Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Deepti Vedala

Building a classification engine for ticket routing in IT support systems

Master's Thesis
Espoo, August 28, 2018

Supervisor: Professor Aristides Gionis
Instructors: Jyrki Tunnela M.Sc. (Information Technology),
Marko Lähde

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Deepti Vedala

Title:
Building a classification engine for ticket routing in IT support systems

Date:	August 28, 2018	**Pages:**	vi + 42
Professorship:	Computer Science, Aalto University	**Code:**	SCI3042
Supervisor:	Professor Aristides Gionis		
Instructors:	Jyrki Tunnela M.Sc. (Information Technology), Marko Lähde		

In any IT support environment, it is important to quickly route support tickets to correct teams. Often, it takes few days to manually classify several hundreds of tickets. This thesis presents a classification engine that provides routing recommendation to specialists for incoming tickets. The classification engine is built using machine learning and software robotics to decrease the amount of human time spent on support ticket classification. Experiments are carried with logistic regression, random forests and extremely randomized trees using historical data. During off-line cross-validation, random forest model performs well with 90% of f1-score and is deployed in production using AWS. The performance of the classification engine is tested in production for two weeks. The deployed model has f1-score of 86%. The f1-scores for the individual groups like level 1, level 23, level 24 are 89%, 88% and 93% respectively. These three groups contribute to almost 90% of total tickets. This thesis presents an approach of how a machine learning model is employed to reduce human time.

Keywords: machine learning, robotics, classification, support tickets

Language: English

Acknowledgements

I am grateful to professor Aristidis Gionis for supervising me during the process of this master thesis. My sincere thanks to Basware Oyj for providing me an opportunity to work on thesis project. I would like to thank my thesis advisors at Basware, Jyrki Tunnela and Marko Lähde for steering me in right direction with their valuable suggestions and time.

Finally, I must express my profound gratitude to my husband Kartiek and to my parents for their continuous support and encouragement throughout my years of study without which this accomplishment would have been impossible.

Espoo, August 28, 2018

Deepti Vedala

Abbreviations and Acronyms

tf	Term Frequency
tsv	tab separated values
csv	comma separated values
idf	Inverse Document Frequency
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
AWS	Amazon Web Services
RPA	Robotic Process Automation
HTTP	Hyper Text Transfer Protocol
ITSM	Issue Tracking System Management
SVM	Support Vector Machine
SVC	Support Vector classification
NuSVC	Nu-Support Vector Classification
LinearSVC	Linear Support Vector Classification

Contents

Chapter 1

Introduction

Basware customer care team receives IT tickets from customers around the globe. It has eight groups that deal with different kinds of problems. Specialists dedicated to each group spend considerable amount of time to manually classify and direct the tickets to correct responsible teams.

Currently, the ITSM system receives emails from customers in eight different languages. Based on these emails, specialists create tickets. The scope of the project is limited to classification of tickets that are in english.

To save human work and time Basware makes use of software automation using RPA. The goal of this thesis project is to augment the software automation and significantly improve ticket resolving times by implementing a classification engine. The core of the classification engine consists of a machine learning model. To build the machine learning model, this thesis uses Scikit-learn, a machine learning library for python programming language. This project has two experimental setups. In setup 1, experiments with logistic regression, random forests and extra trees classifier (also known as extremely randomized trees) on different sets of data is performed to identify the best performing algorithm during off-line cross-validation. In setup 2, the model that performs best on the test data is deployed in production using AWS. RPA uses software robots to communicate with the model deployed in production. For incoming tickets, a software robot interacts with the model and gets the predictions.

1.1 Problem statement

Basware provides various online service channels to the customers for resolving support requests. Customers contact Basware through chat, emails or contact form submissions. Digital self service tools are provided by making use of self-service tools such as online ticketing and knowledge base. Online ticketing tool has the emails/incidents received from the customers, which are further routed to respective support teams for further processing. Support teams are categorized into three major support lines: first level support, second level support and third level support. The domain specialists working for these support lines use knowledge base tool, which has knowledge base articles to resolve the tickets.

Ticket resolution includes three steps as shown in Figure 1.1. The first, is ticket creation which also includes ticket validation. The second step is ticket classification and the final step is ticket processing. Ticket creation is manually done by first level specialists. Specialist checks if the incident has all required information and verifies if certain fields like the company, contact details are correctly set and are not empty, includes error description to notes (detailed description) field, summarizes the incident to summary(description) field and verifies if all the necessary fields are correctly set. After validating all the required fields, specialist then creates a ticket.

As discussed, Basware customer care has three major support lines out of which first level support (level 1) deals with almost 50% of existing tickets. Specialists dedicated to level 1 group, resolve all the tickets that satisfy certain criteria and directs the rest to other groups. There are five different sub categories in second level support: level 21, level 22, level 23, level 24 and level 25. If the ticket do not have sufficient information and if the priority is very high then it is routed to third level. The third level has few subcategories. There are very few tickets that are escalated to third level. Other than these existing groups, there are few other groups to which tickets are rarely escalated. As it is not possible to train a machine learning model with such few number of tickets, this project considers all the subcategories of 3rd level and the rest of the other groups as a new group 'Others'.

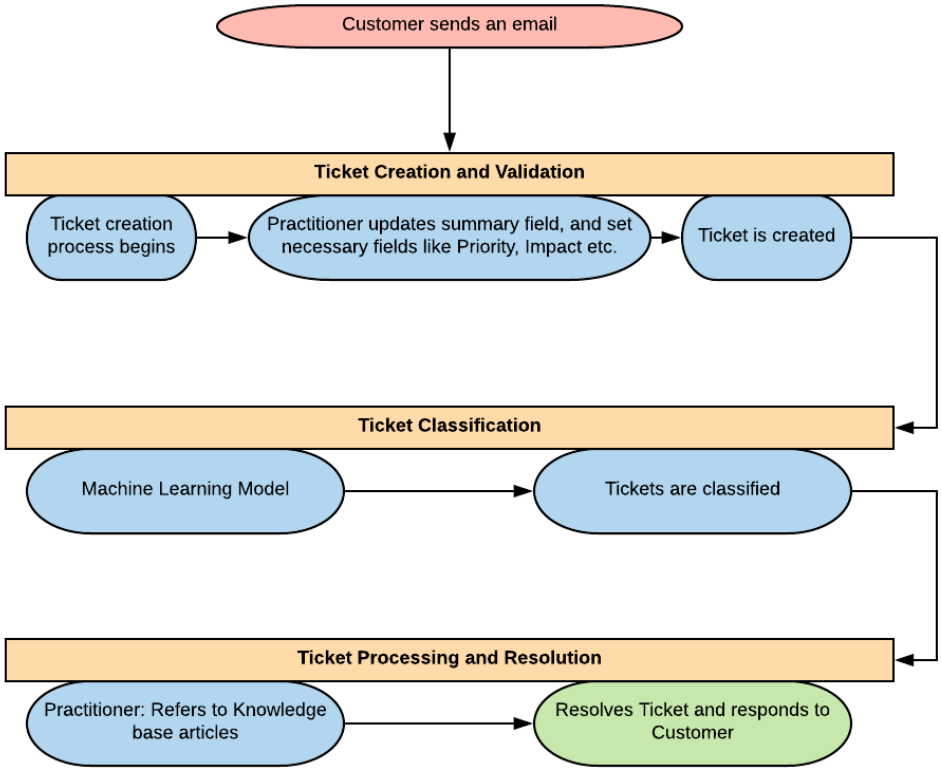

Figure 1.1: Steps involved in the process of resolving support tickets

1.2 My contributions

This thesis addresses the problem with manual ticket routing in IT support systems. Currently, it takes a lot of time and routine work to classify support tickets to correct teams. Based on two year history data, I have experimented logistic regression, random forests and extremely randomized trees on different sets of data in experimental setup 1. I have evaluated the models based on classification metrics like precision, recall and f1-score. The model with best results is deployed in production using AWS. This thesis uses software robots of RPA to communicate between the IT support system and the model deployed in cloud. A test run on real tickets in production is conducted with the model for two weeks. Model evaluation is performed.

1.3 Structure of the thesis

This thesis comprises of six chapters. Chapter 1 is a brief introduction to the problem of classifying support tickets and it also describes how it was addressed earlier. Chapter 2, discusses about different classification techniques in machine learning that are used in this thesis and also gives a brief description on RPA. Chapter 3 describes about the experiments and the methods used to classify support tickets. Chapter 4 evaluates all the methods used in this thesis. Chapter 5 discusses the role of machine learning in addressing the problem and also about the advantages of robotics when combined with machine learning in this context

Chapter 2

Background

2.1 Machine learning

Machine learning is a process of programming machines to improve, achieve optimized performance in solving a problem [1]. A problem might be a classification problem, clustering problem or any kind of machine learning problem. A machine learning problem can be represented using three concepts. First, a machine have to learn data to solve a task 'T'. In present case, the task is to classify the support tickets. Second, these machines require some experience 'E' to resolve the task. Experience in this context refers to the example or training dataset provided. Finally, The performance 'P' of the machine in solving the task is measured [6].

Machines learn, analyze example data and make predictions. In order to make good predictions from machine learning, it is important to have clean data which has more information and less noise. Data cleaning which is also termed as data preprocessing is a crucial step and the procedure of data cleaning differs widely with the type of data used in context. The type of data can be text, numbers, strings, categorical values, continuous values, images, audio files, video files, etc.

