification techniques for multiclass classification algorithms. Mainly, he studied three different approaches in order to solve multiclass classification problem. While first approach was to apply binary classification methods to solve multiclass problem, the second one was decomposition of the binary classification and the third approach was using binary trees to handle multiclass problem.

Kotsiantis (2007), published a study called as “Supervised Machine Learning: A Review of Classification Techniques”, in which he describes supervised machine learning methods in details. The aim was not only explaining and define different methods in general concept, rather than that he described critical features of the mentioned methods. In addition to that, previous studies on machine learning subject were available which enable readers to get citations about different issues. He also analyzed whether integration of two or more algorithms gives better performance or not, and even if integrating or assembling has some advantages, he suggested it for the researchers that try to get best possible accuracy for classification.

Aurangzeb, Baharudin, Lee and Khan (2010) study called as “A Review of Machine Learning Algorithms for Text Documents Classification” was published. They aimed to provide a detailed review for those who want to learn available machine learning algorithms, document representation techniques and feature selection. Some of the

algorithms that they examined are Naive Bayes (NB), Structural Vector Machine (SVM) and K-nearest neighbour (k-NN).

After they describe and compare these algorithms in terms of usage and results, they concluded that still there is a need for more studies on these areas and different techniques should be developed for text mining, classification, semantics, spam filtering in order to get better and accurate results.

A study was published on Journal of Machine Learning Research, (2011), by Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, Duchesnay. This study's called as "Scikit-learn: Machine Learning in Phyton". In this study, they describe Phyton language and its usage both on academical researches and in the industry.

As a machine learning toolbox, they describe the advantages of using Scikit-learn in detail and compared the results with Scikit-learn and other libraries in Phyton such as Mlpy and Pybrain. Depending on their study, they stated that Scikit-learn is more preferable in terms of its easy usage and with its high level language. Besides these advantages, Scikit-learn also provides better and easier comparison for supervised and unsupervised machine learning algorithms.

Tiangi and Carlos (2016), published a comprehensive study which is "XGBoost: A Scalable Tree Boosting System". They described the advantages of using gradient tree boosting system called as XGBoost, they stated that besides providing state-of-art-results and this system enables researchers to get better performance among other methods. Tiangi and Carlos also made contributions to the literature by studying and analyzing on out-of-core computation, cache aware and sparsity- aware learning. They provided a novel sparsity aware algorithm for parallel tree learning. They founded and stated that XGBoost has good performance for real world scale problems even with the minimal resources.

CHAPTER 3

IMPLEMENTED TECHNOLOGIES

3.1. XGBoost Classification Technique (State of the Art)

Recently the importance of machine learning has been recognized in many areas. Machine learning algorithms are everywhere in our lives, from online shopping, smart spam classifiers, advertising systems to fraud detection we come across with any applications of it. (Chen Tianqi and Guestrin Carlos, 2016)

Companies are also benefited from these machine learning algorithms in various areas such as finding target clients via right advertises, detecting the demanded goods and services on the market with demand forecasting systems. (Chen Tianqi and Guestrin Carlos, 2016) Although there are many methods used for machine learning, gradient tree boosting is one of the most preferred technique. Gradient tree boosting is also called as gradient boosting machines. (GBM) (Natekin Alexey and Knoll Alois, 2013)

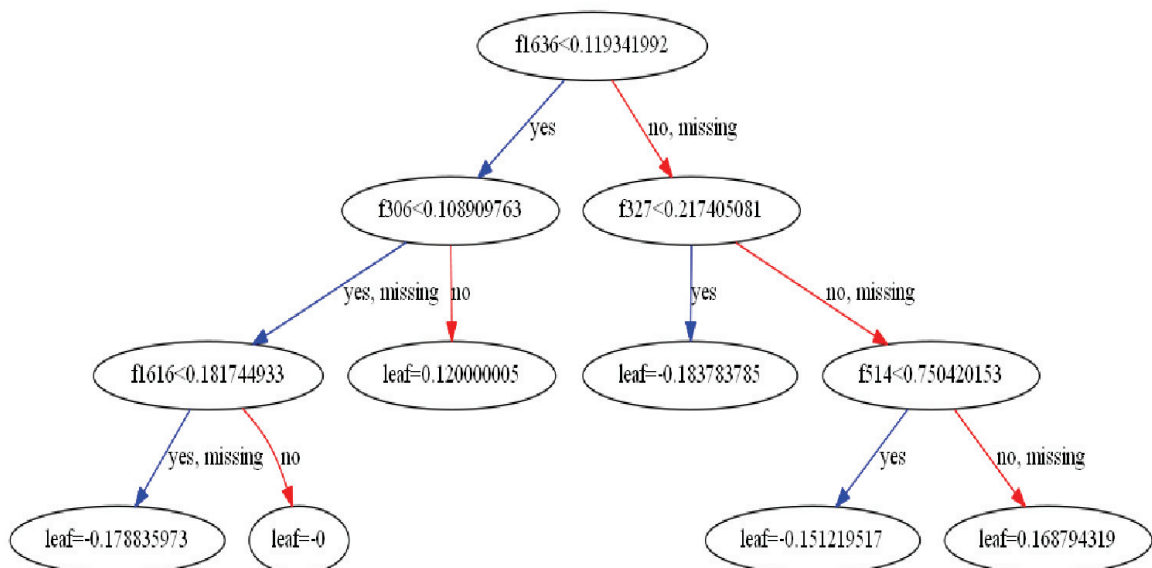


Figure 3.1. A Part of boosting tree that is used in DCS model.

In this context, boosting refers to add new models to the ensemble in sequence, by doing that a new weak model is trained in accordance with the error of the whole learning that is made to completed extent. It was important and necessary to set up a connection with the statistical framework as a result, gradient based formulation of boosting methods has been formed (Friedman Jerome, 2001) These methods hereafter are called as gradient boosting machines. This formulation has been methodological base for the future developments on gradient boosting model. (Natekin Alexey and Knoll Alois, 2013)

Table 3.1 Comparison of major tree boosting systems. (Chen Tianqi and Guestrin Carlos, 2016)

<i>System</i>	<i>exact greedy</i>	<i>app. global</i>	<i>app. local</i>	<i>out-of-core</i>	<i>sparsity aware</i>	<i>parallel</i>
<i>XGBoost</i>	Yes	yes	Yes	yes	yes	yes
<i>pGBRT</i>	No	no	Yes	no	no	yes
<i>Spark</i>	No	yes	No	no	partially	yes
<i>MLlib</i>	No	yes	No	no	partially	yes
<i>H2O</i>	No	yes	No	no	partially	yes
<i>scikit-learn</i>	Yes	no	No	no	no	no
<i>R GBM</i>	Yes	no	No	no	partially	no

XGBoost can be defined as scalable machine learning system for tree boosting which is commonly used by researchers in order to get state of the art results. It is the abbreviation of “extreme gradient boosting” which is mainly preferred and used by data scientists for its high speed and performance. One of the co-creators of XGBoost is Tianqi Chen who studies on building machine learning systems and Carlos Guestrin published an article which gives details on XGBoost and its advantages. In 2016, Tianqi Chen announced that with its innovative system characteristics XG Boost provided approximately ten times faster solutions than other popular machine learning algorithms. (Chen Tianqi and Guestrin Carlos, 2016)

Also, Kaggle is a machine learning competition site, it is an online site for data scientists, they can find and publish data sets, besides being very popular and widely used tool by researchers, XGBoost also mostly implemented by Kaggle competitions. (Titanic Machine Learning from Disaster, 2018)

By downloading XGBoost to your machine, you can have an access from a different inter faces. In particular, it supports below mentioned interfaces (Natekin Alexey and Knoll Alois, 2013):

- Command Line Interface
- C++
- Python
- R
- Julia
- Java and JVM languages like Scala

Advantages of Boosting Trees

XGBoost has been implemented in a variety of machine learning applications, it is observed that the system is much faster than the available popular solutions on a single machine. One of the most distinctive features of this system is its scalability, by means of it the system ranks distributed data with optimum memory usage. Its flexibility in usage, easily accessible features as an open source software, provides high usage rates by the data scientists. (Chen Tianqi and Guestrin Carlos, 2016)

XGBoost can be used in all Windows, Linux and OS X due to its portability and consistence. First, it has been developed to be used Command Line Interface (CLI) nowadays it can be used in many interfaces including Python and Java. (Chen Tianqi and Guestrin Carlos, 2016) There are many more reasons why XGBoost is a widely implemented method for data scientists.

The following table contains the results calculated using pre-processing techniques with the DCS data set. Best scores obtained with the DCS. It should be noted that pre-processing techniques are used only in the algorithm while examining the results.

Table 2, shows XGBoost algorithm score on DBTA dataset by using department classes with DCS using DBTA dataset. The test is performed using 1000 randomly selected requests. 6 requests are deleted in run time in pre-processing. In the test supporter fields: explanation, company and owner columns are used. The model is created with 795 requests. Remained 198 of requests used for the testing. The highest score with this shortened data set, the highest score was obtained by DCS.

Table 3.2. XGBoost algorithm scores on DBTA dataset by using department classes with DCS.

<i>Algorithm type</i>	
<i>Accuracy score</i>	73.86 %
<i>Relation</i>	2018-department-1000.csv
<i>Test mode</i>	Standard test with 994 demand. Supporter information are explanation, company, owner columns.
<i>Date time</i>	2019-05-06 15:47:01.219610
<i>Duration</i>	0:00:07.968443
<i>Demands in set</i>	994
<i>Labels type</i>	Department
<i>Labels count</i>	16
<i>Train data</i>	795
<i>Test data</i>	198
<i>Terms in data</i>	17922
<i>Corpus in set</i>	4239
<i>Stop Words count</i>	519
<i>Model name</i>	model-xgb-department-994-%73.dat
<i>TF-IDF name</i>	tfidf-xgb-department-994-%73.dat

Other Important Features of XGBoost

1. **Regularization:** With XGBoost, it is possible to penalize complex models by L1 and L2 regularization.
2. **Weighted Quantile Sketch:** While other tree based algorithms are able to handle just quantile sketch in other words, data of equal weights; XGBoost is able to give solutions for weighted data.
3. **Block Structure for Parallel Learning:** Besides being known with its quickness, XGBoost also uses multiple cores on CPU which enables it to make split finding and sub-sampling.

