**Final Report for**

AIML Online Capstone Project

(AUTOMATIC TICKET ASSIGNMENT)



:

**Sanket Shetye**

## **INDEX**

[1. **Summary of problem statement, data and findings**](#_heading=h.1t3h5sf)……………………………………………

[**2. Overview**](#_heading=h.44sinio) **……………………………………………………………………………………………...**

[**3. Step-by-step walk through the solution**](#_heading=h.2jxsxqh)**………………………………………………………….**

[**4. Model evaluation**](#_heading=h.17dp8vu)**……………………………………………………………………………………..**

[**5. Comparison to benchmark**](#_heading=h.3rdcrjn)**…………………………………………………………………………**

[**6. Visualizations**](#_heading=h.26in1rg)**…………………………………………………………………………………………**

[**7. Implications**](#_heading=h.z337ya)**…………………………………………………………………………………………...**

[**8. Limitations**](#_heading=h.35nkun2)**………………………………………………………………………………………….…**

[**9. Closing Reflections……………………………………………………………………………….…**](#_heading=h.1ksv4uv)

## **1. Summary of problem statement**

Incident management systems are a key process in business operation continuity. Objective of Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact.

Incident Management process is carried out by L1, L2 and functional groups(L3).

incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within the IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams). This team reviews the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve.

This review process is manual and has following problems:

**Delay in routing incidents:**

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.

**Additional costs and efforts:**

Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

**Poor customer service:**

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.

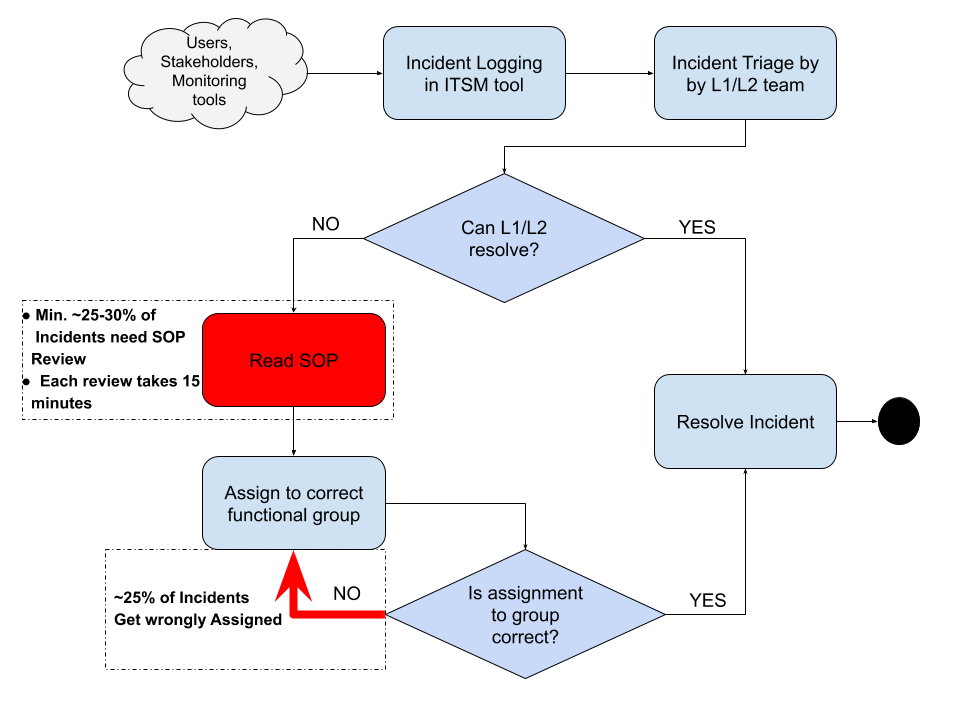
As we had to predict the correct incident group for a given incident out of multiple assignment groups, it's clear that this is a **supervised multi-class classification problem**.

We also made following assumptions:

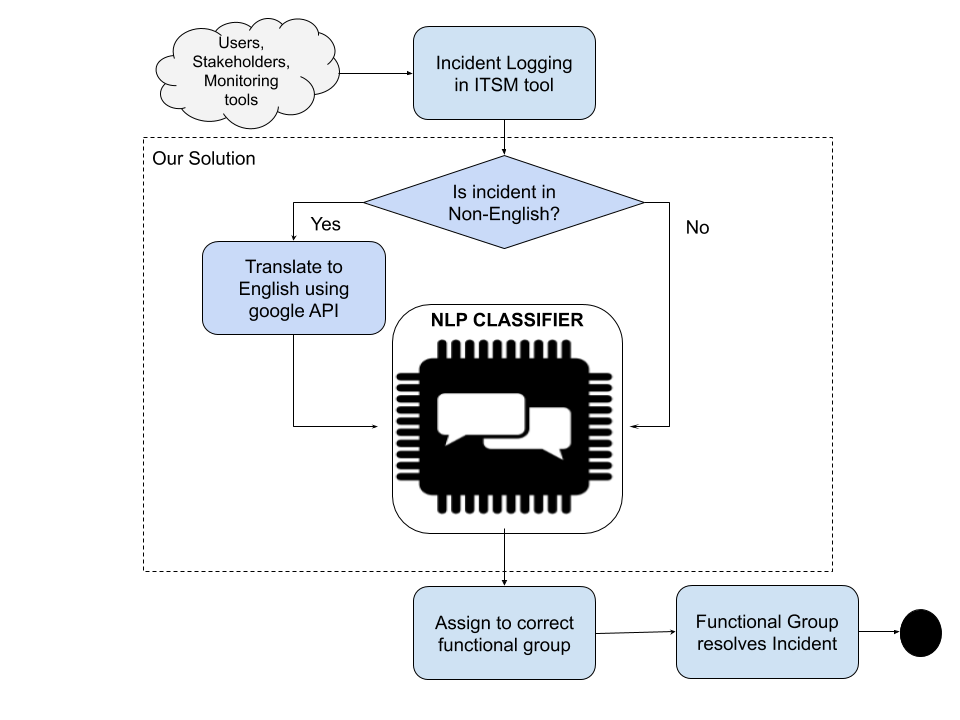
* Assigned group in given dataset is accurate and will be used for learning
* L1 incidents get assigned to grp 0 rest are not L1 incidents.

## **2. Overview**

**Current Process:**

****

**Proposed Process:**

****

**Data Dictionary**

Data is given in the form of excel having 8500 rows with 4 columns as mentioned below

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **No. of entries** | **Data Type** | **Description** |
| Short description | 8492 | object | Title of reported incident |
| Description | 8499 | object | Description of reported incident, many descriptions are in the form of email received from callers |
| Caller | 8500 | object | Entity which reported incident can be various Business and IT Users, End Users/ Vendors or integrated monitoring systems and tools |
| Assignment group | 8500 | object | Target column indicating group which resolved the incident |

**Missing Data Analysis**

The missing value were found using **df.isna().sum()**

**Short description 8**

**Description 1**

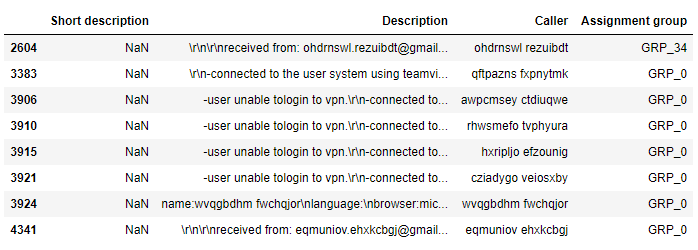
**Caller 0**

**Assignment group 0**

**dtype: int64**

As we can see above 8 records for the short Description and 1 record for Description are missing values

Rows Missing short description:

**

Rows missing description:



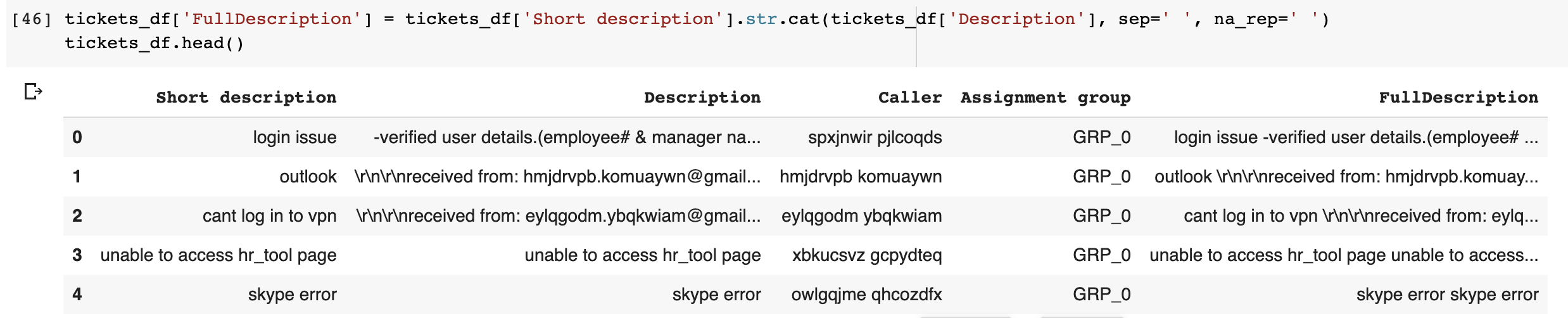
* We combined short description and description text.

***Preprocessing steps***

Pre-processing is an essential, wherein the text is to be cleaned up to bring it to a required format for the information extraction models. This includes normalizing different tenses of words, normalizing synonyms, spell correction etc. There are different ways to preprocess your text. We will be implementing all these.

***Data cleaning***

As described in the EDA step there are 8 missing short descriptions and 1 description value is missing. We will be combining these two columns to have a full description.

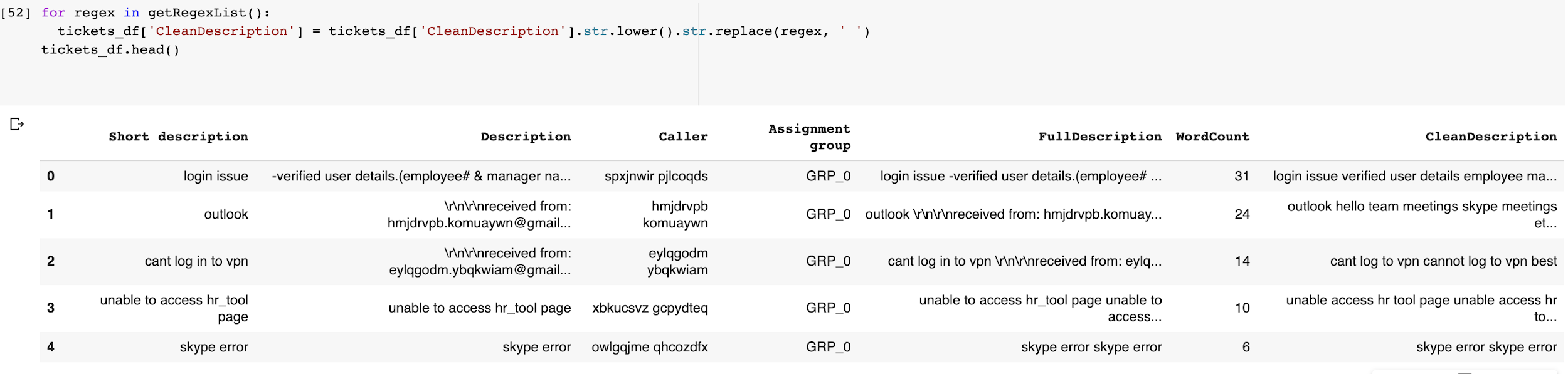


1. **Lowercasing** ALL your text data, although commonly overlooked, is one of the simplest and most effective forms of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with consistency of expected output.
2. **Regex**: The data has to be cleaned by converting all data into lower case and adding a regex list as required to remove unnecessary text such as https:? , numbers, special characters, single alphabets, email tags and email IDs , user names etc.