Currently, the data used in this thesis has text, categorical fields. Preprocessing of data involves different steps for text and categorical data. Text data should be cleaned in such a way that no punctuations, stopwords exist and then encoded to normalized vector format. Categorical data, on the other hand should be converted to respective categorical codes. After completing the preprocessing step, data can be used with suitable algorithms according to the requirement.

Algorithms in machine learning are divided into four broad categories. They are supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning. The Figure 2.1 shows the classification of machine learning algorithms and their sub categories. This thesis addresses the problem of classifying support tickets that belong to more than two categories, by implementing the algorithms that handle multi class classification problems. Multi class classification problem is a sub category of classification in which the learning is supervised. A brief description of supervised learning can be found in Section 2.2.

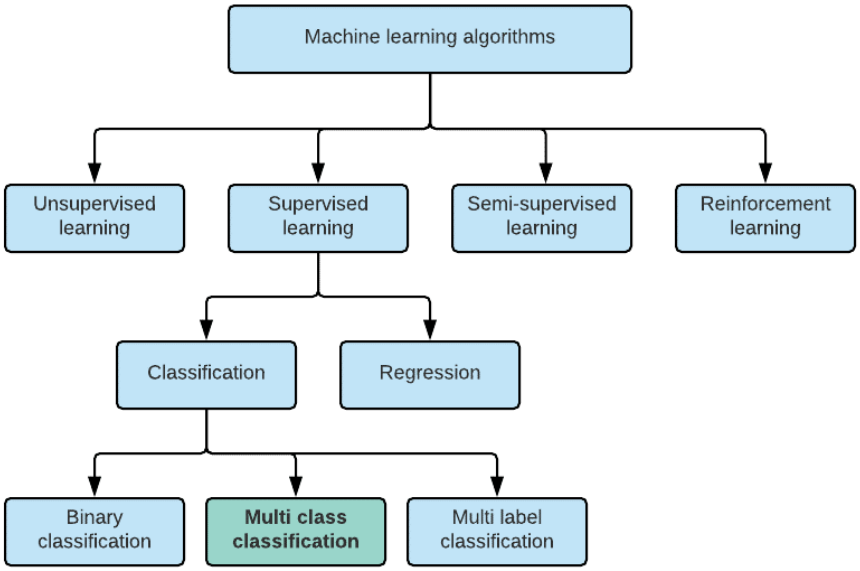

Figure 2.1: Types of machine learning algorithms

2.2 Supervised learning

Supervised learning algorithms are applied on data, which has labeled target variable/variables. In supervised learning, data which has specific features with desired target variables is given as input. Based on such training data, supervised algorithms predict the target variable of the test data [9]. Supervised learning can be divided into two categories – classification, regression.

In classification problems, the target variables are often called categories or labels. The goal of the classification problems is to predict the class/category/label of the target variable. A detailed description of classification algorithms used in this thesis is provided in Subsection 2.2.1. On the other hand, in regression problems, the target variable is a continuous value approximated basing on the input data.

2.2.1 Classification

As discussed, in classification problems, the target variable is categorical and the task is to predict if an observation belongs to certain category. As described in Figure 2.1, there are different kinds of classifications. A classification problem in which the task is to predict a single target value that might belong to one of two existing classes, is a binary classification problem. A problem in which the target value might belong to more than two classes, is a multi class classification problem [9]. On the other hand, if the problem aims at predicting more than one target variable where, each target variable might belong to two or more classes comes under a multi label classification. This thesis focuses on a multi class classification problem, where the target variable to be predicted might belong one of six existing categories. Algorithms that support multi class classification, are applied on current dataset and a detailed description of those algorithms is provided in 2.2.1.1, 2.2.1.2 and 2.2.1.3

2.2.1.1 Logistic regression

Logistic regression is a linear model for classification. It can be used for binomial classification and multi class classification. In multi class classification problems, multinomial logistic regression computes the probability for each class and choose the class that has the maximum probability [3].

Logistic regression uses different solvers like liblinear, lbfgs, sab, sag, newton-cg and few others [5]. Some of those like liblinear, saga support L1 penalization. For multi class classification problems, liblinear uses coordinate descent algorithm, which decomposes the optimization problem in one-vs-regression fashion and uses separate binary classifiers to train all classes. On the other hand, solvers like lbfgs, sag, newton-cg support L2 penalization. They converge faster for high dimensional data. Setting the multi_class parameter to 'multinomial' with these solvers, learns true multinomial logistic regression model and performs better when compared to one-vs-rest setting. Other solvers like sag, uses stochastic average gradient descent algorithm and runs

faster for large datasets.

Regularization solves the problem of overfitting. It adds a regularization term to prevent the coefficients to overfit. There are two types of regularization. Namely, L1 regularization and L2 regularization [8]. L1 regularization is a linear function of weight values that uses the L1 norm (absolute values, Manhattan distance) of weight values. On the other hand, L2 regularization uses the quadratic function of weight values that uses the L2 (square, euclidean distance) of the weight values. Some of the important parameters for logistic regression are listed below [5]

Important hyperparameters for logistic regression

Penalty
This parameter specifies the type of norm used in penalization. For example, L1, L2

Class_weight
As discussed in other classification algorithms, class_weight parameter works same as in other cases. It takes values like 'None', 'balanced' or dictionary values. For balanced mode, the classifier automatically adjusts class weight for each of the existing classes. Class weight of 'None' assigns a weight of one to all classes.

Solver
Specifies the solver to be used. Different types of solvers are newton-cg, lbfgs, liblinear, saga, sag etc. Each of those has their own advantages. Liblinear performs well for small data sets. Sag, saga runs faster for large datasets. Newton-cg, sag, saga, lbfgs can handle multi class classification problems using multinomial loss.

max_iter
Maximum number of iterations taken by the solvers to converge.

multi_class
This parameter takes two values, ovr, multinomial. If the parameter is set to ovr, then a binary problem is for each label. If it is set to multinomial, the classifier tries to minimise the multinomial loss.

2.2.1.2 Random forest classification

Random forest classification technique comes under ensemble learning. Ensemble learning is a powerful machine learning paradigm which trains multiple learners to solve a problem [4]. Unlike general machine learning algorithms, which learn one model from data, ensemble methods combine the predictions/hypothesis of each individual base learner to improve the performance. Base learners are generated from training data based on a learning algorithm like a decision tree or any machine learning algorithm.

Ensembles are constructed in two steps. At first, base learners are generated in either parallel or sequential fashion. Then, the outcomes of all base learners are combined in certain ways. One example approach for combining the results of base learners in classification problems is the majority voting scheme which returns the most frequent vote. In general, for regression problems, the weighted averaging scheme is used which returns the weighted sum. The quality of an ensemble depends on the accuracy, diversity of the base learners [11]. Accuracy of the learners can be measured using different techniques like cross validation, leave-one out etc. Diversity in base learners can be achieved by subsampling the training samples, injecting randomness to learning algorithms etc.

There are a variety of ensemble methods which differ widely in the way base learners are produced and/or in the different combination schemes used. Bagging, boosting are the two most popular ensemble methods.

Boosting
Boosting is a sequential ensemble method in which the base learners are generated sequentially [11]. In sequential methods, the goal is to strengthen the dependence between the base learners. They focus more on improving the performance of the weak learner. Sequential methods boost the overall performance by assigning higher weights to the previously mislabeled examples. Boosting aims at fitting a sequence of weak learners and the final outcome is obtained by combining the final predictions by majority voting scheme in classification and weighted sum in regression. AdaBoost, gradient tree boosting are some of the widely used boosting algorithms [4].

Bagging
Bagging uses bootstrap sampling in training base learners. Bootstrap sampling can be defined as subsampling of training data with replacement and

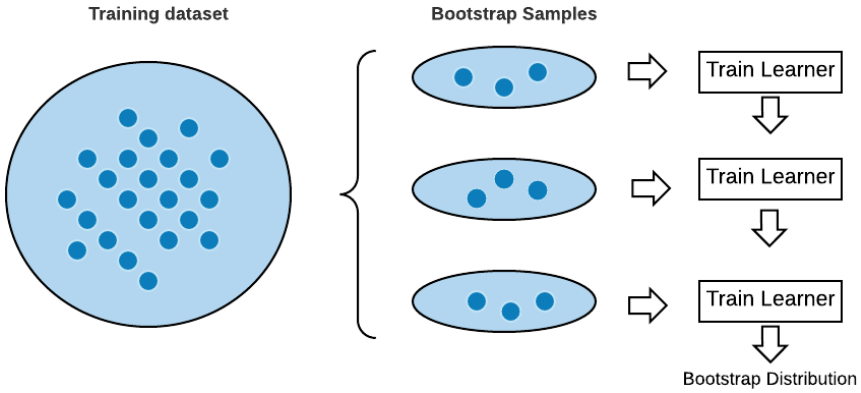

Figure 2.2: Bootstrap Sampling

the size of the sample and the training data set are same [4]. The Figure 2.2 depicts the subsampling of training dataset using bootstrap sampling. Each base learner is trained on a random sample of instances. The output of each individual base learners are aggregated by the ensemble classifier. Thus, the final outcome of the ensemble classifier has better accuracy compared to individual classifiers.

