**Capstone Project Interim Report**

**NLP 1- Automatic Ticket Assignment**

**By,**

**Chaitanya Shahane**

**Prateek Singh**

**Pratheep Kumar C.**

**Rahul Chahel**

**Santosh Sinha**

**Date: 12th December 2021.**

**Mentored by,**

**Somya Rai.**

**Table of Contents**

**Contents**

[1. Summary of Problem Statement, data and findings 1](#_Toc90227201)

[1.1 Summary of Problem Statement: 1](#_Toc90227202)

[1.2 Objective: 2](#_Toc90227203)

[1.3 Current dataset and findings: 2](#_Toc90227204)

[2. Approach to EDA and Data pre-processing 3](#_Toc90227205)

[2.1 Observation from Target variable: 3](#_Toc90227206)

[2.2 Data Pre-processing Approach: 4](#_Toc90227207)

[3. Pre-processing 5](#_Toc90227208)

[3.1 Dropping duplicate tickets: 5](#_Toc90227209)

[3.2 Handling null values and dropping unwanted features: 5](#_Toc90227210)

[3.3 Data Cleansing: 6](#_Toc90227211)

[3.4 Target variable distribution: 8](#_Toc90227212)

[3.5 Language detection and translation: 9](#_Toc90227213)

[3.6 Text analysis: 9](#_Toc90227214)

[4. Modelling 12](#_Toc90227215)

[4.1 Support Vector Machines (SVM) Model: 12](#_Toc90227216)

[4.2 Random Forest Classifier Model: 12](#_Toc90227217)

[4.3 Neural Network Model: 14](#_Toc90227218)

[5. Performance Improvement 15](#_Toc90227219)

6. List of Figures…………………………………………………………………………………. 17

# Summary of Problem Statement, data and findings

## 1.1 Summary of Problem Statement:

In today’s IT world, service management suits hold a key in delivering better quality of customer support and incident management. In Agile world, we call it “Keep Lights ON”. This means we need to provide uninterrupted service and if an unplanned interruption occurs, we need to handle that interruption. Main goal of such service management suits is to provide a quick solution or a workaround to restore the service with immediate effect. Question is “how to handle the distribution of such incidents and decide who will handle which incident”. Handling this distribution of incidents/tickets take lot of man hours and efforts. Process is also prone to error and if mistake happens, it might cost company a customer.

Facts about Incident management system:

* Tickets created by various stakeholders are assigned to service desk teams L1 and L2.
* L1 and L2 will analyse those tickets and decide whether they can solve it initially.
* If L1/L2 cannot solve the ticket then it will categorize, prioritize and escalated to be handled by Functional teams i.e., L3.
* L3 teams will analyse the issue and resolve it.

Side effects of this system-

* L1/L2 teams spend time revieing and analysing tickets
* Almost 25 to 30% tickets need to go through Standard Operating Procedures (SOPs)
* One SOP take almost 15 min
* After SOP, it takes almost 1 FTE effort to assign these tickets to assign to L3 teams.
* As this process is error-prone, almost 25% tickets can get assigned to wrong L3 teams. Therefore, L3 teams spends time assigning tickets to right L3 team.
* During such time-consuming process, few tickets get stuck in queue or miss out. This defeats the purpose of these systems.

Automation in the field of incident handling will be cost-effective. L1/L2 teams will have time for other important tasks. L3 teams will not to spend more time on sending wrongly sent tickets to correct teams.

## 1.2 Objective:

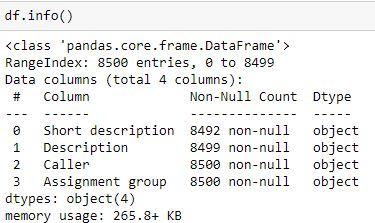
As stated in Chapter 1.1, approximately 25% of the tickets are getting wrongly assigned in the current manual effort-based system. About ~75% of the tickets are currently getting assigned correctly but with is a time consuming and error-prone manual process. To improve on the current process, we need to build an AI-based classifier of the tickets which can assign tickets to correct functional group with at least 85% accuracy. This system will save time of L1, L2 and L3 teams by improving their efficiency and increasing the level of end user satisfaction.