(Chen Tianqi and Guestrin Carlos, 2016)

3.2. Other Models and Scores for Classification

In order to compare BTA Dataset classification results on different algorithms Weka is used. (test tool for ML algorithms) This part of the study compares the test

results. Random Forest Algorithm that is mentioned about its methodology has the second highest scores on the results.

3.2.1 Random Forest

Random forest has been proposed and introduced by Breiman, Leo (2001), with the aim of building a set of decision trees by random selected data. Briefly, random forest can be explained as learning algorithms which builds multiple decision trees and consolidate these trees together in order to obtain more definite results.

Table 3.3. Random Forest Algorithm scores on DBTA dataset by using department classes.

<i>Algorithm type</i>	<i>Random Forest</i>
<i>Accuracy score</i>	64.00 %
<i>Scheme</i>	weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
<i>Relation</i>	2018-department-1000.csv
<i>Instances</i>	999
<i>Test mode</i>	split 80.0% train, remainder test. Bagging with 100 iterations and base learner. weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities
<i>Time taken to build model</i>	0.64 seconds
<i>Time taken to test model on test split</i>	0 seconds
<i>Correctly Classified Instances</i>	130 - 65 %
<i>InCorrectly Classified Instances</i>	70 - 35 %
<i>Kappa statistic</i>	0.4233
<i>Mean absolute error</i>	0.0684
<i>Root mean squared error</i>	0.1754
<i>Relative absolute error</i>	82.7006 %
<i>Root relative squared error</i>	85.9817 %
<i>Total Number of Instances</i>	200

While splitting a node, random forest search for the best feature among other subset of features rather than searching for the most important one. (Breiman Leo, 2001)

Random forest is one of the machine learning algorithms which can be used for both classification and regression problems. The forest here refers to ensemble of decision

trees which are generally applied with the technique of bootstrap aggregating (bagging). Bagging method increases the overall result and enables researchers to avoid from overfitting so that it possible to run many trees. (Biau Gerard, 2012)

Random forest is widely preferred method in Bioinformatics because of the fact that it can deal with small sample size, high-dimensional feature space and complex data structures. (Qi Yanjun, 2012)

3.2.2. Naive Bayes Classifier

Naive Bayes are linear classifiers that can be applied on both binary and multi-class classification problems. This algorithm is called as Naive Bayes as the calculation of the probabilities are being made simpler to be able to observe their calculation. (Xu Shuo, 2016)

The model of Naive Bayes classifiers is based on Bayes theorem of which assumption is that features are mutually independent. The independence assumption is very important and interesting issue in here that despite being unrealistic in real data, the results on Naive Bayes classifier show us that it still performs. (Xu Shuo, 2016)

Naive Bayes classifiers can perform very well for small sample size data. Besides being easy to implemented, it is also fast and accurate that can be applied in different research areas.

Naive Bayes classifiers can be implemented in decision making of treatment processes, spam filtering.

Nevertheless, there are two main disadvantages of Naïve Bayes. One of them is that it leads to poor performance of this classifier. The other one is, in some cases it can lead to strong violation of the independence assumptions, the second is non-linear classification problems of it. (Xu Shuo, 2016)

Where;

$P(c/x)$ is posterior probability of class given predictor

$P(c)$ is prior probability of class

$P(x/c)$ is likelihood which is the probability of predictor given class

$P(x)$ is the prior probability of predictor

$$\text{The Naive Bayes Algorithm: } P(c/x) = \frac{P(x/c).P(c)}{P(x)}$$

Table 3.4. Naïve Bayes Classifier algorithm scores on BTA dataset by using department classes.

<i>Algorithm type</i>	<i>Naive Bayes Classifier</i>
<i>Accuracy score</i>	64.00
<i>Scheme</i>	weka.classifiers.bayes.NaiveBayesMultinomialText -P 0 -M 3.0 -norm 1.0 -lnorm 2.0 –stopwords
<i>Relation</i>	2018-department-1000.csv
<i>Instances</i>	999
<i>Test mode</i>	split 80.0% train, remainder test
<i>Dictionary size</i>	0
<i>Time taken to build model</i>	0 seconds
<i>Time taken to test model on test split</i>	0 seconds
<i>Correctly Classified Instances</i>	94 - 47%
<i>InCorrectly Classified Instances</i>	106 - 53%
<i>Kappa statistic</i>	0
<i>Mean absolute error</i>	0.0827
<i>Root mean squared error</i>	0.204
<i>Relative absolute error</i>	100%
<i>Root relative squared error</i>	100%
<i>Total Number of Instances</i>	200

3.2.3. Decision Tables

Decision tables become a useful tool when dealing with complex data rather than simple structured ones. Even in a situation with complicated logic, it is possible to see each combination of the conditions or choices with decision tables. (Lu Hongjun and Liu Hongyan, 2000)

Decision tables can be generated more easily with grouping and counting facilities, if these are implemented with appropriate grouping and counting, researchers can achieve statistical information related to class distribution over featured values. (Lu Hongjun and Liu Hongyan, 2000)

In order to simplifier the complicated data, in decision tables the conditions are generally recorded as True (T) or False (F). With decision tables, researchers can recognize combination of conditions which wouldn't have been founded in the contrary case. (Lu Hongjun and Liu Hongyan, 2000)

The disadvantage of decision tables is that it doesn't provide completion of test cases which give details about the instruction's gradual processes. If this detailed

information is required, decision table need to be included to test conditions. (Lu Hongjun and Liu Hongyan, 2000)

Table 3.5. Decision Table scores on DBTA dataset by using department classes.

<i>Algorithm type</i>	<i>Decision Table</i>
<i>Accuracy score</i>	63.50
<i>Scheme</i>	weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"
<i>Relation</i>	2018-department-comma-1000
<i>Instances</i>	999
<i>Test mode</i>	split 80.0% train, remainder test
<i>Number of training instances</i>	999
<i>Number of Rules</i>	476 Non matches covered by Majority class. Best first.
<i>Start set</i>	no attributes
<i>Search direction</i>	Forward
<i>Stale search</i>	After 5 node expansions
<i>Total of subsets evaluated</i>	16
<i>Merit of best subset found</i>	61.962
<i>Evaluation (for feature selection)</i>	CV (leave one out)
<i>Feature set</i>	3,6
<i>Time taken to build model</i>	0.06 seconds
<i>Time taken to test model on test split</i>	0 seconds
<i>Correctly Classified Instances</i>	127 - 63.5 %
<i>Incorrectly Classified Instances</i>	73 - 36.5 %
<i>Kappa statistic</i>	0.4222
<i>Mean absolute error</i>	0.0847
<i>Root mean squared error</i>	0.1964
<i>Relative absolute error</i>	99.3132 %
<i>Root relative squared error</i>	96.306 %
<i>Total Number of Instances</i>	200

3.2.4. Random Tree

Random tree is an ensemble of forest (tree predictors) and it is a supervised classifier Random tree adopts bagging idea while constructing decision trees. (Mishra Ajay Kumar and Ratha Bikram Kesari, 2016)

Leo Breiman and Adele Cutler introduced random tree, it is possible to handle both classification and regression problems with this algorithm. The classification process is made as following;

- The classifier receives the input feature vector

- Then, make it classified with each tree in the forest
- Provides the class label which gets the majority.

Regarding regression with random tree, the feedback of the classifier becomes the average of replies over all the tree predictors. Single model trees are combined with random forest idea and consist the random tree. (Mishra Ajay Kumar and Ratha Bikram Kesari, 2016)

Table 3.6. Random Tree Algorithm scores on DBTA dataset by using department classes.

<i>Algorithm type</i>	<i>Random Tree</i>
<i>Accuracy score</i>	50.00
<i>Scheme</i>	weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1
<i>Relation</i>	2018-department-comma-1000
<i>Instances</i>	999
<i>Test mode</i>	split 80.0 % train, remainder test
<i>Size of the tree</i>	968
<i>Time taken to build model</i>	1 seconds
<i>Time taken to test model</i>	1 seconds
<i>Correctly Classified Instances</i>	100 - 50 %
<i>Incorrectly Classified Instances</i>	100 - 50 %
<i>Kappa statistic</i>	0.0785
<i>Mean absolute error</i>	0.0782
<i>Root mean squared error</i>	0.1985
<i>Relative absolute error</i>	94.5088 %
<i>Root relative squared error</i>	97.3207 %
<i>Total Number of Instances</i>	200

3.2.5. ZeroR Classifier

ZeroR classifier is the baseline for both classification and regression problems. It is the simplest classification algorithm which based on the target. ZeroR doesn't have the predictability feature, it just determines the baseline for especially unbalanced data set.

ZeroR chooses the mostly frequent (majority) class as determinant and the algorithm assumes that all predictions are related with this class. (Nasa Chitra, Suman, 2010)

Table 3.7. ZeroR Algorithm scores on DBTA dataset by using department classes.

