**APPENDIX 1**

**AUTOMATIC TICKET ASSIGNMENT**

CAPSTONE PROJECT REPORT

BY

1. Karthik
2. Bhawna
3. Calwin
4. Vishal
5. Rahul
6. Saimadhu

**GREAT LEARNING**

**July 2020**

**Real problem and business value**

One of the key activities of any IT function is to “Keep the lights on” to ensure there is no

impact to the Business operations. IT leverages Incident Management process to achieve the

above Objective. An incident is something that is unplanned interruption to an IT service or

reduction in the quality of an IT service that affects the Users and the Business. The main goal of Incident Management process is to provide a quick fix / workarounds or solutions that

resolves the interruption and restores the service to its full capacity to ensure no business

impact.

In most of the organizations, incidents are created by various Business and IT Users, End Users/ Vendors if they have access to ticketing systems, and from the integrated monitoring

systems and tools. Assigning the incidents to the appropriate person or unit in the support team has critical importance to provide improved user satisfaction while ensuring better allocation of support resources. The assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations.

Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.

**Business Domain Value**

In the support process, incoming incidents are analyzed and assessed by the organization's support teams to fulfill the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings.

Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within the IT Service Management Tool and are assigned to Service Desk teams (L1/ L2 teams). This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. If L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by either Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve

the incidents.

Around ~56% of incidents are resolved by Functional / L3 teams. Incase if vendor support

is needed, they will reach out for their support towards incident closure. L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25 -30% of incidents needs to be reviewed for SOPs before ticket

assignment). 15 min is being spent on SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

**Summary of problem statement**

The objective of the problem classifies the target group, given the short and long description. Which helps the business in solving the tickets issue in time without delay which leads to business impact.

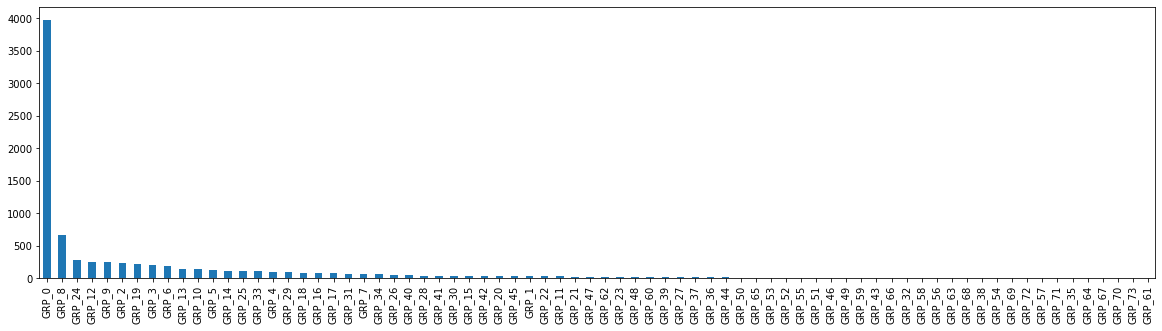
**Summary of EDA & Pre-processing**

For assigning tickets automatically the data set contains **8500** rows and **4** columns.