The algorithm for bagging is as below:
The bagging algorithm:

Input: Dataset D = $(\mathbf{x}_1, y_1)$,$(\mathbf{x}_2, y_2)$, $\cdots$,$(\mathbf{x}_m, y_m)$;
Base learning algorithm ℓ;
Number of learning rounds T.

Process:
for $t = 1, \cdots, T$:
$D_t = Bootstrap(D);$
$h_t = \ell(D_t)$ % Train a base learner h_t from the bootstrap sample
end.

Output: H($\mathbf{x}$) = $argmax_{y \in Y}$ $\Sigma_{t=1}^{T}$ 1(y = h_t ($\mathbf{x}$)) % the value of 1(a) is 1 if a is true and 0 otherwise

Random forests are a good example of bagging technique. They fit numerous decision tree classifiers on different subsamples of dataset. They follow

the majority voting scheme for classification and averaging for regression to improve accuracy, control over-fitting [4]. There a number of parameters and varying each of those parameters lead to different results. Some of the important parameters for random forests are listed below [5].

Important hyperparameters for random forests

n_estimaters
The total number of trees in the forest. The default value of n_estimaters is 10. For larger datasets, an increase in the value of n_estimaters yields better results. But, the results stop getting better after reaching a threshold value.

max_features
Number of features to consider when looking for the best split. The default value is 'auto', for which the max_features value is equal to the square root of total number of features. Other options include log2 (logarithm of total features), float (a defined percentage of features), int (defined number of features) etc.

max_depth
max_depth of the tree takes integer value or it can also be set to None. For large datasets, an increase in value of max_depth leads to more accuracy.

class_weight
class_weight takes values like 'balanced', 'None', or a dictionary. For classification problems that have a class balance do not require class_weight parameter and the value of class_weight parameter can be assigned to 'None'. If the value is 'None', then all classes are supposed to have weight one. But, multi class classification problems with class imbalances need class_weight parameter to balance the imbalance in between different classes. If class_weight is 'balanced', the classifier adjusts the weights of classes inversely proportional to the class frequencies of input data as

bootstrap
The default value for this parameter is set to 'True'. If it is set to true, the classifier uses the bootstrapped samples while building trees.

$$\frac{n\_samples}{(n\_classes * bincount(y))}$$

Another alternative to resolve the class imbalance problem is to pass a dictionary as class_weight parameter. The dictionary should have integer values

as weights assigned to each class.

n_jobs
n_jobs parameter takes an integer. It specifies the number of jobs that run in parallel. If it is equal to -1, the number of jobs is set to number of cores.

2.2.1.3 Extremely randomized trees

Extremely randomized trees, also known as extra trees algorithm is one of the ensemble methods that resembles random forests. Similar to random forests, they build several individual estimators and average the predictions of all estimators to control over-fitting and improve prediction accuracy. They differ from random forests in two aspects. First, unlike random forests, which use bootstrapping for sampling training data, extremely randomized trees use the entire training dataset. Second, to randomly select 'n' number of attributes at a particular node, they chose a random cut-point without taking the target variable into consideration [13]. Extremely randomized trees are one step ahead to random forests in randomizing thresholds while splitting. They focus on decreasing the variance of the model. But, this often leads to higher values of bias.

Extra trees algorithm can be used in classification or in regression tasks. In this context, extra trees classifier is used for handling multi class classification problem. All parameters of extra trees classifier are same as the parameters of random forest classifier except 'bootstrap'.

Hyperparameters of extremely randomized trees that differ from random forests

bootstrap
This parameter is by default set as 'False' where as in random forests, its value is set to 'True'. If the value is false, it means that the classifier does not use bootstrapping for sampling training dataset [5].

2.3 Robotic process automation

RPA, which is termed as robotic process automation is a software technology developed from the concept of automation. Automation can be described as developing a system or programming a workflow that requires no human intervention [14]. RPA makes use of software which mimics human actions in

a system and interacts with various applications to complete various rule-based tasks. RPA is basically designed to perform a variety of rule-based tasks, complex processes. It has virtual assistants that can respond, communicate with other systems as humans.

In Basware, RPA is based on a tool called UiPath. UiPath is an RPA technology vendor who designs the software that automates business processes [14]. The basic components of UiPath RPA platform are UiPath studio, UiPath robot, UiPath orchestrator.

UiPath studio is a development environment of UiPath and can be used to design robotic processes. It is a modeling tool with which one can develop automations visually in less time without requiring high level coding skills [14]. UiPath studio has a variety of activities, which performs an action or a task. Examples of activities include writing text to a text box, selecting radio buttons, clicking buttons to an excel sheet, reading and/or writing a file etc.

Uipath robot is a Windows service that can execute processes or set of activities, designed or developed or recorded using UiPath studio. It is an execution agent which runs the automation projects/processes that are designed in UiPath studio. UiPath robots are managed by UiPath orchestrator. They can run in either attended or unattended environments. Robots running in attended environments works only on human trigger where as the robots that run in unattended environments can work on their own.

UiPath orchestrator is a server based application that helps humans in orchestrating the robots. Orchestrator runs on a server and all the robots are connected to orchestrator. It has a user interface by which humans can create, manage, monitor, deploy resources in the environment. Processes in orchestrator can be scheduled and orchestrator manages the processed that are in the queue.

Chapter 3

Methods and experiments

This chapter focuses on the methods and experiments carried out on the data. It is divided into four sections. The first section describes the data used in this project. The second section presents the steps taken for cleaning the data. The third and fourth sections describe the two different experimental setups employed. The former discusses about the off-line cross-validation with different machine learning algorithms and the later is about the best performing model, deployed in production.

3.1 Data description

Data used in this thesis is collected over a period of 2 years (2016 and 2017). It has information regarding the end-to-end process of resolving ticket, application configuration of the system and meta data of specialists involved in resolving tickets. The total volume of data is 2 gigabytes. It has 280 columns and 381528 rows, where each row represent a single ticket. Tickets are in different languages like English, Finnish, German, Swedish etc. There are 114747 tickets that are in English and the scope of this thesis is limited to English tickets.

It is important to identify the columns that play key role in the process ticket routing. Most of the columns in the data do not contribute to the ticket routing process and are considered to be irrelevant. According to the domain specialists, who are involved in the process of ticket resolution, the three key columns are 'Description', 'Detailed Description' and 'Service CI'. Also, basing on the Basware knowledge repositories for ticket routing, certain columns that provide information regarding the impact, priority, company, product and operation categorizations of the issue raised are also considered

as relevant. The relevant fields are listed in Table 3.1.

The scope of the thesis is limited to Basware customer care unit. Basware customer care unit has seven groups. For all incoming tickets, it is the duty of the specialist to manually process and route the tickets to specific group. Different levels of Basware customer care include, 'level 1', 'level 21', 'level 22', 'level 23', 'level 24', 'level 25', 'Others'. The distribution of tickets to the respective groups can be found in the Figure 3.1

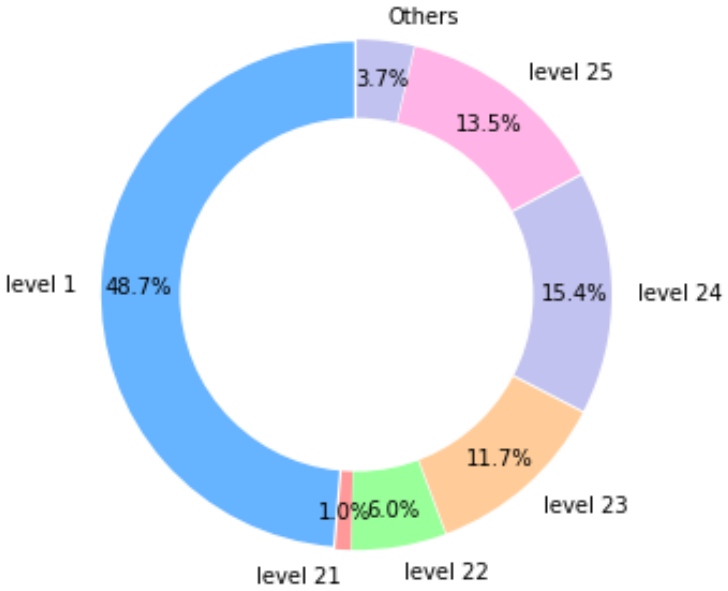

Figure 3.1: Ticket distribution

3.2 Data preprocessing

Initially, database administrators exported data to tsv files. The reason of choosing tsv format is that the columns like description, detailed description are in text format and have punctuations and commas. If we choose csv instead of tsv, the column delimiter might not work well and leads to data corrupting issues. Tab separated files addresses such challenges. Data from 24 tsv files corresponding to two years (24 months) of data is converted to a pandas dataframe. This thesis uses a freely available machine learning library, scikit-learn for Python

Column names	**Type**	**Description of columns**
description	text	summary of the detailed description field
detailed description	text	detailed description of the email/ticket
service CI	string	category of service to which the ticket is assigned
impact	number	It indicates the impact of the ticket.
priority	number	It indicates the priority of the ticket.. It is set by the practitioner while creating the ticket.
product categorization tier 1	string	A categorical string representing the product categorization in tier 1
product categorization tier 2	string	A categorical string representing the product categorization in tier 2
product categorization tier 3	string	A categorical string representing the product categorization in tier 3
assigned group	string	group to which the ticket is routed.
assigned groups	number	ID's representing the group ID's to which the ticket is routed
company	string	respective company
categorization tier 1	string	A categorical string representing the operational category in tier 1
categorization tier 2	string	A categorical string representing the operational category in tier 2
categorization tier 3	string	A categorical string representing the operational category in tier 3

Table 3.1: List of relevant columns and their description

Filtering necessary columns
As discussed, there are 280 columns out of which 14 columns are important. Important columns are filtered accordingly.

