**INTERIM REPORT**

**AI Enabled IT Ticketing Service Tool**

*Submitted By*

*Disha Palan*

*Gloria Kamal*

*Parita Desai*

Table of Contents

[**1.** Summary of problem statement, data and findings 3](#_Toc75957530)

[**Problem Statement** 3](#_Toc75957531)

[**Objective** 4](#_Toc75957532)

[Observations from the given Dataset 4](#_Toc75957533)

[**Solution** 4](#_Toc75957534)

[**2.** Overview of the final process 5](#_Toc75957535)

[Observations from Target Class 5](#_Toc75957536)

[Data Pre-processing 5](#_Toc75957537)

[**3.** Step-by-step walk through the solution 5](#_Toc75957538)

[**Data** 5](#_Toc75957539)

[Summary of the Approach to EDA and Pre-processing 6](#_Toc75957540)

[EDA 6](#_Toc75957541)

[**Findings** 8](#_Toc75957542)

[Other findings 10](#_Toc75957543)

[Further Data Analysis 11](#_Toc75957544)

[Word cloud for Short description 11](#_Toc75957545)

[Word cloud for Description 12](#_Toc75957546)

[Assignment group distribution of top 10 callers as compared to other callers 12](#_Toc75957547)

[Assignment group distribution amongst top 10 callers 13](#_Toc75957548)

[Assignment group distribution 13](#_Toc75957549)

[Feature Engineering 15](#_Toc75957550)

[Data Pre-processing 16](#_Toc75957551)

[Data Cleaning 17](#_Toc75957552)

[NER and POS Tagging 19](#_Toc75957553)

[Deciding Models and Model Building 20](#_Toc75957554)

[Models 21](#_Toc75957555)

[Traditional Models 21](#_Toc75957556)

[Sequential Models 23](#_Toc75957557)

[How to improve your model performance? 24](#_Toc75957558)

[**4.** Comparison to benchmark 24](#_Toc75957559)

[**5.** Implications 24](#_Toc75957560)

[**6.** Limitations 25](#_Toc75957561)

[**7.** Closing Reflections 25](#_Toc75957562)

# Summary of problem statement, data and findings

## **Problem Statement**

In any of the IT industry, incident management plays an important role in delivering quality and timely support to its customers across the globe.

The incidents are generally created by various stakeholders like end users, vendors, IT users, etc. They might not have right information as to which team the ticket should go to. Hence, to improve and retain customer satisfaction, it is very important that the ticket is assigned to the right group of people for faster and appropriate resolution. In many Organizations this is still a manual process. There are few problems with the manual process:

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

L1 / L2 needs to spend time to review Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum 25–30% of incidents needs to be reviewed for SOPs before ticket assignment).

15 mins are 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 and loss of business.

### **Objective**

We are building an AI solution which will enable organizations to classify incidents to the right functional group by implementing the best suited machine learning model and leading to customer satisfaction.

Guided by AI, organizations can reduce the resolution time and focus on more productive tasks. This will overcome and save time with below losses:

1. Time latency due to review of SOPs before assigning to right functional group
2. Incorrect assignments to functional groups
3. Every ticket costs the IT organization $13, despite an average accuracy score (chance of reaching the desired target) of only 40%. Incorrectly assigned tickets bounce between business groups for an average of 21 days before landing in the right place. Cost, latency and accuracy are a huge concern, and lead to poor user experiences.1

## Observations from the given Dataset

* Four columns – Short Description, Description, Caller and Assignment group
* 74 Assignment groups found - Target classes
* Caller names in a random fashion (may not be useful for training data)
* European non-English language also found in the data
* Email/chat format in description
* Symbols & other characters in the description
* Hyperlinks, URLS & few image data found in the description
* Blanks found either in the short description or description field
* Few descriptions same as the short description
* Few words were combined
* Spelling mistakes and typo errors are found

### **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 analyzing 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.

We have tried below pre-modelling and modelling techniques :

**Traditional Models :**

1. Multinomial Naïve Bayes
2. SVM classifier
3. KNN Classifier
4. SGD Classifier
5. Random Forest
6. XGBOOST

**Sequential Models :**

1. Convolutional Neural Networks
2. Bidirectional LSTM Models using both embedding and compared

Transfer Learning : Glove embedding

# Overview of the final process

## Observations from Target Class

* The Target class distribution is extremely skewed
* A large no of entries for GRP\_0 (mounting to 3976) which account for ~50% of the data
* There are groups with 1 entry also. We could merge all groups with small entries to a group to reduce the imbalance in the target. This may reduce the imbalance to some extent.
* The target class were filtered for less than 10 entries and grouped together as there is no much information with the groups individually.

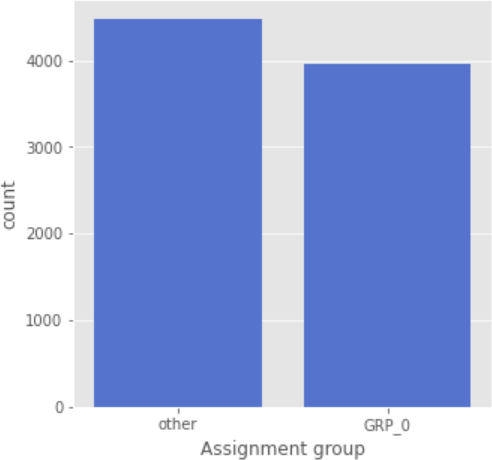
## Data Pre-processing

Below steps have been performed for initial pre-processing and cleanup of data.

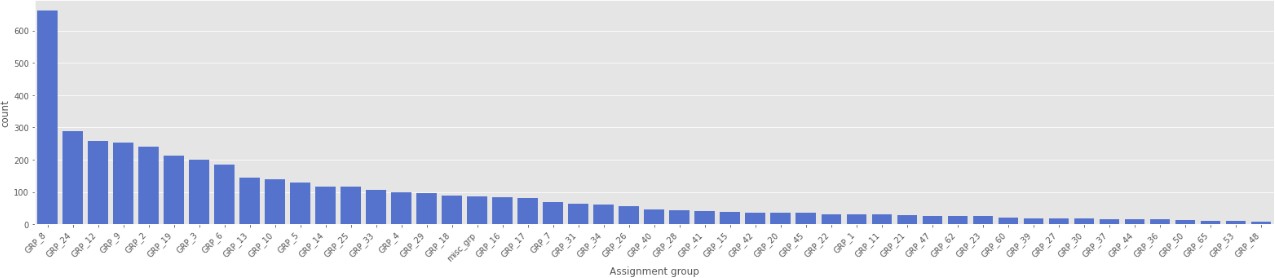
* Dropped the caller field as the data was not found to be useful for analysis
* Replaced Null values in Short description & description with space.
* Merged Short Description & Description fields for analysis
* Contraction words found in the merged Description are removed for ease of word modelling
* Changed the case sensitivity of words to the common one
* Removed Hashtags and kept the words, Hyperlinks, URLs, HTML tags & non-ASCII symbols from merged fields.
* Translating all languages (German) to English
* Tokenization of merged data
* Removal of Stop words
* Lemmatization
* WordCloud created for all available 50 groups to have more information specific to Assignment groups
* Attempted to do spell check
* Created Plot to understand the distribution of words

To further address the imbalance in the target class, we have split the dataset as 2 groups

* A dataset where we resample all the groups to size 660. Here note GRP0 would be downsampled & all other groups would be upsampled.
* We could use 2 separate models. Here one model would be used to classify the GRP0 & a second model would be used to classify the other groups. The dataset from the 1st model contains GRP0 data & all the remaining data combined to a single group, say, ‘Others’. The dataset for the 2nd Model would contain all groups other than GRP0. The dataset here would be resampled again to address the target imbalance if any.