## 1.3 Current dataset and findings:

* Dataset has four features namely “Short description”, “Description”, “Caller” and “Assignment group”.
* “Assignment group” is the target variable where classification of the tickets is stated. It has 74 different classes meaning we 74 different functional groups where tickets can get assigned.
* In “Description”, we can find text in multiple languages which we need to identify.
* Also, multiple special characters and symbols are present in this feature.
* Data in the “Description” feature also includes hyperlinks, image date and URLs.
* There is total 8582 records in this dataset. There are 83 duplicate records/tickets in the dataset which have been removed in the first pre-processing step. “Short description” and “Description” have few null values. Text data is in Raw form currently with multiple languages, numbers, special characters, urls and spelling mistakes.

# Approach to EDA and Data pre-processing

Following is the information about the dataframe ingested in the Notebook:

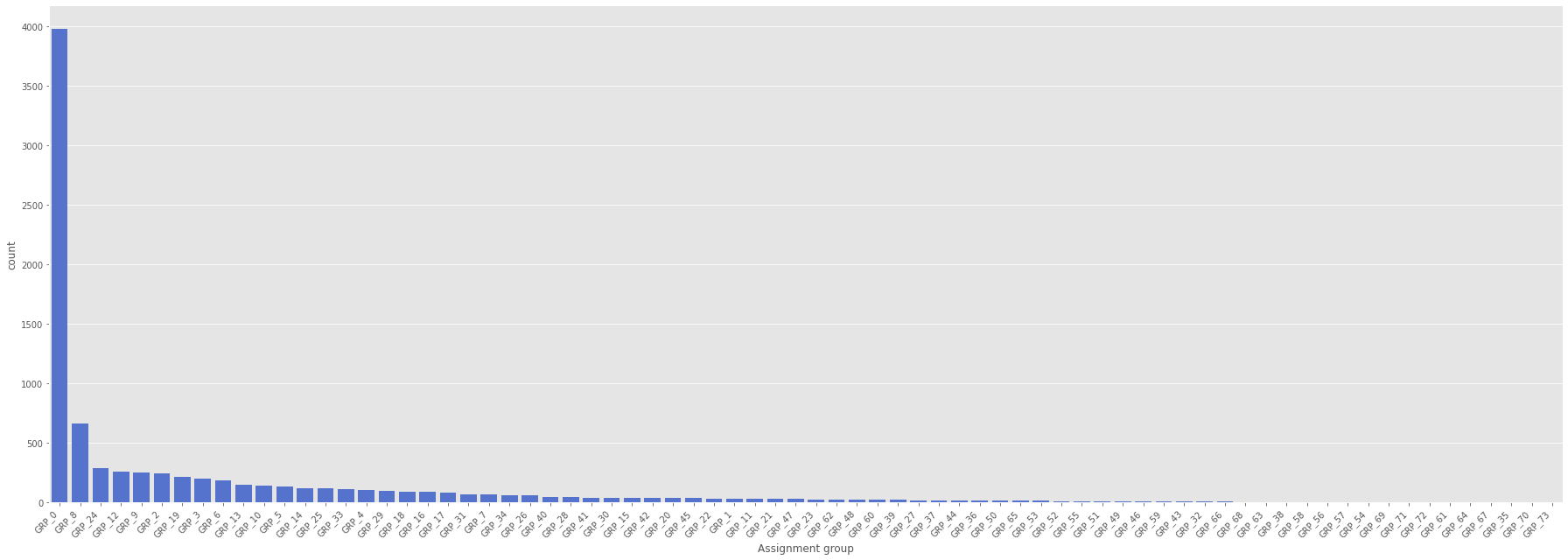


*Figure 1. Dataset information*

From Fig. 1, we can see that we have four features. Considering the data in the all four features, “Assignment Group” is our target variable. It has the data referring to the classification of the tickets. “Description” has the information regarding the issue because of which that particular ticket has been raised. This information will be vital in training our AI-based classifier.