Filter english tickets
The scope of this thesis is limited to classification of English tickets which represent around 50% of total tickets. There are existing language codes for 2016 and 2017 data. Only those rows that correspond to English tickets are filtered.

Remove redundant data
In the two year data, there are certain tickets that are frequently received from the customer. Certain tickets that are completely similar to any of the existing tickets in the data can be removed. Including such repeated tickets do not add any extra value to the machine learning algorithm. It is important to remove those rows from the data.

Filter relevant groups
Currently, the data constitutes of tickets that belong to a variety of groups. Tickets that belong to Basware customer care unit are filtered and all other groups are excluded.

Handling null values
In current dataset, there are certain fields with 'NaN' fields. One way of handling 'NaN' fields, is to exclude rows that have 'NaN' fields. But, in present case, excluding 'NaN' fields would decrease the accuracy of the classifier. Because, as shown in the Figure 3.2, in the column, 'categorization tier 3', almost 53% of data is filled with 'NaN' fields. So, excluding them do not add value in present context. Also, these columns are important in routing a ticket. The better way to handle such data would be to replace the 'NaN' values with the most suitable value.

One way to handle 'NaN' fields is to replace them with the most frequently occurring value. But, using the most frequent values is not apt for columns like Categorization Tier 2, categorization tier 3, product categorization tier 1, product categorization tier 2, product categorization tier 3. Assigning the most frequently occurring category to these groups would mislead the classifier in prediction. Because, for tickets that belong to level 1, the categorization tier 3 field would always be empty and for tickets other than level 1, the values are not empty. So, replacing such empty fields with the most

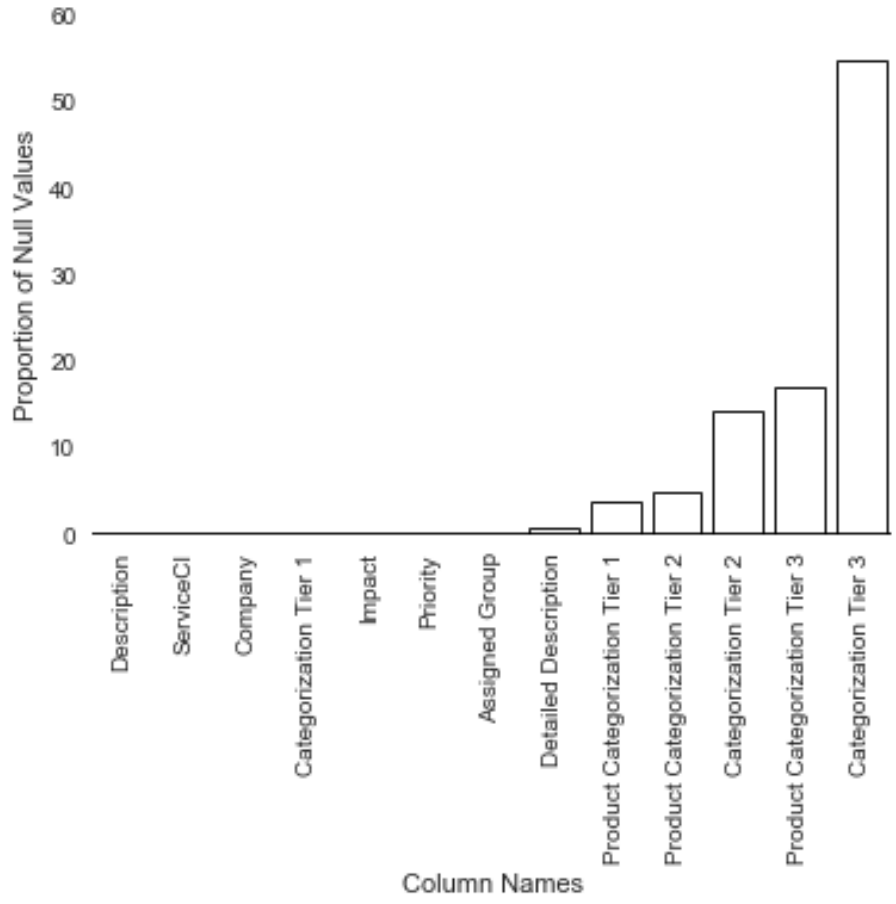

Figure 3.2: Proportion of null values

frequent category misleads the classifier. To address that problem, fields that have 'NaN' values are replaced with a new value, 'MISC' that represents a new category.

Excluding tickets that are assigned to multiple groups
In current data, there are few tickets that are assigned to more than one 'assigned groups'. There is no clear evidence whether they are misdirected to several groups or routed to a specific set of groups to resolve the ticket. Those tickets which have more than one ID in 'assigned group' are excluded from the data.

Categorical encoding
Except 'description' and 'detailed description' columns, all the other columns

have categorical data as strings with datatype 'object'. It is important to encode such categorical strings to numerical categories.

Text processing
There are two text fields in data: 'description' and 'detailed description'. These text fields comprise of characters, punctuations, non-character and extra spaces. Machine learning algorithms cannot be applied directly on text data. Text data should be converted into vector format to make them ingestible for machine learning pipelines [10]. Text data can be represented using bag of words model in which each word is considered as a feature and text in each ticket is transformed into a vector of non-zero elements for all the features.

TfidfVectorizer
Transforming text to quantitative data can be done using transformers like TfidfVectorizer, countVectorizer. In this project, TfidfVectorizer is used to transform text fields and vectorize them using tf-idf [5]. Currently, there are 80,000 tickets. TfidfVectorizer is applied on two text columns and a huge vector of size 80000*180000 dimensions is generated. There are certain parameters in TfidfVectorizer that control the noise by reducing the number of features used.

Some important parameters for TfidfVectorizer

stopwords
It takes a string, list or 'None' as input. If it is a string, the only supported value is 'English'. It checks for the stop list for English language and excludes the stop words from the features. If a list of stop words is passed, all of elements in the list will be removed from the resulting features.

mindf
It ignores the terms that have a document frequency strictly less than the given threshold.

maxdf
It ignores the terms that have a document frequency strictly greater than the given threshold.

use_idf
This parameter is set to true by default. It enables the inverse-document-frequency reweighting.

Combining heterogeneous data
The result of tfidfVectorizer for each text field should be concatenated with the numerical categories. FeatureUnion, is such kind of estimator which concatenates the results of multiple transformers applied on heterogeneous data. The combined result can now be used to train a model using machine learning algorithms like SVM, Logistic regression, Random Forests etc. The Figure 3.3 clearly depicts the data preprocessing steps taken before training the model.

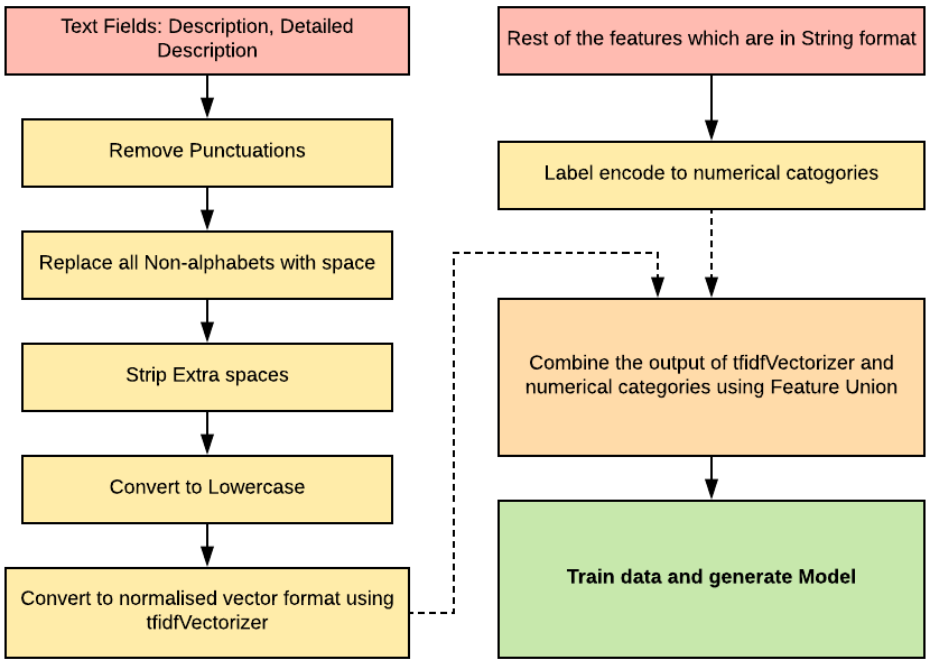

Figure 3.3: Data preprocessing steps

3.3 Experimental setup 1

Basically, this thesis has two experimental setups. This section discusses about the methods used and experiments carried in setup 1. Details regarding setup 2 can be found in Section 3.4. In experimental setup1, classification techniques which support multi class classification like logistic regression,

random forests, extremely randomized trees are chosen and experimented with different sets and volumes of data.