**Distribution of Target class for Other groups**



# Step-by-step walk through the solution

## **Data**

Reference: <https://drive.google.com/open?id=1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ>

The given dataset has below four columns:

1. Short description
2. Description
3. Caller
4. Assignment group

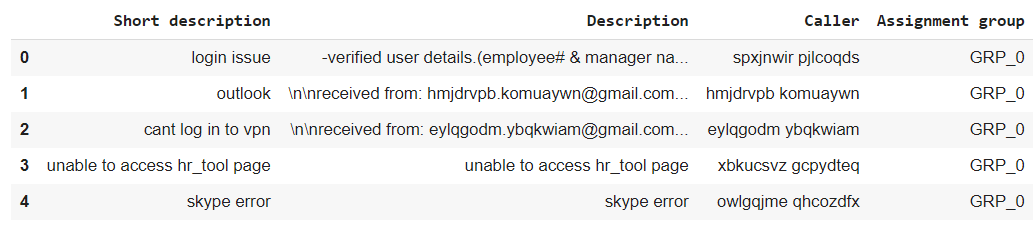
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | | Description | Data type | | |
| **Short description** | | Short description on the problem for which incident is being  raised | | 8492 | non-null object |
| **Description** | | Detailed description of the problem | | 8499 | non-null 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 |

Out of above four columns we have 3 features namely, short description, description and caller and one target group namely assignment group

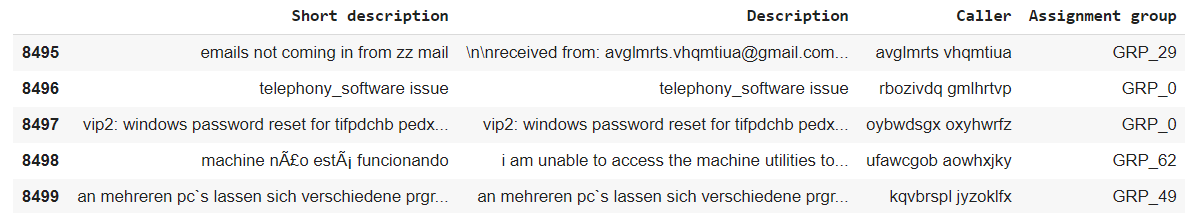
The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

* 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 investigated 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**

Top first records of our dataset :



Last five records of the dataset :



# Summary of the Approach to EDA and Pre-processing

## EDA

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;
  + Determine Optimal Factor Setting

Visually representing the content of a text document is one of the most important tasks in the field of text mining.

We have used SweetViz and Panda profiling to Visualize and analyse our dataset. Below are attached reports for and observations from reports are in next sections. The reports can be viewed in GitHub link - https://github.com/dishapalan02/AI-Enabled-IT-Service-Ticketing-tool/tree/main

**SweetViz**:

Graphical user interface, application

Description automatically generated

**Pandas Profiling:**

A picture containing chart

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Chart, bar chart

Description automatically generated

## **Findings**

From Above two reports we have below observations:

1. Shape of the data - { Rows : 8500, Columns : 4 }
2. Total features - 3

2.1. Short Description - Text

2.2. Description - Text

2.3. Caller - Text

1. Target Column - 1

3.1 Assignment Group - Categorical

1. There are 84 duplicate records in total. Strategy to handle duplicates and the approach taken is defined in the pre-processing section below.
2. New features are required or not needs to be analysed further and also to check if below hidden patterns can be figured out:

   A. Common Issues -> user can be trained if possible   
   B. Common Caller -> May be user needs training or help with hardware or software  
   C. To find if customer is happy with service or needs further imporvement and assistance

#ToDo To find if issue is controllable or not --> Check if possible .

Now let's have a look at individual features:

1. **Short description**

A. Total values - 8492 ( > 99% )

B. Missing values - 8 ( < 1% )

C. Distinct values - 7481 (88%)

D. Mostly occurring value - password reset ( 0.4% )

E. We can also see the number of times each value is being repeated

F. Max length of statement - 159

G. It contains:

 Characters -> Lowercase Letter, Punctuation,   
 Uppercase Letter, Decimal Number,   
 Math Symbol, Math Symbol,   
 Modifier Symbol, Other Number,   
 Other Symbol, Currency Symbol  
   
 Scripts -> Common (ASCII) and Latin

H. Point G indicates that we must translate the texts in the dataset based on the scripts as part of data pre-processing.

1. **Description**

A. Total values - 8499 (> 99%)

B. Missing values - 1 (< 1%)

C. Distinct values - 7817 (2%)

D. Mostly occurring value - it shows "the" (0.7%) but will analyse further after the removal of stop words. But we consider the next which is windows password reset (0.3%)

E. We can also see the number of times each value is being repeated

F. Max length of statement - 13001

G. It contains:

 Characters -> Lowercase Letter, Punctuation,   
 Uppercase Letter, Decimal Number,   
 Math Symbol, Math Symbol,   
 Modifier Symbol, Other Number,   
 Other Symbol, Currency Symbol  
   
 Scripts -> Common(ASCII) and Latin

H. Point G indicates that we have to translate the texts in the dataset based on the scripts as part of data pre-processing.

1. **Caller**

A. Total values - 8500 ( 100% )

B. Missing values - no missing value

C. Distinct values - 2950 ( 35% )

D. Mostly occurring value - bpctwhsn kzqsbmtp (10%)

E. We can also see the number of times each value is being repeated

F. Max length of statement - 30

G. It contains:

 Characters -> Lowercase Letter, Space Separator,   
 Uppercase Letter, Connector Punctuation  
   
 Scripts -> Common(ASCII) and Latin

H. Point G indicates that we have to translate the texts in the dataset based on the scripts as part of data pre-processing.