Short Description feature is of no significant value as the same detail is available in description field hence it is dropped along with “Caller” feature in the dataset.

## 2.1 Observation from Target variable:



*Figure 2. Target Variable distribution*

* All 8500 tickets are assigned into 74 different functional groups.
* 3976 tickets are assigned to only GRP\_0.
* There a number of groups in the current dataset where only single ticket has been assigned. GRP\_0 account for ~50% of the tickets.
* Therefore, we can say that the distribution in target variable is extremely skewed and imbalanced.

## 2.2 Data Pre-processing Approach:

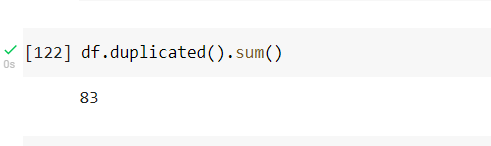
* First, of the pre process step executed should be checking for duplicates records and removing the duplicates.
* Identified duplicates will be dropped and then Null value check can be done on the dataset. Description & Short Description features having null values will be addressed. We can replace “Description” with “Short Description” for the record having Null values. Null values in Short Description feature may not have any impact as this feature could be mostly dropped in next few steps.
* We can try to combine Description & Short Description features & check the prediction accuracy. If this step does not contribute to any gain in accuracy, then “Caller” & “Short Description” features can be dropped.
* Contraction can be removed from the merged feature as it does not bring any value.
* To improve training accuracy, we will convert all texts/descriptions to small case.
* Remove all the special symbols, URLs, non-ASCII characters, HTML tags from merged feature.
* As we observed earlier, there are other languages present in the merged feature other than English. Those languages need to be recognised and translated to English.
* Merged feature needs to be tokenized.
* Apart from Contraction, we can remove the stop words such as ‘are’, ‘or’, ‘and’, ‘is’, etc.
* Do spell-check for merged feature.
* Later Lemmatization should be performed.

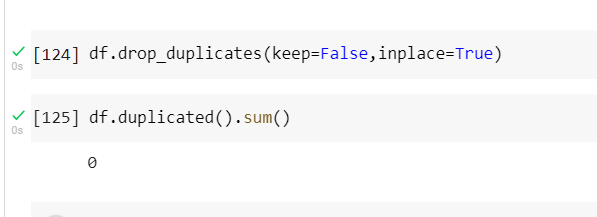
# Pre-processing

## 3.1 Dropping duplicate tickets:

Duplicate records do not help the model building process and if model is built on a training data with lot of duplicate values than algorithm considers those multiple records while training and this could result in overfitting. Best way to proceed is to remove duplicate records.

Given dataset has as many as 83 duplicate records. As mentioned above, those records will be dropped. Following is the snippet from notebook.