<i>Algorithm type</i>	<i>ZeroR</i>
<i>Accuracy score</i>	47.00
<i>Scheme</i>	weka.classifiers.rules.ZeroR
<i>Relation</i>	2018-department-1000.csv
<i>Instances</i>	999
<i>Time taken to build model</i>	0 seconds
<i>Time taken to test model on test split</i>	0.02 seconds
<i>Test mode</i>	split 80.0 % train, remainder test
<i>Correctly Classified Instances</i>	94 - 47 %
<i>Incorrectly Classified Instances</i>	106 - 53 %
<i>Kappa statistic</i>	0
<i>Mean absolute error</i>	0.0827
<i>Root mean squared error</i>	0.204
<i>Relative absolute error</i>	100 %
<i>Root relative squared error</i>	100 %
<i>Total Number of Instances</i>	200

3.2.6. OneR Classifier

OneR classifier is a machine learning algorithm and it is an abbreviation of One rule. It is the improved form of ZeroR algorithm. It is expected to get better output than the ZeroR because of the fact that OneR chooses the one which gives the best output among all training data classes. (Nasa Chitra, Suman, 2012)

OneR algorithm mechanism is as following for each value of the predictor (Nasa Chitra, Suman, 2012)

- First it counts how often each class value takes place
- Find the mostly appeared class
- Make the rule assign that class to this value of predictors
- Each predictor's total error of the rules is calculated
- The predictor with the smallest total error is chosen
- Find the best predictor with smallest total error that using OneR algorithm

OneR algorithm results by using department nominal value is shown in table 3.8. These results are collected with Weka.

Table 3.8. OneR Algorithm scores on DBTA dataset by using department classes.

<i>Algorithm type</i>	<i>OneR</i>
<i>Accuracy score</i>	49.50
<i>Scheme</i>	weka.classifiers.rules.OneR -B 6
<i>Relation</i>	2018-department-comma-1000
<i>Instances</i>	999
<i>Test mode</i>	split 80.0 % train, remainder test
<i>Time taken to build model</i>	0 seconds
<i>Time to test model on test split</i>	0 seconds
<i>Correctly Classified Instances</i>	99 - 49.5 %
<i>Incorrectly Classified Instances</i>	101 - 50.5 %
<i>Kappa statistic</i>	0.0686
<i>Mean absolute error</i>	0.0561
<i>Root mean squared error</i>	0.2369
<i>Relative absolute error</i>	67817 %
<i>Root relative squared error</i>	96.1444 %
<i>Total Number of Instances</i>	200

3.2.7. Other Classification Results

Different algorithms are tested on the study for DBTA Dataset and the results are listed in the Algorithms and Scores figure 3.2. The test done with same text-based data

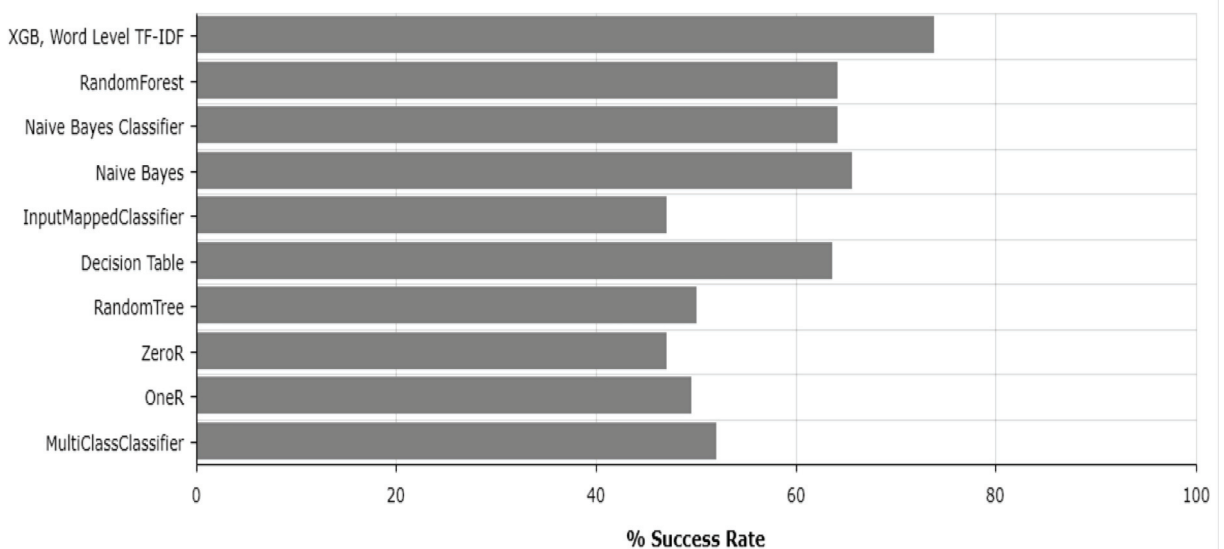


Figure 3.2. Algorithms and Scores using 1000 data from DBTA Dataset with using Weka. (XGB score is observed with DCS)

set with 1000 data. Twenty percent of them were allocated for testing. There exist 16 different department labels. In the data set there were 17922 tokens and 4239 terms.

3.3. Used Other Technologies in DCS (Environments, Packages and Libraries)

Other technologies that are used for development of data classification service explained below such as environments, application interfaces and libraries.

3.3.1 Python Pandas Data Frames

Pandas is a frequently used Python package for data analytics, especially for data processing and analysis. The Pandas package is based on two data structures called series and data frames. It can be described the series as one-dimensional arrays and data frames as two-dimensional matrices. However, Pandas package offers different functions from lists and matrices. Here are the basic operations we can do with Pandas package (Mckinney, 2011)

- Data can be read from files in various formats and printed to files.
- It contains methods to fill in or remove missing values.
- Graphics can be drawn in different types.
- The data can be summarized by grouping the data into groups (like add, counting, average) with group by function.

McKinney, W. believes that Python would be an attractive choice for data analysis applications by designing powerful, user friendly data structures that are coherent with the rest of the Python stack and pandas provides a strong basis upon which a very powerful data analysis ecosystem can be founded. (Mckinney, 2011);

Also in the DCS Application, data collections are stored on Pandas and all process up to model creation done on the Pandas Data frames.

3.3.2. Scikit Learn Application Interface

Scikit-learn is a library in Python that provides many unchecked and supervised learning algorithms. It is compatible to work with NumPy, Pandas and Matplotlib libraries that data scientist are familiar with. (Pedregosa Fabian, 2011)

Some functions provided by Scikit-learn can be listed as follows:

- Regression including Linear and Logistic Regression
- Classification including K-Nearest Neighbors
- K-Means and K-Means ++, including Clustering
- Model selection
- Pre-processing including Min-Max Normalization

Scikit-learn brings out a wide variety of machine learning algorithms using a consistent, task-oriented interface. Therefore, it enables to compare the methods for a given application easily. Scikit-learn also covers these machine learning algorithms while maintaining a user friendly interface tightly combined with Python. (Pedregosa Fabian, 2011)

3.3.3. Matplot Library

Matplotlib is a Python 2D visualization library with its GUI application. Many different output types are supported with the library. The library is used in order to visualize some graphical views that refer to part of boosting trees. (Barrett Paul, Hunter John, Miller J.Todd, Hsu Jin –Hsung, 2005)

3.3.4. Numpy Package

NumPy (Numerical Python) is a math library that allows us to do scientific calculations quickly. It creates the basis of NumPy arrays. The arrays are similar to python lists, but they are more useful than python lists in terms of speed and functionality. NumPy arrays should be homogeneous, which means that all elements in the array must be of the same data type.

One of NumPy's main purposes for data analysis are being the primary container for data to be passed between algorithms beyond the fast array-processing abilities that it provides to Python. NumPy arrays are much more efficient way of storing and manipulating data than the others built-in Python. (Mckinney, 2013)

In DCS, NumPy arrays are only used for graphical representations of model and dataset.

3.3.5. Graphviz Application Interface

Graphviz is a free graphical visualization software. Graph visualization is a way to represent structural information as abstract graphs and network diagrams. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning and various other technical fields.

It is possible to represent simulations of the dynamic behavior of the processes using a free tool like Graphviz. It means, there is no need to spend so much time on developing complex program visualization software always, if it's possible to use simpler solutions. (Riesco Miguel, Fondon Marian D., Alvarez Dario, 2009)

```
Administrator: Anaconda Prompt - conda install graphviz
graphviz-2.38          | hfd603c8_2      37.7 MB
sqlite-3.27.2        | he774522_0      941 KB
-----
Total:                141.3 MB

The following NEW packages will be INSTALLED:

graphviz:      2.38-hfd603c8_2
krb5:          1.16.1-hc04afaa_7

The following packages will be UPDATED:

ca-certificates: 2018.03.07-0 --> 2019.1.23-0
certifi:          2018.8.24-py37_1 --> 2019.3.9-py37_0
conda:           4.5.11-py37_0 --> 4.6.8-py37_0
cryptography:    2.3.1-py37h74b6da3_0 --> 2.6.1-py37h7a1dbc1_0
curl:            7.61.0-h7602738_0 --> 7.64.0-h2a8f88b_2
libcurl:         7.61.0-h7602738_0 --> 7.64.0-h2a8f88b_2
libpng:          1.6.34-h79bbb47_0 --> 1.6.36-h2a8f88b_0
libssh2:         1.8.0-hd619d38_4 --> 1.8.0-h7a1dbc1_4
openssl:         1.0.2p-hfa6e2cd_0 --> 1.1.1b-he774522_1
pycurl:          7.43.0.2-py37h74b6da3_0 --> 7.43.0.2-py37h7a1dbc1_0
qt:              5.9.6-vc14h1e9a669_2 --> 5.9.7-vc14h73c81de_0
sqlite:          3.24.0-h7602738_0 --> 3.27.2-he774522_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
qt-5.9.7          | 92.3 MB | #####4 | 24%
```

Figure 3.3. A view from installation step of Graphviz with Anaconda.

CHAPTER 4

PROPOSED SOLUTION TO THE PROBLEM

4.1. A Sample: Digital Business Tracking Application and Basics of DCS Implementation

The study based on the dataset that is collected from A Digital Business Tracking Application. The next part of the chapter explains the application in detail and shows the demand flow in DBTA and marks the stage which is integrated the “Data Classification Service” to the application.