1. **Assignment Group**

A. Total values - 8500 ( 100% )

B. Missing values - no missing value

C. Distinct values - 74

D. Mostly occurring value - GRP\_0 (47% ~ nearly half of the data --> Hence we can say that target class is highly imbalanced, so needs a strategy to be employed to reduce the bias here)

E. We can also see the number of times each value is being repeated

G. This indicates we can merge few assignment groups with smaller percentage to reduce overall number of categories.

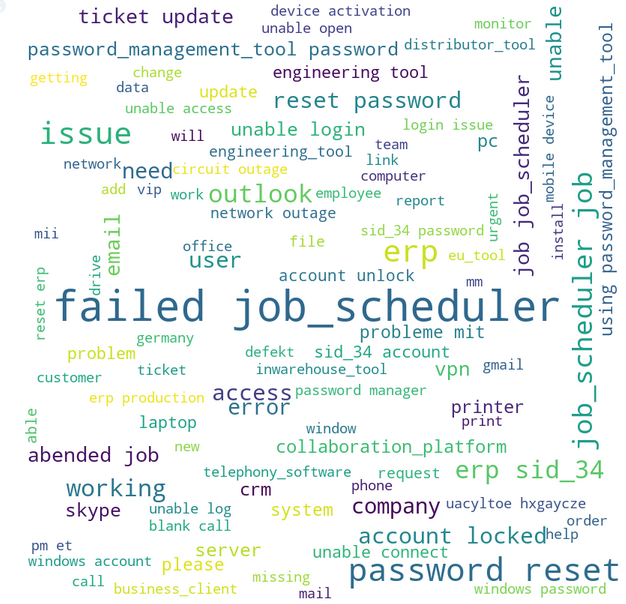
### Other findings

1. There are duplicates which needs to be tackled
2. There are mojibake texts in the description and short description which needs to be processed
3. There are texts belonging to different languages which needs translations
4. There are email ids, blank spaces, dates, numbers which needs to be processed
5. There are missing values to be treated

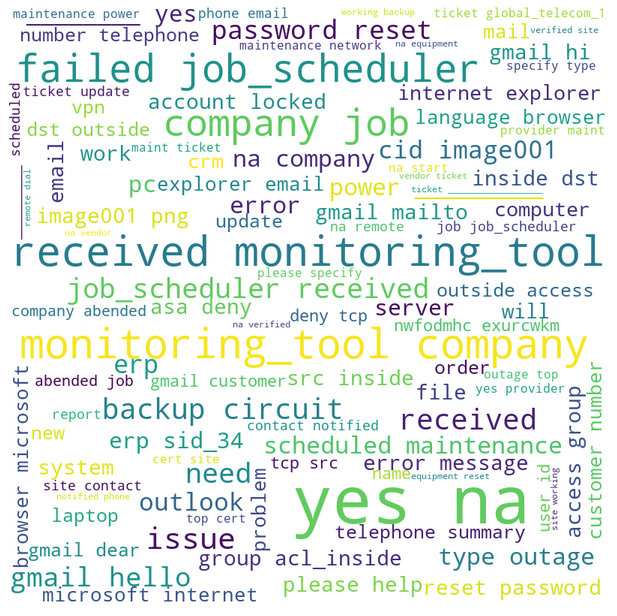
## Further Data Analysis

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

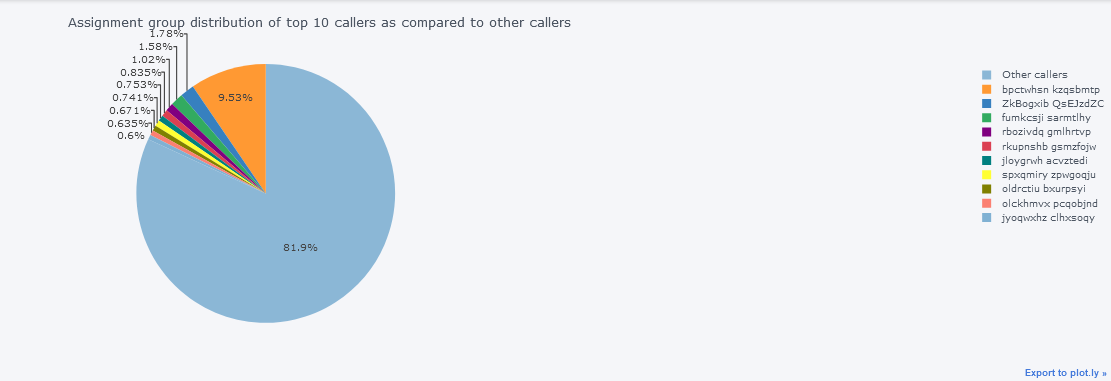
### Word cloud for Short description



### Word cloud for Description



### Assignment group distribution of top 10 callers as compared to other callers



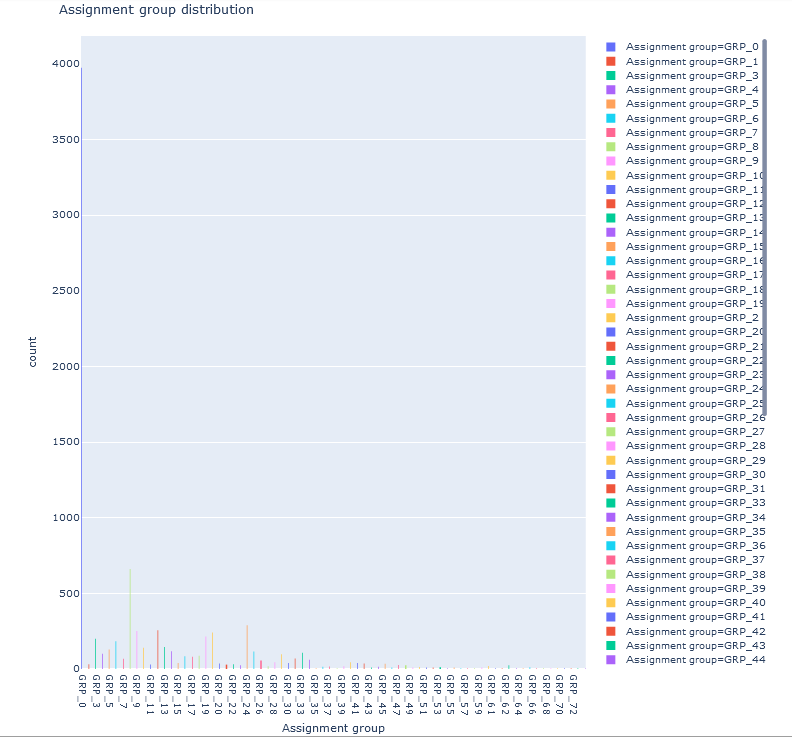
From the above we can say that 9.53% of requests are from Caller "**bpctwhsn kzqsbmtp**". May be this user needs some training or might be facing more issues compared to other callers.

