

Automatic Ticket Assignment

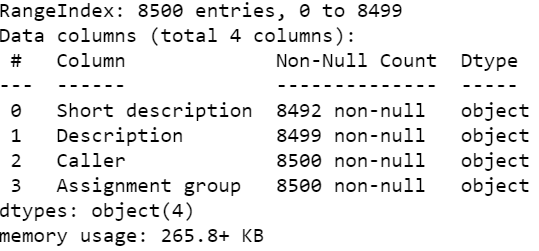
Interim Report

# Team Details

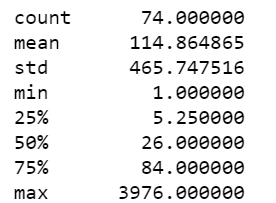
|  |  |
| --- | --- |
| Abhijit Pande  Abhik Kar  Arun Kottakulam  Sayantika Sur | abhijit.a.pande@gmail.com  akar1983@gmail.com  [arun.arunasokan@gmail.com](mailto:arun.arunasokan@gmail.com)  ssurmallick@gmail.com |

# Summary of problem statement, data and findings

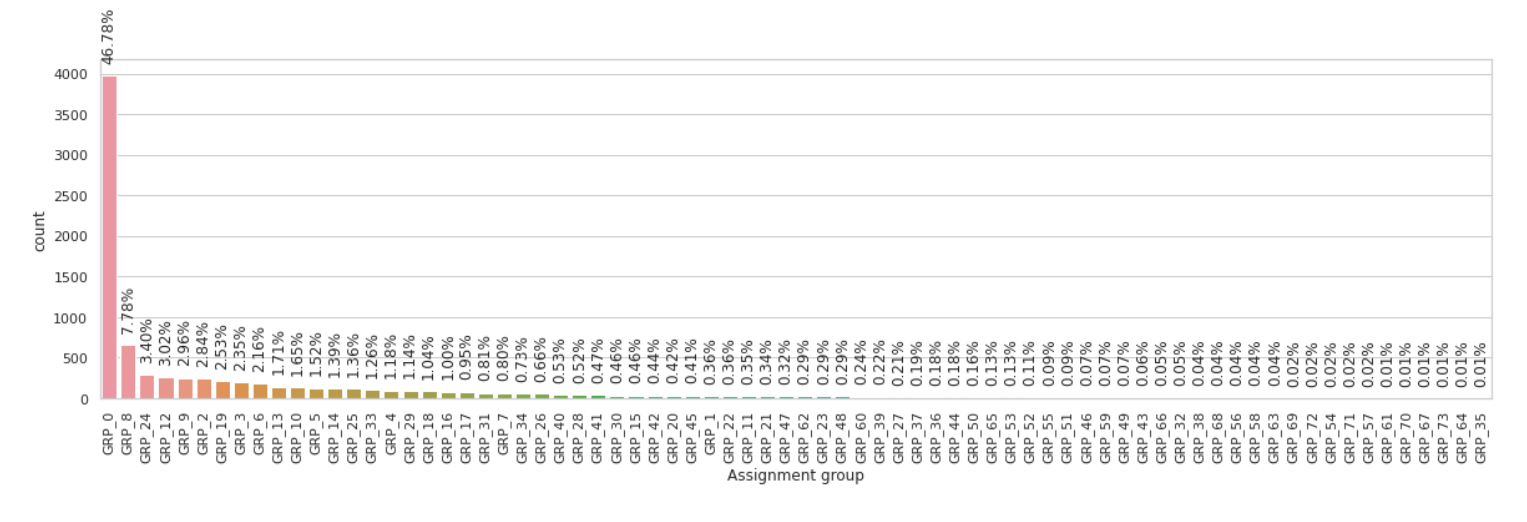
* **Problem Statement** - Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.
* **Abstract** - Applying traditional machine learning and NLP to automatically classify tickets and assign them to the right owner in a timely manner to save effort, increase user satisfaction and improve throughput in the ticketing pipeline of an organization.
* **Summary of Findings & Data** –
  + Data provided in CSV format
  + Total Records - 8500
  + 2950 unique callers
  + 74 unique assignment groups
  + Almost 50% of the data belongs to group 0
  + Data Fields –



* As seen above, data set is containing 4 columns of object type
  + Short description – Summary of the ticket being raised
  + Description – Details of the ticket being raised
  + Caller – User who called to raise the ticket
  + Assignment group – Group in which the ticket is assigned to worked upon.