The test dataset is collected from business tracking application of holding during 2018. The application collects demands that come from internal and external customers IT needs. The dataset is called as DBTA Dataset in the study.

4.1.1. A Digital Business Tracking Application

Large-scale companies have their own help desk organizations. These organizations are responsible for installation, activation and maintenance of technical equipment, also their mission is to support users in the case of possible software and hardware problems. The job definition of these organizations might differentiate at larger scale companies depending on the organization needs. The company that collects the data set, succeeded to be beyond these mentioned concepts by providing technology services to both holding companies and external customers.

This situation led to transfer of the communication needs between the customer and the anonymized Information Technology Company (referred as IT Company) on a more manageable process. Thus, customer will transfer his/her demand by using an application in a quicker way and with a quick assessment; then the demand created by the customer, redirected to the related department in the IT Company. Basically, ‘The Digital Business Tracking Application’ has been developed based on this concept and idea.

When a demand is created on the application by the customer or Information Technology personnel, (hereafter called as IT personnel) the following information are provided;

- Company: The company that is related with the request.
- Demand Scope: Request owner selects demand scope. (Refers to priority of a demand. That can be a critical error that prevents the operation of a factory.)
- Module: User selects related module with the request. (SAP, Software development, User support, Network etc.)
- Title of the request. (in the data set the support column wasn't used)
- Details of the request.
- The owner name and owner company of the demand: Owner information are known because of the LDAP Authentication.

If the request is not within the scope, service duration is defined regarding demand category. Service duration is based on the agreement made with the companies. As per agreement, service durations are as following:

- The authorization process: 1.2 days.
- Error recovery: 1.7 days.
- Sustainance and innovation it is 2.3 days.

These durations are determined by taking consideration of company experiences up to date and customer's requirement. (these periods are guaranteed by contract between customer and the solution provider company) Some requests are considered as 'project'. The time period for requests that within the scope of the project is determined after an effort analysis and prioritization. These requests are not bounded with the service level agreement. (a.k.a. SLA) Those demand that are defined in the scope of project are submitted to manager's approval. The approximate effort of a request that is evaluated as a project; is expected to be over three working days.

Due to the fact that some of holding companies are quoted on stock exchange markets; it is so important that all financial actions and authorization processes between employees have to be in a transparent and traceable structure. For legal legislation reasons; even basic processes are required to be under registration and such reasons lead to difference on completion and assignment processes. The authority requests and authorization processes are directed to related user or managers depending on the request.

The Digital Business Tracking Application offers a broad reporting infrastructure in terms of cost and staff time reports with using information that is stored on demands. In this way, the application allows retrospective inquiry in terms of demand time and cost, that has been invoiced to companies. This capability provided backward accountability and records the past development processes.

4.1.2 Demand flows in Digital Business Tracking Application

Briefly, the process can be explained with five main stages. (there exists other kind of demands are not mentioned because other stages that are not remarkable for the study)

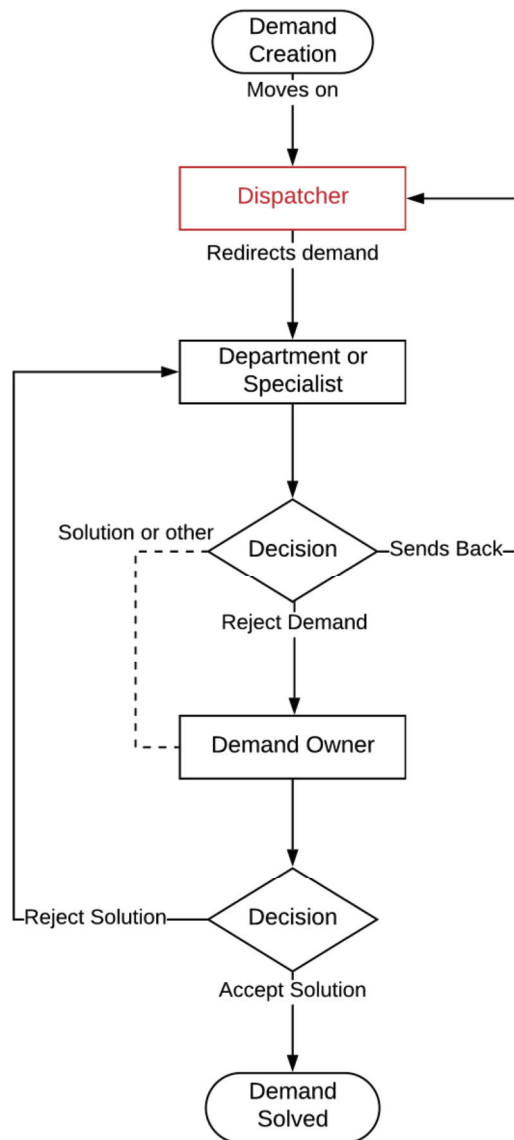


Figure 4.1. Demand flow in A Digital Business Tracking Application. One of them is “Demand Creation”. These demands are created by holding employees and they are directed to the dispatchers.

‘Dispatching’ is the next step of the process. Dispatching on the DBTA system resuming in manual way (by recruiting an employee who is in dispatcher role). The Data Classification Service that is developed during the study is implemented on the dispatching stage which is stated on the ‘Demand flow in A Digital Business Tracking Application’ flow chart. (in the dispatcher stage) The DBTA general flow of demands are as follows shown in figure 4.1.

4.2. Basics of Data Classification Service (DCS)

Data Classification Service, built on the study, implements pre-processing techniques and XGBoost algorithm. Universal interface of ‘Data Classification Service’ provides modular and adaptable structure for different text classification needs in different platforms.

In this study, classification results are referenced to the natural DBTA Dataset. (dataset of A Digital Business Tracking Application where none of the pre-processing methods are applied) The dataset is collected from request of holding companies are engaged in business in food and paint industries (referred as demands). These demands are related with IT needs of the companies which is submitted on the business tracking application of the IT Company during 2018. The dataset includes demands that have different kind of text-based information related with the request such as; request priority, module, owner company, title, explanation and etc. In the dataset, each demand has one specialist and each specialist belongs to a department. In this point of view, the classification problem in the dataset can be described as a multinomial text classification in terms of the specialist and department labels.

The Data Classification Service implements XGBoost classifier. The classifier gets TF-IDF matrix as an input value. In order to create the matrix, features are selected after pre-processing stages. In order to determine optimal number for the count of the features; we observed accuracy scores for the DBTA Dataset using different count of highly used terms from the corpus. The highest score is determined between 2750 and 3000 features. (weight of those features are calculated according to TF-IDF. The method is explained in ‘Weight Calculation Method’ part of the thesis)

The service is tested to create two types of models. One of them is for the specialist and the other is for the departments. After the model creation, the model is stored on the file system using a unique name and accuracy score with a test-card. The test-card includes; 'Test type' for the basic information and supporter columns for the algorithm; 'Demand in set' for the count of demand used to build the model; 'Labels type' for the target field of the model. 'Label count' for count of the nominal values and 'Token and Term' counts for the dataset. (in the test-card figure the responsibility column is shown)

```
Test type      : Standart test with 38949 demand. Supporter informations are ; explanation', 'company', 'type', 'owner' columns.
Date time     : 2019-02-20 08:22:12.106025
Duration      : 0:02:42.262203
Demands in set : 38949
Labels type   : responsibility
Labels count  : 16
Train data    : 33106
Test data     : 5842
Terms in data : 675825
Corpus in set : 43572
Stopwords count: 515
Algorithm type : XGB, Word Level TF-IDF
Model name    : model-xgb-responsibility-38949-%71.dat
Accuracy score : 71.50436419647441
```

Figure 4.2. A test-card that shows the model that accepts responsibility as a target field.

The Data Classification Service accepts text-based, semi-column separated values to create a model. The dataset can be given to the service as a CSV file. After the dataset loaded into the DCS, the next step pre-processing according to the data type can be applied on the set. Then a field from the comma separated values must be selected as a nominal field and the DCS is ready to create a model with given parameters.

The XGB Classifier is achieved its best performance with the given parameters while testing on DBTA dataset:

- base_score = 0.5
- colsample_bytree = 1
- gamma = 0,
- learning_rate = 0.15
- max_delta_step = 0
- max_depth = 5
- min_child_weight = 0.5
- Missing = None

- n_estimators = 100
- Nthread = -1
- Objective = 'multi:softprob'
- Seed = 0
- Silent = True
- Subsample = 1

After the model creation, a test-card is printed as an output to the console. In the test-card, accuracy score for the model is expressed as a percentage. At this stage user is ready for the prediction with using newly created model. (Figure-2 A test-card that shows the model that accepts responsibility label as a nominal value.)

Basically, the service user is able to make a prediction only using the fields: explanation of the request and owner company. The following visual is about the prediction stage:

```

INPUT FIELDS:
  test_explanation      : SAP seyahat portalında, muhasebeleştirme yetkisinin verilmesi konusunda yardımlarınızı rica ederim.
  test_owner           : Emine Tanrıöver
  test_company         : Pınar Su
  test_type            :

OUTPUT FIELDS:
  predicted_specialists : AyXX KaXXXXX
  predicted_department  : İnsan Kaynakları

```

Figure 4.3. An example prediction for specialist and department models with using the Data Classification Service.

The DCS has functions that are able to anonymize determined tokens that takes place in the dataset. The following visual, shows a result about the anonymization of company and employee names in a dataset. (Figure-4) The anonymization can be applied on demand owner, company, explanation or other given fields. By the way, the function is able to give automatic names to the selected columns. The correspondence of the automatically renamed fields are stored on an exported file that includes the matchings.

In conclusion, the service provides programmable interface for detailed pre-processing for dataset operations and prediction. The pre-processing is divided into two steps: 'Token Preprocessor' and 'Cluster Preprocessor'. Both steps are defined in detail at 'Pre-processing Methods Used in DCS' part of the study.