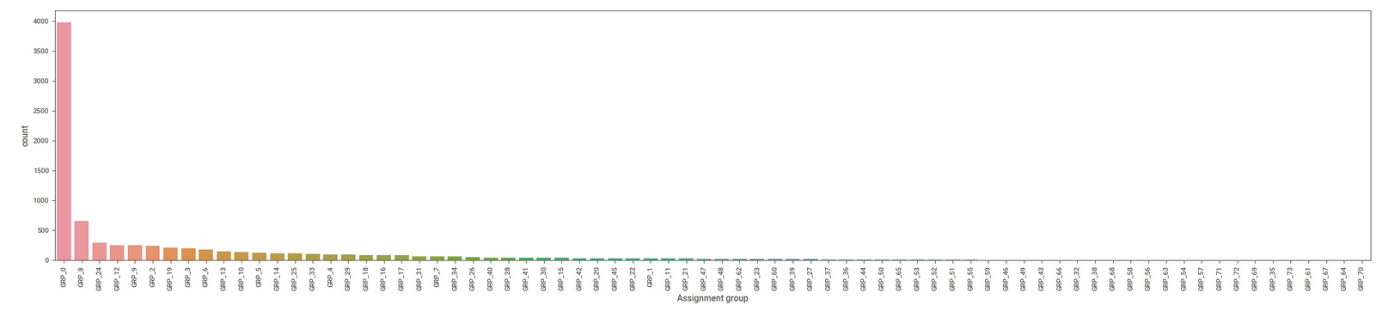
### Assignment group distribution amongst top 10 callers

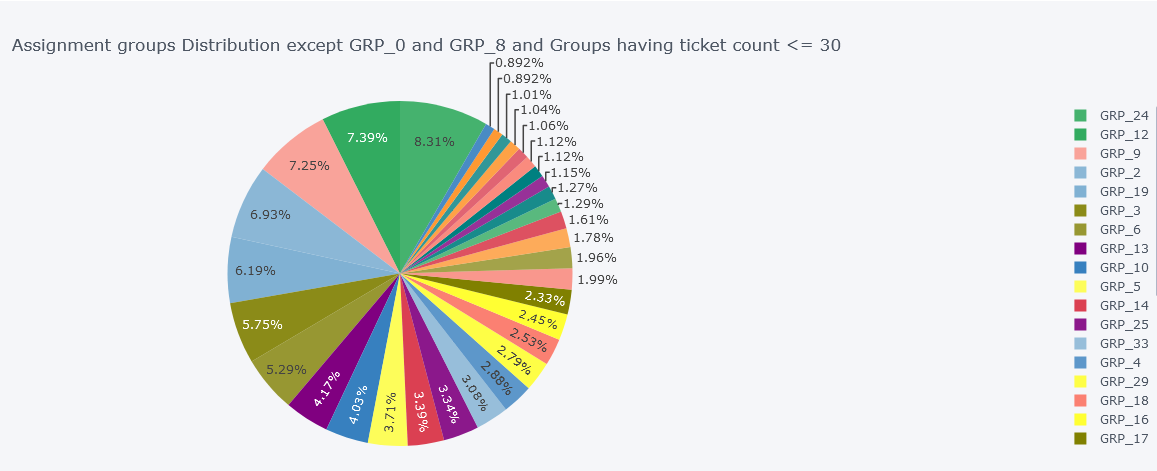


From this we can see the distribution amongst the top 10 callers which indicates that Caller "**bpctwhsn kzqsbmtp**" has created about 52.5% of issues.

### Assignment group distribution



  
From above we see that there is significant class imbalance with GRP\_0 being the majority class. Now let’s see the distribution of classes other than two majority classes GRP\_0 and GRP\_8 and also for those classes which have ticket count less than or equal to 30 (The number 30 is chosen using central limit theorem here)



We can use above figure to define the ranges to see which groups have tickets in below range. This will help us understand the most important groups to focus on. The ranges can be as below:

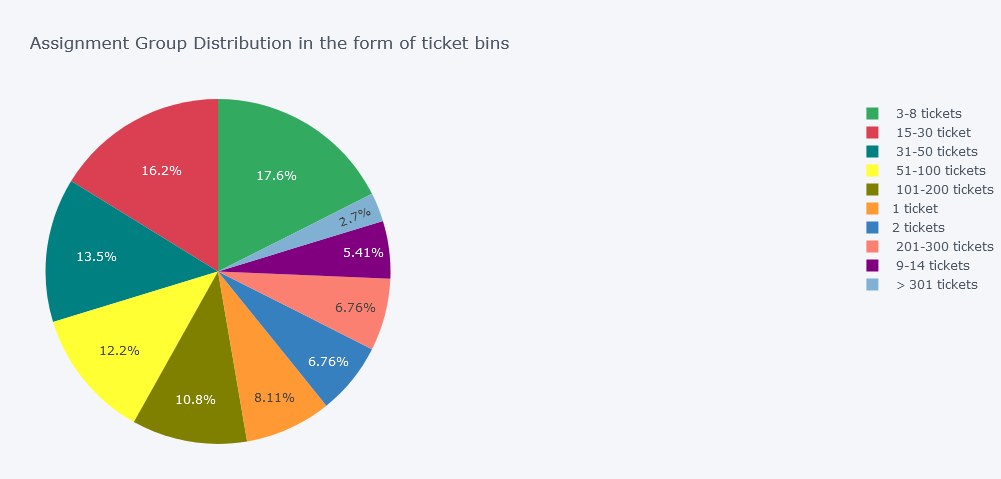
1. 1 ticket
2. 2 tickets
3. 3-8 tickets
4. 9-14 tickets
5. 15-30 tickets
6. 31-50 tickets
7. 51-100 tickets
8. 101-200 tickets
9. 201-300 tickets
10. more than 300 tickets

There are total of 40 Assignment groups which have tickets less than or equal to 30. Let’s see the if we can combine the assignment groups having few tickets and reduce the number of classes for classification.

For this we shall also check if the same kind of issues are handled by other assignment groups in further analysis with which will be a deciding factor to merge the classes.

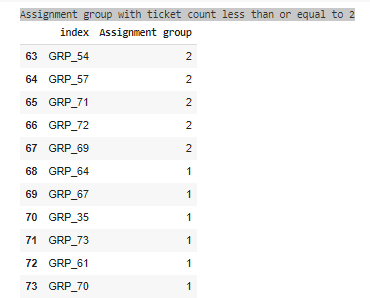
Let's see the assignment group distribution for the range of tickets we see above in the form of pie chart

Now we will see if we can ignore some classes or merge them into some other assignment group considering that other group has capability to resolve the ticket for this assignment group. For this we first divided the classes in some ticket bins and found the distribution as



From the above chart, we can see that Assignment group <= 2 tickets contributes to 14.87% i.e. (8.11 + 6.76%). Let’s analyse this further.

No of Assignment groups with less than or equal to 2 tickets 11



We see above that majority of assignment groups are one with tickets between 3 and 8 i.e. 17.6%. Also, from the above chart, we can see that Assignment group <= 2 tickets contributes to 14.87% i.e. (8.11 + 6.76%).