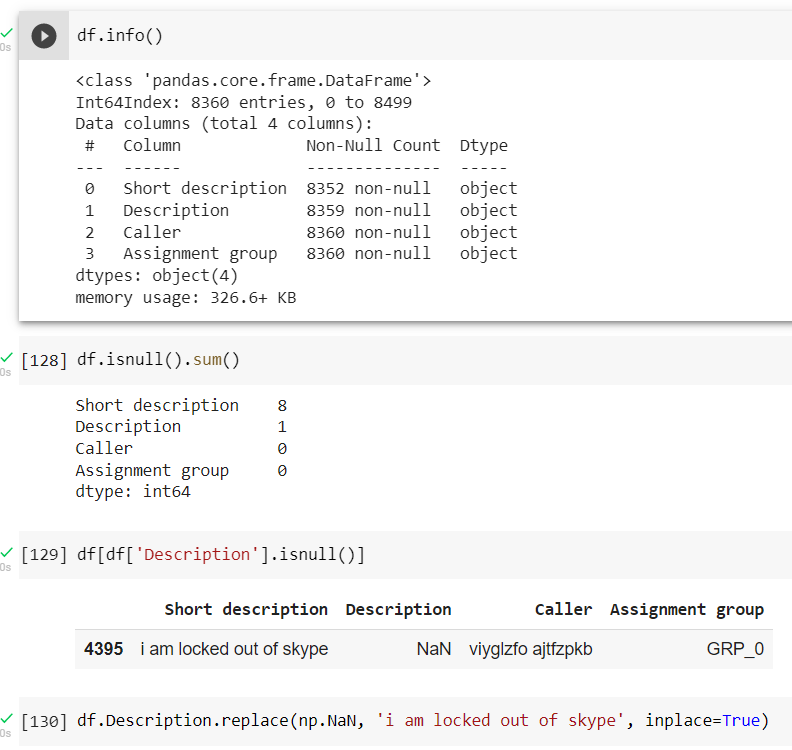
*Figure 3. Duplicates from the data*

## 3.2 Handling null values and dropping unwanted features:

As the next step checked for *NULL* values in the dataset, “Short Description” and “Description” columns have few *NULL* values. “Description” has only one *NULL* and to address this, we will replace that *NULL* with the value from “Short Description” of the corresponding record.

As mentioned earlier, “Short Description” column contains details which are already present in the description column and clubbing them together will lead to duplicity in the train feature. Since short description column will be dropped hence Null values in the Short Description column need not be corrected.

Also, “Caller” column is of no significance and will be dropped from the dataset. Before dropping this feature, a separate dataframe containing unique values of the callers will be created, these unique caller details will be used to remove caller names in the description column by replacing them with blank space.



*Figure 4. Null Values in the dataset*

## 3.3 Data Cleansing:

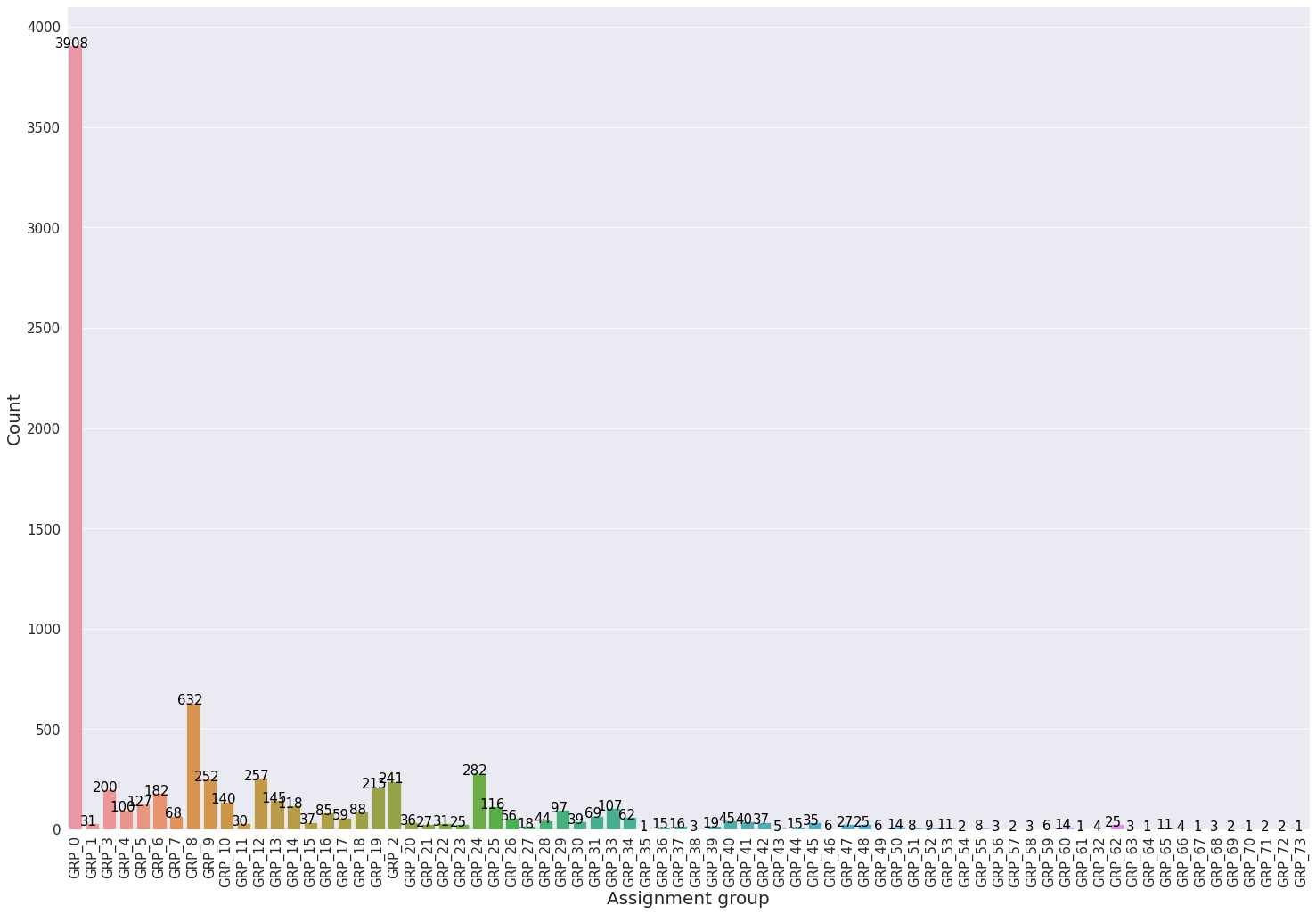
This process involves cleaning unnecessary characters/word/symbol from the train data. In the given dataset, now, there are two features left, “Description” and “Assignment Group” (Target). Text Data in the “Description” will be cleaned.