P	Q	R
explanation	specialist	department
Company_1 Zincir siparişlerini aktarma programında fiyat geçişine denk geldiğinde fiyat bulamama problemi+P32	User_2 Surname_2	Java Uygulan
Digital crm'e giriş yapılamıyor. Atf değerlendiremiyoruz, ekran donuyor. Gaktivite yönlendirilmesi ricasıyla	Emre Şencan	Java Uygulan
PSU Pilot sisteminin 1.11.1 sürümüne geçirilmesi sırasında çıkan problemlerin çözümlenmesini rica ederim.	Emre Şencan	Java Uygulan
Dijital Promosyon sisteminde yıllık anlaşma atflerinde reyon seçilemiyor. Kontrol edilmesini rica ederim.	Emre Şencan	Java Uygulan
BDU ekibine verilecek olan postman eğitimi için açılmış istektir. Emre Şencan' a atanması	Emre Şencan	Java Uygulan
BW için hazırladığımız serviste aktivite tipi ve teşhir türü inaktif olmuştur. Yeniden güncellenmesini rica ederim.	Emre Şencan	Java Uygulan

Figure 4.4. An example of the anonymized words in the dataset. Replacement of the words are red marked.

4.2.1 About the Data Source and Content

The Digital Business Tracking Application dataset (DBTA Dataset) that is collected during 2018 as its mentioned in the previous section. The dataset is created with the contributions of technical and non-technical personnel via requests that are delivered to the name anonymized IT Company. These caused some differences on the explanations about explaining same problems and same requests.

Because of the improper use of terminology by non-technical personnel; in the dataset there exist some bad expressions for the demands. Basically, we can say that in the dataset there exist so many different explanations of same requests or problems. This condition makes difficult the work of classification algorithms.

Some of the explanations include only some features that are meaningless about the requests and they do not contain enough explanations. Also, they are not the requests which are preferred to be seen in dataset. Those kind of records are marked as corrupted and they are eliminated in the run-time for the model creation. (approximately %1 of the records in DBTA dataset)

The demands on the dataset have no department field when it is created. The department field is added to the DBTA Dataset with using external tabular data of the organization that includes specialist and department pairs. However, while matching demands with the related department some problems are encountered on the dataset. For example, the name, surname fields that come from the tabular data may not match with the existed dataset. For example, in the tabular data; there could exist an additional middle name and surname but in the dataset there is only name surname or the surname of a female employee who has a new surname after she married and etc.

Basically, we accepted that they are same person; if there exist two or more same tokens that come from different data sources. However, this does not give a hundred percent accurate results. (Assume that we have ‘Ali İsmail Korkmaz’ from a source and from another source we get ‘İsmail Korkmaz’. According to the logic; total same tokens count are two. They are probably same persons. Even so, there could be a matching between name and middle name fields and that breaks previous logic.)

After we apply the logic to the DBTA dataset, approximately 73% of the records are matched with departments in first iteration. 14% of the records are not found any candidate for department label on the tabular data. Remained part is not matched for any other reason. Thus, there has been an additional column ‘department’ took place in the natural set after the process.

The detailed explanations about the other attributes and statistical background of the dataset are mentioned in the next part.

4.1.1.1. Design of Datasets

The dataset, that is collected from a business tracking application during 2018 is called as natural dataset in this study. The natural dataset contains those fields:

- Demand Id
- Demand Type
- Company
- Owner
- Creator
- Project
- Record Date
- Base Demand Id
- Demand Category
- Demand State
- Demand Scope
- Specialist
- End Date
- Explanation
- Survey

The 'Demand Id' refers the request identification key that is given by the DBTA. 'Demand Type' is not added as a field to the model because it has high importance to determine the departments on by own.

The Demand Type field includes options such as emergency, SAP applications, special business applications and user support that gives an important hint for finding the department. Actually, there exist some tests with using the Demand Type support column and we observed that this field increases the accuracy score approximately 4%. (the difference between the two rates refers to the observation of an arithmetical average of difference for a new model with the field, for four times)

The 'Company' field refers to the demand owner company. Those kind of fields are provided to the dataset from the LDAP when the customer logged on the DBTA. Therefore, there is no need to type the company field on the demand creation form.

'Owner' is an identifier that corresponds the name and surname. Basically its format is: 'name.surname'. This format is a unique identifier for the demand owner.

'Creator' is the creator person of the demand. If there is a need, the DBTA system allows the users to open a demand instead of another employee.

'Project' field is an optional area. It could give an important data for the classifier but it can be selected only by a personal who knows the related projects. The field did not used on the dataset.

'Base Demand' field is used for establishing a hierarchical relation between two demands. A demand can be a base or child demand for another. Those kind of relations are optional and in the DCS the field is not used in any model creation.

'Demand category' is a field that determines the average solution time in service level agreement. If the request is not within the project scope, service duration is defined regarding demand category. Service duration is based on the agreement made with the companies. Its mentioned on 'A Digital Business Tracking Application' part of the study.

'Demand State' gives actual state of the demand. Only solved demands are used in the dataset because if demand is solved that means the demand has find the right department and right specialist for the solution. If a demand incorrectly directed to specialist or department, dispatcher of the department redirects the demand to another specialist and department. This iteration continues until finding the right interlocutors. (the iteration ensures the right collocutor after nth iteration)

Specialist and Department (department is an artificial column) columns are used for target label on this study. In the next section: ‘Dataset Statistics’ has some statistics about these areas.

‘Survey’ field is not a meaningful context for the dataset. This field refers to demand owners score for the solution of the demand.

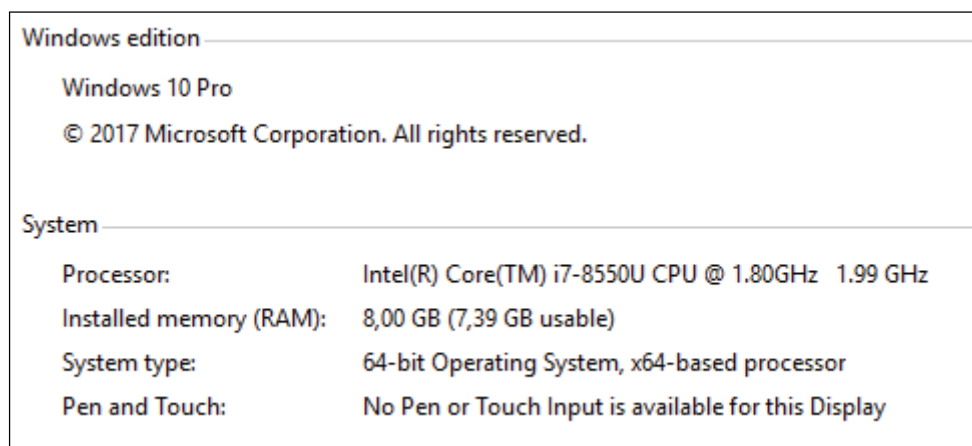
‘Demand Scope’ and Date fields are not used on the study. Those fields are closely related with time and period of the demand.

In conclusion, although there exist many other supporter columns in the DBTA natural dataset; they are not used to preserve universal structure of the data classification service. Hence, unless otherwise noted DCS tests are based on just ‘Explanation’ and ‘Company’ fields.

4.1.1.2. Dataset Statistics

This study based on two different types of dataset in terms of size. One of them is full natural form of the all demands, include approximately 50 032 demands. They are collected during 2018 in the anonymized holding. The set called as DBTA Dataset or natural set in the study. The second dataset includes about 1000 demands. It is considered as a short form of the natural set appropriately to the content characteristics. Both of the datasets are unbalanced, multinomial and text based.

The natural set is used for testing the Data Classification Service in different conditions. The approximate time for building a model with the dataset takes half an hour via the computer:



Windows edition	
Windows 10 Pro	
© 2017 Microsoft Corporation. All rights reserved.	
System	
Processor:	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Installed memory (RAM):	8,00 GB (7,39 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Figure 4.5. The test system specification.

In the pre-processing stage; nearly 12130 of them are filtered by cluster-based filters. Remaining part that is separated for training and testing. %80 of the demands are used to create a model and the remaining 20% were reserved for testing. That means the main model is created using almost 30 296 demands. Tests are made with 7574 demands.

The natural set does not have a homogeneous distribution over specialists and departments. Each demand has 14 attributes. As its mentioned before, department is an artificial column that is added to dataset after all dataset is collected on the DBTA. Only specialist and department columns are accepted as nominal and they are used as target label.

In the natural set, there exist about 130 specialists and 16 department label. Demands on the dataset are not homogeneously distributed on specialist or departments. That means in the natural set a department may have 4 or 19190 demands.

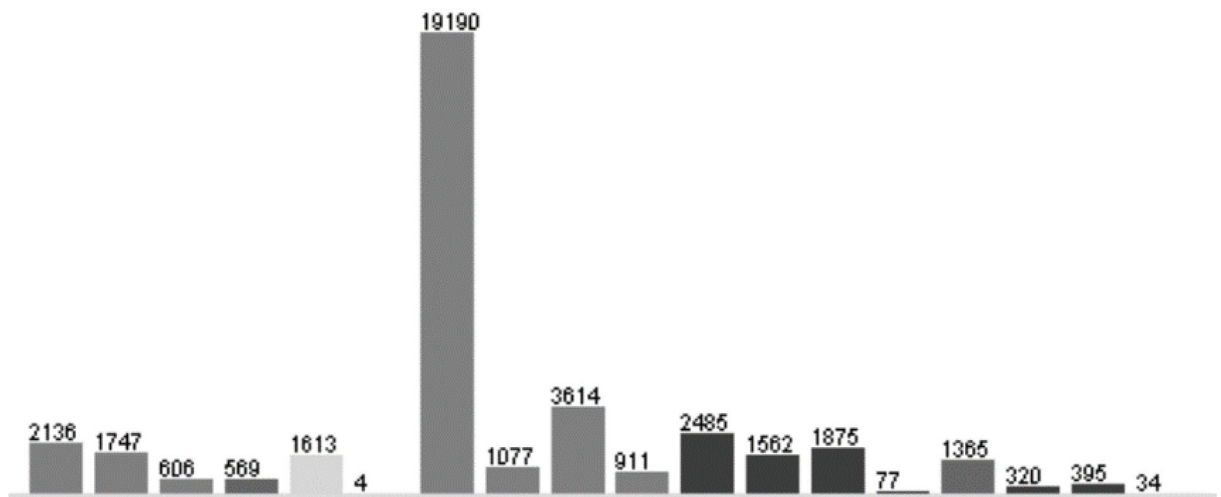


Figure 4.6. The graph shows count of demands on a department in the DBTA dataset. The yellow bar represents a department name called: 'User Support' in the natural set.