* Above data shows the detailed analysis of the Assignment group
* **Visualization –**



We see many groups that have very low number of tickets. This implies we need methods for augmentation and balancing of classes

Figure: Ticket counts per Assignment group

* Above graph depicts that the data is highly skewed. Group 0 is almost covering 50% of the dataset. Below graphs will show the variations for top 20 and bottom 20 categories

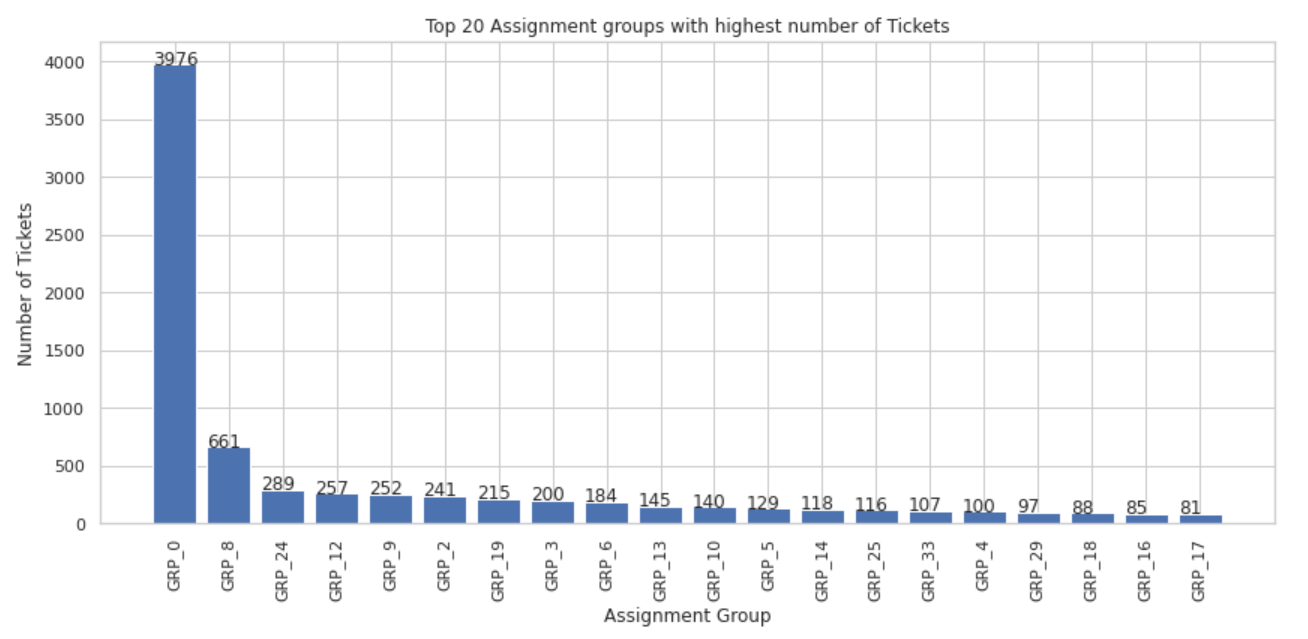