We will create our reg expression list after analyzing the data.



We will put the result of removing the word into new column



1. **StopWords** are the most common words in NLP for ex is, as the, a, an etc. For the purpose of analyzing text data and building NLP models, these StopWords might not add much value to the meaning of the document. For tasks like text classification, where the text is to be classified into different categories, StopWords are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text. Up on removing stopwords, dataset size decreases and the time to train the model also decreases. Removing stopwords helps improve the performance as there are fewer and only meaningful tokens left. Thus, it increases classification accuracy.

We also need to remove words which are occurring frequently but are not contributing to classification. We will also remove them like hello, bye, etc.

Caller name also appears in the description very frequently, we will remove caller from the description to keep words pertinent to issue only. NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text pre-processing. **NLTK has a list of StopWords stored in 16 different languages.**

## **Algorithms:**

**Logistic Regression:** A linear classifier, mostly similar to traditional linear regression, but that fits the output of the logistic function.

**(Multinomial) Naive Bayes:** A Bayesian model that assumes total independence between features. In our case, this means that P("Account") is unrelated to P("login"), which of course is a terrible assumption.

**Random Forest**: Random Forest (as the name might suggest) is the [ensemble](https://www.kaggle.com/wiki/Ensembling) of a large number of decision trees, each trained on a random subset of the input features. They work well when complex feature-relations are involved and are relatively robust to overfitting.

**Support Vector Machines (SVM):** SVM is basically used for binary classification problems in which the given data point belongs to either positive or negative class. In the context of document classification when the documents are represented as data points in a high dimension space, there exist many hyper planes that separates the data points into positive and negative instances. SVM algorithm tries to find the optimum hyper plane with the maximum margin ξ from positive and negative instances

To further analyze the performance of the incident ticket classifier model, ensemble techniques such as **Bagging** and **boosting** are applied on the chosen baseline classifier models.

**Bagging**: It is an ensemble technique which involves building the multiple models of the same type in parallel using the random sub samples of the original data set and then combining the predictions of these models using averaging or majority voting technique.

**Boosting**: Boosting is an ensemble technique that involves creating a sequence of models from the random samples of the training data which tries to rectify the mistakes of the previous predictor classifiers in the sequence. If an instance was misclassified, it tries to increase the weight of this instance and the misclassified instances were given more weightage in the next model.

**LSTM**: For the LSTM model we have used glove encoding. We also tried with Bidirectional LSTM with and without time distribution.

## **3. Step-by-step walk through the solution**

We will cover our approach towards solution by following along usual data science life cycle steps in below diagram

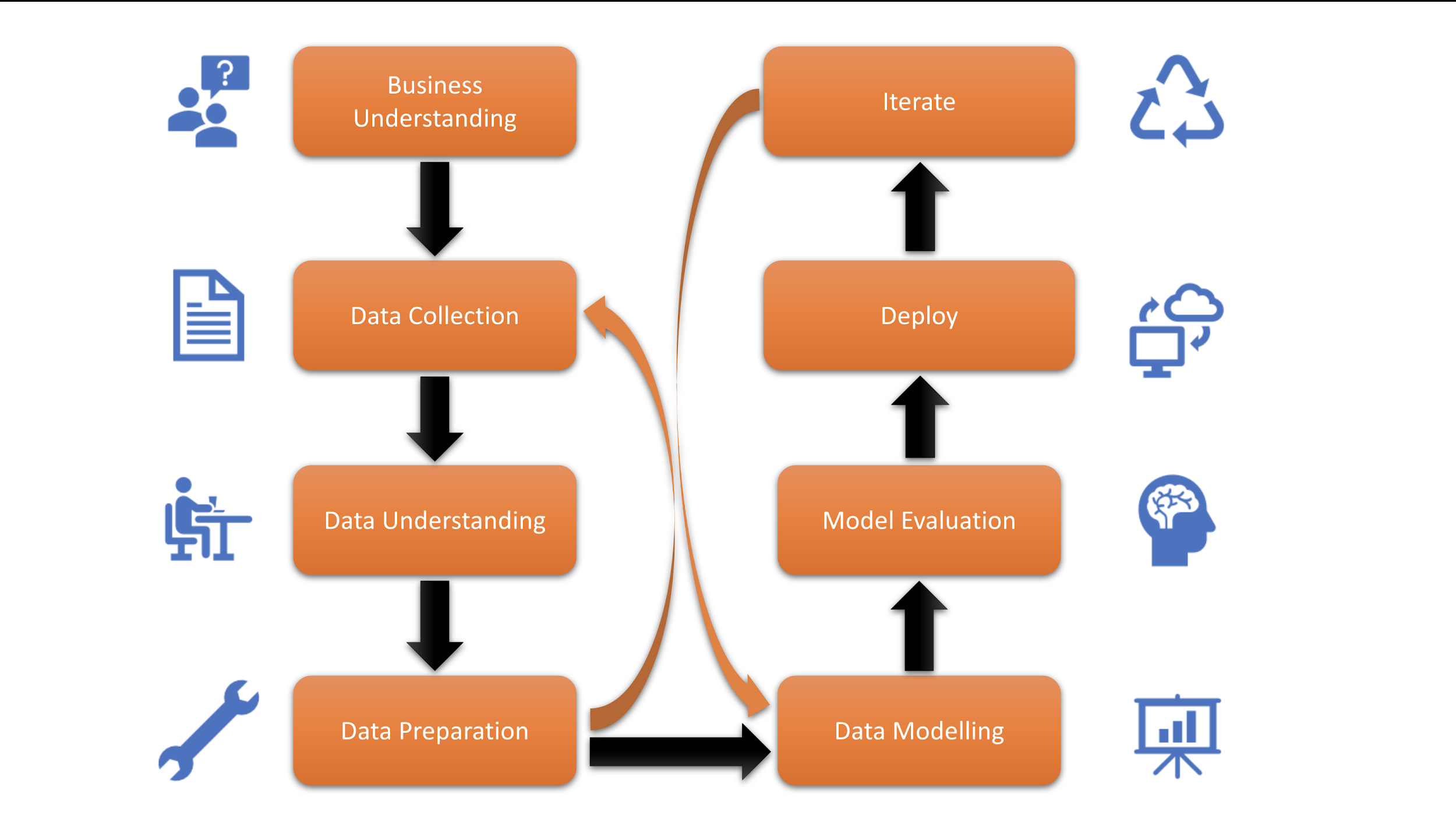


Fig (3). Data science project life cycle steps

1. **Business Understanding**

The problem statement described a very common situation of incorrect or time-consuming routing of tickets to appropriate assignment groups in the organization. As all members in our group were aware of ITSM workflows, we had a good understanding problem to be resolved. We agreed that a software system which takes the incident's description as input, trains itself on existing data using natural language processing techniques, and uses trained models on incoming description to predict the correct group.

This solution design is described in image in section 2.

We decided to go for NLP based solution as it allows following advantages -

* NLP is a sophisticated software approach to understand slang, mispronunciations, misspellings, and other variants in language
* NLP solution will be always available
* It will be faster than humans at classifying text

1. **Data collection**

Dataset for this project was shared by great learning. We used that.

1. **Data preparation**

**Step 1:**  Initial analysis of the unstructured training data for the presence of any class imbalance issues, unwanted features and any other noisy data. After carefully exploring data. We found that main challenges with data were an imbalanced distribution of incidents among groups and incidents in other languages than ‘English’.

The detailed Exploratory data analysis can be found in [**visualization**](#_heading=h.26in1rg) section.

**Step 2:** Handling missing data

We observed that short description and long description were missing some values. We decided to combine the columns.

**Step 3:** Correcting values

We found that the dataset had descriptions in multiple languages so we translated non-English descriptions to English.

**Step 4:** Removing duplicates

We removed the duplicate words.

**Step 5**: for each of the incident ticket description in Training and Test do

1. Extract all the words or features of incident description using tokenization.
2. Remove the stop words from the tokenized words.
3. Remove the special characters, features like date and time using the appropriate pattern recognizers.
4. Remove all the functional words using Parts of Speech (POS) tagging.
5. Remove the entities like user name, phone numbers and email ids using the appropriate pattern recognizers.
6. Perform the stemming of words using a porter stemmer.

**Step 6:** Using the pre-processed incident descriptions in Training, construct the Feature vector representation for each ticket instance based on the Bag of words model.

**Step 7:** Reduce the feature set using the tf idf as a part of dimension reduction.

**Step 8**: Build the classifier model using the chosen base classifiers and ensemble of classifier models.

**Step 9**: Evaluate the different classifiers performance using various performance evaluation metrics and the best performed model is chosen as the predictive model. We decided to use **F1 score** as a key metric to rate our algorithms as both precision and recall is equally important for our classification. Misclassifying incidents are equally costly in both false positive and false negative scenarios for us.

**Step 10:** Feature engineering

We added word count of cleaned description as an added feature.

**Step 11:** For Deep Learning Model we used Glove Embedding.

Initial analysis also revealed that the raw dataset had a huge amount of unwanted features. The details of the features or words present in the training incidents before and after performing data pre-processing step are detailed below

Total Word Count Before Cleaning: **310918**

Clean Word count After Cleaning: **168607**

1. **Data Modelling**

To prepare data suitable to algorithms we have performed various steps.

- The training test split was 80:20 and the validation split were 25% of train data.

- Our target column is the Assignment group that is label encoded using label encoder.

**Approach 1:**

We started with Logistic Regression. Initially, we started with 95% of the data that include groups having 30 or more tickets.

1. Supervised classification algorithms are implemented to evaluate performances comparatively. Commonly used term weighting methods are used to convert text into numerical form. The classification performance varies directly related to the machine learning algorithm, the weighting method and the dataset.