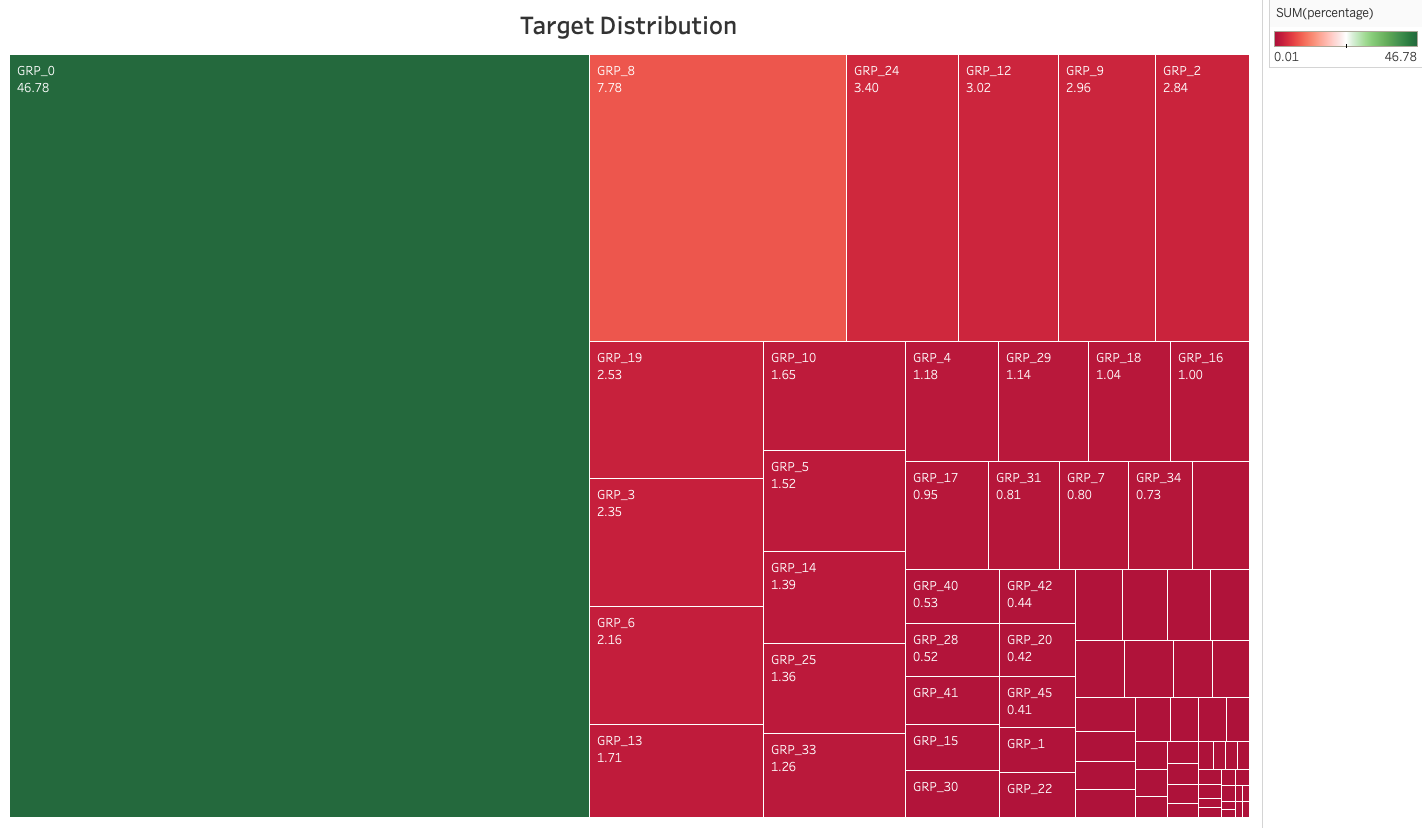
Columns: Description, Short Description, Caller & Group,

Basically it’s a classification problem to correctly determine the group for assigning tickets automatically. There are groups from 0 to 74 (target), below is the target class distribution.

Out of the 8498 data 3975 pertains to the group 0 and 661 pertains to group 8 which is almost 50% of the dataset. Rest of the dataset is below 1% and can be avoided. Many assignment groups have only one test data which can be deleted. Some of the cases it is seen that the description is not in English language and as the data is very few so that can also be deleted. Below is the plot for Assignment group with their count.



Here our challenge is to build a model which has a higher accuracy in spite of the data mainly consisting of only a few groups from the 74 groups.



**Pre-processing**

**2.1 Lower case**

The first pre-processing step which we will do is transform our tweets into lower case. This avoids having multiple copies of the same words. For example, while calculating the word count, ‘Analytics’ and ‘analytics’ will be taken as different words.

**2.2 Removing Punctuation**

The next step is to remove punctuation, as it doesn’t add any extra information while treating text data. Therefore removing all instances of it will help us reduce the size of the training data.

As you can see in the above output, all the punctuation, including ‘#’ and ‘@’, has been removed from the training data.