Figure: Top 20 Assignment groups with highest number of tickets

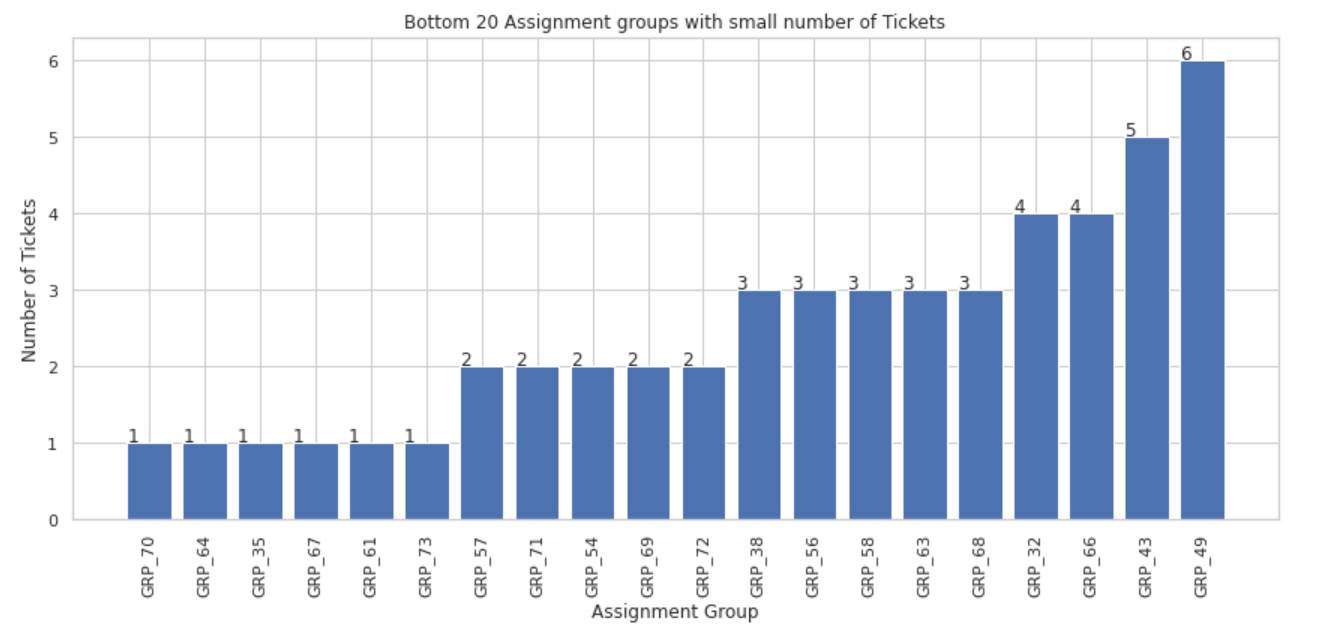


Figure: Bottom 20 Assignment groups with lower number of tickets

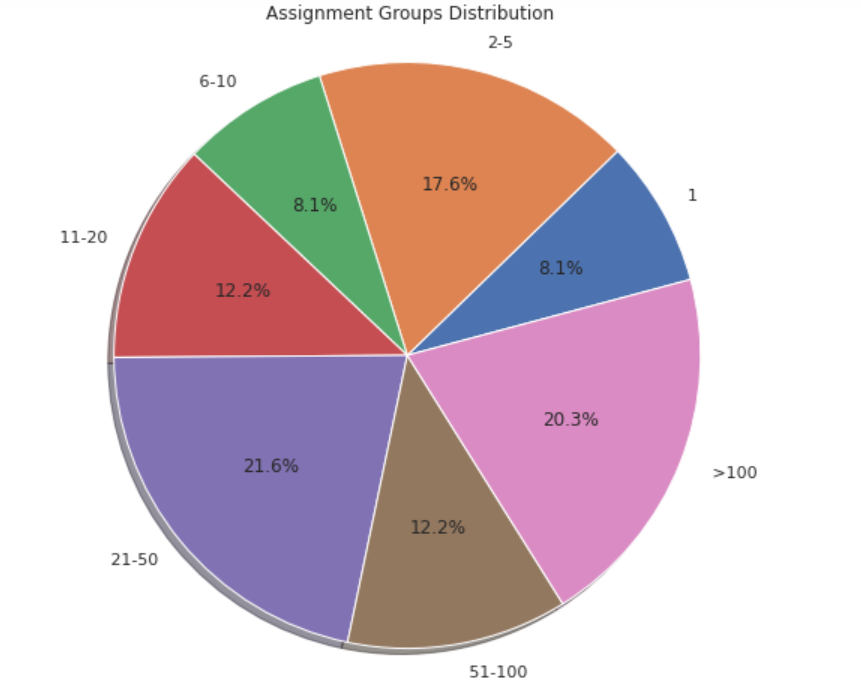
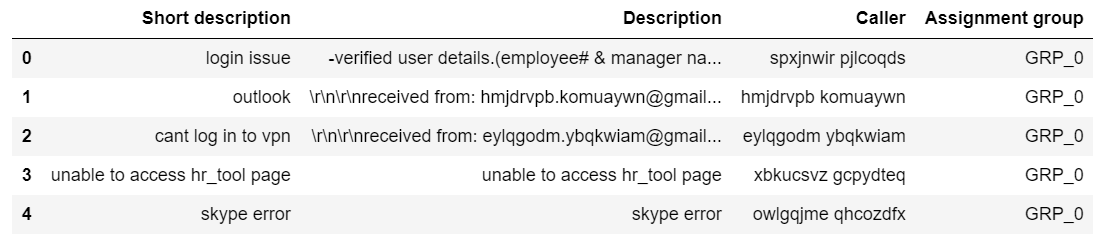


Figure: Pie Chart for Assignment Groups Distribution

**Sample data -**



Our Approach –

* Separate out larger & smaller groups for better data analysis. So we have sliced the dataset into 3 sets keeping this in mind.