Here, Cleaning includes removal of following characters/word/symbol:

1. **HTML tags or characters** – HTML parser is available in the package named **BeautifulSoup**. Using this, we can find HTML tags present in the text and replace it with nothing.
2. There are characters present in the data which are **not ASCII characters**. These special characters need to be removed using a package named **unicodedata.**
3. There are also special characters present which are ASCII characters like #, %, +, etc. These characters can be removed using **regex.**
4. Further step in the cleaning involves removal of **Stop words**. For this purpose, we are using NLTK library’s inbuilt package named **nltk.corpus.stopwords**
5. It has following stop words.  
   {‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’, ‘we’, ‘these’, ‘your’, ‘his’, ‘through’, ‘don’, ‘nor’, ‘me’, ‘were’, ‘her’, ‘more’, ‘himself’, ‘this’, ‘down’, ‘should’, ‘our’, ‘their’, ‘while’, ‘above’, ‘both’, ‘up’, ‘to’, ‘ours’, ‘had’, ‘she’, ‘all’, ‘no’, ‘when’, ‘at’, ‘any’, ‘before’, ‘them’, ‘same’, ‘and’, ‘been’, ‘have’, ‘in’, ‘will’, ‘on’, ‘does’, ‘yourselves’, ‘then’, ‘that’, ‘because’, ‘what’, ‘over’, ‘why’, ‘so’, ‘can’, ‘did’, ‘not’, ‘now’, ‘under’, ‘he’, ‘you’, ‘herself’, ‘has’, ‘just’, ‘where’, ‘too’, ‘only’, ‘myself’, ‘which’, ‘those’, ‘i’, ‘after’, ‘few’, ‘whom’, ‘t’, ‘being’, ‘if’, ‘theirs’, ‘my’, ‘against’, ‘a’, ‘by’, ‘doing’, ‘it’, ‘how’, ‘further’, ‘was’, ‘here’, ‘than’}   
     