In the natural set, as mentioned before; 'demand type, owner company, owner id, creator, project, record date, base demand id, demand category, demand state, demand scope, end date, explanation, survey' attributes are existing. But in the study only 'owner company, explanation' fields are used as a text and scores are calculated with this fields.

After filtering, the unnecessary demands and no needed words (after pre-processing step) there remain 622771 tokens in the set. Corpus of the set includes 43572

piece of terms. About 519 stop words filter is applied after snowball stemmer applied on tokens. (Snowball stemmer is mentioned on ‘Stemming and Lemmatization’ part.)

4.2.2. Pre-processing Methods Used in DCS

The DCS provides, programmable interface for pre-processing. The pre-processing is divided into two main phases that follows each other: ‘Token Preprocessor’ and ‘Cluster Preprocessor’. In order to apply pre-processing steps, the dataset gets into the data structure of Panda data frames.

4.2.2.1. Token-Based Preprocesses

The process begins with the ‘token preprocessor’. The token preprocessor processed on the word pieces that take places in the text body of the requests. These phase has optional steps like: removing unnecessary or corrupted tokens, stemming, stop words and etc. The tokenization step is processed after the demands are written into the data frames.

4.2.2.1.1 Tokenization

This is a well-known step in data science. In the DCS, before the step data column is prepared to the tokenization; some other operations are done like: adding support column to the base raw data; converting the raw data to lower-case; filtering of unnecessary characters and punctuations etc. After the preparation, the raw data is split into the tokens that is stored in a token array. (Tokens are subsequence of a string that is broken into pieces like: words, numbers, phrases, symbols etc. Usually, in the text-based content white space is used to split the raw data.)

4.2.2.1.2 Stemming in Turkish

Stemming is a common approach in natural language processing based on text normalization. Snowball stemmer is a library that is developed by Boulton, Andrew

Macfarlane in 2001 to provide programming interface for language such as: C, Java, Perl and Python etc. (in DCS it is used to get root of the word without postfixes)

In the DCS, Turkish Stemming method of Snowball Stemmer is used. The Snowball Stemmer is based on the Porter's stemming algorithm. (M. Porter, Richard Boulton, Andrew Macfarlane, October 2001)

INPUT DATA:	Dijital Şirket Sistemine Bayi geçişlerimizin başlaması sebebi ile Satış Muhasebe Servisi olarak Dijital Şirket Sistemi Eğitimi ihtiyacı doğmuştur.			
OUTPUT DATA:	dijital şirket	bayi geçiş	başlaması sebep	satış muhasebe servisi olarak dijital

Figure 4.7. After stemming some of the words are deleted by the stop-words filter.

Notice: I believe that more work is needed to improve the Turkish stemmer algorithm.

4.2.2.1.3 Remove Stop-Words

The stop-word list that is used in the DCS is collected with using a basic Turkish dictionary and observing requests that include text-based data.

1	ekleme				
2	ekle				
3	eklenmesi				
4	ile				
5	üzere				
6	fakat				
7	çoğu				
8	gibi				
9	olur				
10	ondan				
11	sn				
12	dikkat				
13	yalnızca				
14	rica				
15	yardım				
16	talep				
17	ricasıyla				
18	sonra				
19	ilgili				
20	hakkında				
21	bana				
22					

Figure 4.8. The figure shows a short list for Turkish Stop-word list.

The stop-word list is stored on a text document as the root of words. The figure 11. Shows a short list sample of the list.

The Stemmer algorithm is applied on the text data that comes from the demand and the stop-words list at run-time. Although we apply Turkish Stemmer to both stop-words and request body, we observed that; some of the terms which comes from the stop-words list are not compatible with the terms that come from the requests data. But the difference is not caused significant change on accuracy score of the algorithm. Therefore, in the study, the incompatibility problem is ignored.

In the Turkish stop-word list archive, we have almost 519 different tokens that are stored.

4.2.2.1.4 Term List and Corpus

In data science, term refers to the root of a token. Each token in dataset represented with a term in the DCS. In the DBTA Dataset there exist approximately 675825 pieces of terms.

Corpus is the set of the terms. In the corpus, there exist just 43572 different pieces. (assumed that each term occurs at most one times in a set) Features of the ML algorithms are selected within the corpus.

4.2.2.2. Cluster-Based Preprocesses

Followed by the token-based preprocess, cluster based filters are applied to the data frames. The aim of the cluster-based filters is to remove the improper data from the dataset at run-time in model creation. The improper data in the dataset is called on the DCS service as ‘corrupted data’.

4.2.2.2.1 Cluster-Based Filters

In the DCS, there exist optional cluster-based filters that remove the data according to demand status, type, state, size and content of the explanation. Cluster based filters remove lines from the data frame at run-time.

One of those filters that removes a line which includes at least one corrupted word or a which has a word count smaller than predefined token size. (Corrupted words can be

selected as word which are smaller than tree chars or includes adherent numerical characters or words consist of character from unknown charset.)

The Data Classification Service pseudo code sample:

1.	Input values are demand collection, demand column
2.	FOR text in demand_collection.at(i, data column)
3.	counter equals zero
4.	FOR EACH letter in text
5.	IF letter is a numeric value
6.	increment the counter
7.	IF length of text is greater counter AND counter is greater zero
8.	IF length of demand_collection.at(i, data column) less than avg. token size
9.	Demand num. equals to demand_collection.at(i, demand column)
10.	IF demand no not in removable demands
11.	append i to removable demands
12.	Demand collection .drop(index of removable demands, delete)
13.	Demand collection reset index(delete, drop)

Figure 4.9. Pseudo code of a DCS function that removes a line which includes at least one corrupted words or a which has word count smaller than average token size. ('i' assumed as an incremental integer)

4.2.2.3. Effect of Filters on BTA Dataset

Pre-process steps affect the performance of the algorithm. In order to measure the effect of each filter, the model is created tree times and the average score is shown in the following tables. (shown by tables 9-10)

In the table 'Base Result' column refers success score without adding related field. Each row from top to bottom in the table is added to the previous row. For example; 'Module info' field added as a parameter means that, 'owner' field already added as an input.

'Filter Affect' refers to the difference after the filter is applied. As can be seen from the table; the effect for module and type fields is very high for cluster-based filters.

Filters usage status in DCS is placed on the ‘Explanation’ column. ‘Default’ means that this filter is applied when the model is created and ‘Not used’ means that the filter is optional.

Actually, the success of the cluster based filters depends on the given dataset. The tests are done on the DBTA natural dataset.

Table 4.1. Shows the average of cluster-based filters on department in DBTA dataset.

Cluster Filters (Avg. %)	Base Result	Filter Affect	After Applied	Expl.
Add owner info	%73	+%1	%74	Default
Add module info	%74	+%5	-	Not used
Add company info	%74	+%2	%76	Default
Filter according to status	%76	+%1	-	Not used
Filter according to type	%76	+%3	%79	Default
Filter according to demand state	%79	+%1	-	Not used

The next table is about token-based preprocessing steps. In the table, assumed that demand explanation, demand owner and company fields are added as an input values for model creation.

Table 4.2. Shows the average of token-based filters on department in DBTA dataset.

Token Filters (Avg. %)	Base Result	Filter Affect	After App.	Expl.
Filter unnecessary characters and punc.	%71	+%2	%73	Default
Turkish stemming	%73	+%3	%76	Default
Remove stop words	%76	+%3	%79	Default
Filter short words	%79	%0	%79	Default
Remove corr. words from data frame	%79	%1	%80	Default

Removing stop words has big effect on the score. The other filter ‘Remove corrupted words from data frame’ deletes meaningless, misspelled words or the words that consisting of letters and numerical characters.

4.2.2.4. Determining Feature Importance

The following figure is a part of a long graph. The graph shows each feature and the importance of the feature for a demand in the model of the DCS.