# Summary of the Approach to EDA and Pre-processing

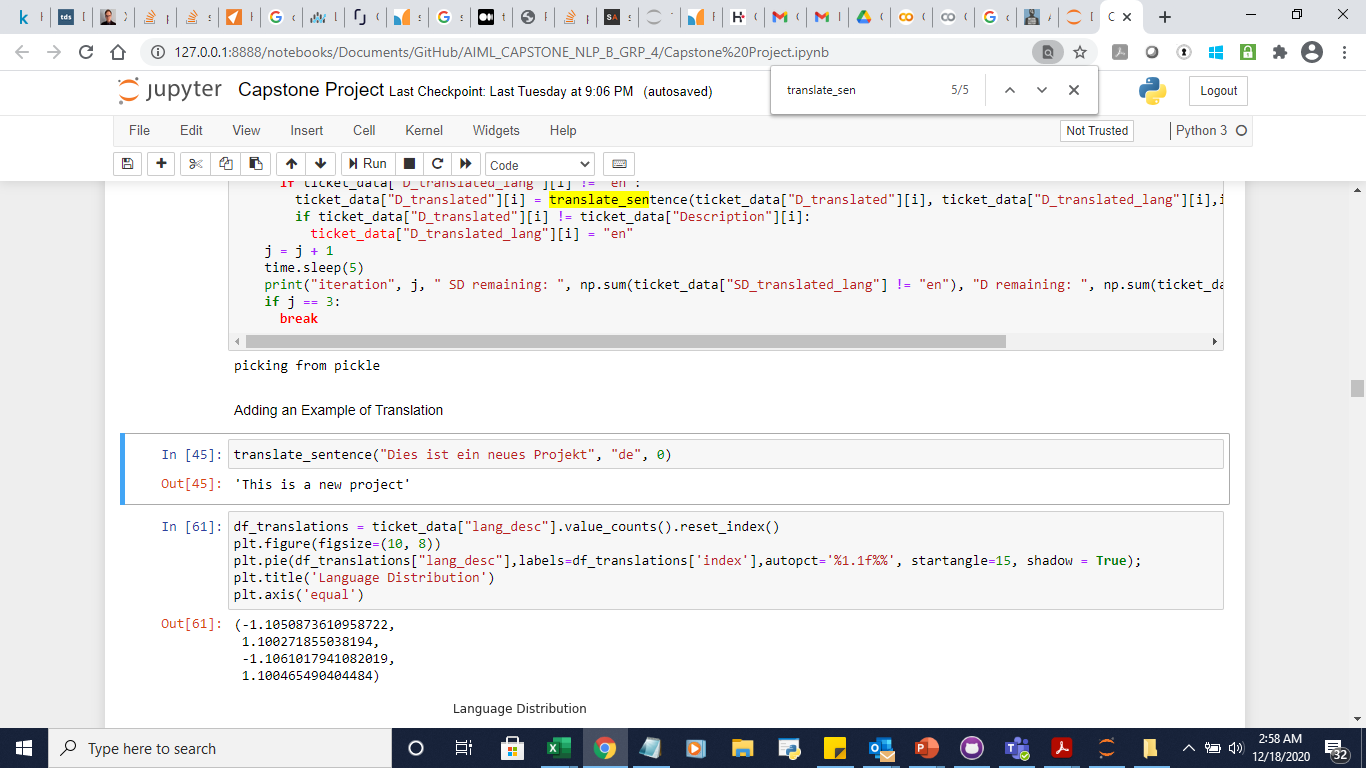
Our EDA is focused on developing the rationale for our next steps. We have attempted to segregate groups based on their size for the below purposes

1. Discovering the need for augmentation and class weighing: Certain groups are very under-represented in the dataset. Thus, we have tried to boost them to over 500 records in each case. Apart from this augmentation, we are also using class weighing to improve the accuracy of our results. Some of these groups, which had fewer than 20 tickets have been set aside for rule-based processing.