Results with assignment groups having 29 or more incidents

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm Name** |  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Linear SVM | TF-IDF | 0.703 | 0.683 | 0.703 | 0.649 |
| **Logistic Regression** | **TF-IDF** | **0.693064** | **0.671427** | **0.671427** | **0.670076** |
| Bagging with logistic | TF-IDF | 0.672 | 0.630 | 0.672 | 0.626 |
| Random Forest | TF-IDF | 0.663551 | 0.676548 | 0.663551 | 0.620025 |
| LSTM | Glove 200d | 0.659287 | 0.634747 | 0.659287 | 0.632272 |
| Logistic Regression with balanced | TF-IDF | 0.655189 | 0.686104 | 0.686104 | 0.658872 |
| Naive Bayes | TF-IDF | 0.609 | 0.533 | 0.609 | 0.499 |
| LSTM- Bidirectional with Time distributed(glove) | Glove 200d | 0.594096 | 0.476811 | 0.594096 | 0.524744 |
| Decision Tree | TF-IDF | 0.590 | 0.561 | 0.590 | 0.567 |
| KNN | TF-IDF | 0.490900 | 0.644951 | 0.490900 | 0.521481  z |
| LSTM- Bidirectional without Time distributed(glove) | Glove 200d | 0.674662 | 0.654896 | 0.674662 | 0.647092 |

1. To analyze if the lower performance of the LSTM was due to lack of data, the linear model and LSTM were trained with varying amounts of training data used.

Result with assignment groups having 50 or more incidents

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm Name** |  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Logistic Regression** | **TF-IDF** | **0.734747** | **0.721630** | **0.721630** | **0.716045** |
| Logistic Regression with balanced | TF-IDF | 0.691830 | 0.733644 | 0.733644 | 0.703430 |
| LSTM |  | 0.710594 | 0.706380 | 0.710594 | 0.686004 |
| Random Forest | TF-IDF | 0.699586 | 0.722811 | 0.699586 | 0.664797 |
| KNN | TF-IDF | 0.590486 | 0.697168 | 0.590486 | 0.577017 |
| LSTM- Bidirectional with Time distributed | Glove 200d | 0.627907 | 0.503161 | 0.627907 | 0.540021 |
| LSTM- Bidirectional without Time distributed | Glove 200d | 0.731266 | 0.697479 | 0.731266 | 0.705236 |

1. Performance of the linear model seems to be saturated after a few more training samples while the LSTM still improves slightly after this point, indicating that the LSTM could potentially perform better given more data.
2. Accuracy further improved by removing the caller name from description.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm Name** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Logistic Regression** | 0.722337 | 0.711079 | 0.711079 | 0.700592 |
| Logistic Regression with balanced | 0.673733 | 0.702686 | 0.702686 | 0.677891 |
| LSTM | 0.718346 | 0.697066 | 0.718346 | 0.694675 |
| Random Forest | 0.698552 | 0.709556 | 0.698552 | 0.666425 |
| KNN | 0.472596 | 0.798180 | 0.472596 | 0.550793 |
| LSTM- Bidirectional with Time distributed | 0.631783 | 0.512073 | 0.631783 | 0.559640 |
| **LSTM- Bidirectional without Time distributed** | **0.731266** | **0.697479** | **0.731266** | **0.705236** |

1. We chose GloVe over word2vec as GloVe utilizes local context-based learning (like word2vec) along with global text statistics (as in classical vector space models such as Latent Semantic Analysis)

We also tried another two approaches of model building but did not pursue them further as the accuracy was lesser than above models.

**Approach 2:**

We used a glove 6B file to create word embeddings of our corpus. We decided to use two level classifiers. First level classifier will be binary and it will determine if a given incident is L1 or not.

Second level classifier will be a multiclass classifier which will classify incoming incidents among groups other than L1.

Combining both level 1 and level 2 classifiers and running it against test samples gave us an average F1 score of 0.58 when considering tickets with incidents groups 50 or above. We decided to not pursue this approach further as other approaches gave more accuracy with considering tickets with incident group 30 or above.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm Name** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Random Forest | 0.569 | 0.597 | 0.569 | 0.576 |

**Approach 3:**

The key to building a good machine learning model is the data it is trained on. Therefore, it is imperative that the training data be clean and balanced. There are several approaches to achieve this.

We leverage augmentation generating synthetic tickets to balance the dataset. We used nlpaug.

We made use of synonym replacement and antonym replacement to generate synthetic tickets

**Synonym Replacement:** Randomly choose *n* words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random. We decided to not to pursue this approach further as accuracy score without augmentation gave us more accuracy. Also some of the synthetic incidents generated drastically misleading incident descriptions.

We have added separate python notebooks for approach 2 and 3.

## **4. Model Evaluation**

As we have seen the data, we got highly imbalanced. There are two common approaches to tackle the problem of extremely imbalanced data. One is based on cost sensitive learning: assigning a high cost to misclassification of the minority class, and trying to minimize the overall cost.

The other approach is to use a sampling technique: Either down-sampling the majority class or over-sampling the minority class, or both.

As we have seen the data we got is highly imbalanced.There are two common approaches to tackle the problem of extremely imbalanced data. One is based on cost sensitive learning: assigning a high cost to misclassification of the minority class, and trying to minimize the overall cost.

The other approach is to use a sampling technique: Either down-sampling the majority class or over-sampling the minority class, or both.

Our final model chosen based on the best F1 score was the Bidirectional LSTM model we have described our selection process in depth in section [comparison to benchmark](https://docs.google.com/document/d/1TPGG7T_n9U2Xi9WrE63auD8cFTPmQ4V-lcgcpmi2DWk/edit#heading=h.fni5igsoqg1).  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_9 (Embedding) (None, None, 200) 1976000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_9 (Bidirection (None, None, 128) 135680

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

global\_max\_pooling1d (Global (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_12 (Dense) (None, 24) 3096

=================================================================

Total params: 2,114,776

Trainable params: 2,114,776

Non-trainable params: 0

**Figure: Summary of Bidirectional LSTM model**

We ran this model for 15 epochs with batch size of 100.

Our early stopping callback function had a min\_delta parameter set to 0.01 and patience was set 3. We monitored validation loss for early stopping.

Below screenshot shows our model in action. Model stopped fitting after 10 epochs at 0.7130 validation accuracy.



Classification report :

**precision recall f1-score support**

**0 0.82 0.93 0.87 820**

**1 0.62 0.26 0.37 19**

**2 0.70 0.59 0.64 54**

**3 0.60 0.52 0.56 29**

**4 0.27 0.17 0.21 18**

**5 0.38 0.50 0.43 10**

**6 0.90 0.95 0.93 20**

**7 0.50 0.42 0.46 19**

**8 0.34 0.26 0.30 42**

**9 0.66 0.45 0.54 55**

**10 0.75 0.94 0.83 51**

**11 0.38 0.38 0.38 16**

**12 0.33 0.14 0.20 7**

**13 0.50 0.33 0.40 15**

**14 0.22 0.15 0.18 39**

**15 0.00 0.00 0.00 15**

**16 0.28 0.22 0.24 23**

**17 1.00 0.07 0.13 14**

**18 0.50 0.17 0.25 18**

**19 0.76 0.52 0.62 31**

**20 0.73 0.44 0.55 36**

**21 0.82 0.43 0.56 21**

**22 0.61 0.89 0.73 127**

**23 0.40 0.17 0.24 48**

**accuracy 0.73 1547**

**macro avg 0.55 0.41 0.44 1547**

**weighted avg 0.70 0.73 0.70 1547**

## **5. Comparison to Benchmark**

No model is universally superior to all other models, and the performance of a model greatly depends on the data used and the nature of the classification/prediction problem solved. This is known as the [No Free Lunch Theorem](https://en.wikipedia.org/wiki/No_free_lunch_theorem).

Our benchmark model was run on all data and without hyper-tuning the parameters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm Name** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Linear-SVM | 0.675294 | 0.651104 | 0.675294 | 0.631339 |
| Random Forest | 0.654510 | 0.616019 | 0.654510 | 0.604028 |
| Decision Tree | 0.530196 | 0.521740 | 0.521740 | 0.521041 |

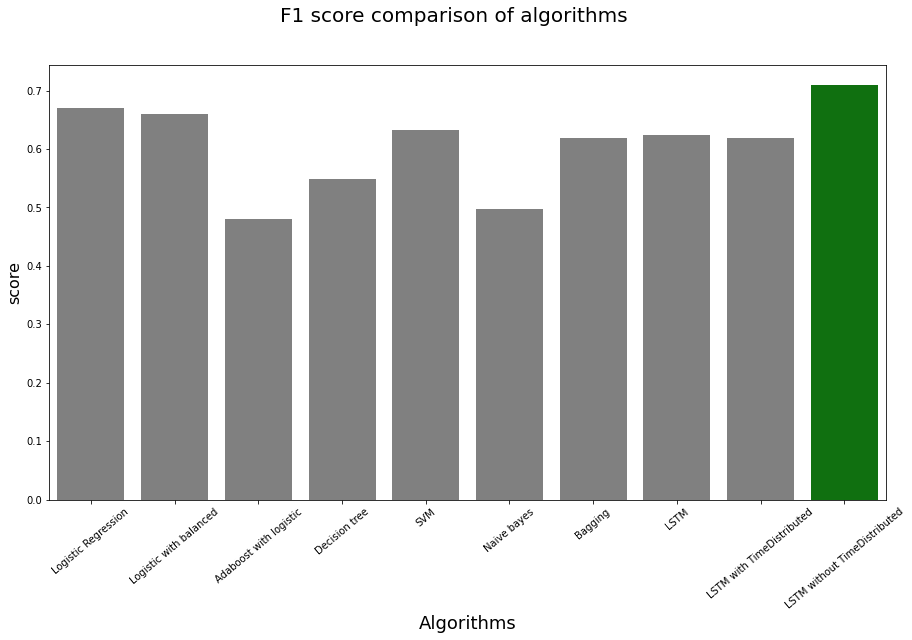
We have performed hyperparameter search for each model: tuning each of its "knobs' ' (number of trees for Random Forest, penalty for Logistic Regression) until we find the optimal ones. For Random Forest we found out the optimal range of hyper parameters by performing Random Grid Search first and then isolated good parameters with grid search cv.