**2.3 Removal of Stop Words**

Stop words (or commonly occurring words) should be removed from the text data. For this purpose, we can either create a list of stop words ourselves or we can use predefined libraries.

**2.4 Common word removal**

Previously, we just removed commonly occurring words in a general sense. We can also remove commonly occurring words from our text data First, let’s check the 10 most frequently occurring words in our text data then take a call to remove or retain.

**2.6 Spelling correction**

We’ve all seen tweets with a plethora of spelling mistakes. Our timelines are often filled with hasty sent descriptions that are barely legible at times.

In that regard, spelling correction is a useful pre-processing step because this also will help us in reducing multiple copies of words. For example, “Analytics” and “analytics” will be treated as different words even if they are used in the same sense.

To achieve this we will use the *textblob* library. If you are not familiar with it, you can check my previous [article](https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/) on ‘NLP for beginners using *textblob’*.

**2.8 Stemming**

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach. For this purpose, we will use *PorterStemmer* from the NLTK library.

**2.9 Lemmatization**

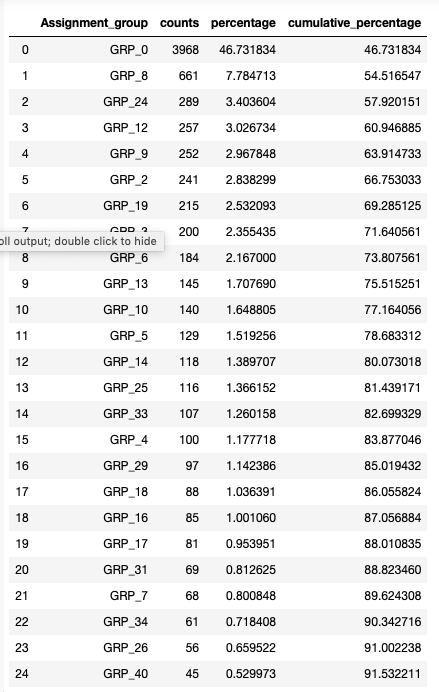
Lemmatization is a more effective option than stemming because it converts the word into its root word, rather than just stripping the suffices. It makes use of the vocabulary and does a morphological analysis to obtain the root word. Therefore, we usually prefer using lemmatization over stemming.

**Features reduction:**

As the caller is not having any significance in model building we have removed it straight away, Then we applied Fuzzy Wuzzy and compared the similarities between short description and description and a new column has been created called **“Derived Description”,** if the similarity is more than 90% derived description will contain only short description**.**

**Target class reduction:**

As we seen in the above image the targets are not equally balanced, in face many target classes are not even having in 1% data.So we merged the assignment group where there is less than 1% data to group no:67 and the total assignment groups reduced to 50 from 74.



So we converted the less popular target classes to single calls “less\_pop” this lead us to reduce the target classes to **50**

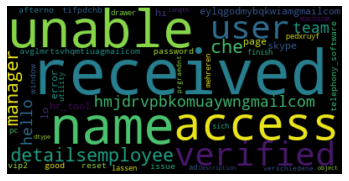
**Word cloud**

We created the word clouds before text cleaning and after cleaning.

Before cleaning:

****

After text cleaning:



**Model Building:**

Before creating the model there are certain steps like the target variable y, we used label encoder and converted to integers and for the X we have tokenized, performed count vectorizer for converting it to vector.