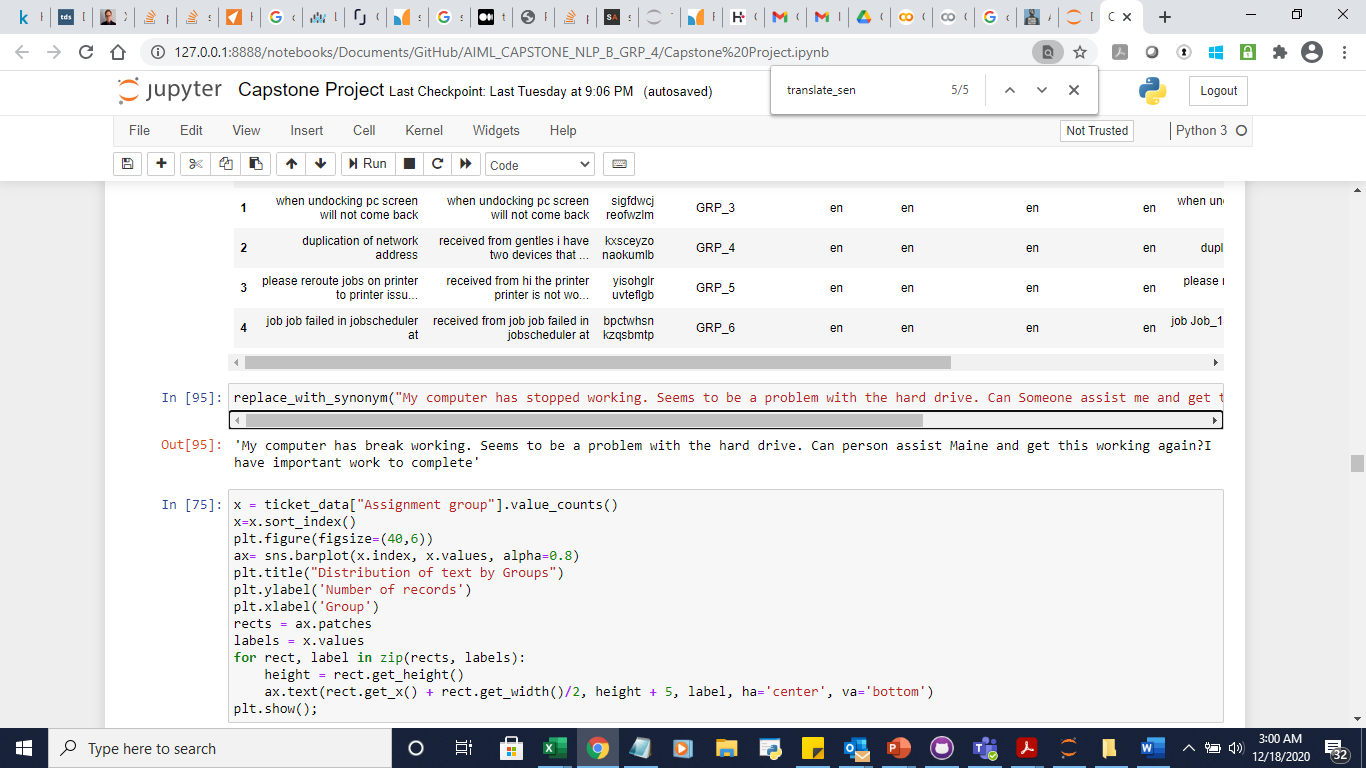
2. Our EDA also helped us determine the need for translations as many records were in different languages.

The steps performed in our preprocessing are as below

* Data Cleaning: We are attempting to first clean the data and make sure the dataset does not have erroneous data. To achieve this, we replace NaN values and standardize the encoding of the data. Also, we remove junk characters through regular expressions and FTFY libraries
* Next, we run a translation function on our data that leverages google translate. We translate our data row by row using the googletrans library. Our function uses an inbuilt backout and wait mechanism to wait for a second before it requests a translation of the same element again. We run this function for the description and the short description separately as we see some language mismatches between them



* Then we go ahead and augment our data. The augmentation uses a synonym replacement function that can replace a random number of words (upto 7) in a record. This allows us to get outcomes as below



* At this stage, the data is fine for deep learning, but not good enough for machine learning. So, we store the dataframe as the deep learning dataframe for all groups with over 20 records.
* Finally, we move on to lemmatization and stop word removal. This allows us to prepare our data for machine learning. The data is then partitioned into the rule based df and the machine learning df

## Visualizations

Figure: Data distribution before translation

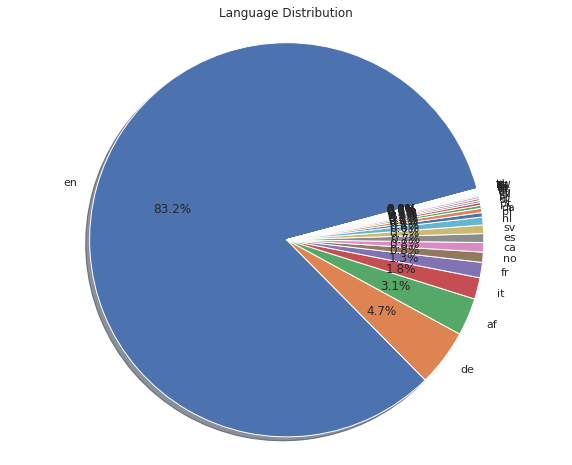


Figure: Data distribution after translation (60% translation successful)

