**Capstone Project - AIML Online Batch**
**2019-2020**

# Automatic Ticket Assignment

**Prepared By:**
**Divya Kamat, Madhushruti Srivastava,**
**Saravana Alagar, Shashank Chandra, Subhau Nath**

**Supervised by:**
**Kapil Chandel**

# 1. Project Overview:

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 the Incident Management process to achieve the above Objective. An incident is something that is an 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 the 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.

In the support process, incoming incidents are analyzed and assessed by 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.

In this capstone project, using a powerful AI / ML technique we will build a classifier that can by analysing text in the incidents and classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

# 2. AS IS Process:

Currently, the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within 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. Incase 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 for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

During the process of incident assignments by L1 / L2 teams to functional groups, there were multiple instances of incidents getting assigned to wrong functional groups. Around ~25% of

Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service.

# 3. Problem Statement:

In most of the IT organizations, the assignment of incidents to appropriate IT groups is still a manual process. 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.

# 4. Solution

This capstone project intends to reduce the manual intervention of IT operations or Service desk teams by automating the ticket assignment process .

The goal here is to create a text classification based ML model that can automatically classify any new tickets by analysing ticket description to one of the relevant Assignment groups, which could be later integrated to any ITSM tool like Service Now

Based on the ticket description our model will output the probability of assigning it to one of the 74 Groups.

# 5. Assumptions

a. In the AS-IS process it's mentioned that around ~54% of the incidents are resolved by L1 / L2 teams and the rest will be resolved as L2. So the assumption is that GRP_0 and GRP_8 which contribute 54% of the tickets are related to L1/L2 teams and the rest of the tickets belongs to L3 teams

b. Since the dataset is very imbalanced, We will be considering a subset of groups for predictions. In 74 groups, 46% of tickets belong to group 1 and 16 groups just have more than 100 tickets, rest of the Assignment groups have very less ticket counts which might not add much value to the model prediction. If we conducted random sampling towards all the subcategories, then we would face a problem that we might miss all the tickets in some categories. Hence, we will be only considering the groups that have more than 100 tickets. Rest of the tickets would be ignored.

Groups with tickets > 100

| | |
|---|---|
| GRP_0 | 3941 |
| GRP_8 | 656 |
| GRP_24 | 289 |
| GRP_12 | 257 |
| GRP_9 | 248 |
| GRP_2 | 239 |
| GRP_19 | 213 |
| GRP_3 | 198 |
| GRP_6 | 182 |
| GRP_13 | 144 |
| GRP_10 | 139 |
| GRP_5 | 129 |
| GRP_14 | 117 |
| GRP_25 | 115 |
| GRP_33 | 104 |

# 6. Approach

The solution is been implemented using below approach:

a. Approach 1 - Using a traditional machine learning algorithm we are classifying the tickets into one of the groups having more than 100 tickets.

b. Approach 2 - It's mentioned that around ~54% of the incidents are resolved by L1 / L2 teams and the rest will be resolved as L3. So the assumption is that GRP_0 and GRP_8 which contribute 54% of the tickets are related to L1/L2 teams and the rest of the tickets belong to L3 teams. In this approach, firstly the ticket would be classified into one of L1/L2 or L3 classes and then it would be further classified into one of the given assignment groups belonging to L1/L2 or L3 teams respectively. In this approach, we have considered assignment groups having more than 50 tickets.

# 7. Data

## Understanding the structure of data

The data files used for this capstone project are available at below google drive location: https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ

- The data set contains 4 columns and all are string columns

| Column | Description | Data type |
|---|---|---|
| **Short description** | Short description on the problem for which the incident is being raised | `8492 non-null object` |
| **Description** | Detailed description of the problem | `8499 non-null object` |
| **Caller** | Email id of the User who raised the problem | `8500 non-null object` |
| **Assignment Group** | IT Support Group to which the Incident log is been assigned to | `8500 non-null object` |

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are *Short description*, *Description* and *Caller*.
- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is *Assignment group*.

- ❖ There are totally 8500 rows
- ❖ There seems to be missing values in Short description and Description columns, which needs to be looked into and handled. There are **8 null/missing values** present in the Short description and **1 null/missing values** present in the description column
- ❖ Caller columns mainly contain the details of the user who raised the incident and is of not much use in our analysis and can be dropped.
- ❖ "Short Description" and "Description" can be concatenated as a single column, so that we won't miss any necessary info about the ticket.
- ❖ Assignment group is our predictor / target column with multiple classes. This is a **Multiclass Classification problem**

# 8. Exploratory Data Analysis

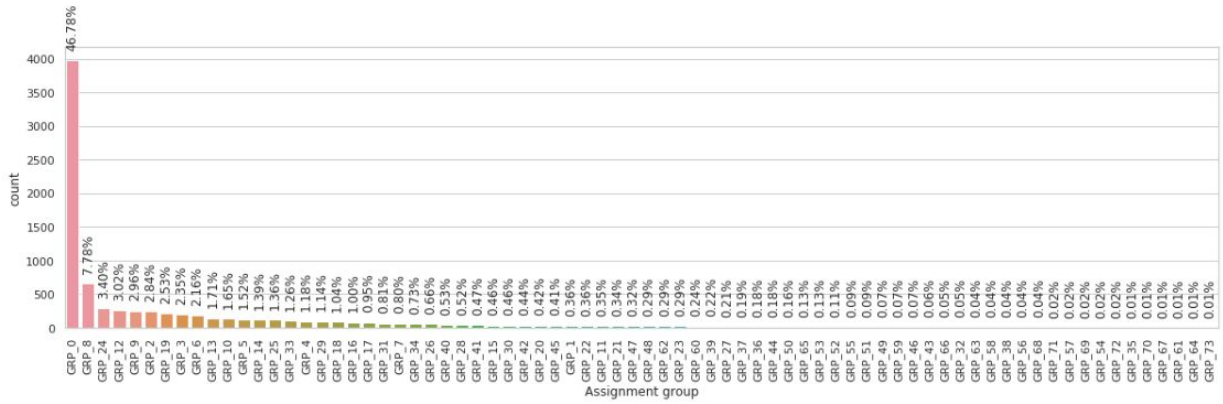
Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

- maximize insight into a data set;
- uncover underlying structure;
- extract important variables;
- detect outliers and anomalies;
- test underlying assumptions;
- develop parsimonious models; and
- determine optimal factor settings

Visually representing the content of a text document is one of the most important tasks in the field of text mining. It helps not only to explore the content of documents from different aspects and at different levels of details, but also helps in summarizing a single document, show the words and topics, detect events, and create storylines.

## Distribution of the Target Column

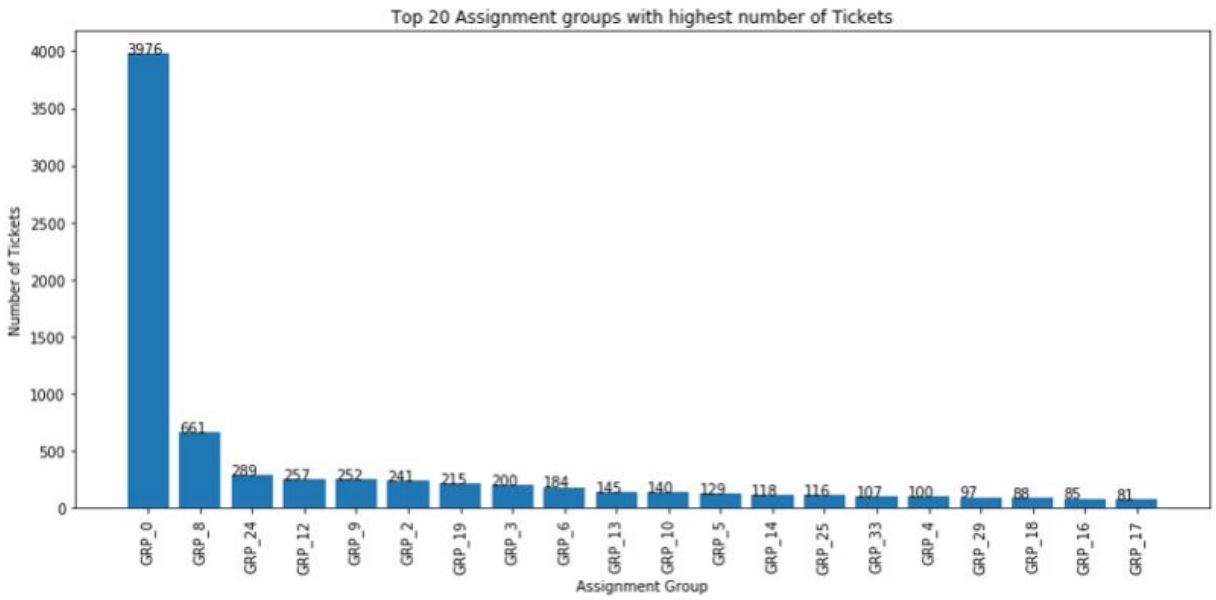
❖ Assignment group contains around 74 different classes



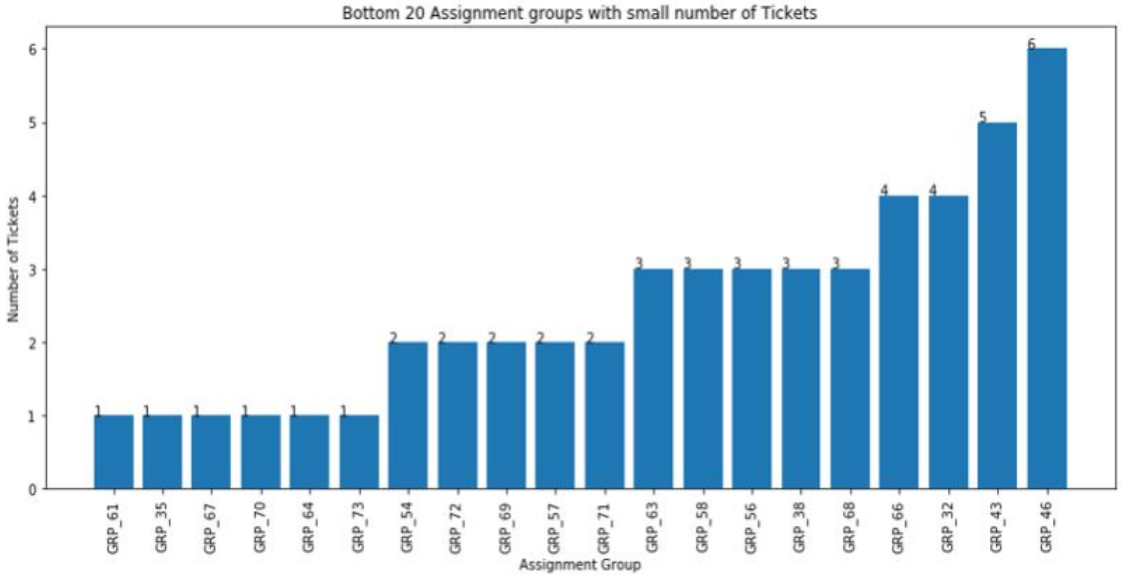

❖ The data is too much biased towards a single group and seems to be highly imbalanced, with majority of incidents are from Group 0 followed by Group 8 , 24 , 12 , 9 , 2 and so on

❖ There are few classes which just have less than 10 incidents pers class and even classes with just 1 or 2 incidents (samples), need to see if we can drop those rows due to the lack of samples representing those classes. They might not be of much help as a predictor

❖ Top 20 Assignment groups having the highest number of tickets for training the data.

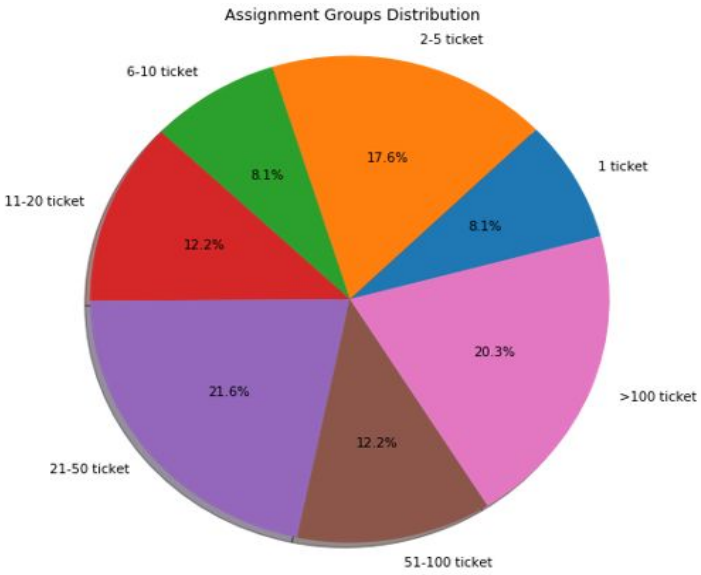



❖ Following are the Tickets with less number of tickets per Assignment groups.