**Models Evaluation:**

**Random Forest Ensemble Model Technique:**

precision recall f1-score support

accuracy 0.66 1700

macro avg 0.29 0.20 0.21 1700

weighted avg 0.60 0.66 0.60 1700

**Total Accuracy:0.65**

**Decision Tree Ensemble Model Technique:**

precision recall f1-score support

accuracy 0.56 1700

macro avg 0.17 0.17 0.16 1700

weighted avg 0.53 0.56 0.54 1700

**Total Accuracy:0.56**

**Ada Boost Ensemble Model Technique:**

precision recall f1-score support

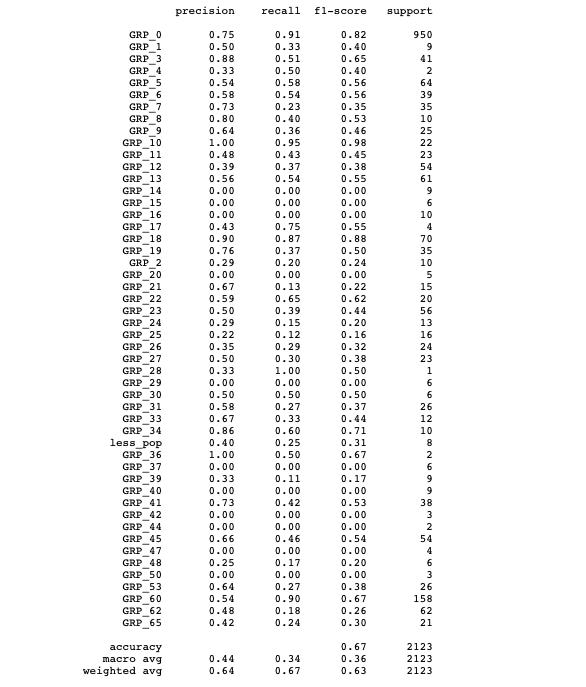
accuracy 0.49 1700

macro avg 0.01 0.02 0.01 1700

weighted avg 0.25 0.49 0.32 1700

**Total Accuracy:0.49**

**Logistic Regression Model Technique:**



**Total Accuracy:0.67**

**Model Accuracy details**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Model | Train accuracy % | Test accuracy% |
|  | Logistic Regression | 67 | 65 |
|  | Decision Tree | 92 | 57 |
|  | Ada Boost | 46 | 48 |
|  | Random Forest | 92 | 66 |
|  | LSTM | 83 | 60 |

**Upsampling:**

We did the upsampling to improve the accuracy. After upsampling below are the updated accuracies.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Model | Train accuracy % | Test accuracy% | F1 Score |  |
|  | Logistic Regression | 78 | 70 | 71 |  |
|  | Decision Tree | 94 | 92 | 92 |  |
|  | Ada Boost | 33 | 50 | 50 |  |
|  | Random Forest | 94 | 92 | 93 |  |

**Hyper-Parameter Tuning:**

* **Optimizers-** Adam, rmsprop, sgd
* **Activation Functions-** Softmax, relu, sigmoid
* **Learning rate-** 0.01 and 0.001
* **Vocab size-** 2000 to 20000
* **Epochs-** 5 to 20
* **Loss function-** sparse\_categorical\_crossentropy, binary\_crossentropy, categorical\_crossentropy
* **Metrics-** accuracy

**Final Parameters Selection:**

* **Optimizers-** Adam
* **Activation Functions-** Softmax, relu
* **Learning rate-** 0.01 and 0.001
* **Vocab size-** 2000 to 20000
* **Epochs-** 5 to 20
* **Loss function-** sparse\_categorical\_crossentropy
* **Metrics-** accuracy

**Final Model Selection:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **accuracy%** | **val\_accuracy%** | **loss** | **val\_loss** | **epochs** |
| 1. **LSTM** | 82 | 62 | 0.69 | 1.56 | 10 |
| 1. **LSTM(glove embedding)** | 96 | 97 | 0.14 | 0.13 | 20 |
| 1. **GRU** | 93 | 60 | 0.24 | 1.72 | 20 |

The final model we selected is “LSTM with glove embedding” with the following:

**epochs** - 20

**Train Accuracy** – 96%

**Val\_accuracy** - 97%

**Optimizer** - Adam

**Activation function** – softmax

**Loss function** - sparse\_categorical\_crossentropy

**Metrics** – accuracy

**Future Scope and Limitations:**

We have proposed the ticket assignment engine that uses an ensemble of machine learning techniques. Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.

However, there is still some scope for improvement of the system. The system can also be enhanced to handle concept drift better. However in datasets with high concept drift this method may not give good results over the long run.

We can build chatbots

**Code links**

Link: <https://github.com/saimadhu-polamuri/automatic_ticket_assignment>