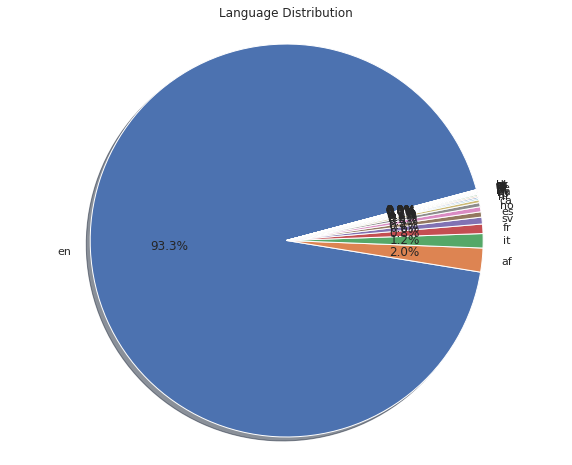


Figure: Data distribution before augmentation

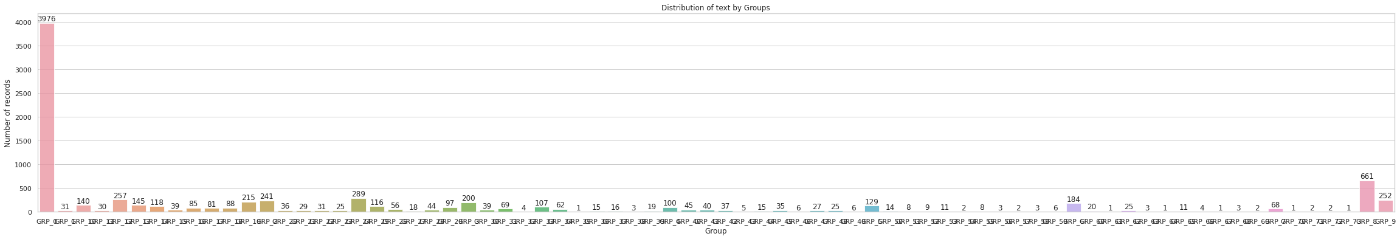
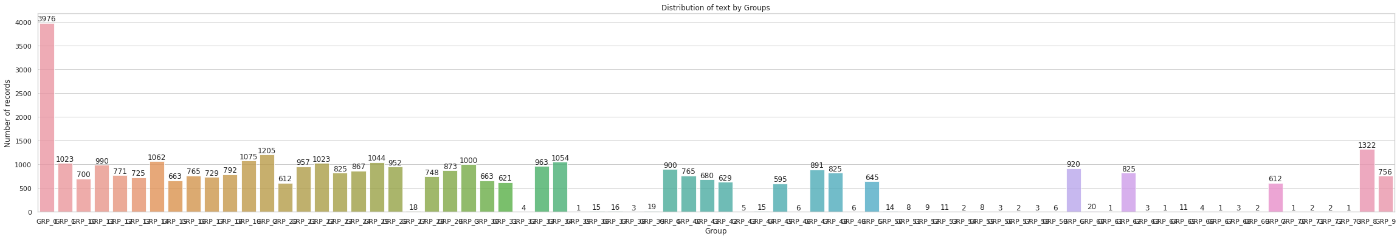


Figure: Data distribution after augmentation (groups not augmented are segregated for rule-based assignment)