Also, after comparing the descriptions and understanding that other groups have capability to resolve same tickets. We found that we can either merge or ignore below assignment groups – mostly ignored based on above analysis. We will decide this later post feature engineering

A picture containing text, bird, plant

Description automatically generated

**DATA INCONSISTENCIES**

## Feature Engineering

***Null value check and treatment***

***Removal of Duplicates***

***N-gram analysis***

***Word cloud for each assignment group***

**IMPUTATION**

**Null value treatment strategies** : We see there are around 9 records with null values. Hence, we can use below strategies to replace null values:

1. Drop the records with null values
2. Replace null value (NaN) with empty string
3. Look for same description in any other record and then replace the corresponding shor description or in case of description has null value then search for same short description and replace description with corresponding description

We have gone with approach 2 that we will replace the null values with empty string. This is to make sure we do not loose descriptions with dropping and with approach 3 we might end up creating further duplicates inside.

* Hence our NULL/Missing value treatment replaces the NaN cells with just an empty string

**Checking null value after treatment** : There are no null values inside our dataset now

We also saw during analysis that there are multiple scripts inside and Latin characters. Let's see if we have texts from different languages. For this first we have created only single column for both short description and description and Named it as Ticket\_description

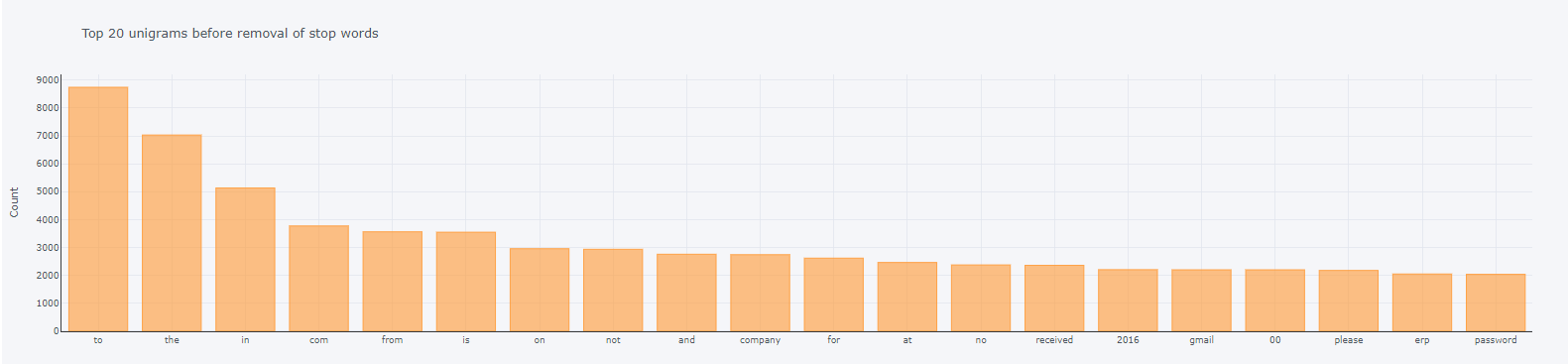
**Short description and Description are merged into one single feature.**

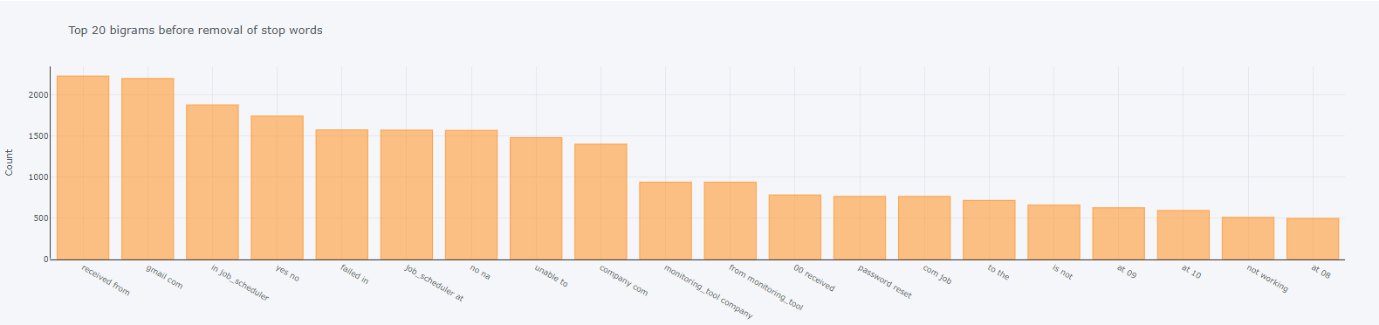
Approach is to combine the ‘Short Description’ & “Description’ field with the assumption that the vocabulary from the ‘Short Description’ could help in model accuracy. But found that combining these two fields could lead to combining non-english “short Description’ to ‘Description’ in english or vice versa. This posed a problem of many combined entries to be not translated. To further improve the translation process, we attempted to measure the impact of model accuracy on dropping the ‘Short Description. By dropping the ‘Short Description’ field we only observed a minor drop in model accuracy, ~1% drop & hence concluded to proceed with dropping ‘Short Description’.