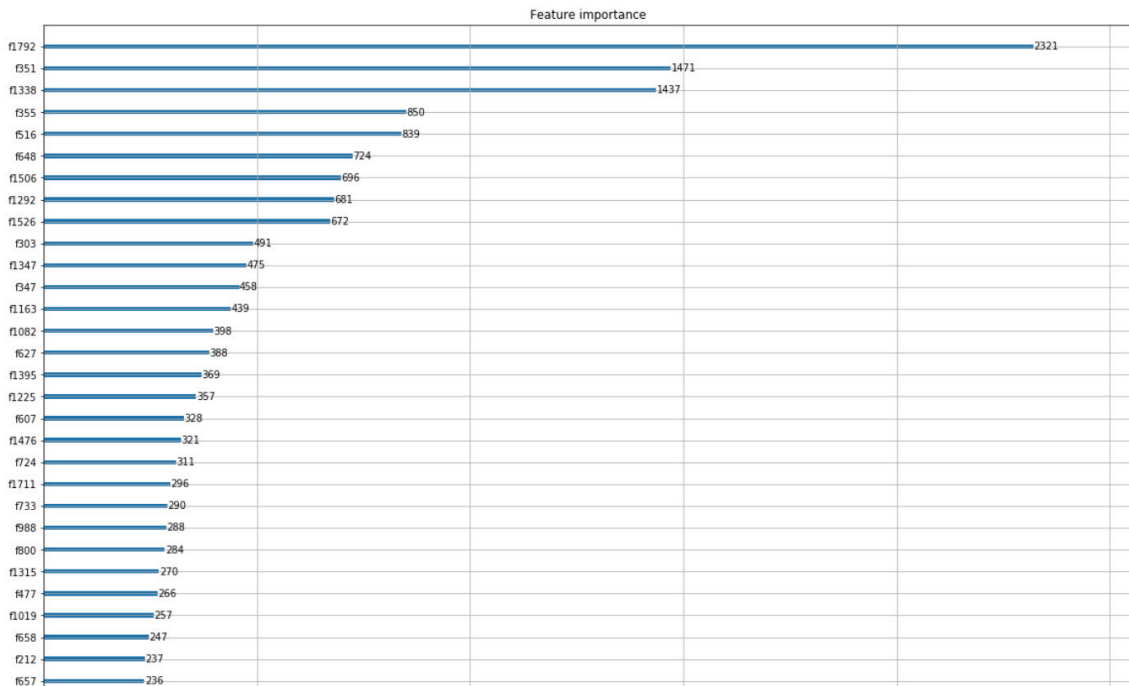


Figure 4.10. A graph that represents feature importance on DBTA Dataset.

The horizontal bars on the graph represent a group of terms that are selected as a feature on the DBTA dataset. Each bar corresponds a feature in the dataset. Most frequently, repeated terms in the set are determined as feature. In the DBTA Dataset, in order to get highest accuracy the feature count has to be between 2750 and 3000. (The model created four times in order to observe changes related with the feature count. The change observation is started from 500 peace and its lasted up to 3500 peace of feature. Optimal feature count is determined around between 2750 and 3000 for the dataset.)

I believe that; optimal feature count for each dataset can be calculated; if token count and corpus size are known. There is no need for any other information.

4.2.2.5. Weight Calculation Method

In order to calculate weight of a request; the feature or features that is located in the request has to be determined. Calculation of weight of a feature is strongly related with these features and their frequency.

Basically, if a term repeats itself many times in a request, that means the feature is important for the demand. On the other hand, if a term repeats itself again and again in other requests, that decreases the importance of the request. That's a well-known concept in data science as: **term frequency x inverse document frequency**. (a.k.a. TF-IDF)

In the DCS, Sci-kit library is used to calculate the TF-IDF values. The basic formula of the calculation is: $TF-IDF = TF(t,d) \times IDF(t)$ According to this calculation;

- The Term Frequency is occurrence times of a feature in a request.
- The Inverse Document Frequency: $IDF(t) = \log ((1 + n) / (1 + df(t))) + 1$ where n is the count of the total demands and $df(t)$ is the number of the documents that includes the feature in the dataset.

(Juan Ramos, 2016)

Aim of the weight calculation is converting feature frequency to numerical values which refers to weight of the features for each document. The feature weight used in XGBoost classification algorithm to create a model.

4.2.3. Demo of Data Classification Service

The Data Classification Service integrated with a basic test graphical user interface. The demo interface is done to make the service tests easier. The interface is built with ‘Tkinter’ library on Python.



Data Classification Service

Data Classification Service

İstek Açıklaması :

İstek Sahibi :

İlgili şirket :

Predict Clear Form

Predicted Department :

Predicted Specialist :

Figure 4.11. Graphical user interface of the Data Classification Service demo.

Basically, the interface gets demand explanation, owner and company fields as input values and returns predicted department and specialist of the given requests. (The figure 4.11 shows the demo interface of the service.)

The Data Classification Service will be running on a centralized application server and will be communicating with clients with using Rest architecture via a JSON object.

The prediction in the interface is made in two stages. One of them is prediction of the department for the given input. After department is determined, the second stage is specialist prediction. In the stage, according to the predicted department a new model is created from the dataset and prediction is done with the new modal. (There could be an alternative approach to do both predictions. As we know, each specialist belongs to a department. After a specialist predicted for demand, department can be found from the relation.)

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Manual Measuring Accuracy Score for Data Classification Service

The given prediction scores in the study calculated programmatically according to one to one relation that comes from the dataset. For the DBTA dataset, each demand has a one specialist and one department. If prediction result is not the same with the given person or department which is stated on the test record, program marks the record as a false prediction. However, the prediction could be correct if roles or responsibilities in the organization are same.

The situation decreases accuracy score when it is calculated programmatically. For example, in the network and security group of name-anonymized company, there exist five personnel in similar roles and two of them working on VPN related issues. If a request related with the first person (in the test record) and the model predicts the second person as the specialist, the record is assumed as a false prediction by the program.

There are similar roles/responsibilities in the departments as well as the specialists. Some of the departments are specialized in a sector. For example, some of the SAP departments in the name-anonymized company, work in only food industry and some other departments work for only construction industry. However, technically both groups have the same roles/responsibilities.

In manual measuring, the scenario which decreases the accuracy score is eliminated. Each prediction results are evaluated one by one and they are put into five different categories. (Figure 5.1 and Figure 5.2.) While dark green bar refers to same label with the test record, red bar is an alternative label for the request. (alternative department or specialist) The yellow bar refers to false predictions. Blue bar refers to total correct predictions and lastly light green bar refers wrongly directed demands. (left to right bars color: dark green, red, yellow, blue, green)

For the manual testing, one hundred pieces of demands randomly selected from the DBTA dataset.

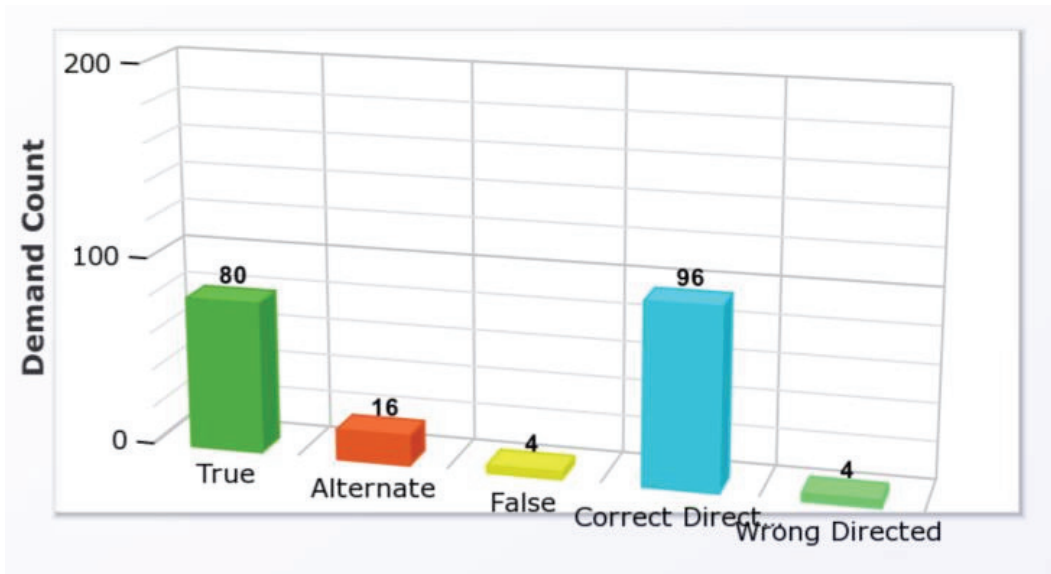


Figure 5.1. Manually evaluated prediction results for department. (done with 100 demands)



Figure 5.2. Manually evaluated prediction results for specialist. (done with 100 demands)

5.2. Auto Generated Scores for Data Classification Service

The data classification service generates a test-card automatically after model creation. The card contains short information about test such as date-time, test duration (model creation time), demands count in the set (after cluster-based filters remove some of the records in run-time), labels type (target field), labels count (total different labels in

the dataset), train-test data counts, terms in data (includes redundant terms count in train data), count of the corpus, stop words count, algorithm type and accuracy scores. Some of them are listed below:

.....

Test type : Standard test with 38949 demand. Supporter information are ;
explanation', 'company', 'owner' columns.
Date time : 2019-02-20 08:19:29.768076
Duration : 0:02:59.099823
Demands in set : 38949
Labels type : department
Labels count : 17
Train data : 33106
Test data : 5842
Terms in data : 675825
Corpus in set : 43572
Stopwords count: 515
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-38949-%70.dat
Accuracy score : 70.03251754235838

.....

.....

Test type : Standard test with 38949 demand. Supporter information are ;
explanation', 'company', 'owner' columns. Two SAP group is concated.
Date time : 2019-02-20 08:22:12.106025
Duration : 0:02:42.262203
Demands in set : 38949
Labels type : department
Labels count : 16
Train data : 33106
Test data : 5842
Terms in data : 675825
Corpus in set : 43572
Stopwords count: 515
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-responsibility-38949-%71.dat
Accuracy score : 71.50436419647441

.....

.....

Test type : Standard test with 35218 demand. Supporter information are;
'explanation', 'company', 'owner' columns.
Date time : 2019-03-03 17:44:17.178881
Duration : 0:13:38.968016
Demands in set : 35218
Labels type : department
Labels count : 17
Train data : 29935
Test data : 5282
Terms in data : 623303
Corpus in set : 36558
Stopwords count: 516
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-35218-%72.dat
TF-IDF name : tfidf-xgb-department-35218-%72.dat
Accuracy score : 72.40204429301534

.....

.....