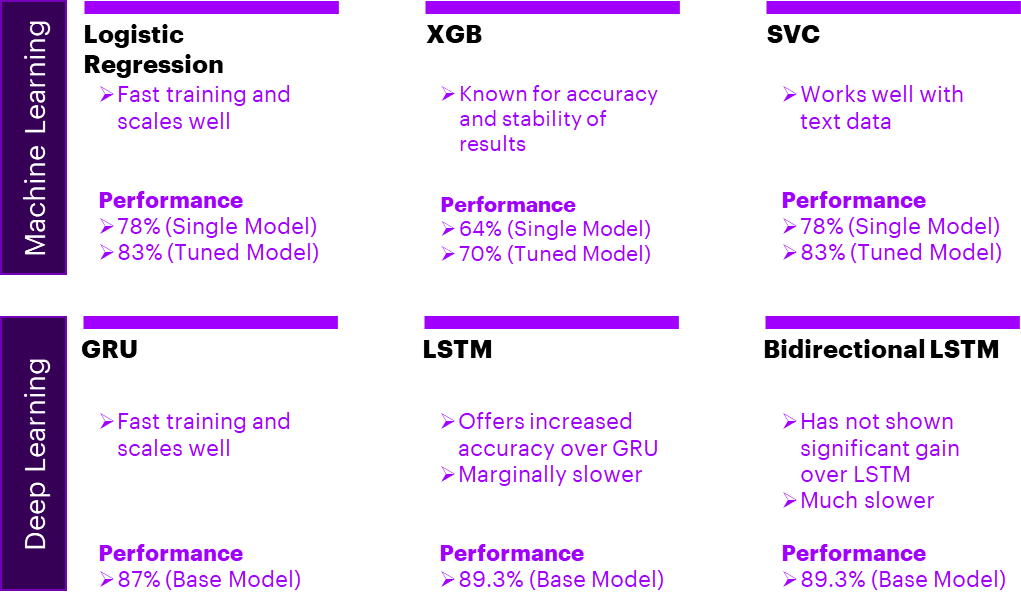
After completing this preprocessing, we moved on to feature engineering and vocabularization for machine learning and deep learning dataframes.

We use the TFIDF Vectorizer to convert the machine learning dataframe into a TFIDF matrix and then generate 2000 features to represent the valuable datapoints in a vocabulary of around 24000. This allows us to generate accurate predictions.

For deep learning, we run a tokenizer to convert the vocabulary into integer sequences and then proceed with the model building

# Deciding Models and Model Building

We have chosen the below models for our analysis



## Machine Learning

1. In our experience, we have seen Logistic regression scale very well to datasets of large sizes. Hence, our first model is logistic regression. With some tuning we can provide it a large feature set and still obtain accuracy over 80%

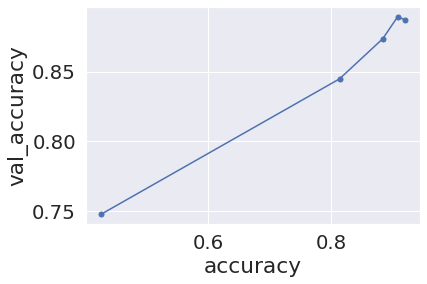
2. Our second model is XGB. This is a much-favored model for all Kaggle competition participants especially when dealing with large datasets. XGB provides better memory and execution performance compared to other tree-based models and hence is a natural choice for such a tabular dataset.

3. Our third model is the SVC. While this does present a degree of scaling challenges, SVC’s are very accurate at segregating text data and have show a base performance of nearly 5-10% greater than logistic regressions for smaller data samples. However, we have struggled to get it to run at higher sample sizes and are resorting to stratified training to reduce the quadratic time expansion

## Deep Learning

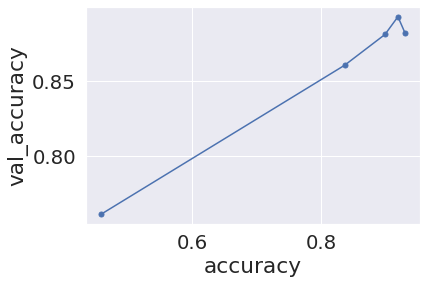
1. Our first attempt is an LSTM which is a natural choice for text data. This model allows us to carry forth the correlation of the previous words and some degree of context across an entire record. This has provided us almost 90% accuracy in our first runs and we are likely to improve it further significantly as we improve upon our initial data

Figure: LSTM performance (Training Accuracy vs Validation Accuracy)



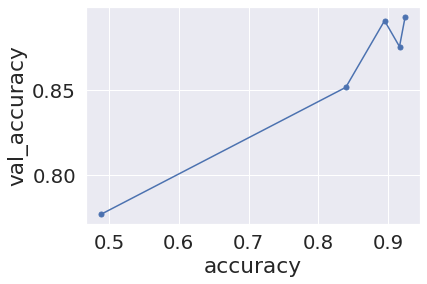
2. We have also attempted the GRU as a deep learning model, which works almost the same way as LSTMs but provides marginally lower accuracy and significantly lower training time.