   These words do not add any value to our classification. Hence, these are removed.
6. **Lemmatization** – This process removes the words which are form of another word. E.g., went, going, gone are forms of the word ‘go’. Using **nltk.step.WordLemmatizer** , we can remove such words.
7. Removing **URLs** with **regex.** This was done using the following regular expression: r'^https?:\/\/.\*[\r\n]\*'
8. **Email Ids** – This is also achieved using **regex**: r'\S+@\S+'
9. Unnecessary **new line** characters are removed which do not contribute to modelling process.
10. As mentioned before characters like ‘**#**’ are removed. With current tradition of hash tagging, we can remove hashtag and keep the text attached to it by using **regex**.
11. We also need to look for foreign characters. Characters which are **not** **Unicode**. Also, we have few unreadable characters are removed by using **regex**.
12. Stripping extra spaces from start & end of the sentences, also regex is used to remove extra spaces in between words & sentences.
13. In the ticket description, text such as "received from:", "from:", "to:", "subject:", "sent:", "ic:", "cc:", "bcc:", "issue resolved." appears regularly. These text pattern have been removed using regex. Also, caller names are removed by referencing with the list of unique callers saved earlier.

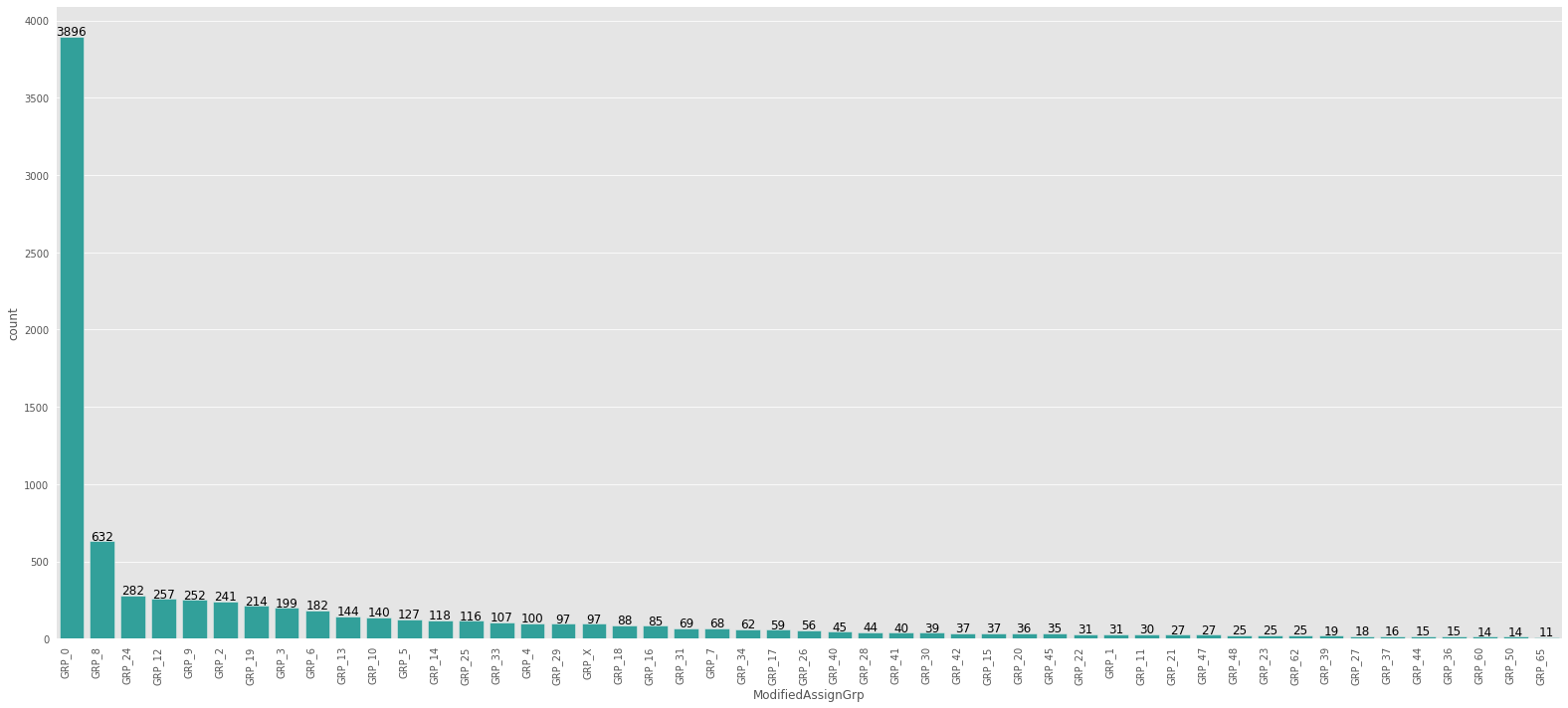
## 3.4 Target variable distribution:

We have already observed that the distribution in Target variable is extremely skewed. After cleaning the data, out of 8360 records 3908 records belong to GRP\_0. We can visualize the distribution from following Fig. 5.



*Figure 5. Target variable distribution after data cleaning*

This skewness of data will result into overfitting with very good accuracy for “GRP\_0” and other classes will have less accuracy on testing data. It can be observed that there are multiple Groups with just one ticket or less than 10 tickets. These groups will not help in classification as the data is too small for model to learn anything meaningful.



*Figure 6. Target variable distribution with 49 groups*

To decrease the skewness of the Target variable, we can combine these groups together to make one group named ‘GRP\_X'. Now there are total 49 groups left. Fig. 6 will display that there are no groups with less than 10 tickets assigned.

## 3.5 Language detection and translation:

If given dataset is observed, it can be concluded that there are multiple languages used in the “Description” feature apart from English language.

**Language detection** -Language of the word from given Corpus can be detected by using python package **langdetect**, detect() function available in the package can be used to detect the language from the supplied text. In fact, there are words from 29 other languages present in the given Corpus. Most of the words are German apart from English language. Once language is detected, we need to translate those words to English to make the Corpus in same language i.e., English.

**Language translation** – For language translation, google library named **deep\_translator** is used. A function named GoogleTranslator() is used to detect and translate text in another language into English language.

Graphical user interface

Description automatically generated with medium confidence

*Figure 7. Language detection and translation*

From Fig. 7, it can be observed that other languages apart from English has been detected and translated into English later.

## 3.6 Text analysis:

Text analysis will be performed on the cleansed & pre-processed data, which is used for training of the model. To perform Text analysis, we need to break each document i.e. ticket description into a list of words. These words need to be tokenized and should be converted to lowercase. For this purpose, python package named **gensim.utils** is used, **simple\_preprocess**() function is used from this package to tokenize these words.

While performing the Text analysis, one must also consider that words can work in a group for example, bigrams, trigrams, etc. When words are analysed as a group, then the sequence of the words is used for understanding the meaning & prediction. E.g., a trigram “please turn page” means something different than the word” turn” individually. Such words/phrases should be considered in training the model, the package **genism** has these phrases pre-compiled,function **genism.models.Phrases** can used to create these bigram and trigrams.

Another aspect of Text analysis is to find the frequency of the word in each of the groups or classes. After cleaning the documents of the corpus, the most frequent and important words in each class can be plotted using WordClouds. WordClouds is the technique which can help in visualizing the frequency of the words by its size and brightness in the WordCloud. Maximum of top 100 words can be visualized in a WordCloud.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |

*Figure 8. Bigram Wordcloud*

From Fig. 8, WordCloud on the right side suggest that words like ‘failed’, ‘circuit’, ‘issue’, etc. are most frequent & important words.

To observe the length of the documents or ticket description, a plot can be made with the number of description v/s ranges of the number of words called as ‘Bins’. From following Fig. 7, same can be visualized.

Fig. 9 suggests that ~8000 records are in the range of 0-100, word length. There are ~7 records which have description length between 300-500 words. No words found in the frequency range above this.

Chart

Description automatically generated

*Figure 9. Word distribution per bin*

# Modelling

The task at hand is classification based on the ticket description and we started first with traditional models. The dataset used in the current modelling is the raw dataset without any resampling and with existing target class skewness.

## 4.1 Support Vector Machines (SVM) Model:

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labelled training data for each category, they’re able to categorize new text. This is a fast and dependable classification algorithm that performs very well with a limited amount of data. SVM model was hyper tuned using the Grid-search to determine most suitable hyper parameter values.