Listing 1: Python code for splitting data into training and test sets

```
from sklearn.model_selection import train_test_split

#Split data into train, test datasets
x_train, x_test, y_train, y_test = train_test_split(
                twoYearData.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12]],
                twoYearData.iloc[:, [8]],
                test_size=0.33)
```

3.3.1 Off-line cross-validation

Training and test datasets
Initially, the data is randomly split into two sets, training data and test data. Training data and test data are divided in such a way that two thirds and one third of data respectively. An example code for such random split of data into training and test datasets can be found in Listing 1. The four variables x_train, y_train, x_test and y_test represent the training and test datasets. y_test, y_train datasets have a single column 'assigned_group', which is the target value to be predicted. x_train and x_test includes all the fields in the data except the target variable as listed in Listing 1. A machine learning model is trained on the x_train, y_train datasets as in Listing 2. The same model is then used on test dataset to predict the target variable.

The reason for splitting data is to evaluate the performance of the classifier on a new unseen dataset [12]. The new unseen dataset in this case is the test set. A machine learning pipeline is constructed, which uses a scikit 'FeatureUnion' fuction to vectorize the text features and then combine them with the categorical features on the training data. In the next step, the pipeline generates models on the combined dataset using different machine learning algorithms. The predictions made by the model can be optimized by tuning the hyper-parameters of the classification algorithms. It is often hard to manually tune the parameters with a random guess. This thesis uses a sci-kit 'GridSearchCV' function to address such problem.

Grid Search
Grid searchCV uses cross-validation and performs an exhaustive search on a grid of hyperparameter values to determine the model with best performance. Hyperparameters are passed as arguments to the estimators. As discussed,

the values of hyperparameters can be changed to optimize the performance of the model. It is often recommended to search in the hyperparameter space that have the best cross validation score [7]. An example code for finding the best fit can be found in Listing 2. The returned list of best values for the hyperparameters can also be found in Listing 2

Cross-Validation
Cross-Validation is a statistical technique used in machine learning context to get the best estimate of error [12]. This thesis uses a 5-fold cross-validation, which splits the data into five approximately equal parts and each partition in turn is used as testing and remainder is used for training.

Listing 2: Python code for finding the best parameters using GridSearchCV

```
pipeline = Pipeline([('features',Datafeatures),
                ('classifier', RandomForestClassifier(random_state = 0))])

hyperparameters = {'classifier__max_depth': [10, 50, 70, 90, 100, 200],
'classifier__min_samples_leaf': [2],
'classifier__n_jobs': [-1],
'classifier__n_estimators': [10, 50, 70, 90, 100, 200],
'features__Description__tfidf__max_df': [0.4, 0.5, 0.7, 0.9, 0.95],
'features__Description__tfidf__min_df': [0.004, 0.001, 0.01, 0.1],
'features__Detailed_Description__tfidf__min_df':[0.004, 0.001, 0.01, 0.1],
'features__Detailed_Description__tfidf__max_df':[0.4, 0.5, 0.7, 0.9, 0.95],
'features__Detailed_Description__tfidf__token_pattern': [r'[a-zA-z]+']}

model = GridSearchCV(pipeline, hyperparameters, cv=5)
model.fit(x_train, y_train)
predic = model.predict(x_test)
model.best_params_
model.refit


####################### Output: Best Parameters ########################

{'classifier__max_depth': [90],
'classifier__min_samples_leaf': [2],
'classifier__n_jobs': [-1],
'classifier__n_estimators': [100],
'features__Description__tfidf__max_df': [0.9],
'features__Description__tfidf__min_df': [0.004],
'features__Detailed_Description__tfidf__min_df':[0.004],
'features__Detailed_Description__tfidf__max_df':[0.7],
'features__Detailed_Description__tfidf__token_pattern': [r'[a-zA-z]+']}
```

Case 1
As shown in Table 3.2 different experiments are conducted for three different

Case	Volume	'detailed description' field	CS level25
1	2 years	Excluded	Included
2	2 years	Included	Included
3	2 years	Included	Excluded

Table 3.2: Datasets used in different cases

cases. There are two text fields as mentioned in Table 3.1. As the 'detailed description' field has lot of noise, case 1 excludes the 'detailed description' field to observe the performance of different classifiers. Logistic regression, random forests, extra trees classifier are trained on the data. The performance of these algorithms is tabulated in Table 4.3.

Case 2
Case 2 includes the 'detailed description' field to observe if the classifier would perform better in this scenario. All three classifiers are trained on the data and the performance of these classifiers is tabulated in Table 4.5. There is an interesting observation from the confusion matrices of each classifier. Table 4.6 represents the confusion matrix of the random forest classifier on two year data set including 'detailed description' field. From the table, it is clear that, good amount of 'level 25' tickets are classified to 'level 1' and vice versa.

After further investigation, it was found that the reason for this kind of merge in between these two groups is because of having different consoles for incoming tickets. Practitioners dealing with the tickets that arrive to console A, do not have access to assign it to 'level 25'. So, those tickets are assigned to the 'level 1' that deals with the similar kind of tickets.

Case 3
To address the merge of tickets in between 'level 1' and 'level 25', case 3 excludes 'level 25' group from the data. The three classifiers are trained on remaining data. The results can be seen in the Table 4.7. Basing on results, it can be concluded that the performance of extra trees classifier and random forest classifier are better compared to logistic regression. Random forest has the best scores for precision , recall and f1-score.

3.4 Experimental setup 2

In experimental setup 2, this thesis focuses on deploying the best performing machine learning model in production environment, testing its performance

on real tickets, evaluating the performance of the model. The model that shows best results during off-line cross-validation on test data is deployed in production. A test run on two weeks of tickets is performed using the same model. The performance of the model can be seen in Figure 4.4.

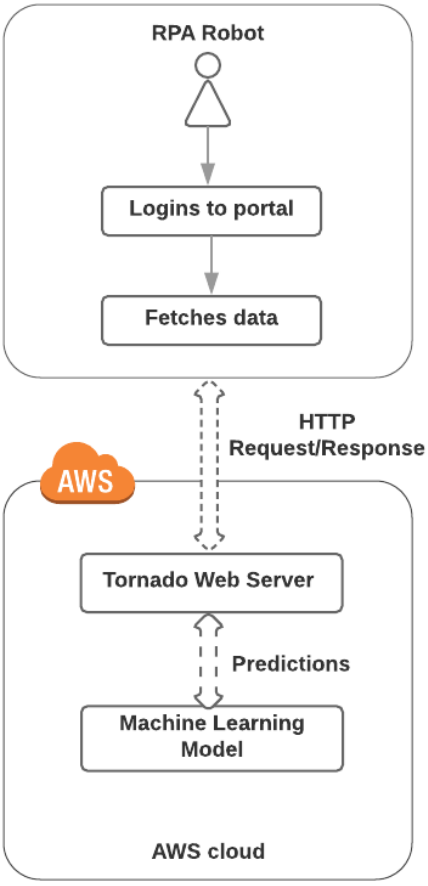

Figure 3.4: Process workflow of machine learning combined with robotics

3.4.1 Production deployment

Basware makes use of RPA robots to add a language code to existing support tickets of ITSM portal. This thesis extends the functionality of the existing robot to get the predictions of the model for all incoming tickets. The robot is

scheduled in orchestrator and runs accordingly. It fetches the necessary fields of an incoming ticket and generates a HTTP request to the web server. This project uses Tornado, a powerful scalable web server for handling the HTTP requests. AWS (Amazon web services) hosts the Tornado web server. The machine learning model and other required files are stored in scikit pickled format and are deployed to AWS. When a HTTP request is send by robot, Tornado web server handles the request, get the predictions, responds to the robot. The Figure 3.4 gives an overview of the end-to-end process of classifying the support tickets.

Chapter 4

Evaluation

In Chapter 3, we discussed how different models are generated on training dataset, how the generated models are used to make certain predictions on test data set, how models are tuned using grid search. But, a performance measure of these developed models is required in order to choose a model that suits best to the current requirement. Model evaluation plays a prominent role in building generalized models, tuning them, and choosing the best model by using numerical metrics [12]. Usually, these metrics vary depending upon the type of problem. For example, classification problems have metrics different from that of the regression and clustering problems.

4.1 Classification metrics

Classification performance of a model can be measured using a variety of metrics. As the data used in this thesis deals with a multi classification problem, a subset of metrics that suit well for current purpose should be identified. This thesis uses confusion matrix, precision, recall and f1-score in evaluating the models for multi class classification.

4.1.1 Confusion matrix

Confusion matrix, is one of the popular metrics for evaluating classification models. It is a matrix representation of data where each cell c[i, j] represents the instances that are classified with label j, when it actually belongs to label i [2]. Each column in the matrix represents the count of instances classified based on the predictions of the model and each row represents the count of instances based on the actual labels [12]. The diagonal elements, c[i, i] are

	A	B	C	D	E	F
A	**15103**	1	314	276	729	117
B	0	**278**	0	42	0	0
C	262	0	**1750**	0	3	30
D	110	18	3	**3708**	66	14
E	200	0	6	76	**4832**	35
F	267	5	77	235	154	**482**

Table 4.1: An example of a confusion matrix

the instance counts of the number of correct classifications for a respective group and the off-diagonal elements represent the misclassified instances. Confusion matrix visualizes different types of errors made by the classifier. The below list of concepts describes more about the general type of errors made by the classifiers [2]. An example of confusion matrix can be seen in Table 4.1.