The performance of all the models are evaluated and compared against individual baseline classifier models using various classification performance metrics such as Accuracy, Precision, Recall and F-score.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm Name** | **Approach** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| SVM | **TFID** | 0.737332 | 0.744358 | 0.737332 | 0.692530 |
| Logistic Regression | **TFID** | 0.722337 | 0.711079 | 0.711079 | 0.700592 |
| Logistic Regression with balanced | **TFID** | 0.673733 | 0.702686 | 0.702686 | 0.677891 |
| LSTM | Glove Embedding | 0.718346 | 0.697066 | 0.718346 | 0.694675 |
| Random Forest | **TFID** | 0.698552 | 0.709556 | 0.698552 | 0.666425 |
| KNN | **TFID** | 0.472596 | 0.798180 | 0.472596 | 0.550793 |
| LSTM- Bidirectional with Time distributed | Glove Embedding | 0.631783 | 0.512073 | 0.631783 | 0.559640 |
| LSTM- Bidirectional without Time distributed | Glove Embedding | 0.731266 | 0.697479 | 0.731266 | 0.705236 |

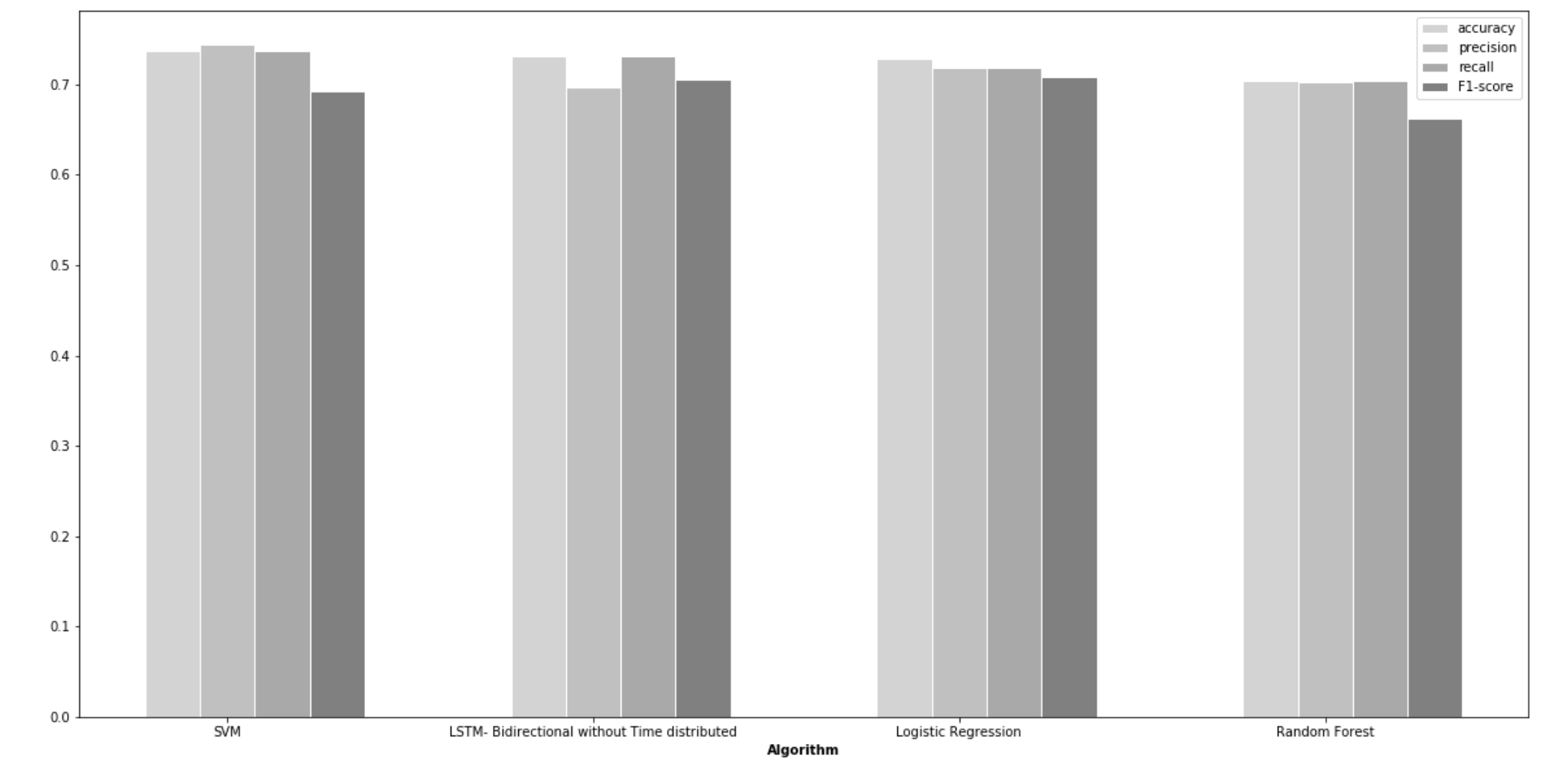
From the above results shown, we could see that Logistic Regression models perform well in comparison with the base classifier models on the test data.

F1 score bar graph of all models

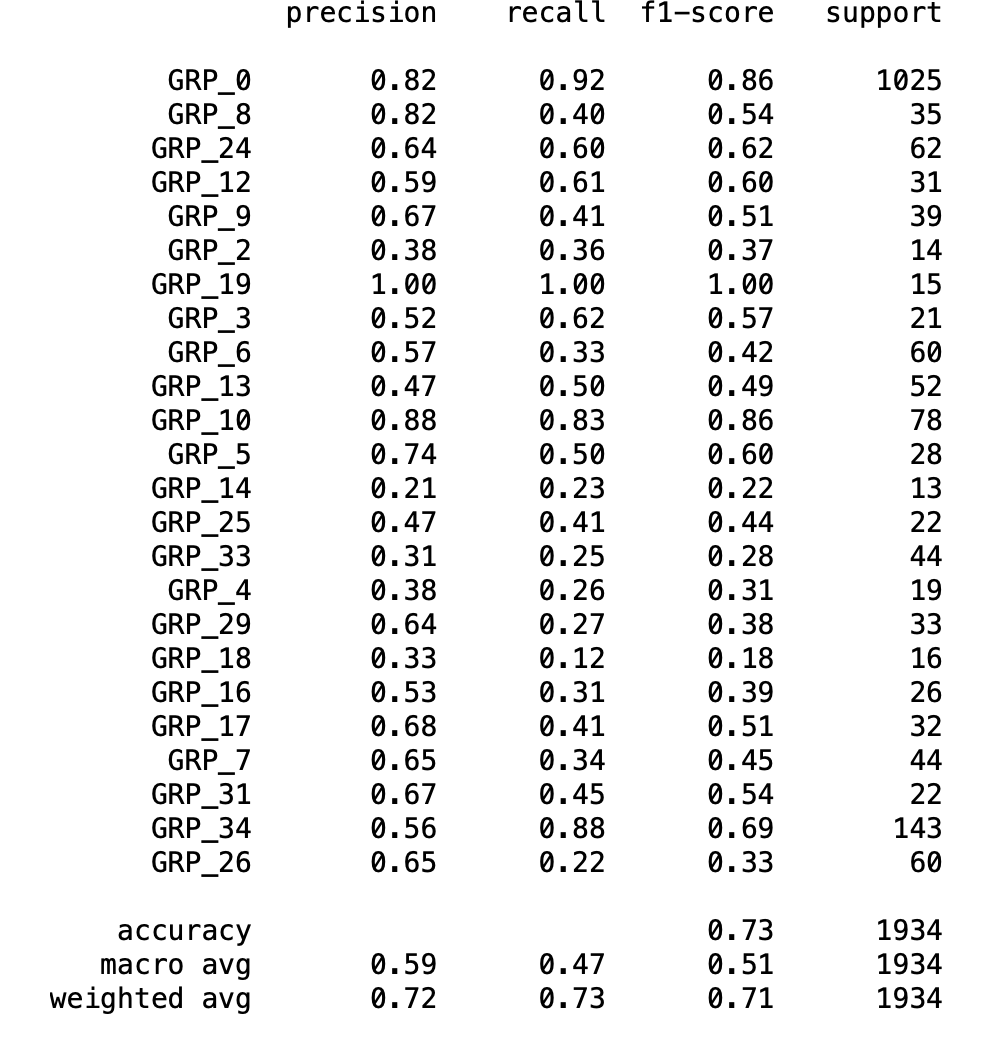


As we can see LSTM without Timedistributed has performed slightly better than logistic regression and it’s our best model.