**Word Clouds for each assignment group **

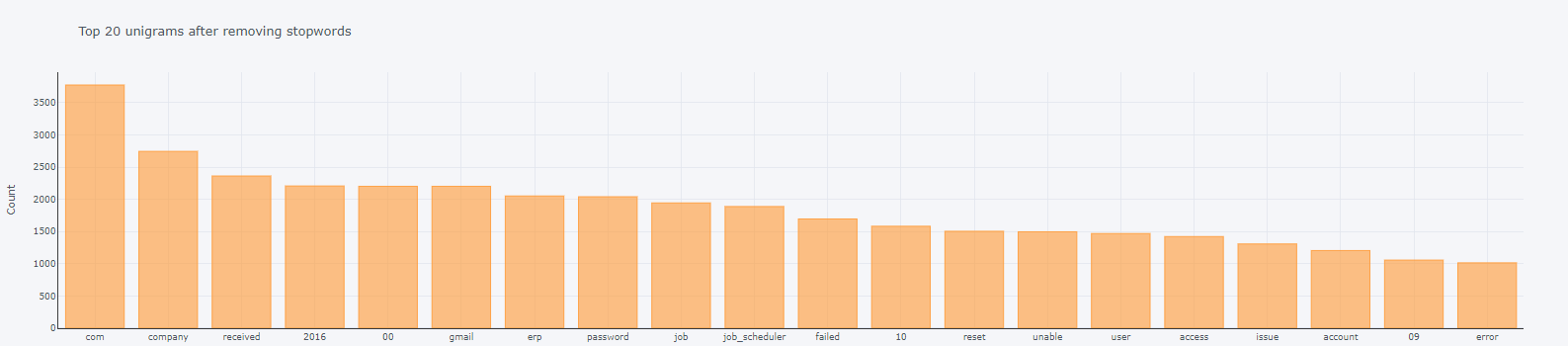
**N-gram Analysis**

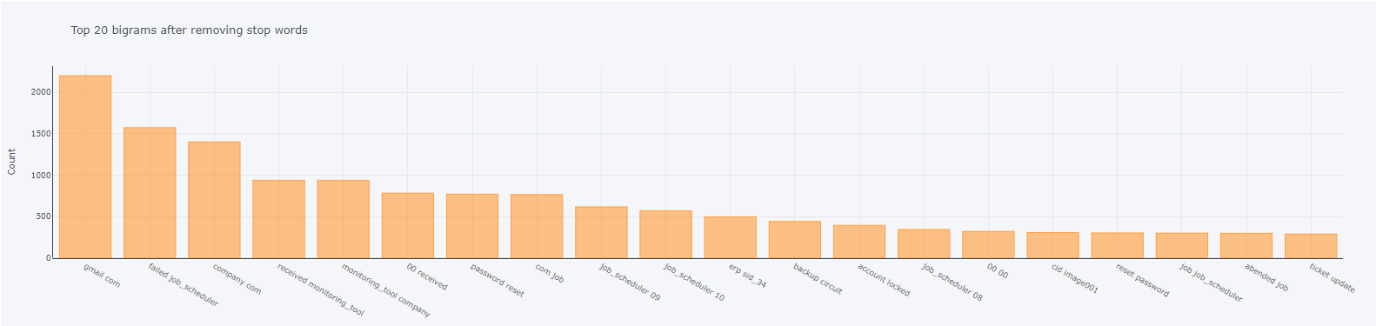
N-gram, Unigram and Bigram analysis N grams are used to describe number of words as observations as unigram means singly worded, bigram means 2-worded phrase, and trigram means 3-worded phrase. This analysis is performed before and after the removal of stop-words to look for any unexpected changes





As we can see that there are few English stopwords in the Ticket\_Description feature other than the default ones provided by NLTK library, decision is to extend the default list and then remove them from the Ticket\_Description feature and visualize unigrams and bigrams post that.





## Data Pre-processing

***Translation***

***Detecting different languages***

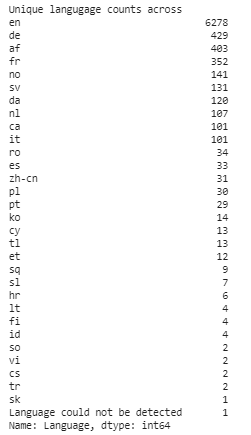
***Fix Encoding***

* **Fix encoding**

First verify mojibake text cleaning by taking 1 sample. We observe that the Mojibake texts were fixed. Further which all languages are there is dataframe is checked. langdetect library is used.

* **Detecting Language**

 Most of the descriptions are in English language followed by German

****

German language is found from the dataset. Attempted using many libraries such as googletrans, textblob, goslate, etc for translation of non-english entries to english, but found that all of them had size limitations & was unable to proceed with translation. To overcome this limitation, a wordlist of non-english words was formed from the dataset. All the rows from the Description column filtered using German wordlist have been translated to English language by passing to a Google translator.

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 include:

* 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 daily, 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

More precisely a function was used to call in chunks of Data and file is pickled for future use and then final set was prepared. We did translations before removing duplicates and pickled it so we are removing the translations also for duplicate rows and retaining others to merge later. After translations we have about 95% data which is translated correctly. There's still 5% data which are Common Nouns, etc and are left untranslated.

## Data Cleaning

A function clean\_text() has been created to clean up the unwanted information from initial observations. Further actions taken are to :

1. Convert the text to lowercase
2. Removing punctuation marks and other special characters
3. Removing numbers as converting them into corresponding words will result into dominance otherwise
4. Removing blank spaces, horizontal tab spaces, new line breaks with single space.
5. Removing stop words, parse terms and other particular words.
6. Removing email addresses as it will not add any significant value into the expected analysis.

***Removal of stop words***

***Lemmatization***

***Tokenization***

* **Stop words 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.
* **Stop Words (Removal) NLP**

Stop words are the most common words in any natural language. To analyze text data and build NLP models, these stop words might not add much value to the meaning of the document.

Consider this text string – “There is a pen on the table”. Now, the words “is”, “a”,

“on”, and “the” add no meaning to the statement while parsing it. Whereas words like “there”, “book”, and “table” are the keywords and tell us what the statement is all about.

* **Text Processing with Unicode:** Encode the string, to make it easier to be passed to language detection API.

**Lemmatization**

Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing.

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma.

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is like Stemming but it brings context to the words. So, it links words with similar meanings to one word.

Here we have preferred Lemmatization over Stemming because lemmatization does morphological analysis of the words.

* **Lemmatization with NLTK**
  + Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word.
  + Advantage of lemmatization is that it is more accurate. So if you’re dealing with an NLP application such as a chatbot or a virtual assistant where understanding the meaning of the dialogue is crucial, lemmatization would be useful. But this accuracy comes at a cost.
  + Because lemmatization involves deriving the meaning of a word from something like a dictionary, it’s very time-consuming. So most lemmatization algorithms are slower compared to their stemming counterparts.
* **Spell Check**

We have used pySpellchecker to perform spell check on the data. But there were few technical words which were also corrected with this function. Eg. Hostage for hostname, sky for skype, wife for wifi, etc. So a set of exceptional words have been loaded with such IT related technical words.

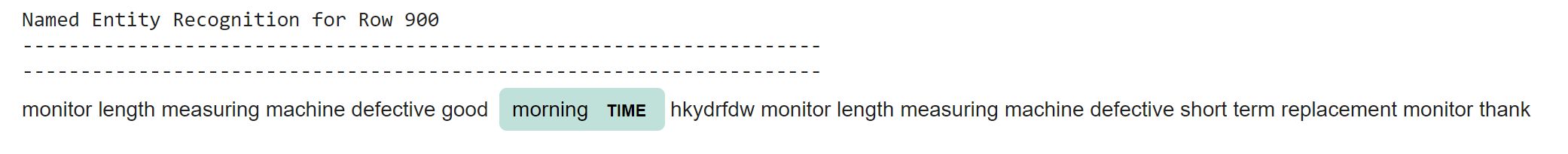
Found that performing spell check was a time consuming process. Although spell check helped in reduction of vocabulary size, it did not help in model accuracy improvement. Hence decided against applying spell check as it did not provide any substantial improvement to the whole process.