True positives
True positives are the relevant instances that are correctly classified by the classifier as relevant. As shown in Table 4.1, all the diagonal elements that are in bold, represent the true positives corresponding to each class. For example, consider the diagonal element corresponding to column 'A', which has a value of '15103' and is in bold. It indicates that '15103' instances which actually belong to class 'A' are correctly classified with label 'A'.

False positives (type 1 errors)
False positives are the irrelevant instances that are misclassified as relevant. In confusion matrix, all the elements in a column excluding the diagonal element represent the false positives. For example, in Table 4.1, all the elements in column 'A', except the diagonal element represent the false positives of class 'A'.

True negatives
True negatives are the irrelevant instances that are correctly identified as irrelevant.

False negatives (type 2 errors)
False negatives are the relevant instances that are misclassified as irrelevant. In confusion matrix, all the elements in a row excluding the diagonal element represent the false negatives. For example, in Table 4.1, all the elements in

the first row, excluding the diagonal element represent the false negatives of class 'A'. Those instances which have a true label as class 'A', are misclassified to other classes by the classifier.

This thesis uses a classification report that gives a text report of important classification metrics like precision, recall, f1-score for each class. For multi classification problems, the notions of precision, recall and f1-score are applied independently for each class/label as described above. In addition, classification report provides a computed average of the precision, recall, f1-scores for all the classes basing on the weighted (weighted by considering the number of true instances for each class) macro average across the classes as shown in Table 4.2. The metrics in Table 4.2 correspond to the confusion matrix that is tabulated in Table 4.1.

4.1.2 Precision

Precision is a measure of result relevancy. It is the ratio of the instances that were correctly classified as relevant to the total number of instances that were classified as relevant.

$$Precision = \frac{TP}{TP+FP}$$

As shown in Table 4.2, metrics related to the precision of each individual class is listed as column 1. Those individual precision values for each class are calculated with the above mentioned formula.

4.1.3 Recall

Recall is the percent of relevant instances that were correctly classified as relevant. It can be calculated as shown below.

$$Recall = \frac{TP}{TP+FN}$$

In Table 4.2, the recall metrics for each class are listed in column 2.

4.1.4 f1-score

f1-score is the harmonic mean of precision and recall. It provides a balanced optimization score of both precision and recall. f1-score is the F-beta score, where beta=1.0. It can be calculated as below:

$$F1Score = \frac{2*Precison*Recall}{Precision+Recall}$$

	Precision	Recall	F1 Score	Support
A	0.95	0.91	0.93	16540
B	0.92	0.87	0.89	320
C	0.81	0.86	0.83	2045
D	0.85	0.95	0.90	3919
E	0.84	0.94	0.88	5149
F	0.71	0.40	0.51	1220
Average/Total	**0.90**	**0.90**	**0.89**	**29193**

Table 4.2: An example of a classification report

The f1-score metrics corresponding to the Table 4.1 are listed as column 3 in Table 4.2.

4.2 Results

4.2.1 Experimental setup 1

As discussed in Chapter 3, there are three cases in experimental setup 1. In all three cases, the data is experimented with three different algorithms: logistic regression, extra trees classifier and random forests.

Case 1
In case 1, data excludes the 'detailed description' field. Algorithms like logistic regression, extra trees classifier and random forests are applied on the data. After tuning the hyperparameters, the best results obtained for each classier are tabulated in a single table. Table 4.3, gives an overview of results.

The results are tabulated in such a way that each row represents individual scores of precision, recall and f1-score for each group with respect to the algorithm used. The last row gives the average of individual scores for all groups. Basing on these results, it is clear that the average scores of precision, recall, f1-score are high in case of extra trees classifier compared to the rest. For certain groups like level 1, level 23 and level 24 the individual scores exceeded 80% where as, the individual scores for the rest of the groups are pretty low.

Case 2
In case 2, data includes the 'detailed description' field. The results of all

Case 1

Class	**Logistic regression**			**Extra trees**			**Random forests**			Support
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	
level 1	0.77	0.91	0.83	0.92	0.70	0.79	0.88	0.73	0.80	14603
level 21	0.89	0.42	0.57	0.77	0.90	0.83	0.76	0.76	0.76	99
level 22	0.85	0.47	0.60	0.64	0.85	0.73	0.48	0.84	0.61	1350
leve1 23	0.88	0.90	0.89	0.90	0.94	0.92	0.88	0.94	0.91	3290
level 24	0.83	0.87	0.85	0.84	0.94	0.89	0.83	0.94	0.88	4543
level 25	0.56	0.29	0.38	0.47	0.71	0.57	0.51	0.56	0.53	3958
Others	0.00	0.00	0.00	0.19	0.37	0.25	0.16	0.22	0.18	434
Avg/total	**0.76**	**0.78**	**0.76**	**0.82**	**0.77**	**0.78**	**0.79**	**0.76**	**0.77**	28277

Table 4.3: Performance of three classifiers for different groups in case 1

	level 1	level 21	level 22	level 23	level 24	level 25	Others
level 1	**10159**	2	528	212	618	2705	379
level 21	0	**89**	0	10	0	0	0
level 22	78	0	**1152**	2	2	106	10
level 23	18	21	1	**3099**	22	118	11
level 24	40	0	4	57	**4254**	110	78
level 25	735	1	97	44	57	**2818**	206
Others	31	2	5	35	85	114	**162**

Table 4.4: Confusion matrix of extra trees classifier in case 1

Case 2

Class	Logistic regression			Extra trees			Random forests			Support
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	
level 1	0.80	0.91	0.85	0.80	0.93	0.86	0.85	0.85	0.85	16360
level 21	0.96	0.75	0.85	0.91	0.85	0.88	0.89	0.84	0.86	347
level 22	0.87	0.66	0.75	0.91	0.72	0.81	0.84	0.83	0.83	2043
level 23	0.88	0.94	0.91	0.87	0.94	0.90	0.88	0.96	0.92	3920
level 24	0.89	0.91	0.90	0.88	0.92	0.90	0.85	0.93	0.89	5224
level 25	0.65	0.47	0.55	0.70	0.42	0.53	0.60	0.59	0.59	4628
Others	0.67	0.26	0.38	0.96	0.24	0.38	0.81	0.32	0.46	1235
Avg/total	**0.80**	**0.81**	**0.80**	**0.82**	**0.82**	**0.80**	**0.82**	**0.82**	**0.81**	33757

Table 4.5: Performance of three classifiers for different groups in case 2

three classifiers are tabulated in Table 4.5. Based on the results, it is evident that the performance of random forest classifier is better compared to the rest.

The Figure 4.1 presents the comparison between the results of case 1 and 2 for all three algorithms. The plot contains two subplots, subplot1 on the upper half and the subplot2 on the lower half. Subplot1 is a factor plot that compares the performances of three algorithms in two different cases with respect to the metrics like precision, recall and f1-score. The bars that are in blue, represent the metric scores in case 1 and the bars in green are the scores of classifiers in case 2. For the subplot1, the scale of y-axis range from 0 to 1. The only difference between the subplot 1 and 2 is that the subplot2 is the zoomed version of subplot1, which visualizes only the necessary details by rescaling the y-axis. From the plot, it is also clear that the results of case 2 are better than that of case 1.

The confusion matrix for the best performing algorithm, random forest in this case is tabulated in Table 4.6. From the table, it is evident that a huge proportion of level 1 tickets are misclassified as level 25 and vice versa. The reason for this is that the training data has tickets that are misclassified by the specialists due to lack of access to certain groups.

Case 3

The issue with the merge of tickets between level 1 and level 25 groups can be addressed in two ways. One approach would be to remove all the tickets that are misclassified by the specialists to these groups. Second approach is to exclude all the tickets that belong to level 25.

Proceeding with the first approach is highly impossible for this kind of data.