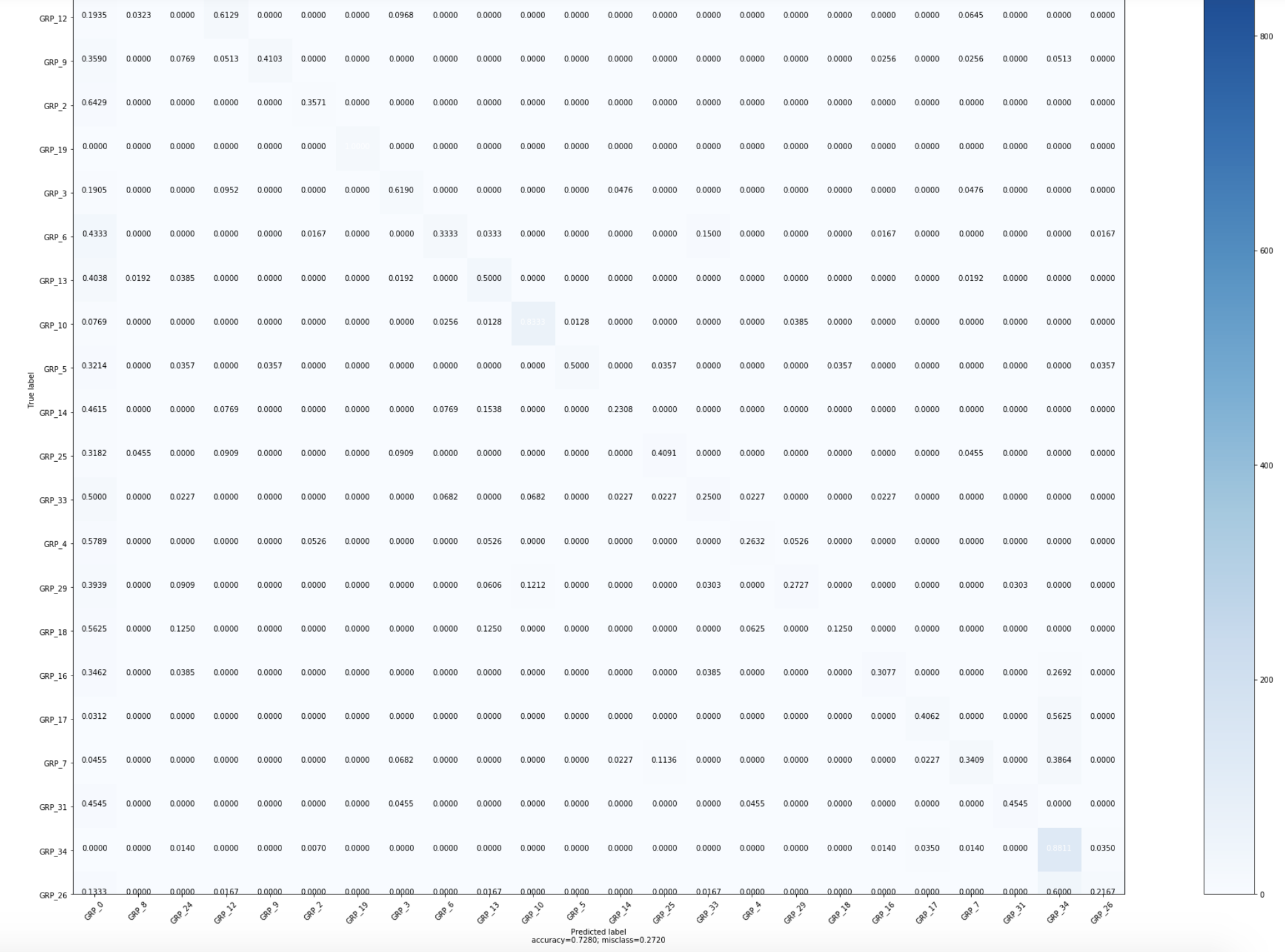
Top 4 Models



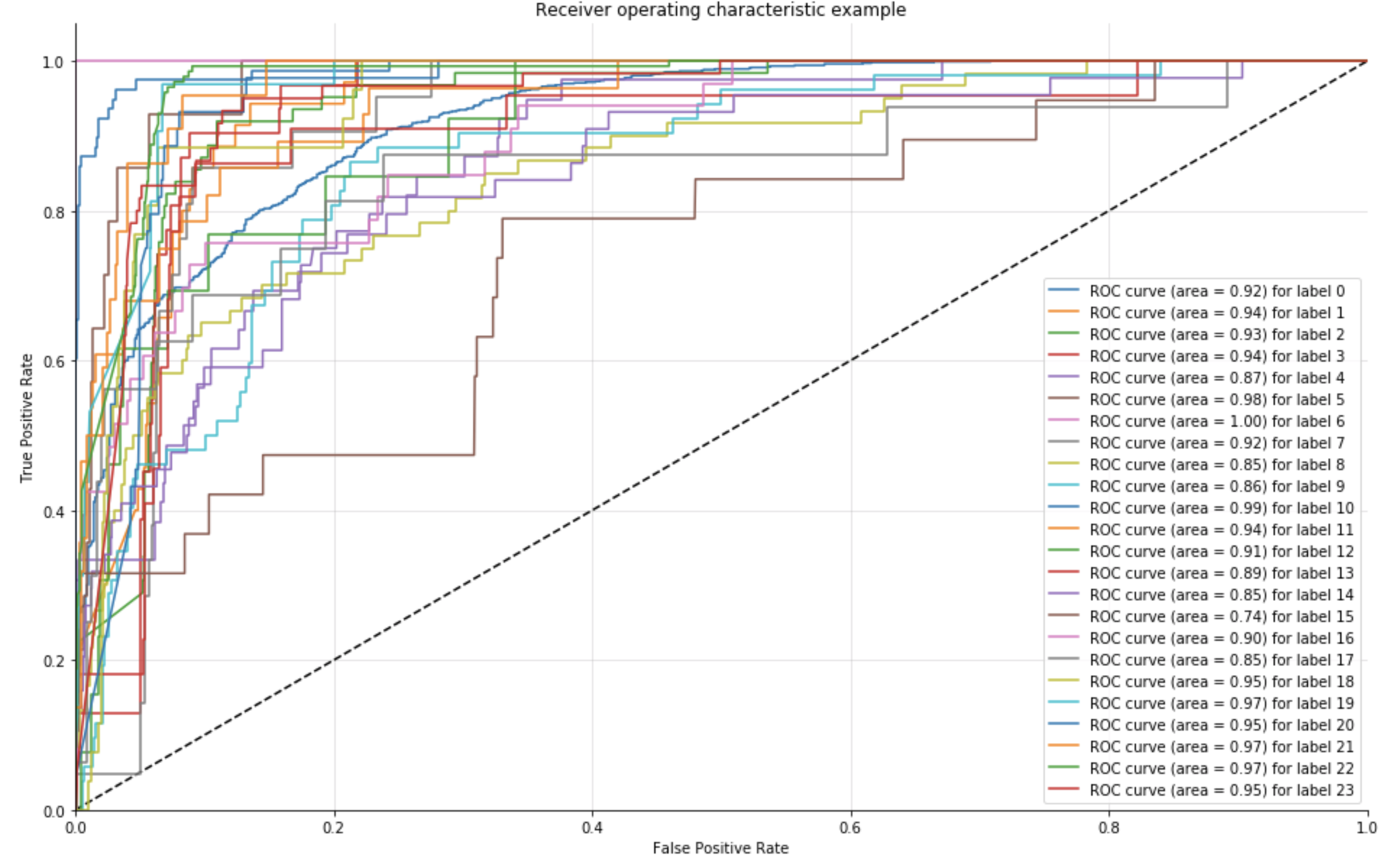
**Classification Report for Logistic Regression**



**Confusion Matrix plot for Logistic Regression**

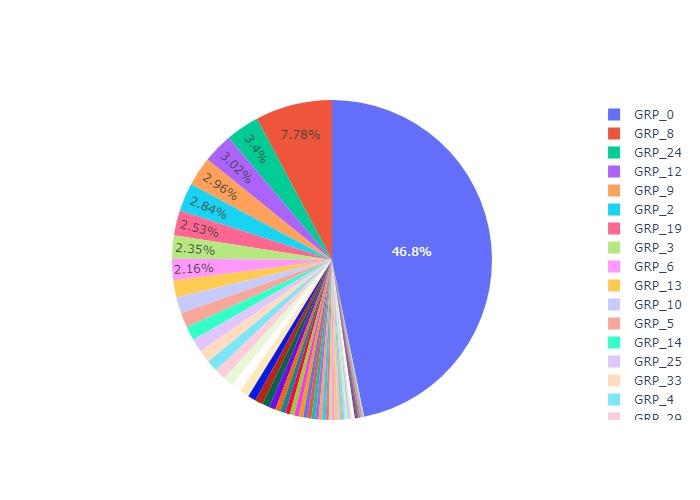


**ROC Curve for Logistic Regression**



## **6. Visualizations**

1. **Incident distribution as per assignment group count**

******

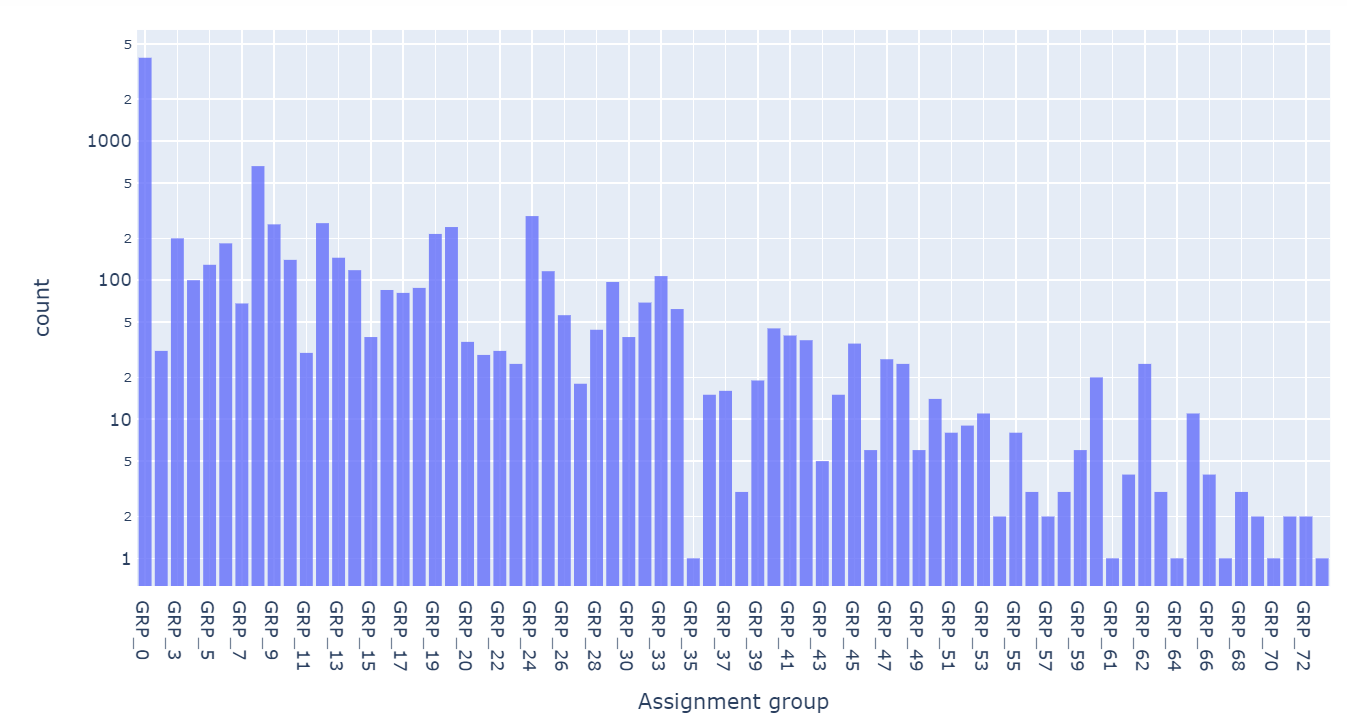
As we can see group 0 and 8 almost constitute 54.58% of incidents. They are representing L1, L2 distribution. Overall, our target column distribution looks imbalanced.

1. **Distribution of Incidents by L3 assignment group count**

******

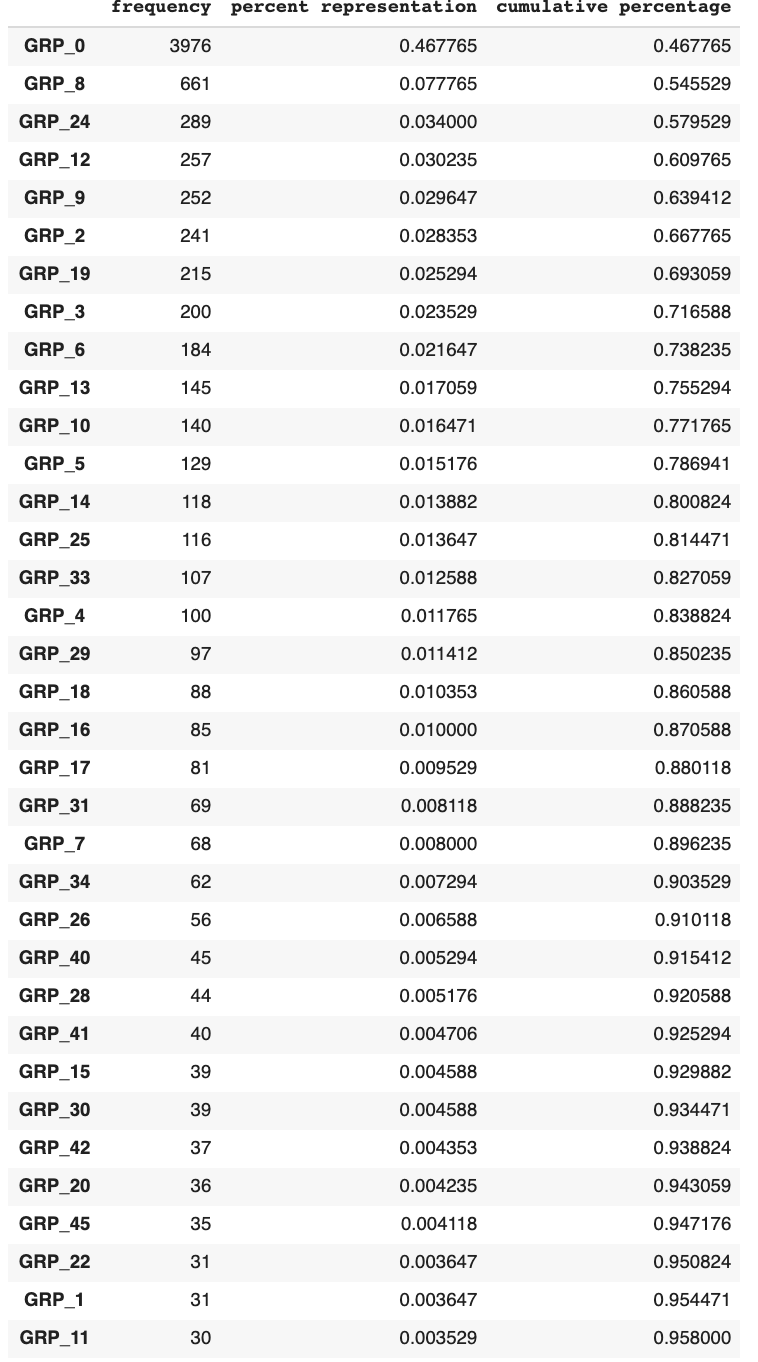
This distribution looks balanced. GRP\_24 has the highest share of incidents. There are few groups which have 1 or 2 incidents, we can drop them as they won’t be much help in classifier.