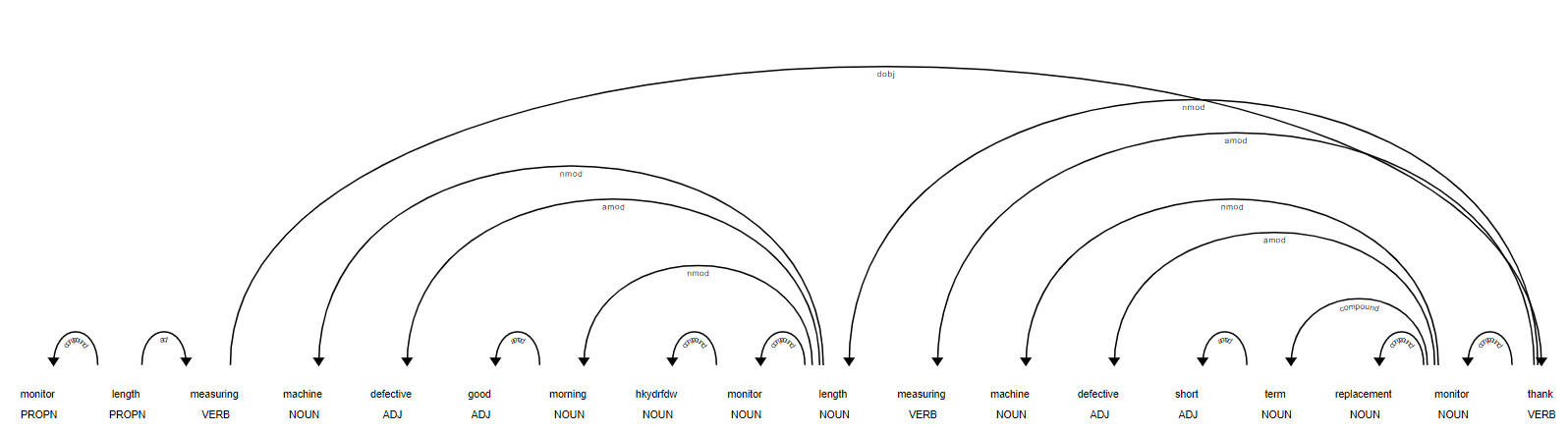
* **Tokenization**: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we want. A sentence tokenizer can be used to find the list of sentences and a Word tokenizer can be used to find the list of words in strings. Tokenization breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the model for the NLP.

o Raw text: I ate a burger, and it was good.

Tokenized text: [’ I’, ’ate’, ’a’, ’burger’, ‘,’, ‘and’, ’it’, ’was’, ’good’, ‘.’]

## NER and POS Tagging





Diagram

Description automatically generated

# Deciding Models and Model Building

Before building the model below steps were performed

Label Encoding

Splitting into train and test (80:20)

Treating class imbalance

* **Label Encoding**

o In label encoding in Python, we replace the categorical value with a numeric value between **0 and the number of classes minus 1.**

o To understand label encoding with an example, let us take COVID-19 cases in India across states. If we observe the below data frame, the State column contains a categorical value that is not very machine-friendly and the rest of the columns contain a numerical value. Let us perform Label encoding for State Column.

The Assignment group before and after class balancing is shown as below:

Chart, pie chart

Description automatically generatedChart, pie chart

Description automatically generated

## Models

As this is a classification problem where we need to classify the assignment groups using the ticket description and short descriptions. We have selected below traditional models and sequential models

As the target class is completely skewed, various models have been tried with the below set of datasets to compare each performance. Datasets used for each model are:

* Raw data with the target class without any sampling
* Resampled data where all the target classes are sampled with a count of 660. (Eg. Grp\_0 is down sampled and other groups are up sampled)
* Model with Two datasets: Model 1 with Grp\_0 & Model 2 with all other groups except Grp\_0 and Model 2 is resampled

### Traditional Models

Trying different model architectures by researching state of the art for similar tasks.

1. Multinomial Naïve Bayes Classifier

The Multivariate Event model is referred to as **Multinomial Naive Bayes**. **Naive Bayes** is based on **Bayes**' theorem, where the adjective **Naïve** says that features in the dataset are mutually independent. Occurrence of **one** feature does not affect the probability of occurrence of the other feature

multi\_nb\_clf = MultinomialNB(alpha=0.25)

multi\_nb\_clf = OneVsRestClassifier(multi\_nb\_clf)

1. SVC Classifier

Support vector machines (**SVMs**) are particular linear **classifiers** which are based on the margin maximization principle. They perform structural risk minimization, which improves the complexity of the **classifier** with the aim of achieving excellent generalization performance.

Classifier: OneVsRestClassifier(estimator=LinearSVC(C=1.0, class\_weight=None, dual=True,

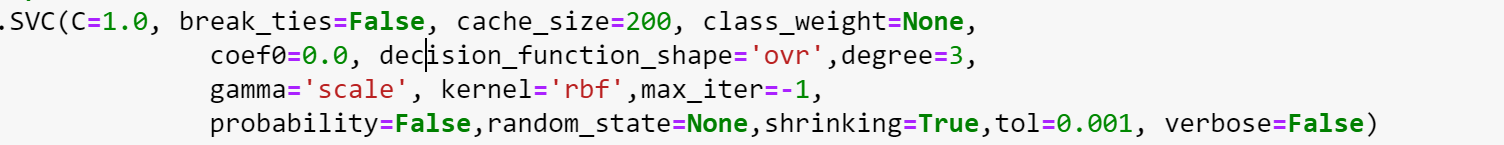
fit\_intercept=True, intercept\_scaling=1,

loss='hinge', max\_iter=1000,

multi\_class='ovr', penalty='l2',

random\_state=42, tol=0.0001,

verbose=0),

n\_jobs=None)

1. KNN Classifier

The k-nearest neighbours (**KNN**) **algorithm** is a simple, supervised **machine learning algorithm** that can be used to solve both **classification** and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows

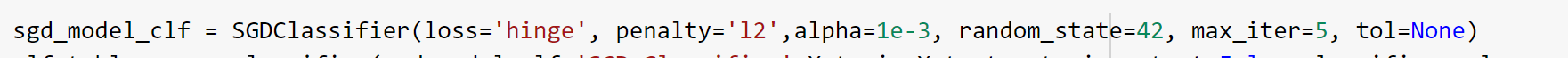
knn\_model\_clf = KNeighborsRegressor(algorithm='auto', leaf\_size=30, n\_jobs=None, n\_neighbors=3, p=2, weights='uniform')

knn\_model\_clf = OneVsRestClassifier(knn\_model\_clf)

1. SGD Classifier

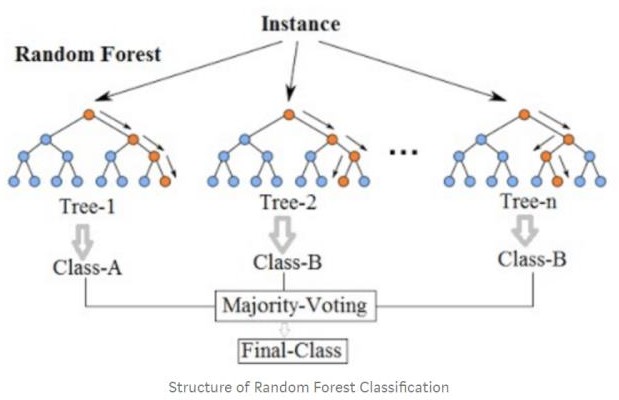
**SGD Classifier** is a linear **classifier** (SVM, logistic regression, a.o.) optimized by the **SGD**. You can think of that a machine learning model defines a loss function, and the optimization method minimizes / maximizes it.

sgd\_model\_clf = SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random\_state=42, max\_iter=5, tol=None)

clf\_table = run\_classifier(sgd\_model\_clf,'SGD Classifier',X\_train,X\_test,y\_train,y\_test,False,classifier\_columns,clf\_table)