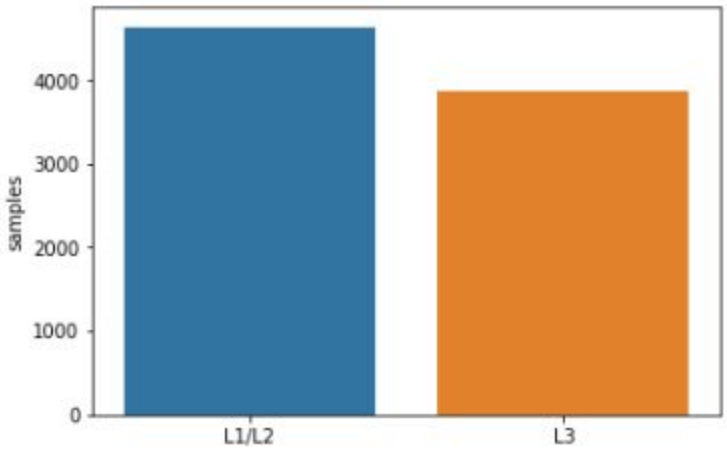

❖ Distribution of tickets available in the dataset based on Assignment Groups.

| | Description | Ticket Count |
|---|---|---|
| 0 | 1 ticket | 6 |
| 1 | 2-5 ticket | 13 |
| 2 | 6-10 ticket | 6 |
| 3 | 11-20 ticket | 9 |
| 4 | 21-50 ticket | 16 |
| 5 | 51-100 ticket | 9 |
| 6 | >100 ticket | 15 |

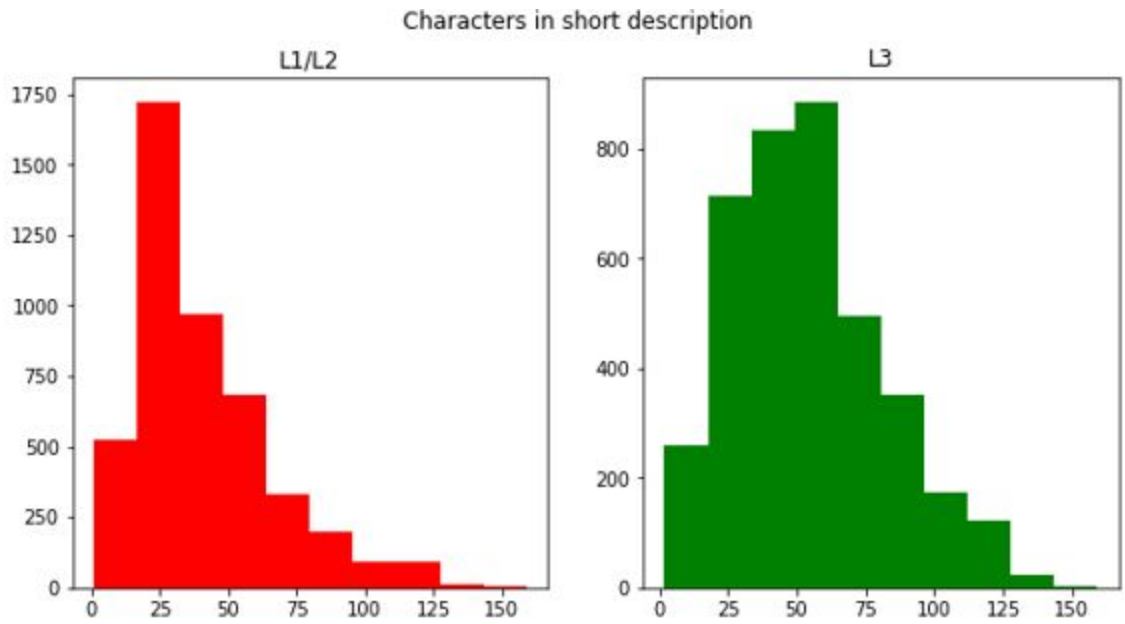



- ➢ We see that there are 6 Assignment Group's for which just have 1 ticket in the dataset
- ➢ There are 15 Assignment group's which have more than 100 tickets. Only 20% of the Assignment groups have greater than 100 tickets.

**Number of samples by functional groups L1/L2 or L3.**

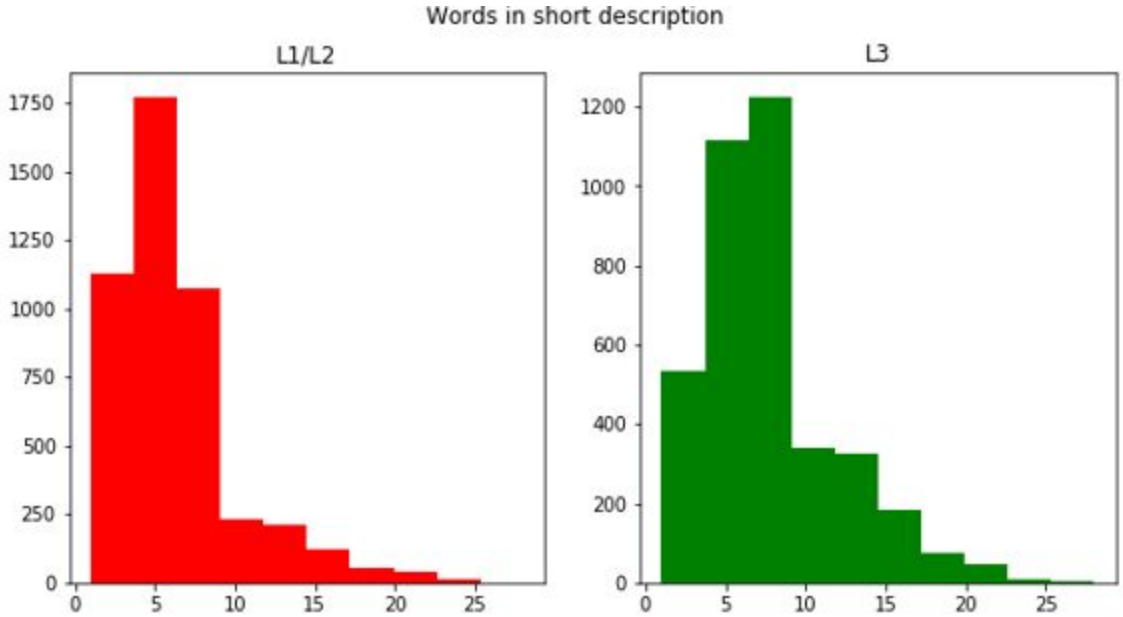



**Character count in Short description by functional groups L1/L2 or L3.**



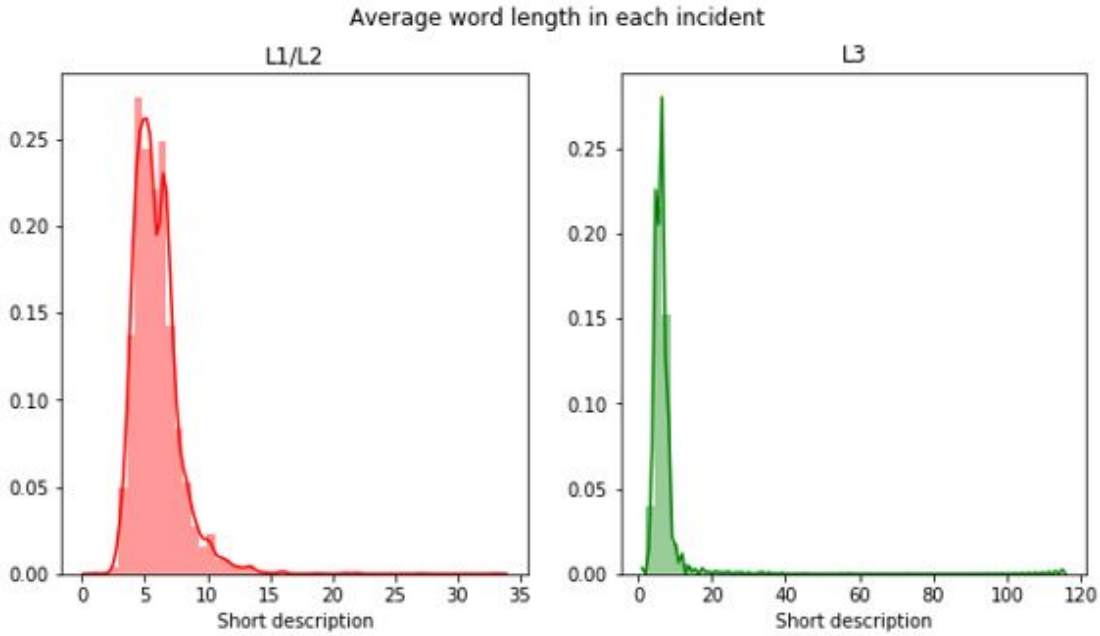

Character in L1/L2 tickets look to be slightly lower than L3 tickets

**Word count in Short description by functional groups L1/L2 or L3.**



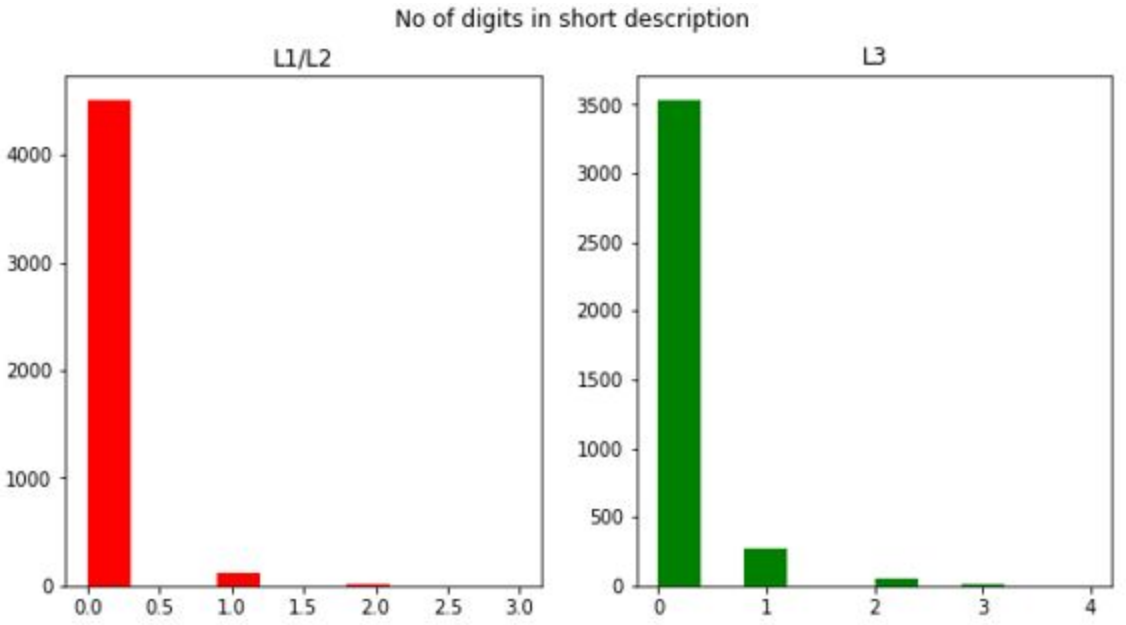

Word count in L1/L2 tickets look to be slightly lower than L3 tickets

**Average word length in each ticket by functional groups L1/L2 or L3.**

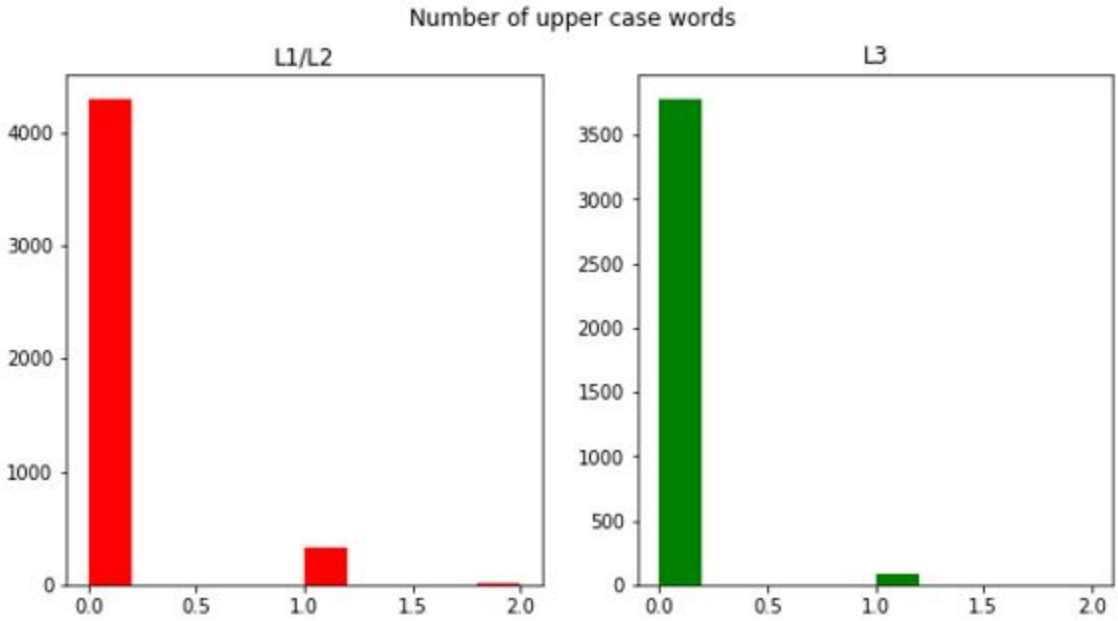



Average word length is more in L1/L2 tickets and the distribution is normal compared to L3 tickets. There seems to be length words in L3 tickets

**Digits count in short description by functional groups L1/L2 or L3.**



**Upper Case count in short description by functional groups L1/L2 or L3.**



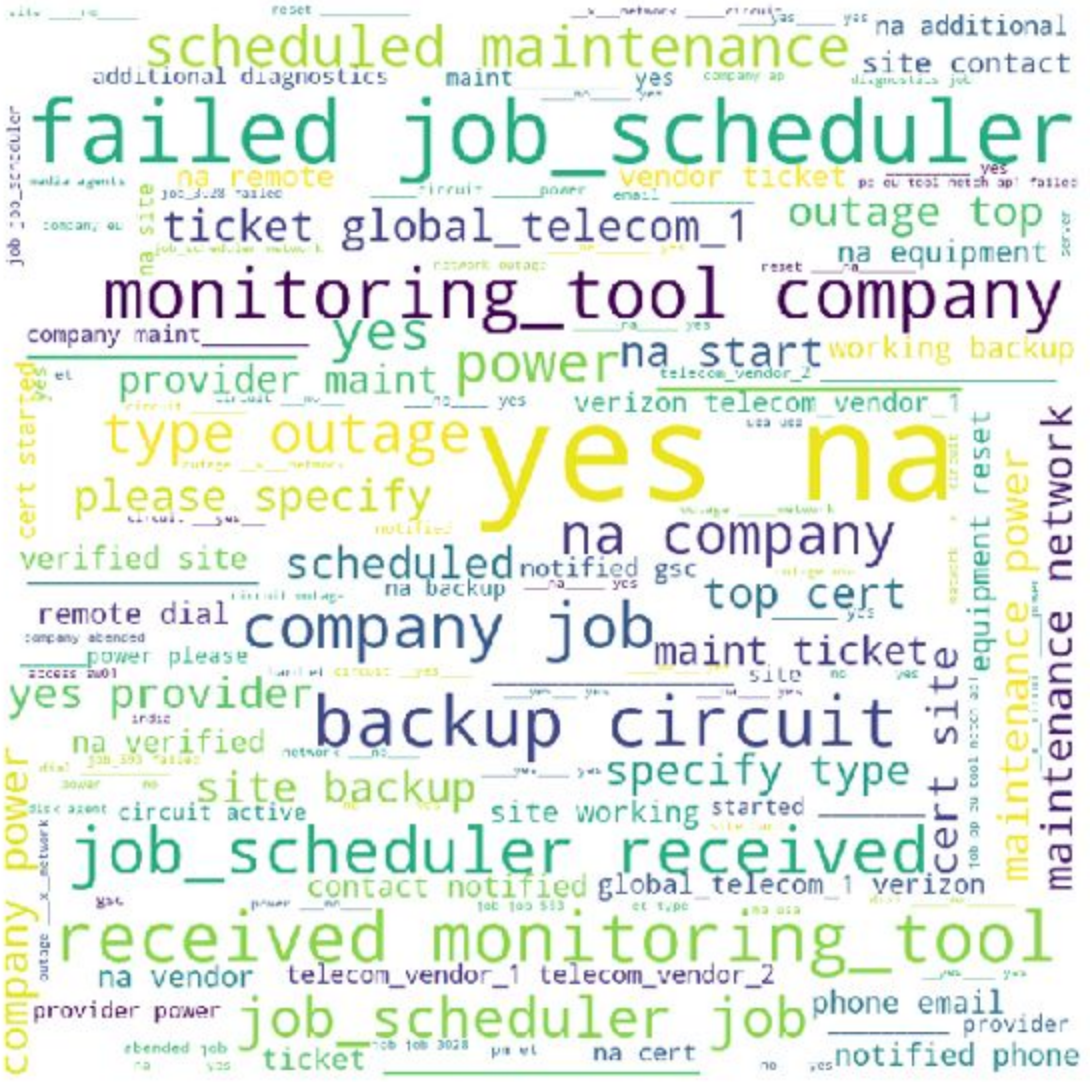
**Let's use word cloud to view the type of tickets in top 4 Assignment groups**