1. **Count distribution of groups**



We can see quite a few groups towards the right have **very less (<=4)** incidents assigned to them. We can remove these rows from our classifier. GRP\_24 has highest L3 incidents

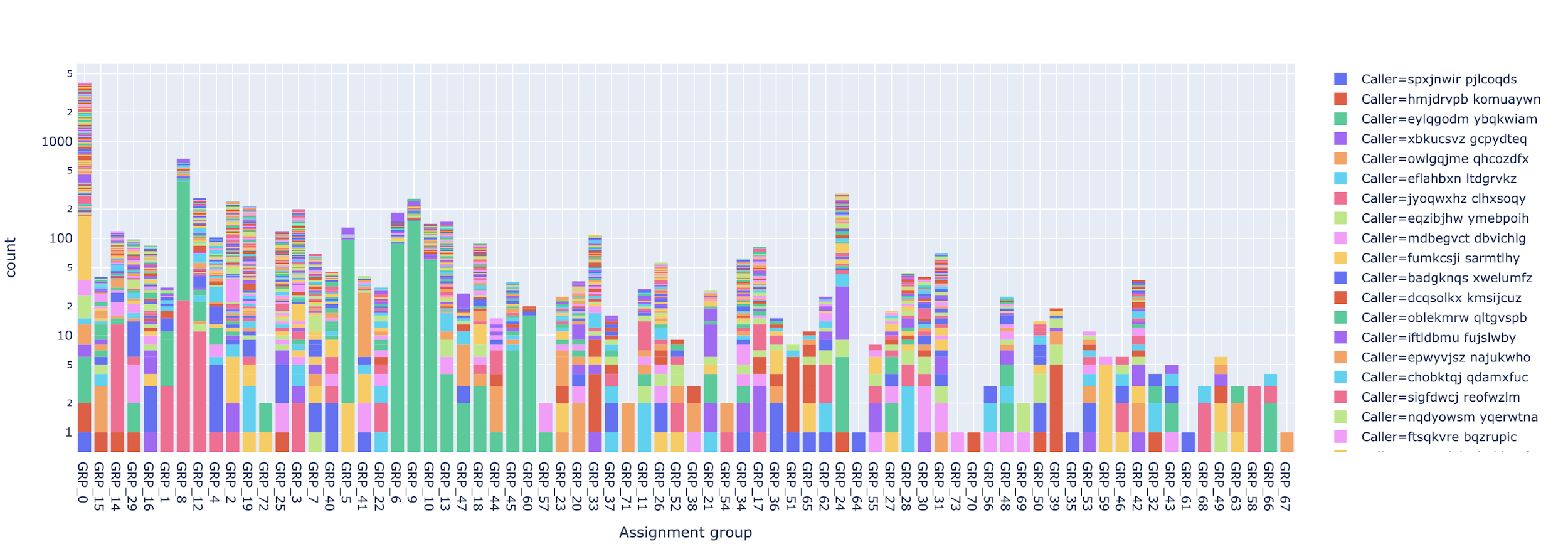
Further analysis shows 95% of tickets are distributed in 35 groups.



1. **Incident caller distribution**

Caller ‘bpctwhsn’ has created the greatest number of incidents. We can see that most incidents are created by unique callers.

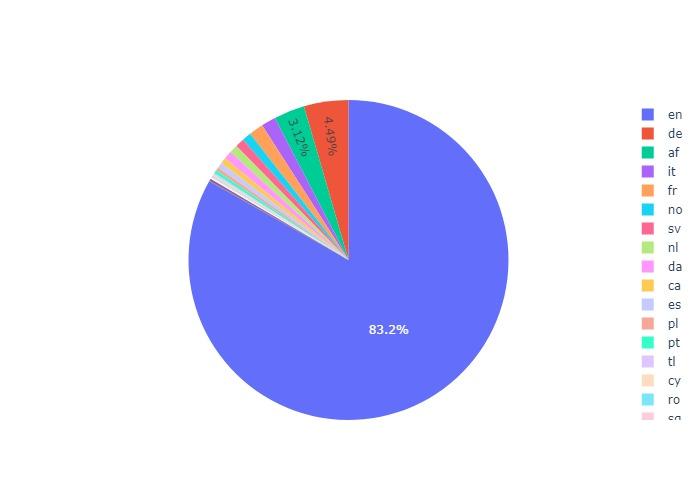
Now we want to see if a particular user ticket is getting assigned to the same group every time.



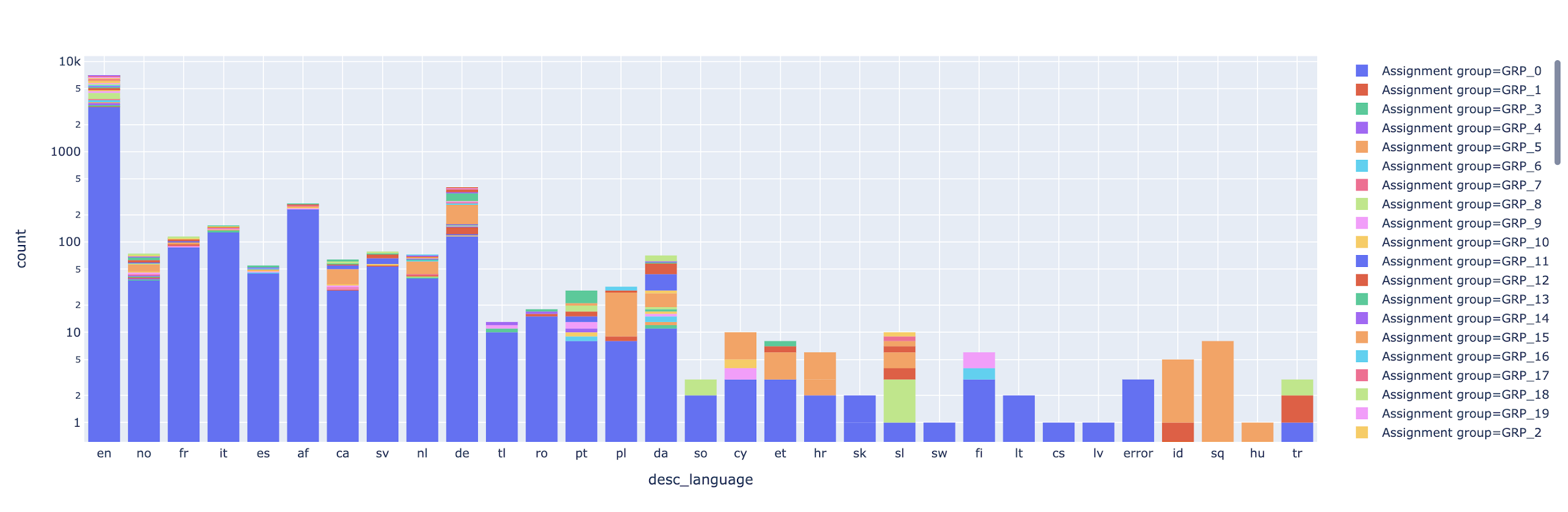
We don’t see any relation between the assignment group and the caller.

4. **Detected Language distribution of description in incident**

We have used the “langdetect” python package for detecting language.

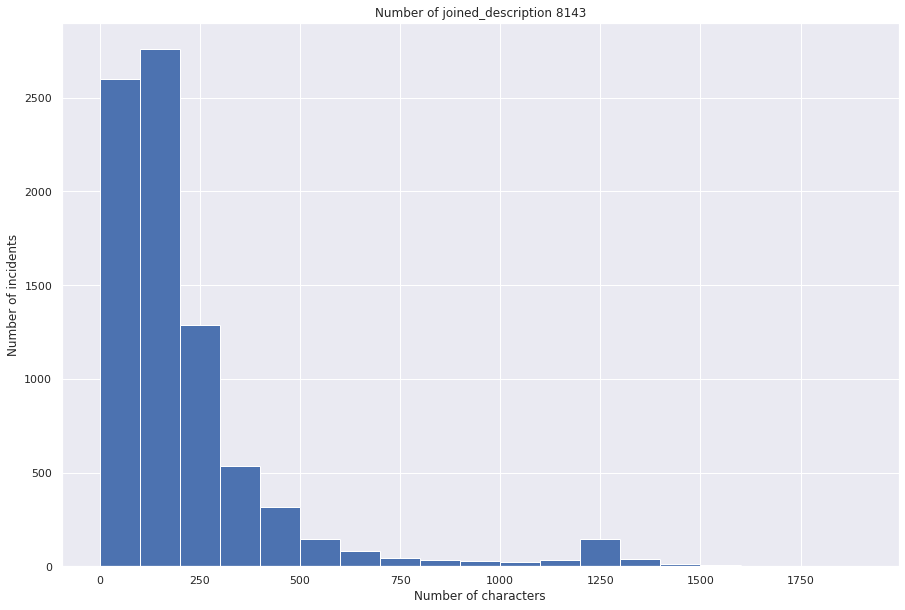


We can see ‘English’ is predominant language followed by ‘German’ and ‘Afrikaans’. We want to further see that if there is a relation between language and assignment group.



We couldn’t find any relation between the assignment group and language.