Figure: GRU performance (Training Accuracy vs Validation Accuracy)



3. Our third model is the bi-directional LSTM and this takes us almost 4 times the training time of the GRU. This was an experimental choice as we needed to see if bidirectional inferences have any impact on prediction accuracy. We have discovered only 0-0.5% of an accuracy change while almost doubling the time of the LSTM. In such a case, it would make sense to continue further with the LSTM itself. Also, it can be inferred that unless it is a translation task, typical LSTM should suffice for deep learning on text

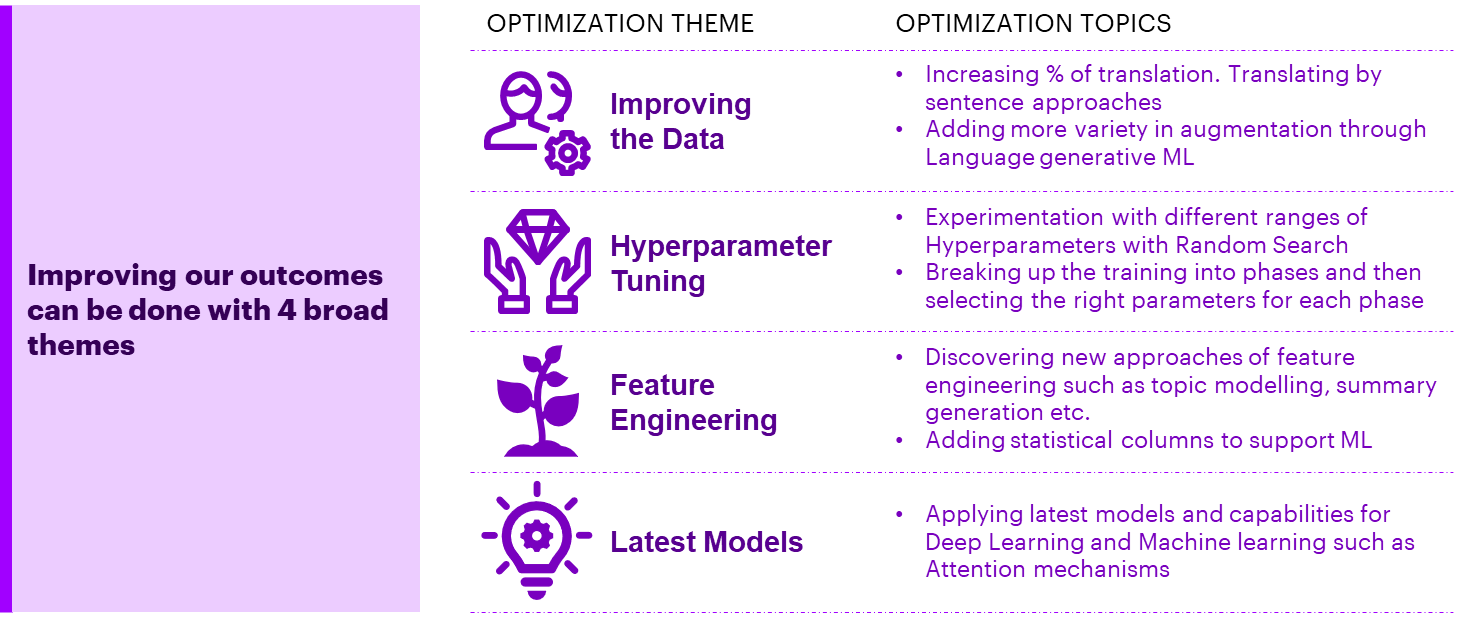
Figure: Bi-Directional LSTM performance (Training Accuracy vs Validation Accuracy)



* Potential Models that have success in classifying vs have shown great performance (Models based on experience, based on outside information)
* Explanation of the models attempted
* Graphs, Timing charts that present evidence of Model performance and comparatives to build a narrative towards a type of models (in this deep learning)
* Hyperparameter tuning and its outcomes with evidence and visuals
* The Progress that we have made

# How to improve your model performance?

We have explored multiple ways to improve our model performance. Some of them are listed below

1. Currently, our use of google translate and row by row translation creates a greater scope for failures of certain records. Such records while low in number still can affect our final accuracy. To improve this, we are planning to move to a sentence by sentence translation model, which though slower, will allow us to provide much cleaner training data to feed to our ML/DL

2. Our current augmentation is solely focused on using synonyms for replacements. While this can be a simple and effective approach, it does not train the machine on indirect relationships. Dropping words, scattering sentences could be additional approaches to try. Additionally, we could use generative language models to generate records

3. We have attempted phased fitting of models to different parameters of data. The idea is to slow down the learning rates, as the model training levels increase. This could be a potential way to sharpen the learning at the top end. Also, it allows us to individually Hyperparameterize different phases of training

4. We are also interested in Developing new supporting features that can augment our current data and introduce variables that can increase the predictive power of our models. This could involve the use of statistical as well as topic modelling/summary generation techniques

5. Lastly, we have not used transformers, attention mechanisms on our data yet. This could be another way to improve our prediction accuracy.