**Word Cloud for tickets with Assignment group 'GRP_0'**


- Analysis on GRP_0 which is the most frequent group to assign a ticket to reveals that this group deals with mostly the maintenance problems such as *password reset*, *account lock*, *login issue*, *ticket update* etc.
- Maximum of the tickets from GRP_0 can be reduced by self correcting itself by putting automation scripts/mechanisms to help resolve these common

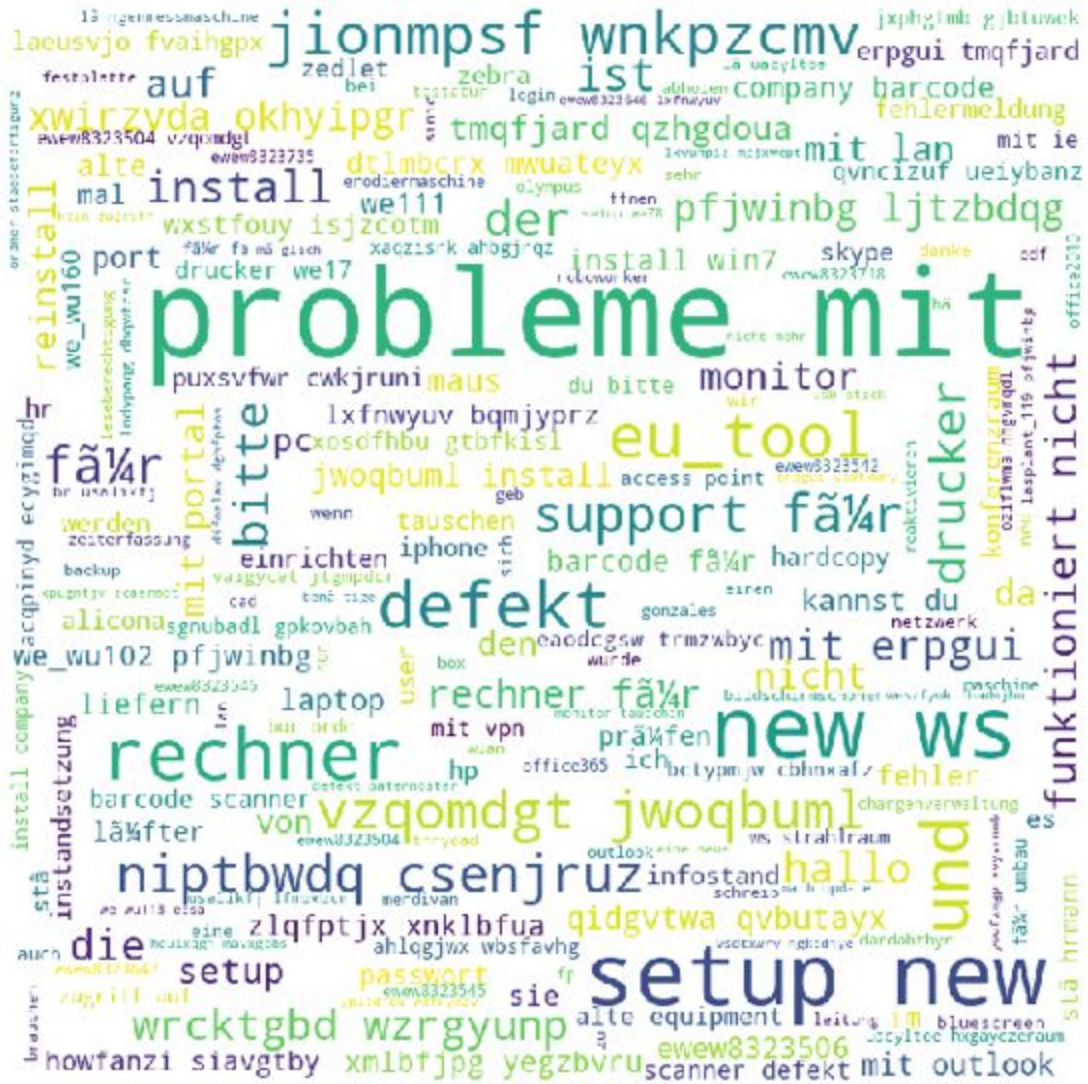
maintenance issues. This will help in lowering the inflow of service tickets thereby saving the person/hour efforts spent and increasing the business revenue.

**Word Cloud for tickets with Assignment group 'GRP_8'**



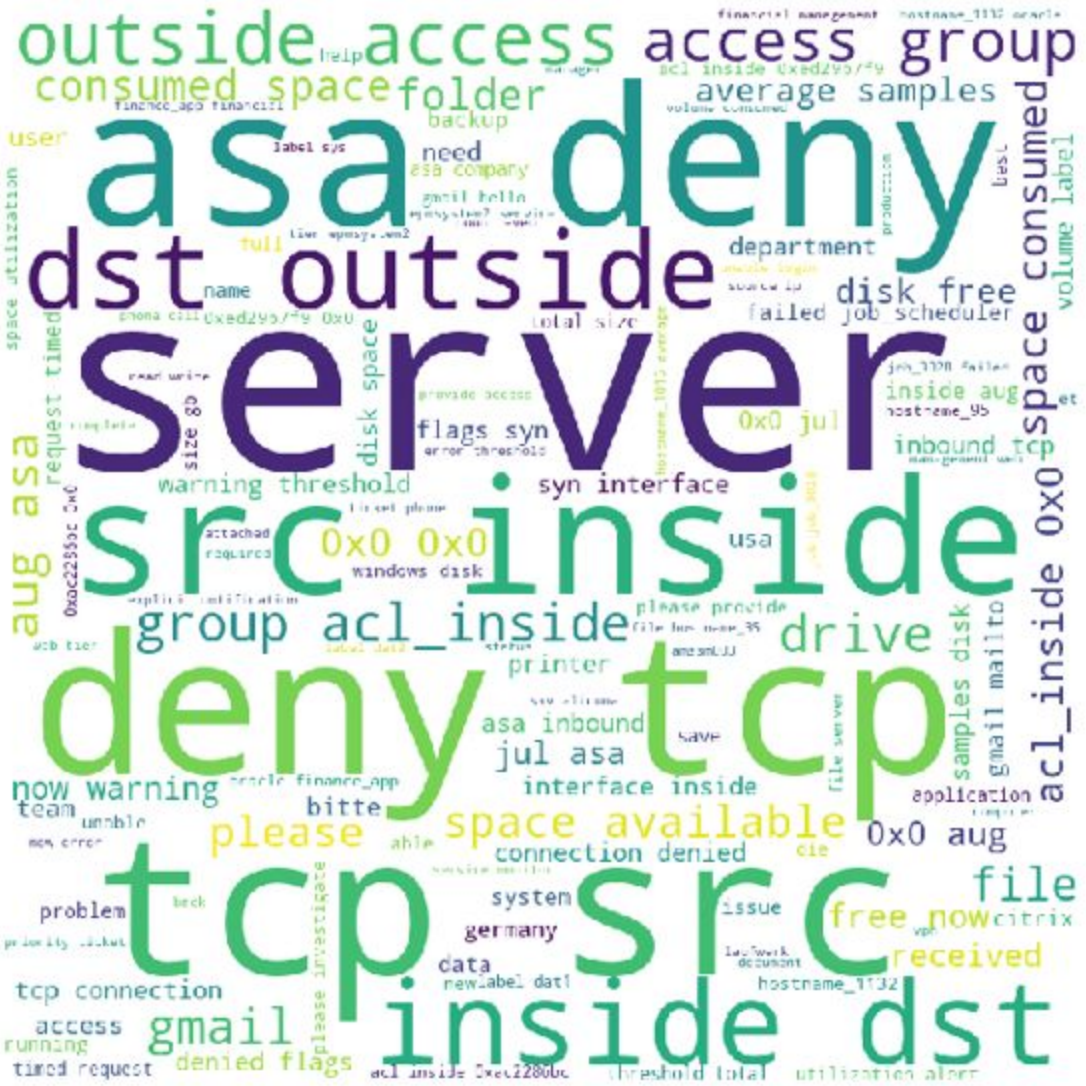

- GRP_8 seems to have tickets related to outage, job failures, monitoring tool etc

**Word Cloud for tickets with Assignment group 'GRP_24'**




- GRP_24 - Tickets are mainly in german, these tickets need to be translated to english before passing it to our model.

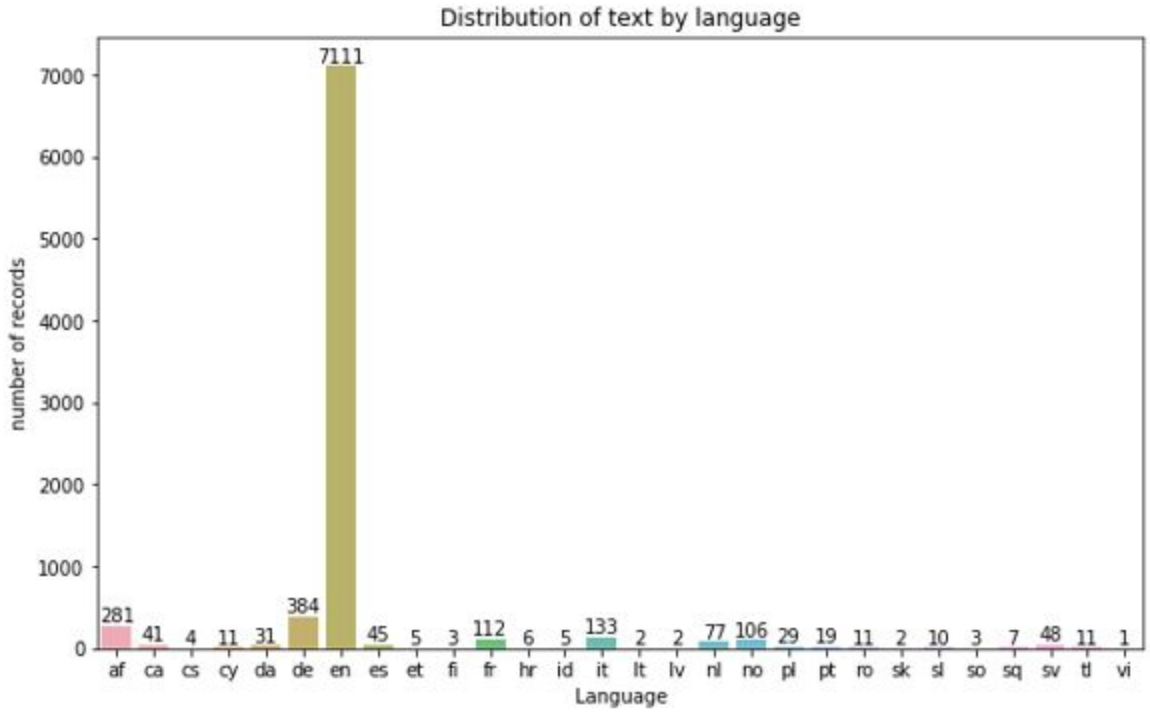
**Word Cloud for tickets with Assignment group 'GRP_12'**




- GRP_12 contains tickets related to systems like  disk space issues, t network issues like tie out, citrix issue, connectivity timeout etc.

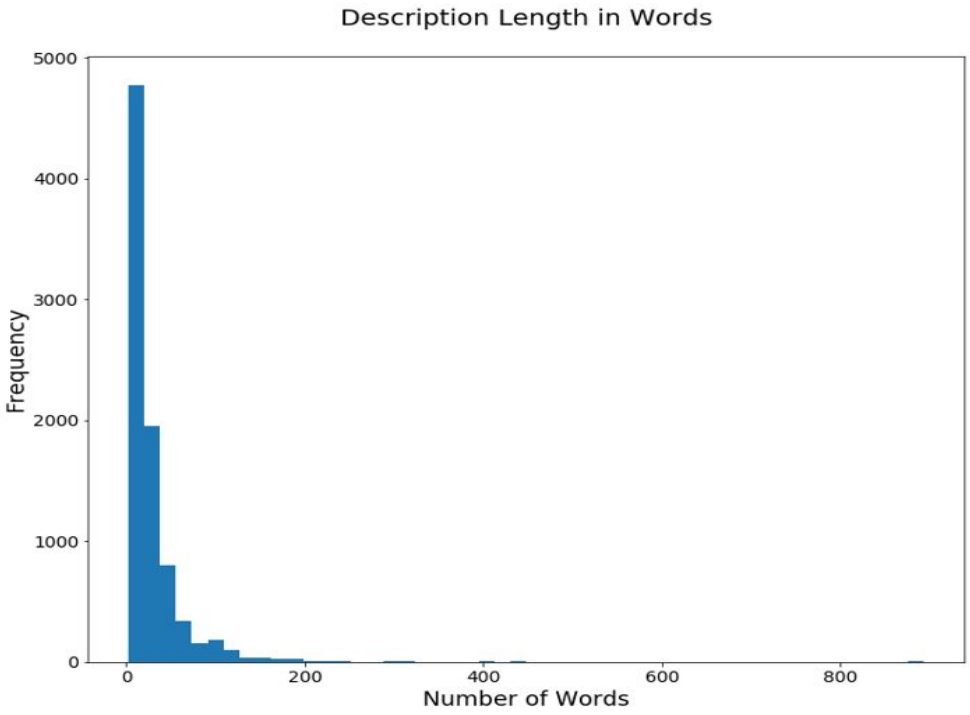
- It's indicative from the n-gram analysis and the word cloud is that the entire dataset speaks more about issues around
    - **password reset (1246 times)**
    - **fail job_scheduler (1614 times)**
    - **outlook (948 times)**
    - **login (861 times)**
    - **job fail (897 times)**