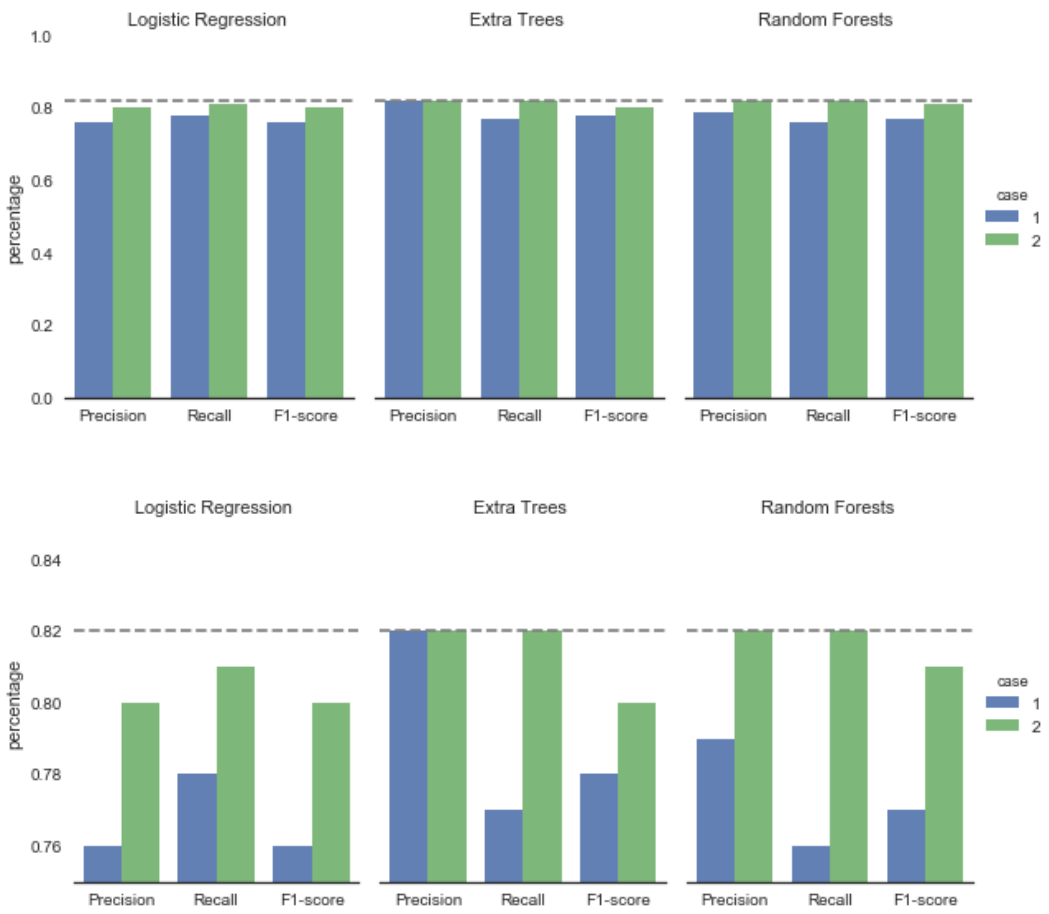

Figure 4.1: Performance comparison of three classifiers in case 1 and 2

	level 1	level 21	level 22	level 23	level 24	level 25	Others
level 1	**13939**	0	193	228	621	1334	45
level 21	0	**292**	0	54	1	0	0
level 22	247	30	**1695**	1	4	79	17
level 23	55	30	1	**3683**	58	92	1
level 24	152	0	2	69	**4869**	109	23
level 25	1741	0	39	49	73	**2717**	9
Others	180	7	92	234	131	194	**397**

Table 4.6: Confusion matrix of random forest classifier in case 2

Case 3

Class	Logistic regression			Extra trees			Random forests			Support
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	
level 1	0.90	0.96	0.93	0.95	0.91	0.93	0.94	0.93	0.93	16540
level 21	0.90	0.81	0.85	0.92	0.87	0.89	0.92	0.85	0.88	320
level 22	0.87	0.73	0.79	0.81	0.86	0.83	0.86	0.82	0.84	2045
level 23	0.87	0.91	0.89	0.85	0.95	0.90	0.86	0.94	0.90	3919
level 24	0.87	0.87	0.87	0.84	0.94	0.88	0.85	0.93	0.89	5149
Others	0.86	0.27	0.41	0.71	0.40	0.51	0.79	0.35	0.48	1220
Avg/total	**0.89**	**0.89**	**0.88**	**0.90**	**0.90**	**0.89**	**0.90**	**0.90**	**0.90**	29193

Table 4.7: Performance of three classifiers for different groups in case 3

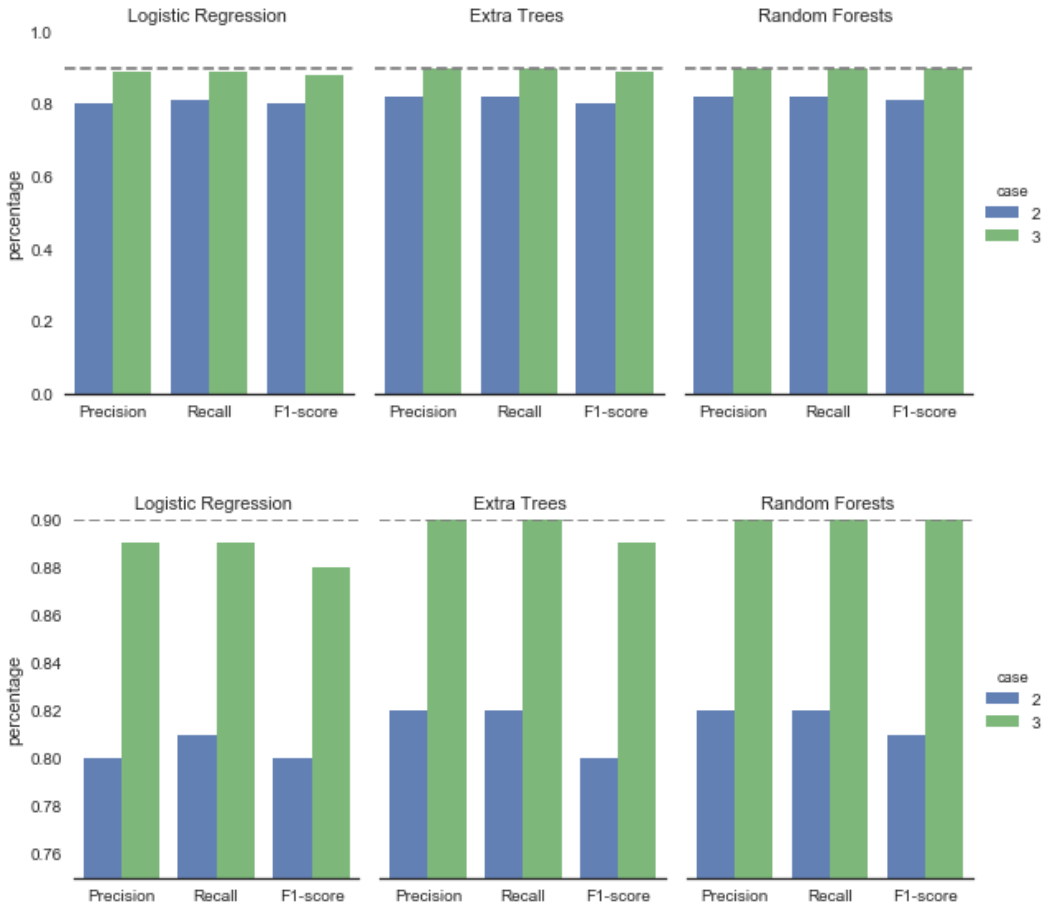

Figure 4.2: Performance comparison of three classifiers in case 2 and 3

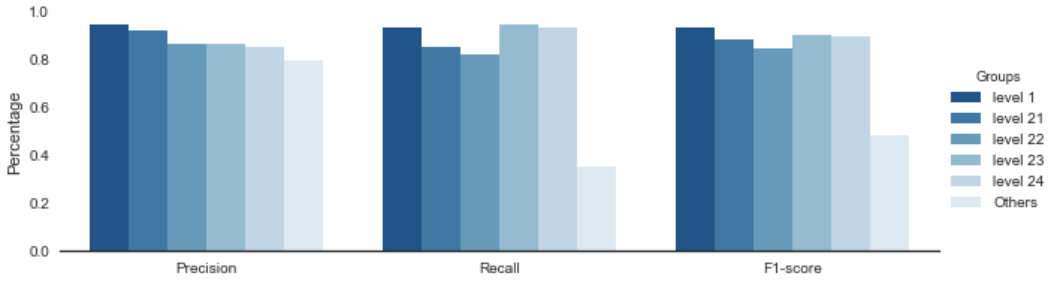

Figure 4.3: Performance of random forest classifier for different groups in case 3

Because, it is not possible to identify such tickets that are misclassified.This thesis proceeds with approach 2 as the total number of tickets that belong to level 25 are less in number. Hence, the scope of case 3 is limited to Basware customer care unit excluding the level 25 group.

The data is experimented with logistic regression, extra trees and random forests. The individual scores of each classifier with respect to the metrics for each group are tabulated in Table 4.7. In the table, the last row that is in bold, represents the average of scores for all groups with respect to each classifier. It is clear that the average scores are high for the random forest classifier when compared to the rest. Figure 4.2 compares the performance of all classifiers for case 2 and 3. Figure 4.2 is similar to Figure 4.1 in all aspects expect that Figure 4.2 compares the performance of classifiers in case 2 and 3. It is evident that the performance of all three classifiers is improved in case 3 compared to case 2. Random forest in case 3 exhibits an average score of 90% over all the groups for all the metrics. For random forest classifier, the individual metric scores of each group are plotted in Figure 4.3

4.2.2 Experimental setup 2

As discussed in Chapter 3, the experimental setup 2 has the best performing model running in production on real tickets. Random forest model is the best performing model in off-line cross-validation. It is deployed in production and its performance is tested for two weeks over 614 tickets. The predictions of the classifier are compared against the human predictions. The classification metrics can be found in Table 4.8. In the table, the rows that are in bold have better scores of precision, recall and f1-score.