Test type : Standard test with 35218 demand. Supporter information are;
'explanation', 'company', 'owner' columns. Two SAP group is added.
Date time : 2019-03-03 17:45:43.244133
Duration : 0:15:05.033268
Demands in set : 35218
Labels type : department
Labels count : 16
Train data : 29935
Test data : 5282
Terms in data : 623303
Corpus in set : 36558
Stopwords count: 516
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-responsibility-35218-%71.dat
TF-IDF name : tfidf-xgb-responsibility-35218-%71.dat
Accuracy score : 71.20954003407155

.....

.....

Test type : Standard test with 35218 demand. Supporter information are; 'explanation', 'company', 'owner' columns. With using 1500 feature.
Date time : 2019-03-15 12:10:52.008758
Duration : 0:09:30.511163
Demands in set : 35218
Labels type : department
Labels count : 17
Train data : 29935
Test data : 5282
Terms in data : 622771
Corpus in set : 36557
Stopwords count: 519
Feature count : 1500
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-35218-%73.dat
TF-IDF name : tfidf-xgb-department-35218-%73.dat
Accuracy score : 73.61347719098997

.....

.....

Test type : Standard test with 35218 demand. Supporter information are; 'explanation', 'company', 'owner' columns. After XGBoost tuning from the default parameters.
Date time : 2019-03-15 12:00:49.167261
Duration : 0:10:27.290093
Demands in set : 35218
Labels type : department
Labels count : 17
Train data : 29935
Test data : 5282
Terms in data : 622771
Corpus in set : 36557
Stopwords count: 519
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-35218-%78.dat
TF-IDF name : tfidf-xgb-department-35218-%78.dat
Accuracy score : 78.40242286579596

.....

.....

Test type : Standard test with 35218 demand. Supporter information are; 'explanation', 'company', 'owner' and 'type' columns.
Date time : 2019-03-15 13:29:27.290634
Duration : 0:09:39.496597
Demands in set : 35218
Labels type : department
Labels count : 17
Train data : 29935
Test data : 5282
Terms in data : 622771
Corpus in set : 36557
Stopwords count: 519
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-35218-%75.dat
TF-IDF name : tfidf-xgb-department-35218-%75.dat
Accuracy score : 75.75241340147643

.....

.....

Test type : Standard test with 35218 demand. Supporter information are; 'explanation', 'company', 'owner' columns. After XGBoost tuning from the default params.
Date time : 2019-03-15 13:54:32.992369
Duration : 0:11:11.122564
Demands in set : 35218
Labels type : department
Labels count : 17
Train data : 29935
Test data : 5282
Terms in data : 622771
Corpus in set : 36557
Stopwords count: 519
Feature count : 2750
Algorithm type : XGB, Word Level TF-IDF
Model name : model-xgb-department-35218-%79.dat
TF-IDF name : tfidf-xgb-department-35218-%79.dat
Accuracy score : 79.61385576377057

.....

.....
Test type : Standard test with 35218 demand. Supporter information are;
'explanation', 'company', 'owner' columns. After XGBoost tuning from the default
parameters.

Date time : 2019-03-15 14:07:33.818095

Duration : 0:11:55.078383

Demands in set : 35218

Labels type : department

Labels count : 17

Train data : 29935

Test data : 5282

Terms in data : 622771

Corpus in set : 36557

Stopwords count: 519

Feature count : 2750

Algorithm type : XGB, Word Level TF-IDF

Model name : model-xgb-department-35218-%80.dat

TF-IDF name : tfidf-xgb-department-35218-%80.dat

Accuracy score : 80.37100132500473
.....

CHAPTER 6

FUTURE WORKS AND CONCLUSION

The dataset is collected from archive from requests that are sent in 2018. Each request called as a demand in the dataset. Each demand in the dataset, that is already solved in the corporate company; belongs to a department and specialist personnel. In the company, there exist 16 different departments. Each department differentiated from the content because of unique or rarely used words. (those kind of words are selected as feature for the model)

For example; if a demand contains a feature like; VPN, awebsite.net, foreign access, OTP and etc. that increases the probability about the demand belongs to a department related with ‘network’.

Although the demand is likely to be associated with a department related with network, there could be exceptional cases too. In other words; any demand includes these key words in its explanation but the demand may be belong to some other departments.

Similar conditions apply to specialist classes. By the way, there exist some specialists who has the same responsibility with someone works in another department. In order to avoid those kind of confusions; we offered to do a dispatching mechanism according to the responsibilities instead of department or specialist labels.

One of our goals at the beginning of the study were to develop a public service with a universal interface. The study has a universal and text-based interface but the service has no graphical interface to manage the functions and it is not opened to public use.

- In order to open the DCS to publicity, the service has to be launched from a web-based graphical user interface. The GUI has to provide following functions:
- Uploading text-based dataset to the service.
- Updating the dataset if there exist new data or there is need to delete existed data.
- Manageable pre-processing operations. According to dataset service user can decide to apply a pre-processing step. These processes must be optional.
- Manageable tuning options for XGBoost model.

- Selecting nominal value and built a new model with uploaded data via the GUI.
- Defining web services related with the model and providing public or private access to the service via a security token.

The main purpose of the web-based GUI is enable users to create their model and prediction service using basic web-interfaces with those functions which are ready to serve via using text-based programmable python interfaces.

We've mainly emphasized the advantages of DCS classification service which has been developed within the scope of this study. Although there exist many advantages of the system, it has disadvantages too. One of them is the necessity to keep the data set updated. Depending on rapidly changing conditions and needs in industrial area, the organization of the departments and responsibilities of employees may have to be changed. In Turkey, it has been calculated that an average turnover rate on IT companies are approximately 15%. (Brumley Sara, 2019)

If we consider the situation in this context, classification model that has been prepared with the data of year 2018, (in case data is not updated) it means that the model won't be served to 15% of the employees during 2019. Depending on the changing conditions and needs, it can be required to re-organize the roles and responsibilities of departments that takes place within the institution.

When both advantages and disadvantages are being considered, we suggest institutions to pass hybrid solution depending on our study's results. Besides having DCS service integration, our hybrid solution proposal is, to have at least one dispatcher on the departments of companies. (in our company of which DBTA data set is being received, each department has its own dispatcher or dispatchers who are able to make dispatch to his / her team members or to the other departments)

In this way, the possible demands that might be dispatched incorrectly by DCS's can be addressed to correct person by the dispatcher. (mis-dispatching rate is approximately 18% on label of department). Also, it will be possible to include newly joined person to team or newly formed team in the institutions to the data set that is intended for preparing the next model.

For proper functioning of the system; updating the model with certain intervals is correct approach. It is predicted according to the study that this interval should be between two or three months. On DBTA system, approximately 15.000 new demands will be

created within this time period. These demands will be directed to about 118 specialists and 16 different departments.

As well as departments and specialists that are renewed in according to the new requirements; there could be departments which are no more necessary for the institution or specialists who left the companies. It is necessary to omit these departments or specialists from data set which is ordered by date priority. Therefore, the model will stay updated and will provide better performance.

All these processes are proceeding on a data set prepared in specified text format and it is eligible to be managed easily using the DCS interfaces. The DCS is designed in order to have universal and coherent interface as far as possible. Thus, it is aimed to implement the service on manual classification practices in different platforms.

It is planned that “A Digital Business Tracking Application” which is sampled many times in this study will become integrated with “Data Classification Application” in 2020.

REFERENCES

- Barrett Paul, Hunter John, Miller J.Todd, Hsu Jin -Hsung, Greenfield Perry, Matplotlib Lib- A portable Python Plotting Package, Astronomical Data Analysis Software and Systems XIV ASP Conference Series Volume 347, (2005), [13]
- Biau Gerard, Analysis of a Random Forest Model, Journal of Machine Learning Research 13, (2012), 1063-1095
- Breiman Leo, Random Forests, University of California, (2001), 5-32
- Brumley Sara, Small Business - Chron.com, Accessed 30 May 2019
- Chen Tianqi and Guestrin Carlos, XGBoost: A Scalable Tree Boosting System, (2016)
- Friedman Jerome H, Greedy Function Approximation: A Gradient Boosting Machine, Feb 1999 (modified March 15, 2000, April 19, 2001)
- Juan Ramos, Using TF-IDF to Determine Word Relevance in Document Queries (2016)
- Kaggle Competition: Titanic Machine Learning From Disaster, (2018)
- Lu Hongjun and Liu Hongyan, Decision Tables: Scalable Classification Exploring RDBMS Capabilities, published in VLDB, (2000)
- Martin Porter, Richard Boulton, Andrew Macfarlane, Snowball: A language for stemming algorithms, M.F. Porter, October (2001)
- McKinney Wes, Python for Data Analysis, Agile Tools for Real World Data, 2013 [15]
- McKinney, Wes Pandas: A Foundational Python Library for Data Analysis and Statistics, (2011)

Mishra Ajay Kumar and Ratha Bikram Kesari, Study of Random Tree and Random Forest Data Mining Algorithms for Microarray Data Analysis, PG Department of Utkal University, India, (2016)

Nasa Chitra, Suman, "Evaluation of Different Classification Techniques for WEB Data, International Journal of Computer Applications, August (2012), Volume 52, No:9

Natekin Alexey and Knoll Alois, Gradient Boosting Machines, A Tutorial, (2013)

Pedregosa Fabian, Varoquaux Gael, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Olivier, Blondel Mathieu, Prettenhofer Peter, Weiss Ron, Dubourg Vincent, Vanderplas Jake, Passos Alexandre, Cournapeau David, Brucher Matthieu, Perrot Matthieu, Duchesnay Edouard, Scikit Learn: Machine Learning in Python, Journal of Machine Learning Research 12, (2011), 2825-2830

Qi Yanjun, Random Forest for Bioinformatics, Nec Labs America, (2012), 307-323

Riesco Miguel, Fondon Marian D., Alvarez Dario, Using Grapviz as Low Cost option to Facilitate the Understanding of Unix Process System Calls, Volume 224, (2009), 89-95

Xu Shuo, Bayesian Naive Bayes Classifiers to Text Classification, (2016)