Graphical user interface, text, application

Description automatically generated

*Figure 10. SVM Modelling*

## 4.2 Random Forest Classifier Model:

Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Chart, radar chart

Description automatically generated

*Figure . Random Forest Modelling*

*Figure 11. Random forest modelling*

**Observation from our experiments:** Random Forest Model using raw data seems to perform better as compared to Random Forest Model with PCA done. It also affirms the understanding that PCA uses linearity in the data to reduce dimensionality, whereas the problems such NLP uses the non-linearity of feature.

Graphical user interface, text, application, email

Description automatically generated

*Figure 12. Random Forest Implementation*

## 4.3 Neural Network Model:

Neural Network model is provided word vectors as the input, where a document of a corpus is converted into matrix of numbers by using statistical NLP techniques such a One hot coding, BoW or TF-IDF. Ticket descriptions have been converted into term frequency vectors post pre-processing and then fed into a Neural Network model. Performance of the NN model is comparable to traditional models as the accuracy of classification on validation data is compared but a huge overfit is seen in the results as the train accuracy achieved is much higher compared to test accuracy.

Table

Description automatically generated

*Figure 13. Model fitting for Neural Network*

# Performance Improvement

As it mentioned in pre-processing description above, target class in the dataset is highly unbalanced and skewed. The imbalance of the dataset would be addressed such that data in the minority classes will be up-sampled and majority class data will be down-sampled to create a single balanced dataset. Other this previous strategy we can divide dataset in two dataframes one containing only Grp\_0 and rest of the group’s in a resampled dataframe.

Chart

Description automatically generated

*Figure 14. Skewed distribution of Target variable*

After addressing the target class imbalance, traditional models and LSTM based models will be tried with the resampled datasets & the performance will be compared for each of the models. Following three Datasets will be used for the modelling are:

* Dataset1: Raw data with the target class without any sampling
* Dataset2: Resampled data where all the target classes are sampled with a count of ~630. (E.g., Grp\_0 is down sampled and other groups are up sampled)
* Dataset3: Model with Two datasets: Model 1 with Grp\_0 & Model 2 with all other groups except Grp\_0 and Model 2 is resampled

Other than traditional models, Bidirectional LSTM models using word2vec and glove embeddings will be executed on Dataset 1 & Dataset2.

**Word Embedding:**

Machine Learning and Deep learning algorithms are incapable of processing strings or plain text in their raw form, word embeddings are used to convert the texts into numbers. There may be different numerical representations of the same text. It tries to map a word using a dictionary to a vector.

We will be experimenting with below 2 types of embeddings in our models with the dimension as 100.

*Word2Vector Embedding:*

Word2Vec models are shallow, two-layer [neural networks](https://en.wikipedia.org/wiki/Neural_network) that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large [corpus of text](https://en.wikipedia.org/wiki/Text_corpus) and produces a [vector space](https://en.wikipedia.org/wiki/Vector_space), typically of several hundred [dimensions](https://en.wikipedia.org/wiki/Dimensions), with each unique word in the [corpus](https://en.wikipedia.org/wiki/Corpus_linguistics) being assigned a corresponding vector in the space.

*GloVe (Global Vectors) Embedding:*

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

1. **List of Figures**

|  |  |  |
| --- | --- | --- |
| No | Name | Page No. |
| 1 | Dataset Information | 3 |
| 2 | Target variable information | 3 |
| 3 | Duplicates from the data | 5 |
| 4 | Null values from the dataset | 6 |
| 5 | Target variable distribution after data cleaning | 8 |
| 6 | Target variable distribution with 49 groups | 8 |
| 7 | Language detection and translation | 9 |
| 8 | Bigram WordCloud | 10 |
| 9 | Word distribution per bin | 11 |
| 10 | SVM Modelling | 12 |
| 11 | Random Forest Modelling | 13 |
| 12 | Random Forest Implementation | 13 |
| 13 | Model fitting with Neural Networks | 14 |
| 14 | Skewed distribution of the target variable | 15 |