**Distribution of text by language**

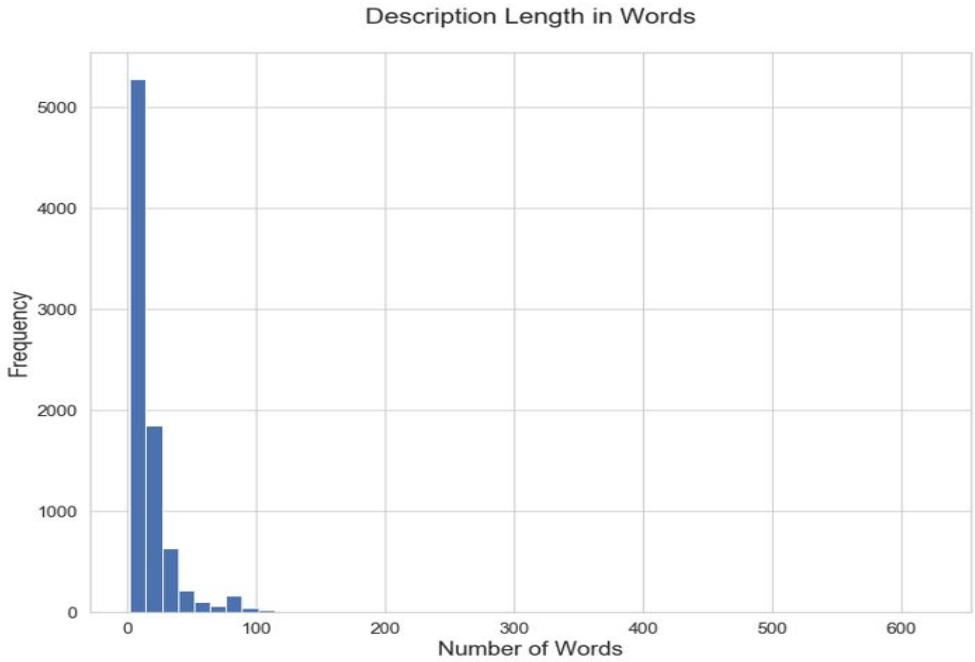



- We can see that most of the tickets are in english, followed by tickets in German language. We need to translate these into english.
- We will be using google translate package to translate, however there is a limitation on the number of requests that google translate API can accept per day. So we translated those in batches and saved the translated file to disk.

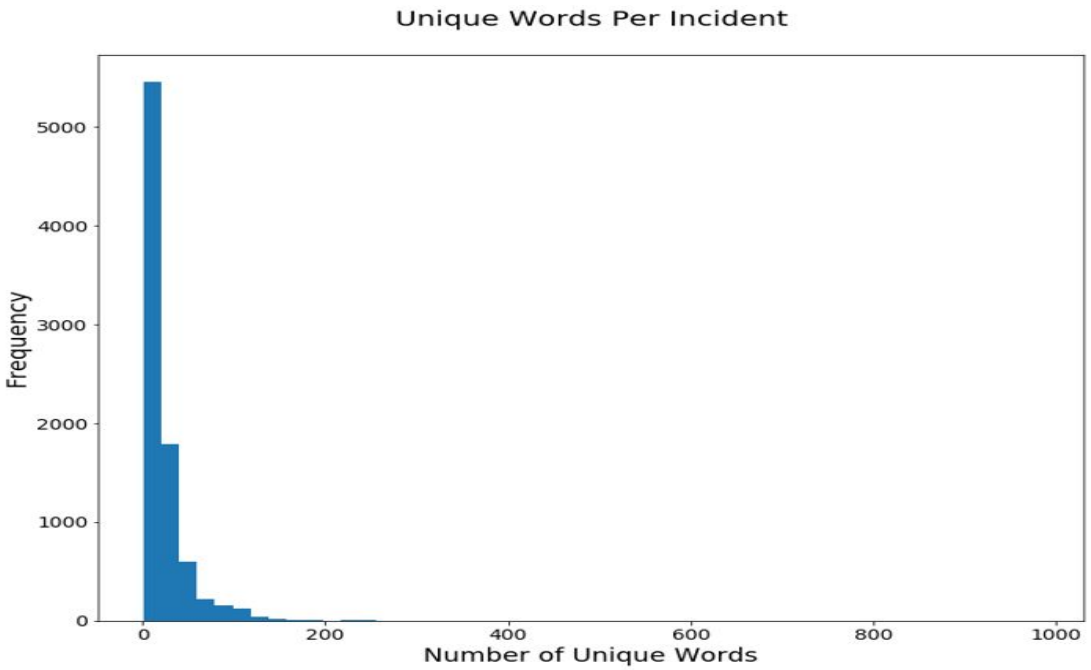
## Distribution of description word counts



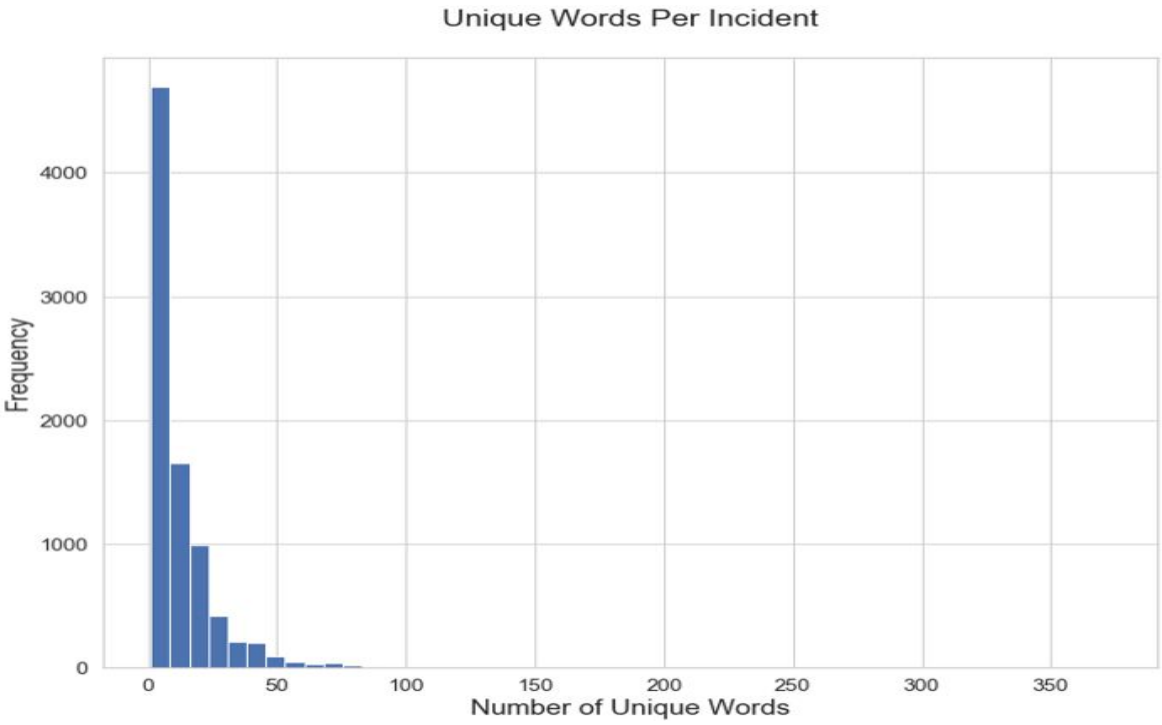

Few tickets have longer descriptions, more than 800 words, below is the distribution of words after cleaning the text - removing punctuation, stopword , digit etc



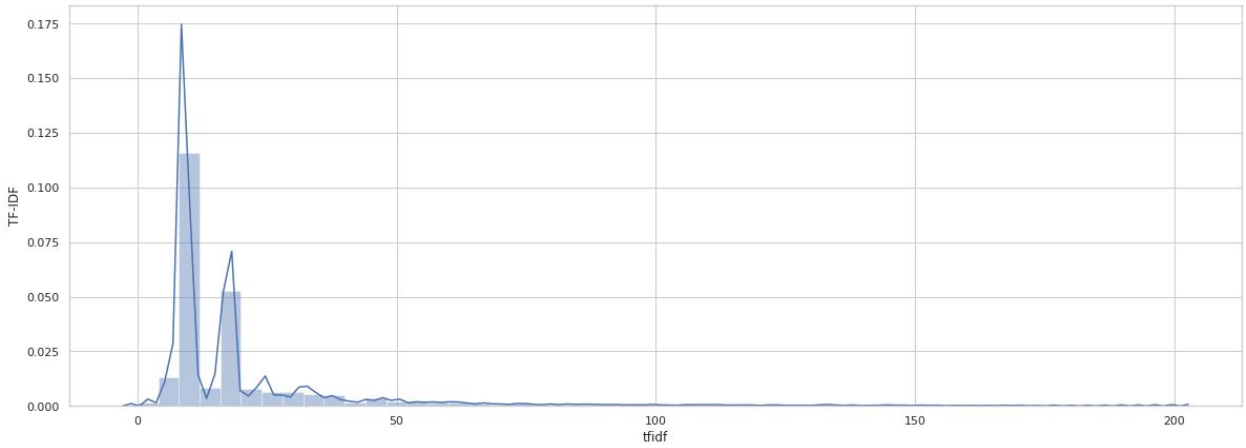
### Distribution of unique words per ticket



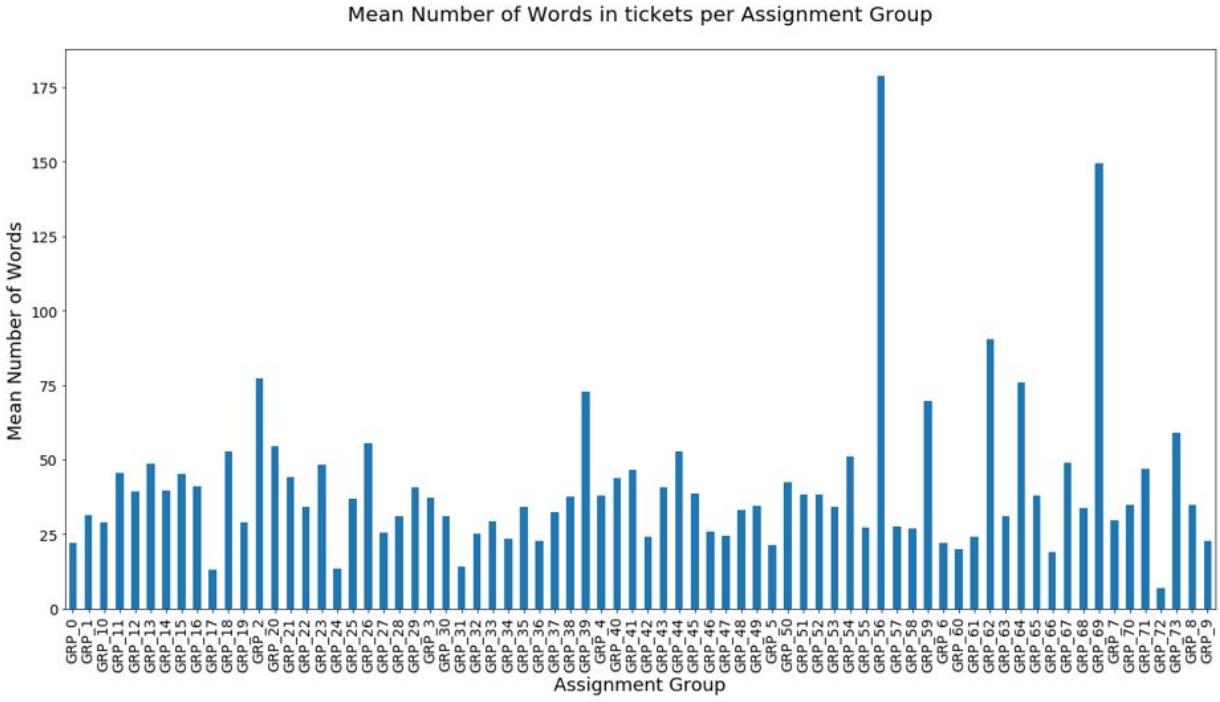

When we plot this into a chart, we can see that while the distribution of unique words is still skewed, it looks a bit similar to the distribution based on total word counts we generated earlier. Below is the distribution of unique words after cleaning the text - removing punctuation, stopword , digit etc.



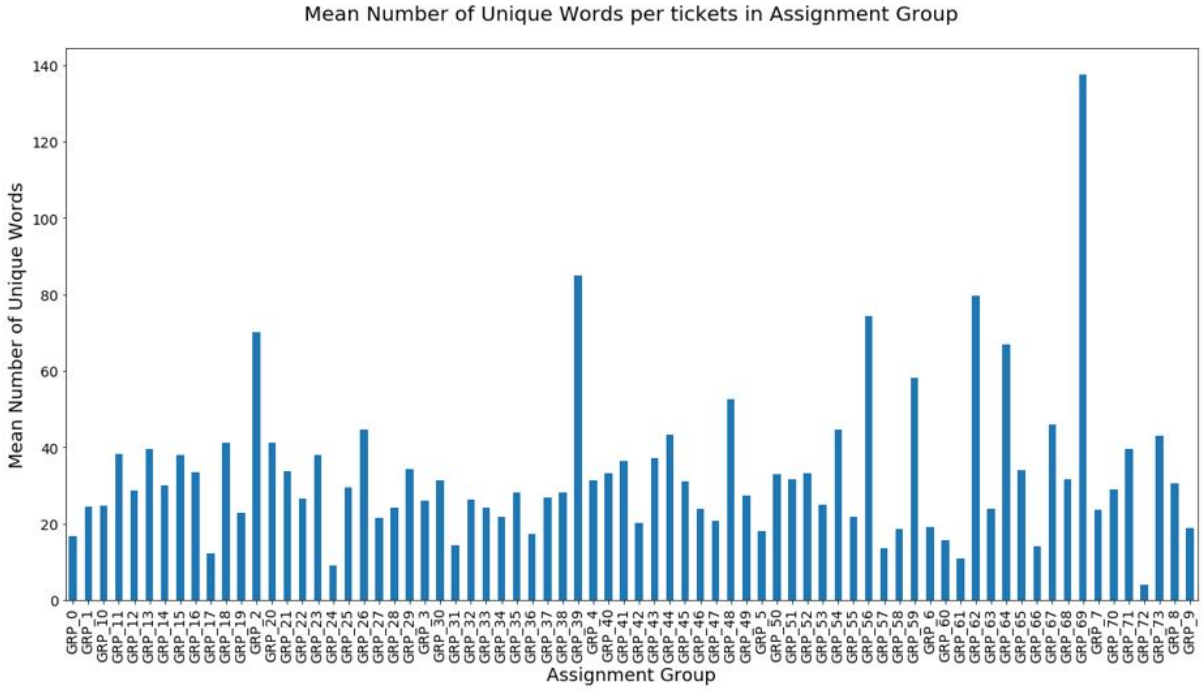
## TF-IDF Distribution of words:




## Mean Number of Words in tickets per Assignment Group



**Mean Number of Unique Words in tickets per Assignment Group**




# 9. Text Data Preprocessing

## 1. Missing values

There are missing values in the dataset, within 'Short description' and 'Description' columns, let's view the missing values. There are **8 null/missing values** present in the Short description and **1 null/missing values** present in the description column

## 1.1. Missing values in Short description column

| | Short description | Description | Assignment group |
|---|---|---|---|
| 2604 | NaN | \r\n\r\nreceived from: ohdrnswl.rezuibdt@gmail... | GRP_34 |
| 3383 | NaN | \r\n-connected to the user system using teamvi... | GRP_0 |
| 3906 | NaN | -user unable tologin to vpn.\r\n-connected to... | GRP_0 |
| 3910 | NaN | -user unable tologin to vpn.\r\n-connected to... | GRP_0 |
| 3915 | NaN | -user unable tologin to vpn.\r\n-connected to... | GRP_0 |
| 3921 | NaN | -user unable tologin to vpn.\r\n-connected to... | GRP_0 |
| 3924 | NaN | name:wvqgbdhm fwchqjor\nlanguage:\nbrowser:mic... | GRP_0 |
| 4341 | NaN | \r\n\r\nreceived from: eqmuniov.ehxkcbgj@gmail... | GRP_0 |

## 1.2. Missing values in Description column

| | Short description | Description | Assignment group |
|---|---|---|---|
| 4395 | i am locked out of skype | NaN | GRP_0 |

## 1.3. Imputation:

- We have various ways of treating the NULL/Missing values in the dataset such as
  - Replacing them with empty string
  - Replacing them with some default values
  - Duplicating the Short description and Description values wherever one of them is Null
  - Dropping the records with null/missing values completely.
- We're not choosing to drop any record as we don't want to lose any information. And as we're going to concatenate the Short description and Description columns for each record while feeding them into NLP, we neither want to pollute the data by introducing any default values nor bias it by duplicating the description columns.
- Hence our NULL/Missing value treatment replaces the NaN cells with just empty string.

### 2. Data Cleaning

Before we start with any text analytics we need to pre-process the data to get it all in a consistent format.We need to clean, tokenize and convert our data into a matrix. Some of the basic text pre-processing techniques includes:

- Translation: A small number of tickets were written in German. Hence, we used the Google translate python api to convert German to English to generate the input data for the next steps. However, the google translator api can only process a limited number of texts on a daily basis, so we translated the text in batches and saved the file for further processing.
- Make text all **lowercase** so that the algorithm does not treat the same words in different cases as different
- **Removing Noise** i.e everything that isn't in a standard number or letter i.e Punctuation, Numerical values
- Removing extract spaces
- Removed punctuations
- Removed words containing numbers
- **Tokenization**: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.
- **Stopword Removal**: Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words
- **Lemmatization**

## 10. Feature Engineering

We can concatenate Short Description and Description to form a single column named 'New_Description' and use it as a predictor

## 11. Deciding Models and Model Building

*Since we need to classify the tickets among one of the assignment groups, this is a multiclass classification problem.*

*We used Downsampling because the dataset is very much imbalanced. In 74 groups, 46% of tickets are in group 1 and 16 groups just have more than 100 tickets. If we*

*conducted random sampling towards all the subcategories, then we would face a problem that we might miss all the tickets in some categories. Hence, we conducted sampling towards groups that have more than 100 tickets.*

- BenchMark Model

*We will be using classification algorithms, to start with we have used below basic Machine Learning algorithms:*

- *Multinomial NB*
- *Linear Support vector Machine*
- *Logistic regression*
- *Xgboost*

- **Multinomial NB**

```
accuracy 0.6718861209964413
f1 score 0.7837546719316676
              precision recall  f1-score   support

           0    0.69      1.00      0.82      795
           1    0.00      0.00      0.00      28
           2    0.75      0.18      0.29      51
           3    0.00      0.00      0.00      29
           4    0.00      0.00      0.00      24
           5    0.00      0.00      0.00      43
           6    0.75      0.12      0.21      48
           7    1.00      0.21      0.34      58
           8    0.00      0.00      0.00      23
           9    0.00      0.00      0.00      40
          10        0.00      0.00      0.00      21
          11        0.00      0.00      0.00      26
          12        1.00      0.03      0.05      37
          13        0.55      0.92      0.68      132
          14        0.00      0.00      0.00      50

    accuracy                           0.67      1405
   macro avg        0.32      0.16      0.16      1405
```

```
weighted avg        0.56      0.67      0.56      1405

[[795   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 17   0   0   0   0   0   0   0   0   0   0   0   0  11   0]
 [ 36   0   9   0   0   0   1   0   0   0   0   0   0   5   0]
 [ 28   0   0   0   0   0   0   0   0   0   0   0   0   1   0]
 [ 21   0   2   0   0   0   0   0   0   0   0   0   0   1   0]
 [ 43   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 42   0   0   0   0   0   6   0   0   0   0   0   0   0   0]
 [ 46   0   0   0   0   0   0  12   0   0   0   0   0   0   0]
 [ 23   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 39   0   0   0   0   0   1   0   0   0   0   0   0   0   0]
 [ 21   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  3   0   0   0   0   0   0   0   0   0   0   0   0  23   0]
 [ 11   0   0   0   0   0   0   0   0   0   0   0   1  25   0]
 [ 11   0   0   0   0   0   0   0   0   0   0   0   0 121   0]
 [ 14   0   1   0   0   0   0   0   0   0   0   0   0  35   0]]
```

- **Linear Support vector Machine**

```
accuracy 0.7217081850533807
f1 score 0.7948898452920107
              precision recall  f1-score   support

           0    0.74      1.00      0.85      795
           1    0.00      0.00      0.00      28
           2    0.73      0.31      0.44      51
           3    0.80      0.14      0.24      29
           4    0.40      0.08      0.14      24
           5    1.00      0.09      0.17      43
           6    0.93      0.29      0.44      48
           7    0.89      0.59      0.71      58
           8    1.00      0.30      0.47      23
           9    0.00      0.00      0.00      40
          10        1.00      0.10      0.17      21
          11        0.00      0.00      0.00      26
          12        0.86      0.32      0.47      37
          13        0.56      0.92      0.70      132
```

```
          14        1.00      0.10      0.18       50

    accuracy                           0.72      1405
   macro avg        0.66      0.28      0.33      1405
weighted avg        0.71      0.72      0.65      1405

[[792   0   2   0   0   0   0   1   0   0   0   0   0   0   0]
 [ 16   0   0   0   0   0   0   0   0   0   0   0   1  11   0]
 [ 26   0  16   0   2   0   0   0   0   0   0   0   0   7   0]
 [ 22   1   0   4   0   0   0   0   0   0   0   0   1   1   0]
 [ 19   0   2   0   2   0   0   0   0   0   0   0   0   1   0]
 [ 39   0   0   0   0   4   0   0   0   0   0   0   0   0   0]
 [ 33   0   0   0   0   0  14   1   0   0   0   0   0   0   0]
 [ 23   0   0   0   0   0   0  34   0   1   0   0   0   0   0]
 [ 15   0   0   0   0   0   0   0   7   0   0   0   0   1   0]
 [ 37   0   0   0   0   0   1   2   0   0   0   0   0   0   0]
 [ 18   0   1   0   0   0   0   0   0   0   2   0   0   0   0]
 [  3   0   0   0   0   0   0   0   0   0   0   0   0  23   0]
 [  3   0   0   0   0   0   0   0   0   0   0   0  12  22   0]
 [  9   0   0   0   1   0   0   0   0   0   0   0   0 122   0]
 [ 12   0   1   1   0   0   0   0   0   0   0   0   0  31   5]]
```

Linear support vector algorithm seem to perform better than Multinomial NB.

- **Xgboost**

```
accuracy 0.7293447293447294
f1 score 0.7683327811648981
              precision    recall  f1-score   support

           0       0.81      0.95      0.87       794
           1       0.36      0.18      0.24        28
           2       0.54      0.49      0.52        51
           3       0.53      0.34      0.42        29
           4       0.38      0.12      0.19        24
           5       0.40      0.23      0.29        43
           6       0.85      0.48      0.61        48
           7       0.80      0.55      0.65        58
           8       0.71      0.43      0.54        23
```

```
           9       0.37      0.17      0.24        40
          10       0.67      0.29      0.40        21
          11       0.00      0.00      0.00        26
          12       0.67      0.28      0.39        36
          13       0.55      0.92      0.69       132
          14       0.70      0.14      0.23        51

    accuracy                           0.73      1404
   macro avg       0.56      0.37      0.42      1404
weighted avg       0.70      0.73      0.69      1404

[[754   3  11   3   2   5   1   3   2   7   0   0   2   0   1]
 [  9   5   0   2   0   0   1   0   0   0   0   0   0  11   0]
 [ 15   0  25   0   1   1   1   0   0   0   1   0   0   7   0]
 [  9   6   0  10   1   0   0   0   0   0   0   0   2   1   0]
 [ 14   0   3   0   3   1   0   0   1   0   0   0   0   1   1]
 [ 29   0   1   0   0  10   0   1   0   2   0   0   0   0   0]
 [ 24   0   1   0   0   0  23   0   0   0   0   0   0   0   0]
 [ 20   0   0   0   0   0   0  32   1   3   2   0   0   0   0]
 [ 12   0   0   0   0   0   0   1  10   0   0   0   0   0   0]
 [ 24   0   0   1   0   4   1   3   0   7   0   0   0   0   0]
 [ 10   0   1   0   0   3   0   0   0   0   6   1   0   0   0]
 [  1   0   0   0   0   0   0   0   0   0   0   0   1  23   1]
 [  2   0   0   1   0   0   0   0   0   0   0   0  10  23   0]
 [  5   0   3   0   1   0   0   0   0   0   0   1   0 122   0]
 [  8   0   1   2   0   1   0   0   0   0   0   0   0  32   7]]
```

- **Logistic regression**

```
accuracy 0.7537366548042704
f1 score 0.7742423923879348
              precision recall  f1-score   support

           0    0.85      0.91      0.88       795
           1    0.59      0.36      0.44       28
           2    0.50      0.45      0.47       51
           3    0.76      0.55      0.64       29
           4    0.53      0.38      0.44       24
           5    0.48      0.37      0.42       43
           6    0.60      0.60      0.60       48
           7    0.94      0.76      0.84       58
           8    0.73      0.70      0.71       23
```

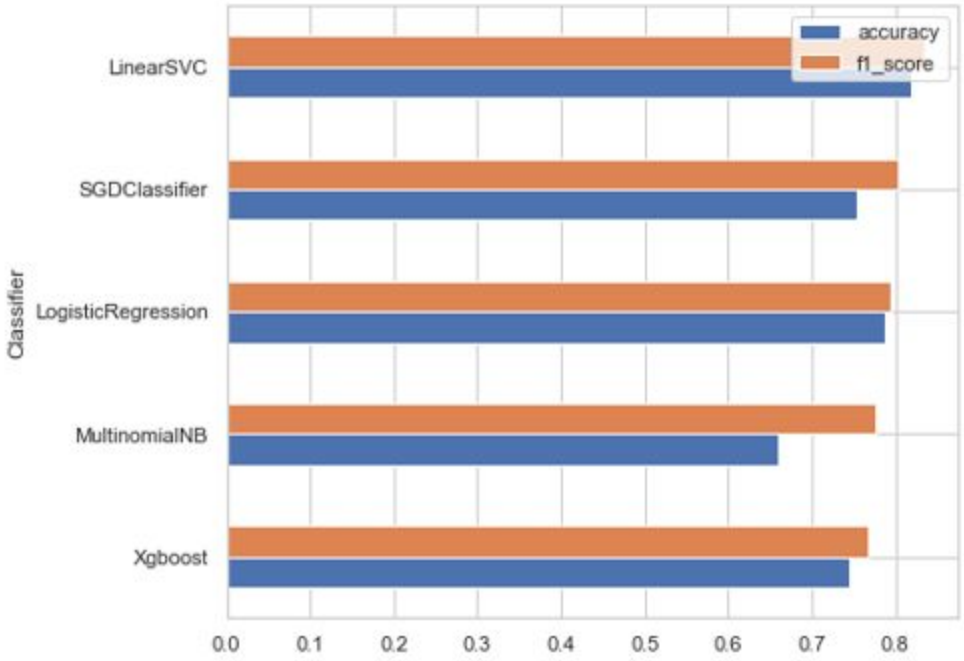
```
           9    0.43      0.40      0.42       40
          10         0.62      0.38      0.47       21
          11         0.00      0.00      0.00       26
          12         0.69      0.30      0.42       37
          13         0.56      0.93      0.70      132
          14         0.79      0.22      0.34       50

    accuracy                           0.75      1405
   macro avg         0.60      0.49      0.52      1405
weighted avg         0.74      0.75      0.73      1405
```

```
[[727   2  13   2   4  12  15   0   5   9   3   0   2   0   1]
[  6  10   0   0   0   0   0   0   0   0   0   0   1  11   0]
[ 12   0  23   0   3   1   1   1   0   3   1   0   0   6   0]
[  7   3   0  16   0   0   0   0   0   0   0   0   2   1   0]
[ 11   0   1   0   9   0   1   0   1   0   0   0   0   1   0]
[ 21   0   0   0   0  16   0   0   0   6   0   0   0   0   0]
[ 17   0   2   0   0   0  29   0   0   0   0   0   0   0   0]
[ 11   0   1   0   0   0   0  44   0   2   0   0   0   0   0]
[  6   0   0   0   0   0   1   0  16   0   0   0   0   0   0]
[ 18   0   0   0   0   3   1   1   0  16   1   0   0   0   0]
[  9   0   1   0   0   1   0   1   0   1   8   0   0   0   0]
[  1   1   0   0   0   0   0   0   0   0   0   0   0  23   1]
[  1   0   0   2   0   0   0   0   0   0   0   0  11  22   1]
[  3   0   4   0   1   0   0   0   0   0   0   1   0 123   0]
[  5   1   1   1   0   0   0   0   0   0   0   0   0  31  11]]
```

| Classifier | accuracy | f1_score |
|---|---|---|
| **Xgboost** | 0.744847 | 0.766310 |
| **MultinomialNB** | 0.658849 | 0.775546 |
| **LogisticRegression** | 0.788202 | 0.794654 |
| **SGDClassifier** | 0.754087 | 0.803002 |

| | | |
|---|---|---|
| **LinearSVC** | 0.818053 | 0.833642 |




*LinearSVC gives better performance with*
***accuracy*** `0.833642`
***f1 score*** `0.818053`

Although, it seems like the call is biased towards GRP_0 which has a majority of samples.