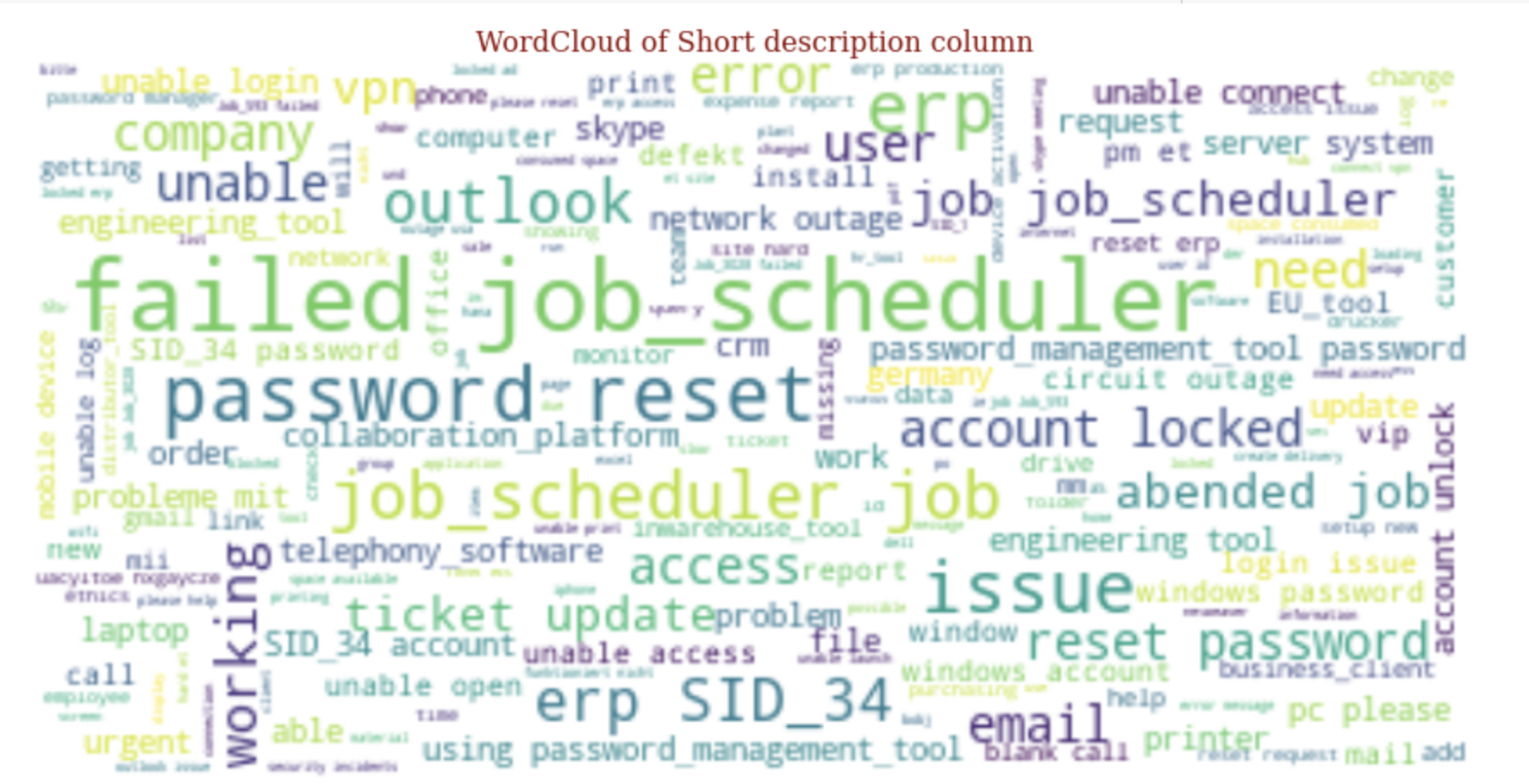
1. Character count distribution among incidents

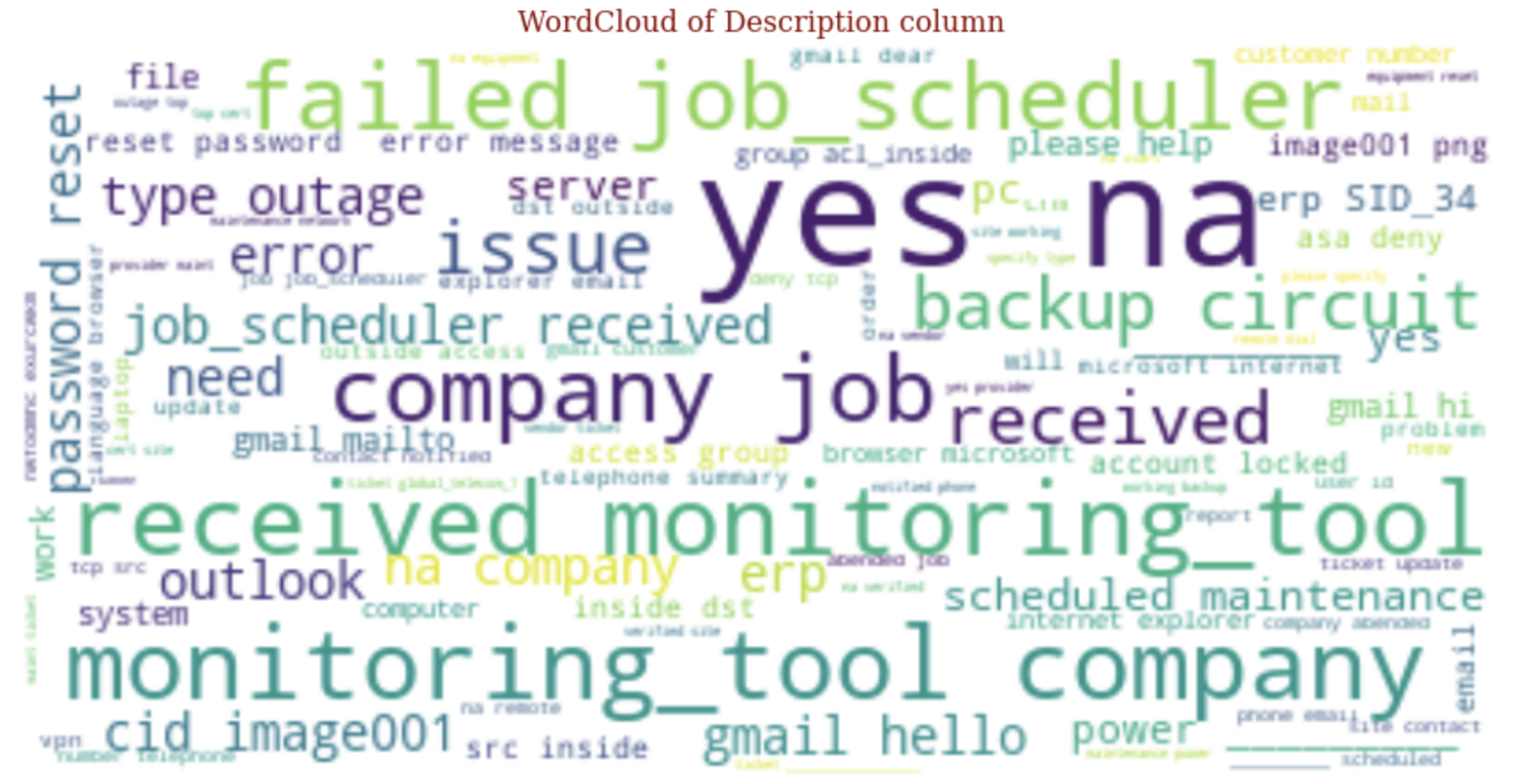


Majority of incidents descriptions have less than 500 characters. We can see few incidents have more than 1000 characters. We can investigate them more closely.

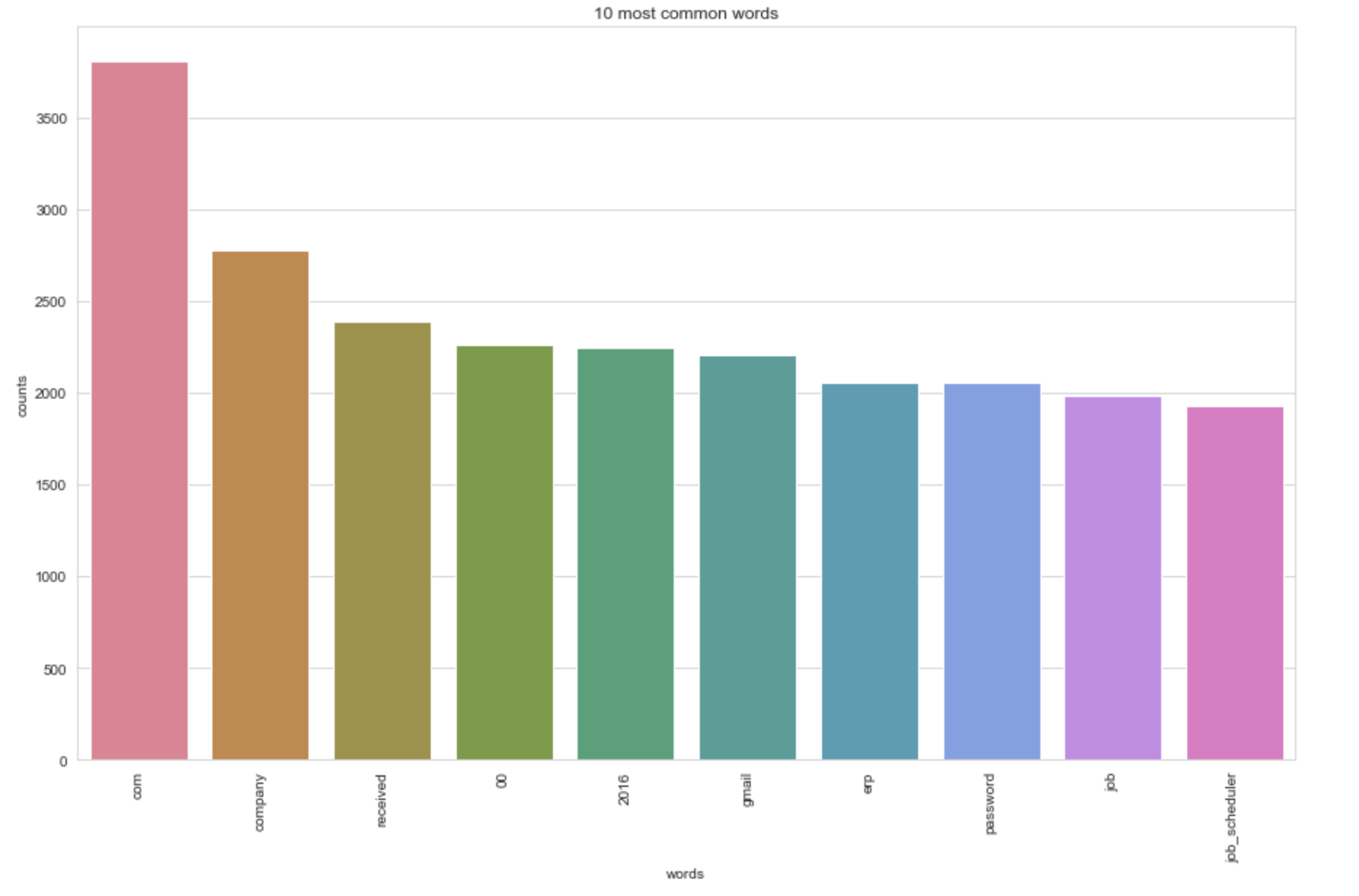
1. **Text Visualization**

Word cloud is used to represent the frequency of each word and hence we want to check how word cloud looks like for unprocessed data.