1. Random Forest Classifier

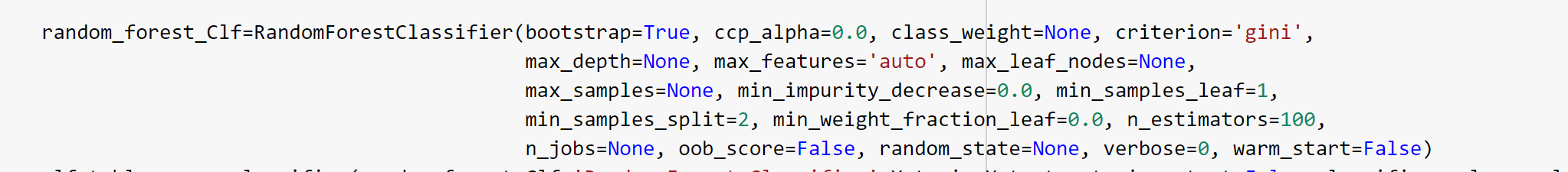
Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.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



random\_forest\_Clf=RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None, criterion='gini',                              max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

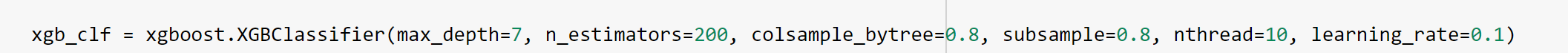
                                         max\_samples=None, min\_impurity\_decrease=0.0, min\_samples\_leaf=1,

                                         min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0, n\_estimators=100,

 n\_jobs=None, oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

1. XGBOOST

**XGBoost** is an **algorithm** that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. **XGBoost** is an implementation of gradient boosted decision trees designed for speed and performance.

xgb\_clf = xgboost.XGBClassifier(max\_depth=7, n\_estimators=200, colsample\_bytree=0.8, subsample=0.8, nthread=10, learning\_rate=0.1)

Metrics for these models along with the observation on how changing different hyper parameters leads to change in the final evaluation metric. We got below training and test accuracies and F1 score (“Training duration??”)

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | **Train Accuracy** | **Test Accuracy** | **F1 Score** |
| Multinomial NB Classifier | 69.57% | 59.89% | 70.91% |
| SVC Classifier | 90.01% | 66.72% | 71.98% |
| KNN Classifier | 76.33% | 62.62% | 65.60% |
| SGD Classifier | 72.95% | 62.38% | 70.78% |
| Random Forest Classifier | 95.36% | 63.75% | 71.36% |
| XGBOOST | 92.70% | 64.52% | 69.78% |

### Sequential Models

1. CNN

A Convolutional Neural Network (ConvNet/**CNN**) is a **Deep Learning** algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

cnn\_model = build\_CNN(word\_index\_g,embeddings\_matrix)

clf\_table = run\_classifier(cnn\_model,'CNN',X\_train\_g,X\_test\_g,y\_train,y\_test,True,classifier\_columns,clf\_table,preTasks=False)

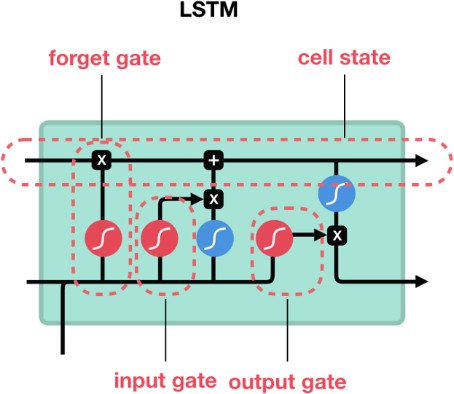


1. LSTM

**Bi-directional LSTM Model**

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.

In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.



lstm\_model = build\_LSTM(word\_index\_g,embeddings\_matrix)

clf\_table = run\_classifier(lstm\_model,'LSTM',X\_train\_g,X\_test\_g,y\_train,y\_test,True,classifier\_columns,clf\_table,preTasks=False)



|  |  |  |  |
| --- | --- | --- | --- |
| **Classifier** | **Train Accuracy** | **Test Accuracy** | **F1 Score** |
| CNN |  |  |  |
| LSTM |  |  |  |

**Treating Class Imbalance :**

Following strategies were tried to treat class imbalance

1. Random Over sampler and under sampler

2. SMOTE

3. Random over sampler

4. Resampler

After trying all the above startegies we selected random over sampler as it was giving good accuracy. With SMOTE we faced the issue due to large dataset and SMOTE using KNN in the end it was always giving us error.

# How to improve your model performance?

* Still we are working with balancing the classes (We are also trying SMOTE for this)
* We want to use PCA as there seems to be too much difference between training and test accuracies
* Plotting the accuracy graphs is in progress

# Comparison to benchmark

From the given problem description, we could see that the existing system is able to assign 75% of the tickets correctly.

So our objective here is to build an AI-based classifier model to assign the tickets to right functional groups by analysing the given description with an accuracy of at least 85%.

From the prediction results we see that the GRU model based on the resampled data can achieve an accuracy of 91.24% which is above our benchmark.

# Implications

Although this model can classify the IT tickets with 91.24% accuracy, to achieve better accuracy in the real world it would be good if the business can collect additional data around 300 records for each group.

# Limitations

As part of Data pre-processing, we had grouped all assignment groups with less than 10 entries as one group (misc\_grp) which had reduced the Target class from 74 to 50 groups. While applying this model in the real world there could be additional intervention required to classify the tickets if it has been classified as misc\_grp by our model. Since the number of elements reported under misc\_grp are less, we expect this intervention to be done less often.

# Closing Reflections

We found the data was present in multiple languages and in various formats such as emails, chat, etc bringing in a lot of variability in the data to be analyzed. The Business can improve the process of raising tickets via a common unified IT Ticket Service Portal which reduces the above-mentioned variability. By doing this, the model can perform better which can help businesses to identify the problem area for relevant clusters of topics.

**Sources :**

1. https://towardsdatascience.com/predict-it-support-tickets-with-machine-learning-and-nlp-a87ee1cb66fc
2. ??

**Libraries Used :**

1. Pandas profiling is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates a report with all the information easily available
2. numpy : Library needed for numerical calculations
3. matplotlib.pyplot : Needed for graphs and charts during data analysis
4. seaborn : Needed for graphs and charts during data analysis
5. plotly : Visualization library to generate graphs
6. cufflinks : Required to link plotly to pandas dataframe and add the iplot method
7. plotly.io
8. Sweetviz is an open-source Python library that generates beautiful, high-density visualizations to kickstart EDA (Exploratory Data Analysis) with just two lines of code. Output is a fully self-contained HTML application. Needed for EDA nad generating report
9. Sklearn
10. Tensorflow
11. Keras
12. ftfy : Fixes Text For You :- Library to detect and fix Mojibakes.It fixes Unicode that’s broken. The goal of ftfy is to take in bad Unicode and output good Unicode.
13. langdetect : Detects language of a given text
14. translate-api
15. wordcloud : Needed to display word clouds
16. pandas\_profiling : Needed for EDA and generting reports