- Class Imbalance
  - Even after downsampling the data we see that the predictions are biased towards GRP_0 which has a majority of samples.
  - Imbalance causes two problems:
    - Training is inefficient as most samples are easy examples that contribute no useful learning signal;
    - The easy examples can overwhelm training and lead to degenerate models.
  - A common solution is to perform some form of hard negative mining that samples hard examples during training or more complex sampling/re weighing schemes.In order to handle the imbalance problem we used class_weight option so that weightage is given to classes with lower samples, we used class_weight='balanced' option

**After setting the class_weight to balanced, we re-trained various classifiers and below are the results:**

| Classifier | accuracy | f1_score |
|---|---|---|
| Xgboost | 0.681592 | 0.661371 |
| SGDClassifier | 0.728500 | 0.708847 |
| MultinomialNB | 0.658849 | 0.775546 |
| LogisticRegression | 0.788202 | 0.792341 |
| LinearSVC | 0.797441 | 0.797100 |

Although the scores have reduced, we are now ensured that the predictions are not biased and weightage is given to classes with less samples.

**We could see LinearSVC perform better with below score**

**accuracy** 0.797441

**f1 score** 0.797100

- Word2vec embedding

Next, we also tried using pre trained word embedding, but the only challenge was that we could not find any embeddings trained on ITSM data. We used the glove model with 100d for this, and then used logistic regression and Xgboost to train the model. But, the scores were poorer than the benchmark model.

- Doc2vec

Next, we tried vector space modelling Doc2Vec embedding with Logistic regression and Xgboost model, we used two techniques here

- Distributed memory and
- Distributed bag of words
- Concatenated both these models for prediction

Below are the model performance, obtained from vector space modelling:

| Classifier | accuracy | f1_score |
|---|---|---|

| | | |
|---|---|---|
| Doc2Vec (dm) - LogisticRegression | 0.421464 | 0.381583 |
| Word2Vec - LogisticRegression | 0.467662 | 0.415653 |
| Doc2Vec (dbow) - LogisticRegression | 0.567875 | 0.520968 |
| Doc2Vec (dbow + dm) - LogisticRegression | 0.566453 | 0.521671 |

Although,'Doc2Vec' is a more advanced model in NLP rather than 'Tf-Idf', in our case, it is not giving proper results. We have tried with a linear & boosting based classifier respectively.

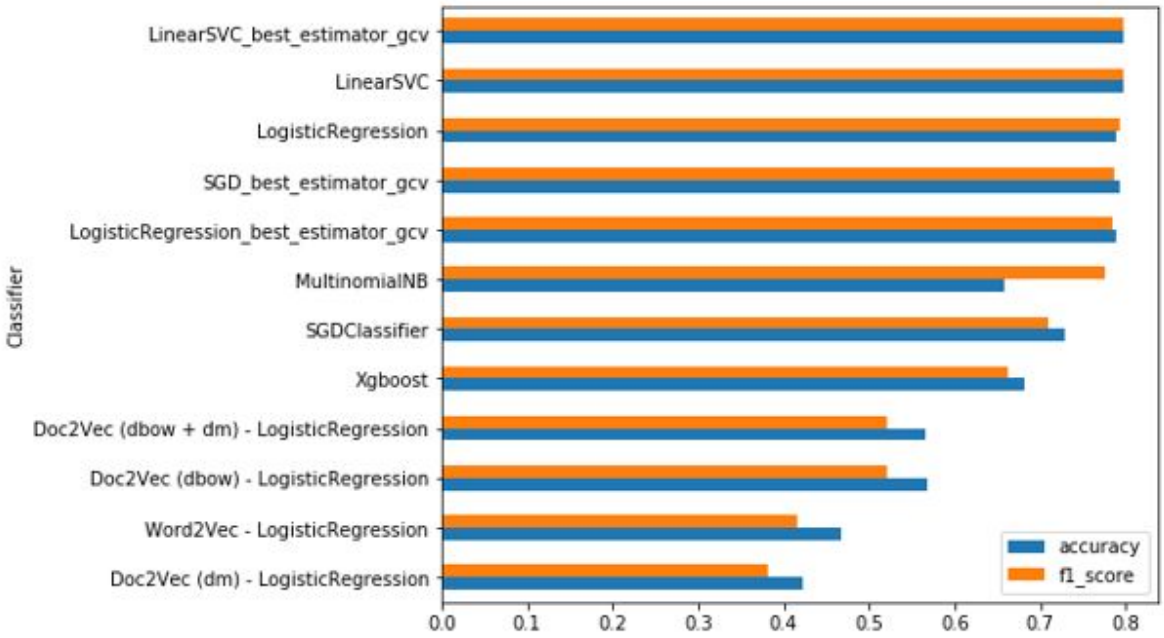
In our dataset, 'texts' are domain-specific. Furthermore, 'Doc2Vec' model is more suitable for very well written grammatically correct texts. In our case, texts are quite rough in nature.It is also proven in various examples and Data Scientist's experiments that though 'Tf-Idf' model is inferior as compared to 'Doc2Vec', it still gives better results while classifying very domain specific texts.

# 12. HyperParameter Tuning

We then tuned the models with different parameters and then compared the results and picked the one with the best values and below is the scores for Approach1.

| Classifier | accuracy | f1_score |
|---|---|---|
| Doc2Vec (dm) - LogisticRegression | 0.421464 | 0.381583 |
| Word2Vec - LogisticRegression | 0.467662 | 0.415653 |
| Doc2Vec (dbow) - LogisticRegression | 0.567875 | 0.520968 |
| Doc2Vec (dbow + dm) - LogisticRegression | 0.566453 | 0.521671 |
| Xgboost | 0.681592 | 0.661371 |
| SGDClassifier | 0.728500 | 0.708847 |
| MultinomialNB | 0.658849 | 0.775546 |

| | | |
|---|---|---|
| LogisticRegression_best_estimator_gcv | 0.788202 | 0.784387 |
| SGD_best_estimator_gcv | 0.792466 | 0.787450 |
| LogisticRegression | 0.788202 | 0.792341 |
| LinearSVC | 0.797441 | 0.797100 |
| LinearSVC_best_estimator_gcv | 0.797441 | 0.797100 |




In most cases results were pretty similar but for some of the models, Linear SVC and Logistic regression performed much better (especially after applying hyperparameters) so at some point we decided to work with Linear SVC and Logistic regression only.

# 13. Approach 2

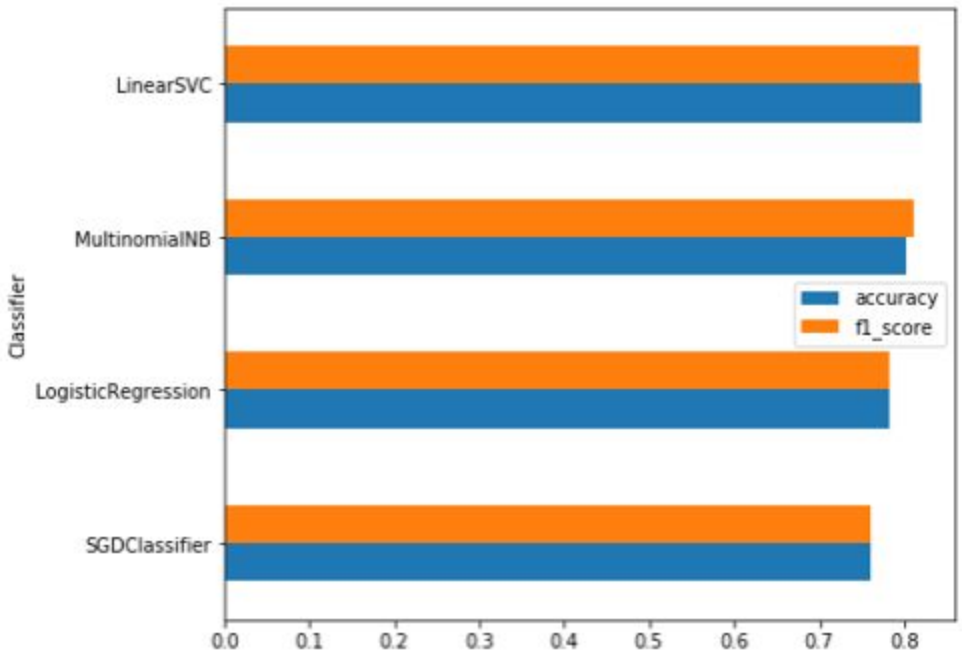
In the AS-IS process it's mentioned that around ~54% of the incidents are resolved by L1 / L2 teams and the rest will be resolved as L3. So the assumption is that GRP_0 and GRP_8 which contribute 54% of the tickets are related to L1/L2 teams and the rest of the tickets belongs to L3 teams

So firstly, the ticket would be classified into L1/L2 or L3 classes and then it would be further classified into one of the given assignment groups.

Since the dataset is very imbalanced, we considered a subset of groups for predictions. In 74 groups, 46% of tickets belong to group 1 and 16 groups have more than 100 tickets and around 22 groups have more than 50 tickets, rest of the Assignment groups have very less ticket counts which might not add much value to the model prediction. If we conducted random sampling towards all the subcategories, then we would face a problem that we might miss all the tickets in some categories. Hence, we considered the groups that have more than 50 tickets in this approach.

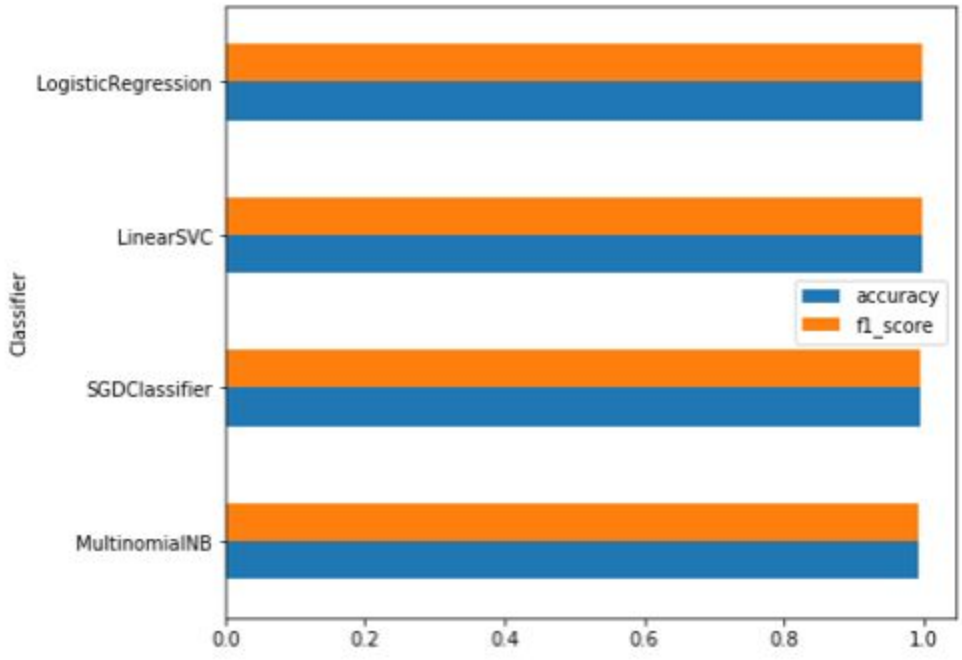
- We first created a model to classify the given tickets as l1/l2 or l3 tickets, we found that Linear SVC was giving a better score.

| Classifier | accuracy | f1_score |
|---|---|---|
| **SGDClassifier** | 0.761320 | 0.759826 |
| **LogisticRegression** | 0.782665 | 0.782090 |
| **MultinomialNB** | 0.803364 | 0.812145 |
| **LinearSVC** | 0.819534 | 0.818434 |



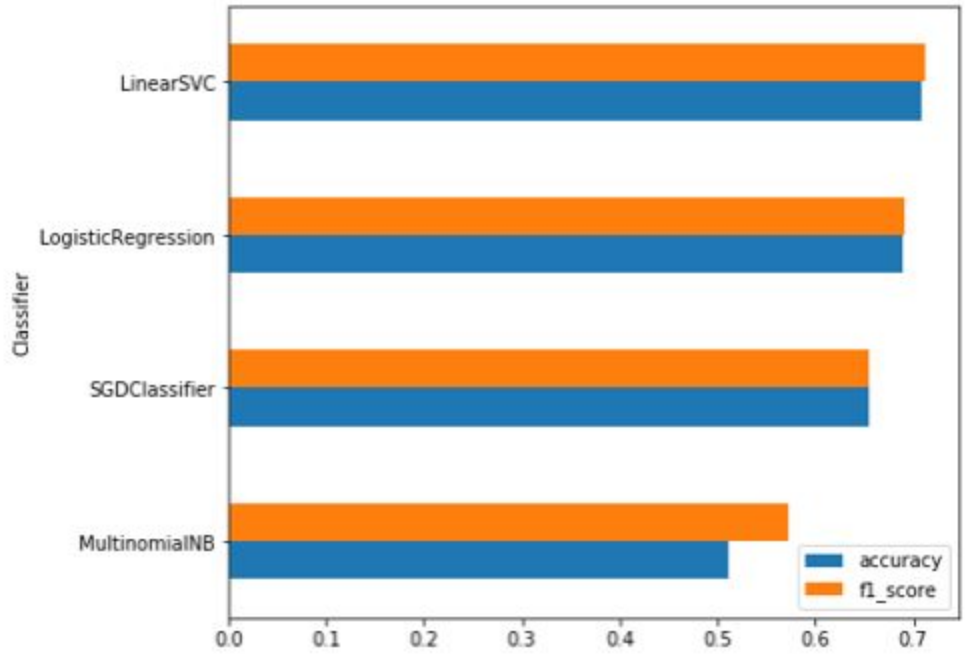

Next, another model was trained considering only the l1/l2 level of incidents consisting of GRP_0 and GRP_8.

| Classifier | accuracy | f1_score |
|---|---|---|
| **MultinomialNB** | 0.992416 | 0.992505 |
| **SGDClassifier** | 0.994583 | 0.994592 |
| **LinearSVC** | 0.996750 | 0.996766 |
| **LogisticRegression** | 0.996750 | 0.996766 |