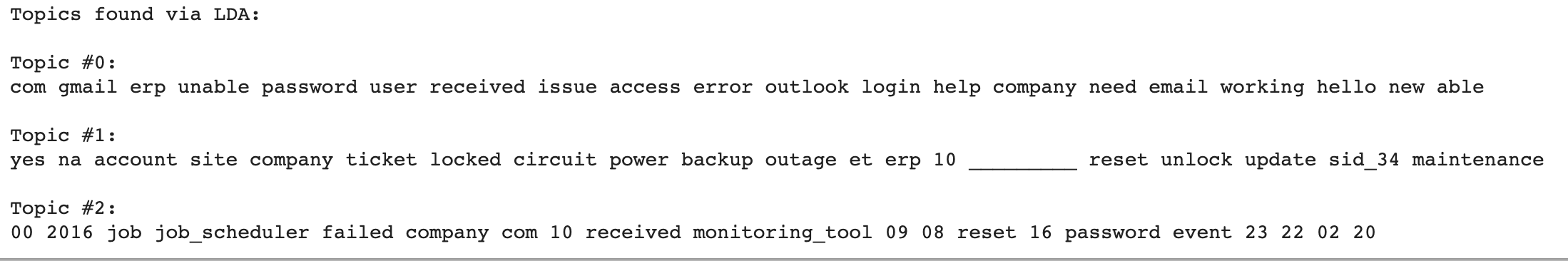


We will take a look of most common word before data cleanup using library CountVectorizer



As we have seen from the above chart there are words that we need to remove in our preprocessing step. For topic modelling we have used LatentDirichletAllocation

Model. Most common topic before preprocessing are

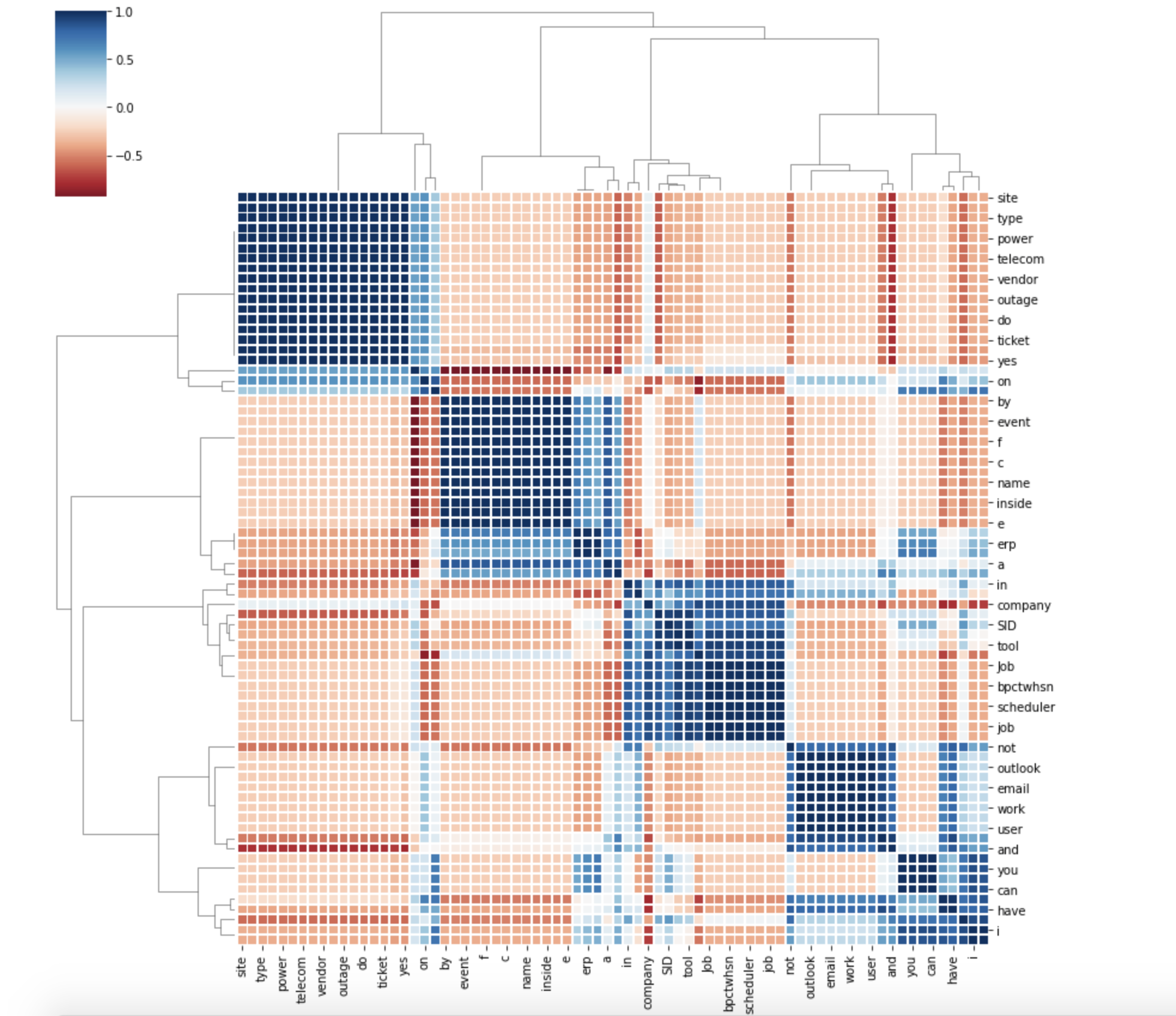


#### **ClusterMap**

ClusterMap helps in finding the pairwise correlation of all columns in the dataframe. Any na values are automatically excluded.

ClusterMap clusters both columns and rows and adds dendrograms to show the clustering of similar keywords in this case.

Hierarchical clusters help to order data by similarity and displays similar content next to one another for even more depth of understanding the data



## **7. Implications**

Existing manual classification systems are complicated and cluttered with too many categories for helpdesk to choose from. After spending endless hours going through tickets, agents will often end up either assigning tickets to the ‘Other’ category to sort them faster or assigning tickets to an inappropriate category. So with the existing solutions, it is very difficult to route tickets to correct team members in real-time.

Our solution (automated ticket classification approach) avoids the above-mentioned problem,

firstly, because machine learning tools will only assign a tag that we’ve previously defined, and

secondly because ticket categorization models will assign these tags automatically, without our need to scroll through a long list of tags.

Our automatic ticket classification system utilizes Natural Language Processing techniques which helps machines process, understand, and potentially generate human language in a fast and cost-effective way. Using text classifiers, we can tag tickets as and when they drop into the help desk. For example, we might categorize tickets by language, topic, or specific channel (Twitter, email, live chat, etc.), and route them to the correct team member or department based on these tags.

The benefits of our automated ticket classification below:

**Real-time Analysis and Response**

One of the biggest advantages of our model is that ticket categorization tools can work around the clock. So, whether a customer sends a complaint in working hours, or late night, (it doesn’t matter what time it is), our automated ticket classification can send a response in real-time, 24/7.

**Scalability**

Ticket classification with machine learning automatically tags hundreds of support tickets in seconds, as opposed to hours if done manually by human agents. It allows the agents to focus on more fulfilling tasks, and avoid heavy and repetitive workloads.

**Cost Effective**

Our automation software can streamline the incoming communication, and help the IT team avoid dealing with redundant tasks that eat up productive time. Our Intelligent system can handle mundane tasks, while the engineers can focus on dealing more serious issues.

**End-to-end visibility**

Our automated solution helps the organization to gain detailed insights into the service level performance and understand when and why the bottlenecks occur. The newly obtained data can be then used to improve the team’s performance.

**Recommendations**

1. An ensemble-based classification engine that uses supervised machine learning to understand the nature of the problem from free unstructured text and assign accurately. The choice of ensemble is based on the results of study performed with various machine learning and deep learning models as presented in section 3.
2. A rule engine to i) handle domain specific content missed by the ensemble classifier and ii) ensure business continuity. The rules are designed to strategically combine with machine learning methods for effective disambiguation of classes. Rules are specified through a customer independent framework.
3. An effective retraining strategy which ensures that the models are up-to-date with the changes that happen over time in the ticket as well as resolver group organization. In short keeping the model up to date with the ticket received.

## **8. Limitations**

1. Minimum 50 tickets are required.
2. We can tune decision threshold parameter to get better accuracy.
3. Pre-processing of incoming ticket before feeding it to the model.
4. Real-world training data tends to have categorical imbalances, the typical result after training a classification model on real-world service tickets is that dominant classes tend to have better prediction accuracies. In contrast, the model almost always misclassified minority classes as one of the dominant categories, resulting in false positives in the dominant categories.
5. For learning deep neural networks, tf-idf representation being extremely sparse is not useful.
6. We can try n-gram approach.
7. We can use [TSNE](http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html)l to **visualize** high-dimensional data. Based on finding we can use PCA for dense data or Truncated-SVD for sparse data) to reduce the number of dimensions.

## **9. Closing Reflections**

Automated assignment of helpdesk tickets results in considerable saving in human effort for large companies having clients across geographies. It reduces the time taken for ticket assignment and at the same time minimizes human error. This enables the companies to focus more on innovation and core business needs. Based on our results as summary, we have an estimate of human effort saving, across all accounts. Assuming that a human agent takes about 3 min to read and assign each ticket, the net savings (in min) for an account can be calculated as: Si = T ×3.

Below are main takeaways from our evaluation results above:

1. The most important observation is that our deep learning algorithms work best if the number of tickets per group is 50 or more. Also, time distributed layer is not useful in this problem as we do not have any temporal attribute to the data.
2. Also, it was observed that models using glove embeddings were less accurate than TFIDF measuring. Since ‘tf-idf’ is a simple scoring system and can identify topics from input words easily it suited our incident classifier problem more.
3. We can see that simple machine learning algorithms like Logistic regression, Random Forest are often better than more computationally expensive deep learning algorithms in the task of ticket assignment. However, it must be noted that LSTM accuracy increases with the size of the dataset and with a very large training data size (more than 50 records per group) CNN and LSTM start outperforming MLP.

**Conclusion and Future Work**

1. We will have a ticket assignment engine that uses an ensemble of machine learning techniques combined with a rule engine to perform automated dispatch.
2. Adding support for images and attachments.
3. Chatbot can be built to auto resolve tickets based on knowledge base.
4. Deployment of model in docker or cloud-based engine.

**Appendix:**

We have submitted four python notebooks along with the report.

1. **Model Benchmarking.ipynb** - describes evaluation of different algorithms on clean dataset
2. **Two\_Step\_Classifier\_Approach\_2.ipynb** - describes result of two level classifier on clean dataset
3. **Data Augmentation Approach 3.ipynb** - describes result of data augmentation approach on clean dataset
4. **Interim\_eda.ipynb** - contains EDA diagrams and data cleaning steps