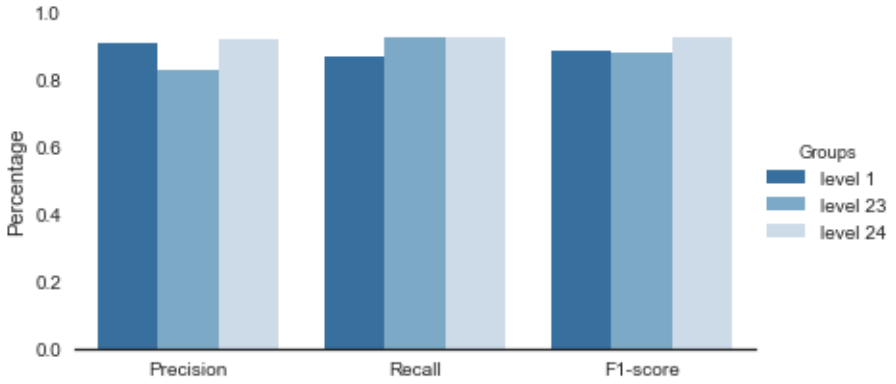

Figure 4.4: Performance of the model in production for level 1, level 23, level 24 groups

	Precision	Recall	F1 Score	Support
level 1	**0.91**	**0.87**	**0.89**	**275**
level 21	0.00	0.00	0.00	6
level 22	0.46	0.50	0.48	12
level 23	**0.83**	**0.93**	**0.88**	**155**
level 24	**0.92**	**0.93**	**0.93**	**153**
Others	0.00	0.00	0.00	13
Average/Total	0.86	0.87	0.86	614

Table 4.8: Classification report of the model in production

The average scores over all groups exceed 85%. The f1-scores exceed 88% for the level 1, level 23 and level 24 groups, which share approximately 95% of the total tickets. For the rest of the groups which constitute less than 5% of tickets, the predictions are pretty low. But, it is not appropriate to measure the performance of the classifier for these groups at this point of time as they do not have sufficient number of tickets to evaluate the model. The Table 4.9 represent the confusion matrix of the model in production for 614 tickets. The Figure 4.4 shows the performance of the classifier for which the scores are better.

	level 1	level 21	level 22	level 23	level 24	Others
level 1	**240**	0	7	13	6	9
level 21	0	**0**	0	6	0	0
level 22	5	0	**6**	0	1	0
level 23	9	0	0	**144**	2	0
level 24	5	0	0	4	**143**	1
Others	4	0	0	6	3	**0**

Table 4.9: Confusion matrix for the model in production

Chapter 5

Recommendations

This chapter discusses about the model performance deployed in production and the possibilities to address the issues found in production. It also suggests some of the alternatives to overcome the problems faced in production. It also summarizes the advantages of using robotics in present scenario. Last but not the least, a brief discussion on the future enhancements or alternative approaches to reduce the ticket processing time are provided.

5.1 Discussion

Based on the results obtained for different models, random forest model that performed best during off-line cross-validation is deployed in production. It is trained on data excluding the level 25 group. In production, for all incoming tickets the classifier predicts the most suitable group, out of the six existing groups on which it was trained. The performance of the classifier is evaluated on the tickets that actually belong to the six groups. Tables 4.8, 4.9 that correspond to the deployed model explain the same.

However, in production, there are almost 10% of total tickets which belong to level 25. On observing the predictions of classifier for level 25 tickets, it was quite surprising that almost 95% of level 25 tickets are classified as level 1. That means, for every 100 tickets that are classified as level 1, 70% belong to level 1 and the rest belong to level 25. So, specialists dealing with level 1 tickets start working on the tickets that are classified as level 1 by the classifier. They open the tickets and try to process them. True level 1 tickets are resolved in no time and the remaining can then be assigned to level 25. For the majority of tickets, the classifier has reduced the time taken to route the tickets to correct teams and also forward them to next levels when needed.

The average time spent on each ticket previously is in minutes and has been reduced to few seconds by the classifier.

5.2 Machine learning combined with robotics

In the existing system, support tickets are manually processed by specialists. There are about 35 dedicated specialists for classifying tickets that are in English. They login to system and continuously monitor the incoming tickets and assign them to the most suitable group. It takes few days for them to manually process several hundreds of tickets. There is another team of specialists, whose aim is to resolve the classified tickets. So, the time taken for an incoming ticket classification should be reduced in order to speed up the entire process of resolving tickets. This thesis combines machine learning with robotics to speed up the process of ticket classification.

As discussed, Basware uses RPA to automate business tasks. RPA has robots to handle repetitive rule based tasks without human intervention. Robots reduce human labor and eventually saves lot of time and money. But, they cannot make intelligent decisions. In present case, it is not possible for a robot to classify support tickets based on certain rules. There are several thousands of rules that should be taken into consideration while classifying tickets and it is not feasible to automate those rules.

On the other hand, a robot can be made smarter by combining it with machine learning. Software robot of RPA is like arms and legs whereas the machine learning part is the brain. A machine learning model that is able to predict the most probable group for an incoming ticket is built and deployed in cloud. Robot communicates with the model by generating a HTTP request. Robot fetches the predictions of the classifier and writes results. This entire process takes few seconds and several hundreds of tickets can be classified in just few minutes.

The results obtained in this project clearly shows that the random forest model deployed, performs well for certain groups that contribute large portion of tickets. In the present context, the process of ticket classification using machine learning combined with robotics reduces the time spent on ticket classification to a good extent. Hence, the combination of machine learning and robotics is recommended for speeding up the process of ticket classification.

5.3 Possible extensions

This thesis can be extended in a few ways. The current classifier predicts the most probable group for an incoming ticket. Based on the results, it is possible that the predictions might go wrong for few cases. This might be because the probabilities of certain groups might be a bit closer and the classifier just returns the group that has more probability. An alternative approach would be to consider the most probable group only if it exceeds a certain threshold probability and for the rest, the classifier returns the two most probable groups. This approach might increase the true positive rate.

There are a few more possible additions to existing work. They include automating the ticket creation and ticket resolution processes. In case of ticket creation process, for each incoming email, a specialist should manually create a ticket by assigning each of the existing fields in the system with respective values. It takes few minutes for creation of each ticket. Upon talking to specialists, it was clear that 50% of the incoming emails are related to trivial cases and consumes almost 50% of the time in the process of ticket creation. For such trivial redundant cases, it is possible to automate the work in order to save human time. Machine learning combined with robotics can address this problem. In the first step, machine learning can be used in identifying the type of ticket basing on the text. For trivial tickets like 'request to password reset', the robot sets required fields in the system with respective values basing on certain rules. In long term, this solution of using machine learning with robotics for ticket creation saves human time and cost.

Another possibility would be to automate the ticket resolution process. Domain specialists for each group resolve tickets basing on knowledge base articles. There are several hundreds of rules in ticket resolution process that cannot be automated. But, for some of the simple cases, respective knowledge base articles can be used as historical data. A machine learning model can be developed using such historical data, which can suggest the related knowledge base articles for each of the incoming emails. this model should be more advantageous for IT support teams as it eliminate the time taken for the ticket creation, classification and resolution. At-least this approach should be applicable for 50% of the most trivial cases.

Chapter 6

Conclusions

This thesis builds a classification engine that provides a recommendation to specialists for routing support tickets in Basware ITSM system. It is crucial to reduce the amount of time spent on ticket routing process. Manual ticket routing includes routine work and takes at least few days to route several hundreds of support tickets. Various experiments conducted on different sets of historical data using logistic regression, random forests and extremely randomized trees. From the results presented in Chapter 4, it is evident that the random forest model performs well with an average of 90% f1-score over all groups during off-line cross-validation. This model is deployed in production using AWS to evaluate the performance of the model on real tickets. This thesis uses software robots of RPA to communicate between the IT support system and the deployed model. Evaluation of the model is performed for two weeks of real tickets. The model has an average f1-score of 86% for all groups. The f1-scores for the level 1, level 23, level 24 groups that contribute 90% of total tickets are 89%, 88%, 93% respectively. This classification engine is quick in providing recommendation to the specialists and reduced the time spent on routing each ticket from minutes to seconds.

Few possible ways to extend this work include automating ticket creating and resolution processes for the most frequent and trivial cases. This can reduce the total time spent on 50% of tickets and speed up the entire process of ticket resolution.

Bibliography

[1] Alpaydin, E. *Introduction to Machine Learning.*, vol. Third edition of *Adaptive Computation and Machine Learning*. 2014.

[2] Bird, S., Klein, E., and Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[3] Bishop, C. M. Pattern recognition and machine learning (information science and statistics), 2006.

[4] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[5] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122.

[6] Garreta, R., and Moncecchi, G. Learning scikit-learn : Machine learning in python, 2013.

[7] Hussain, Z., Mueller, J., and Massaron, L. *Python for Data Science For Dummies*. –For dummies. Wiley, 2015.

[8] Jurafsky, D., and Martin, J. H. *Speech and language processing*. Harlow : Pearson Education cop. 2014.

[9] Maglogiannis, I. *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*. Frontiers in artificial intelligence and applications. IOS Press, 2007.

[10] Ozdemir, S., and Susarla, D. *Feature Engineering Made Easy: Identify unique features from your dataset in order to build powerful machine learning systems.* Packt Publishing, 2018.

[11] Rokach, L. *Pattern Classification Using Ensemble Methods.* World Scientific Publishing Co Pte Ltd, 2014.

[12] Sarkar, D., Bali, R., and Sharma, T. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems.* Apress, 2017.

[13] Subramanian, G. *Python Data Science Cookbook.* Packt Publishing, 2015.

[14] Tripathi, A. M. *Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool ? UiPath.* Packt Publishing Ltd, 2018, 2018.