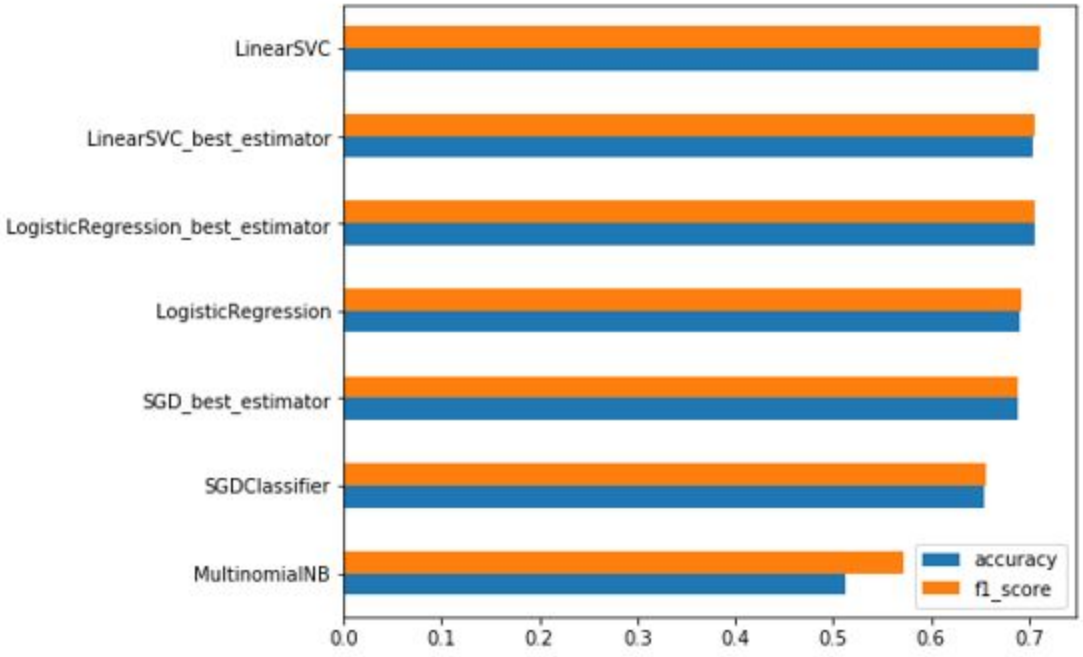

Finally, a third model was trained considering only l3 level of tickets.

| Classifier | accuracy | f1_score |
|---|---|---|
| **MultinomialNB** | 0.512039 | 0.572078 |
| **SGDClassifier** | 0.654896 | 0.655273 |
| **LogisticRegression** | 0.690209 | 0.692030 |
| **LinearSVC** | 0.709470 | 0.712074 |




We also used hyperparameter tuning, to identify the best classifier with best parameters. We found that LinearSVC was performing the best among all the other classifiers.

| Classifier | accuracy | f1_score |
|---|---|---|
| **MultinomialNB** | 0.512039 | 0.572078 |
| **SGDClassifier** | 0.654896 | 0.655273 |
| **SGD_best_estimator** | 0.688604 | 0.688376 |
| **LogisticRegression** | 0.690209 | 0.692030 |
| **LogisticRegression_best_estimator** | 0.706260 | 0.705392 |
| **LinearSVC_best_estimator** | 0.704655 | 0.706799 |
| **LinearSVC** | 0.709470 | 0.712074 |




We also tried keras implementation with focal loss as a loss function to handle the class imbalance problem, which helps in giving more weightage to groups with less samples, but the results were not satisfactory.

Finally, Logistic Regression gave better performance with hyperparameter tuning and this model would be used for classifying the L3 tickets into one of the groups.

accuracy 0.706260

f1 score 0.705392

# 14. Model Deployment

Deployment of machine learning models or putting models into production means making your models available to the end users or systems. Here we have deployed the application using Flask Web application.

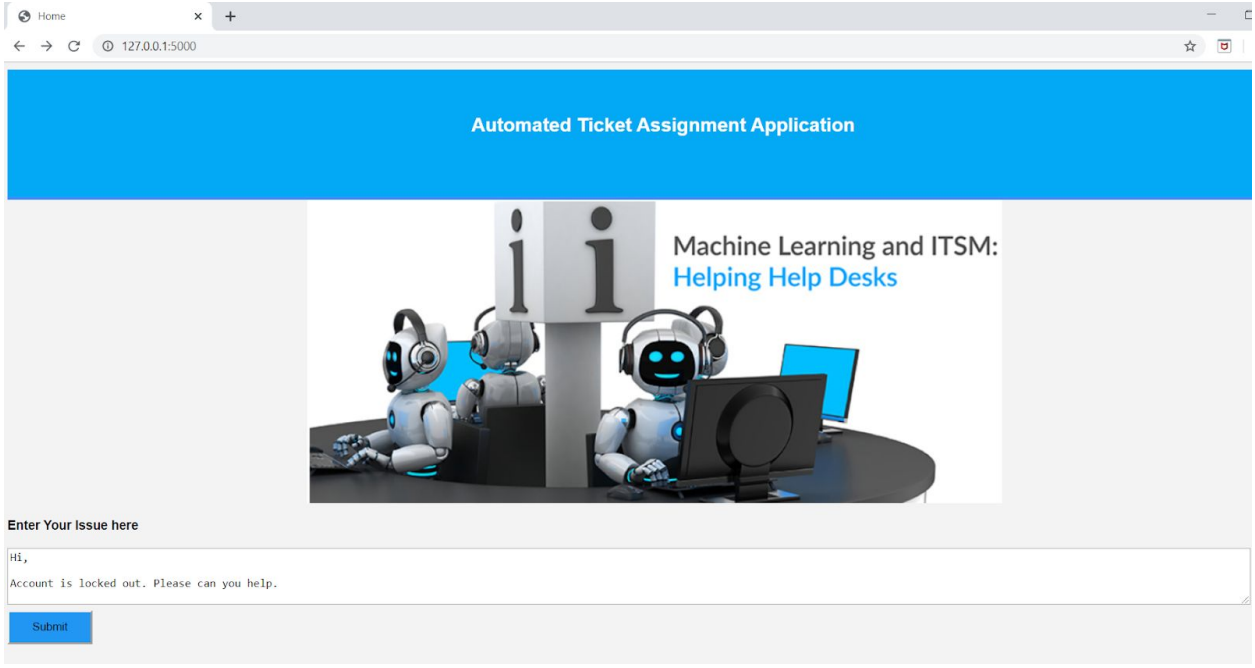
Following are the steps for model deployment:

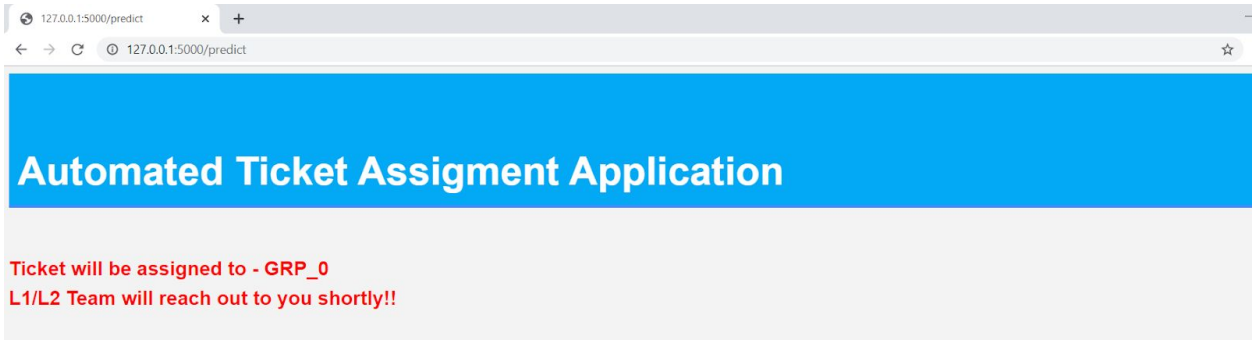
- First, we serialize the model using joblib (i.e persist a python object on the disk that can be transferred anywhere and later de-serialized (read) back by a python script)
- Write a Flask Application which has below parts

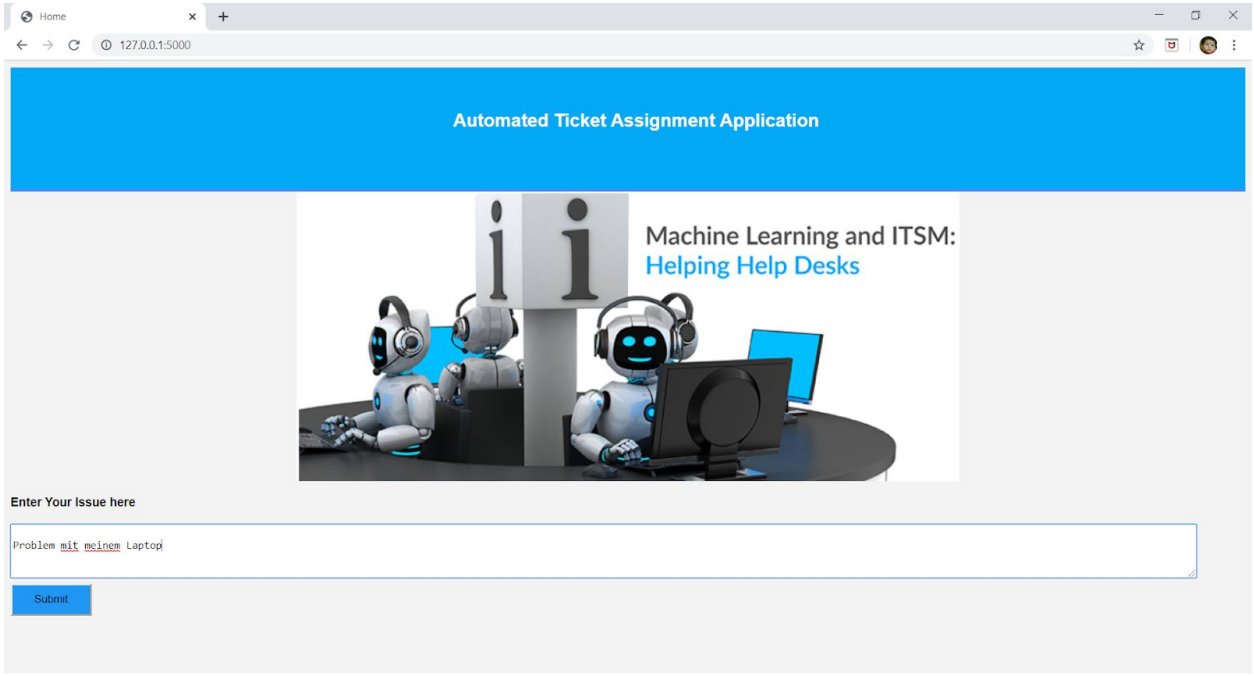
- app.py — This contains Flask APIs that receives ticket details through GUI or API calls, computes the predicted value based on our model and returns it.
  - Load serialized model
  - This creates a route that receives input via GUI
    - preprocess the text by cleaning,
    - removing stop words and
    - translating the text to english
  - Then it uses the trained model to make a prediction, and returns that prediction, which can be accessed through the API endpoint.
- HTML/CSS — This contains the HTML template and CSS styling to allow users to enter issue/ticket details and displays the predicted assignment group

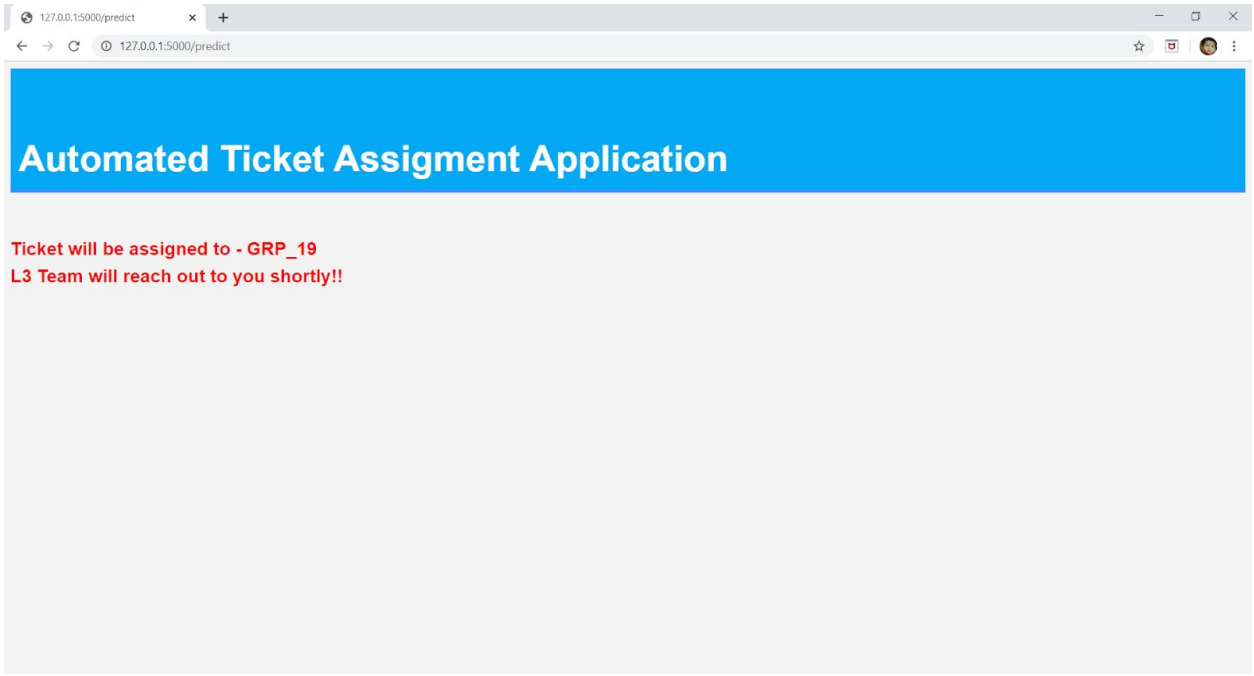
- Web application is hosted on localhost http://127.0.0.1:5000/

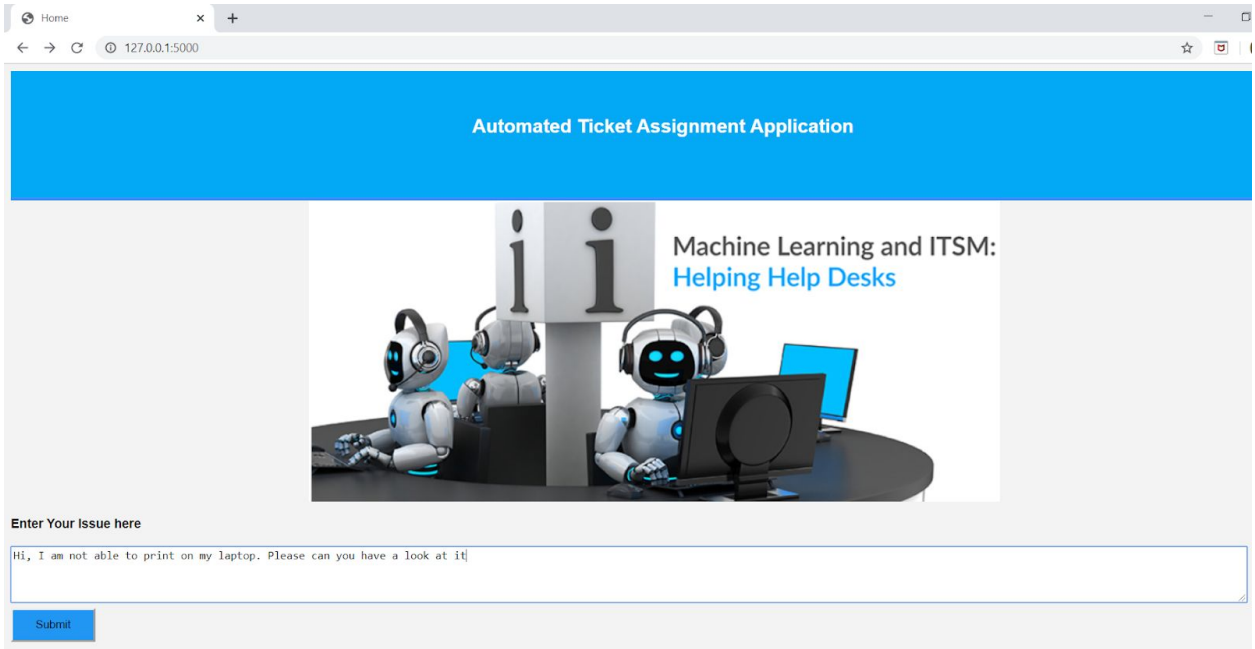
Below are few snapshots of web application host on Flask Framework:

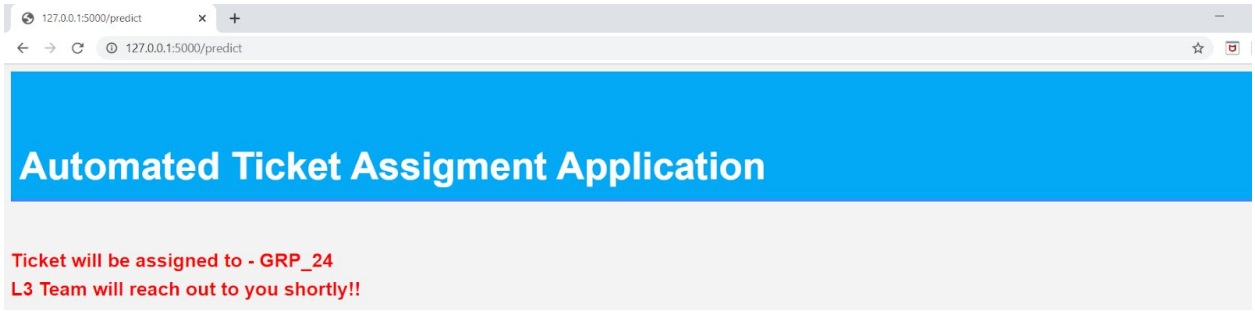


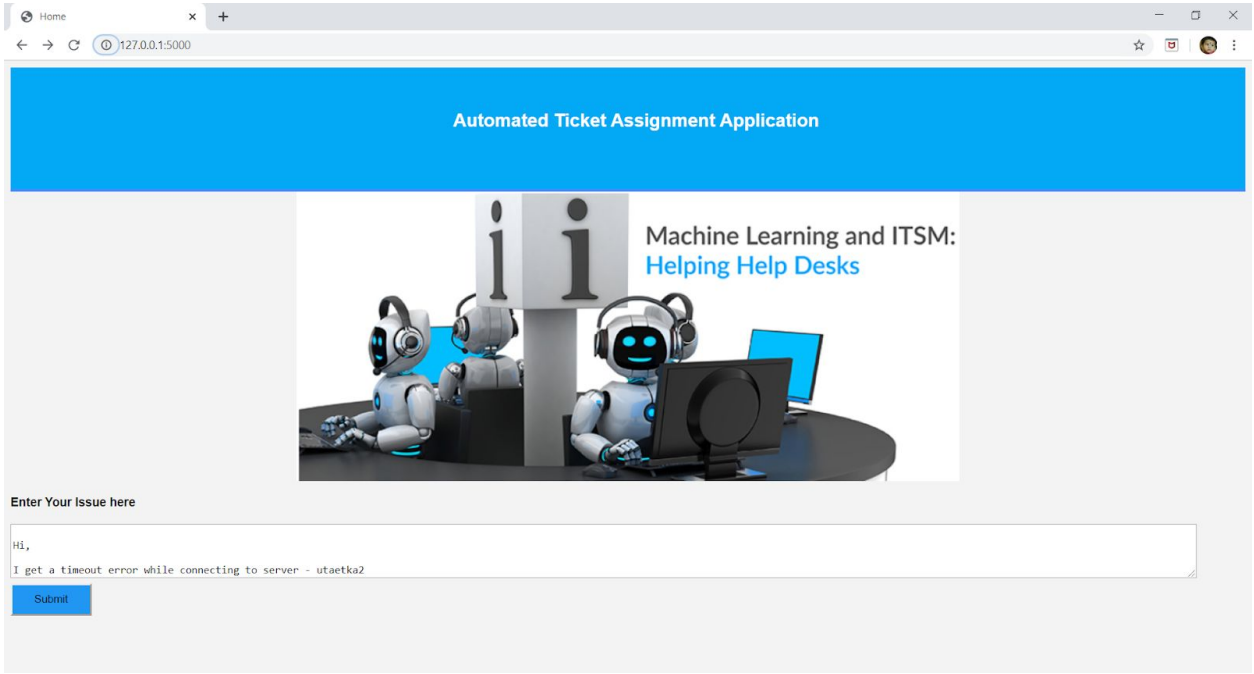

127.0.0.1:5000/predict
127.0.0.1:5000/predict
Automated Ticket Assigment Application
Ticket will be assigned to - GRP_0
L1/L2 Team will reach out to you shortly!!

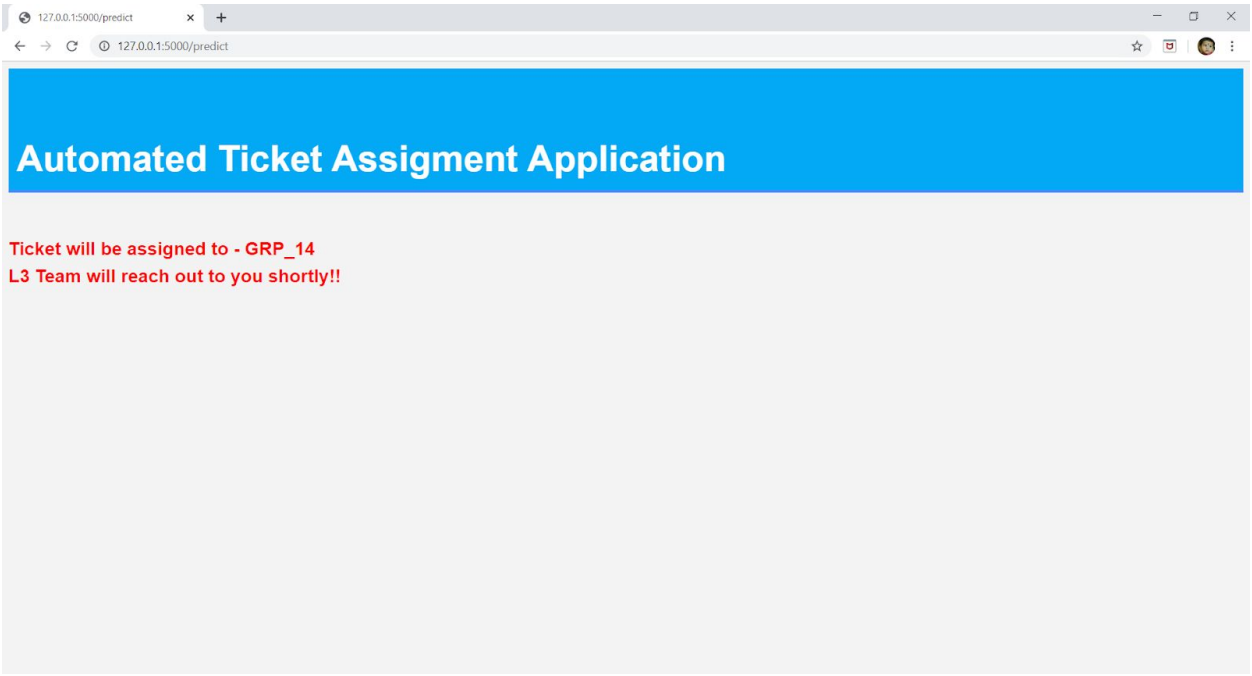

Home
127.0.0.1:5000
Automated Ticket Assignment Application
Machine Learning and ITSM:
Helping Help Desks
Enter Your Issue here
Problem mit meinem Laptop
Submit


127.0.0.1:5000/predict
127.0.0.1:5000/predict
Automated Ticket Assigment Application
Ticket will be assigned to - GRP_19
L3 Team will reach out to you shortly!!


Home
127.0.0.1:5000
Automated Ticket Assignment Application
Machine Learning and ITSM:
Helping Help Desks
Enter Your Issue here
Hi, I am not able to print on my laptop. Please can you have a look at it
Submit


127.0.0.1:5000/predict
127.0.0.1:5000/predict
Automated Ticket Assigment Application
Ticket will be assigned to - GRP_24
L3 Team will reach out to you shortly!!


Home
127.0.0.1:5000
Automated Ticket Assignment Application
Machine Learning and ITSM:
Helping Help Desks
Enter Your Issue here
Hi,
I get a timeout error while connecting to server - utaetka2
Submit




# 15. Limitations and Future Work

This application does not consider samples with lower ticket counts, as a future scope of work,

- We could collect more data around groups with lower ticket samples.
- We could also use any pre-trained word embeddings specific to ITSM tickets.

# 16. Conclusions

We first analysed the dataset provided to us, understood the structure of the data provided - number of columns, field , data types etc.

- We did Exploratory Data Analysis to derive further insights from this data set and we found that
    - Data is very much imbalanced, there are around ~45% of the Groups with less than 20 tickets.
    - Few of the tickets are in foreign language like German
    - The data has a lot of noise in it, for eg- few tickets related to account setup are spread across multiple assignment groups.

- We performed the data cleaning and preprocessing
    - Translation: A small number of tickets were written in German. Hence, we used the Google translate python api to convert German to English to generate the input data for the next steps. However, the google translator rest api can only process a limited number of texts on a daily basis, so we translated the text in batches and saved the file for further processing.
    - Make text all lowercase so that the algorithm does not treat the same words in different cases as different
    - Removing Noise i.e everything that isn't in a standard number or letter i.e Punctuation, Numerical values
    - Removing extract spaces
    - Removed punctuations
    - Removed words containing numbers
    - Stopword Removal: Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words
    - Lemmatization
    - Tokenization: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.
- We then ran a basic benchmark model using the cleaned and preprocessed dataset
    - Since the dataset is very imbalanced, We considered a subset of groups for predictions. In 74 groups, 46% of tickets belong to group 1 and 16 groups just have more than 100 tickets, rest of the Assignment groups have very less ticket counts which might not add much value to the model prediction. If we conducted random sampling towards all the subcategories, then we would face a problem that we might miss all the tickets in some categories. Hence, we considered the groups that have more than 100 tickets.
    - We trained the data using below models:
        - Multinomial NB
        - Linear Support vector Machine
        - Logistic regression
        - Xgboost
- Logistic regression gives better benchmark performance with
    - accuracy 0.7537366548042704
    - f1 score 0.7742423923879348

- Even after downsampling the data we see that the predictions are biased towards GRP_0 which has a majority of samples.

- Imbalance causes two problems:

    - Training is inefficient as most samples are easy examples that contribute no useful learning signal;
    - The easy examples can overwhelm training and lead to degenerate models. A common solution is to perform some form of hard negative mining that samples hard examples during training or more complex sampling/re weighing schemes.In order to handle the imbalance problem we used class_weight=balanced hyperparameter while training the model, which tells the model to "pay more attention" to samples from an under-represented class.
- Although, the accuracy and f1_score went down. This ensured that the classes were being correctly classified with lesser number of misclassification and good precision/recall scores for all the classes

- Next, we also tried using pre trained word embedding, but the only challenge was that we could not find any embeddings trained on ITSM data. We used the glove model with 100d for this, and then used logistic regression and Xgboost to train the model. But, the scores were poorer than the benchmark model.

- Then, we also tried vector space modelling using Doc2Vec with DistributedBOW and Distributed Memory approach, though ‘Doc2Vec’ is a more advanced model in NLP rather than ‘Tf-Idf’, but still in our case, it is not giving proper results. We have tried with a linear & boosting based classifier respectively. Furthermore, ‘Doc2Vec’ model is more suitable for very well written grammatically correct texts.
- In most cases results were pretty similar but for some of the models, Linear SVC and Logistic regression performed much better (especially after applying hyperparameters) so at some point we decided to work with Linear SVC and Logistic regression only.

- We also tried an alternative approach, as it's mentioned that around ~54% of the incidents are resolved by L1 / L2 teams and the rest will be resolved as L3. So the assumption is that GRP_0 and GRP_8 which contribute 54% of the tickets are related to L1/L2 teams and the rest of the tickets belongs to L3 teams
- we used Approach 2 where the ticket would be classified into L1/L2 or L3 classes and then it would be further classified into one of the given assignment groups.
- We first created a model to classify the given tickets as l1/l2 or l3 tickets, we found that Linear SVC was giving a better score.

- Next, another model was trained considering only the l1/l2 level of incidents consisting of GRP_0 and GRP_8.
- Finally, a third model was trained considering l3 level of tickets.

- We also used hyperparameter tuning, and tuned the models with different parameters and then compared the results and picked the one with the best values. We found that LinearSVC was performing the best among all the other classifiers.
- We also tried keras implementation with focal loss as a loss function to handle the class imbalance problem, which helps in giving more weightage to groups with less samples, but the results were not satisfactory.

  In our dataset, ‘texts’ are domain-specific. In our case, texts are quite rough in nature. Though ‘Tf-Idf’ model is inferior as compared to ‘Doc2Vec’, it still gives better results while classifying very domain specific texts.

# 17. Benefits

- Machine learning model is able to predict the assignment groups instantaneously for the new tickets
- The prediction is accurate for ~78% of the tickets
- Machine learning-based automation of triaging has the ability to reduce the load on the triage team to a greater extent