# TABLE OF CONTENTS

# Project Description

## 1. Abstract

Due to the rise of usage of virtual systems, support ticket systems have come into prominence. Addressing the issue tickets to appropriate person or unit in the support team has critical importance in order to provide improved end user satisfaction while ensuring better allotment of support recourses.

The assignment of help ticket to appropriate group is still manually performed. Especially at large organizations, the manual assignment is not applicable sufficiently. It is time consuming and requires human efforts. There may be mistakes due to human errors. Also, resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response time which result in end user satisfaction deterioration. Multiple-choice systems which provide the user to choose the related categories or unit within defined categories may seem like better, but the systems are not useful because of those users, especially new users which have never used the system before, usually have no idea about the related category or department. Also, users do not want to fill long ticket forms which are needed to identify the issue. In this study, an extension to ITS for auto-addressing the issue ticket to the relevant person or unit in support team is proposed. In this system, bag of word approach, machine learning techniques and neural network (LSTM) which proven performance in text processing are used. The recommended method provides high quality user support and boosts end-user satisfaction. It reduces manual efforts and human errors while ensuring high service levels and improved end-user satisfaction.

## 2. The Real Problem

One of the key activities of any IT function is to "Keep the lights on" to ensure there is no impact to the Business operations. IT leverages Incident Management process to achieve the above Objective. An incident is something that is unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business. The main goal of Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact.

In most of the organizations, incidents are created by various Business and IT Users, End Users/ Vendors if they have access to ticketing systems, and from the integrated monitoring systems and tools. Assigning the incidents to the appropriate person or unit in the support team has critical importance to provide improved user satisfaction while ensuring better allocation of support resources. The assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations. 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.

## 3. Project Description

In this capstone project, the goal is to build a classifier that can classify the tickets by analysing text.
Details about the data and dataset files are given in below link,
https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ

### Milestone 1:

Pre-Processing, Data Visualization and EDA Overview

- Exploring the given Data files
- Understanding the structure of data
- Missing points in data
- Finding inconsistencies in the data
- Visualizing different patterns
- Visualizing different text features
- Dealing with missing values
- Text preprocessing
- Creating word vocabulary from the corpus of report text data
- Creating tokens as required

### Milestone 2: Model Building

**Overview**

- Building a model architecture which can classify.

- Trying different model architectures by researching state of the art for similar tasks.
- Train the model
- To deal with large training time, save the weights so that you can use them when training the model for the second time without starting from scratch.

**Milestone 3:**

- Test the Model, Fine-tuning and Repeat Overview
- Test the model and report as per evaluation metrics
- Try different models
- Try different evaluation metrics
- Set different hyper parameters, by trying different optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc..for these models to finetune them
- Report evaluation metrics for these models along with your observation on how changing different hyper parameters leads to change in the final evaluation metric.

**Project Objectives**

The objective of the project are

- Learn how to use different classification models.
- Use transfer learning to use pre-built models
- Learn to set the optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc.

## 4. Existing System

Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams). This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. Incase L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by either Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve the incidents. Around ~56% of incidents are resolved by Functional / L3 teams. Incase if vendor support is needed, they will reach out for their support towards incident closure.

L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

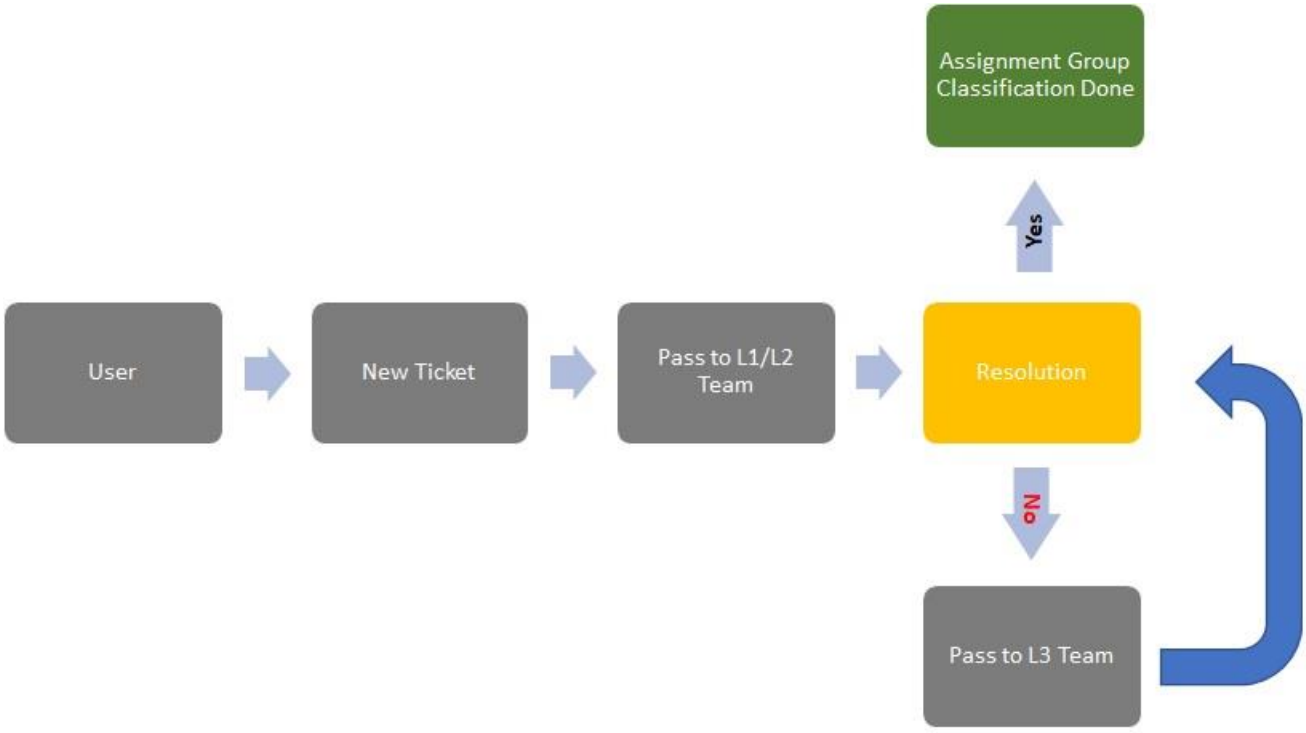


**Fig – 1.1**

## 5. Existing System Problem

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.

## 6. Proposed System

In the support process, incoming incidents are analyzed and assessed by organization's support teams to fulfill the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings.

Guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

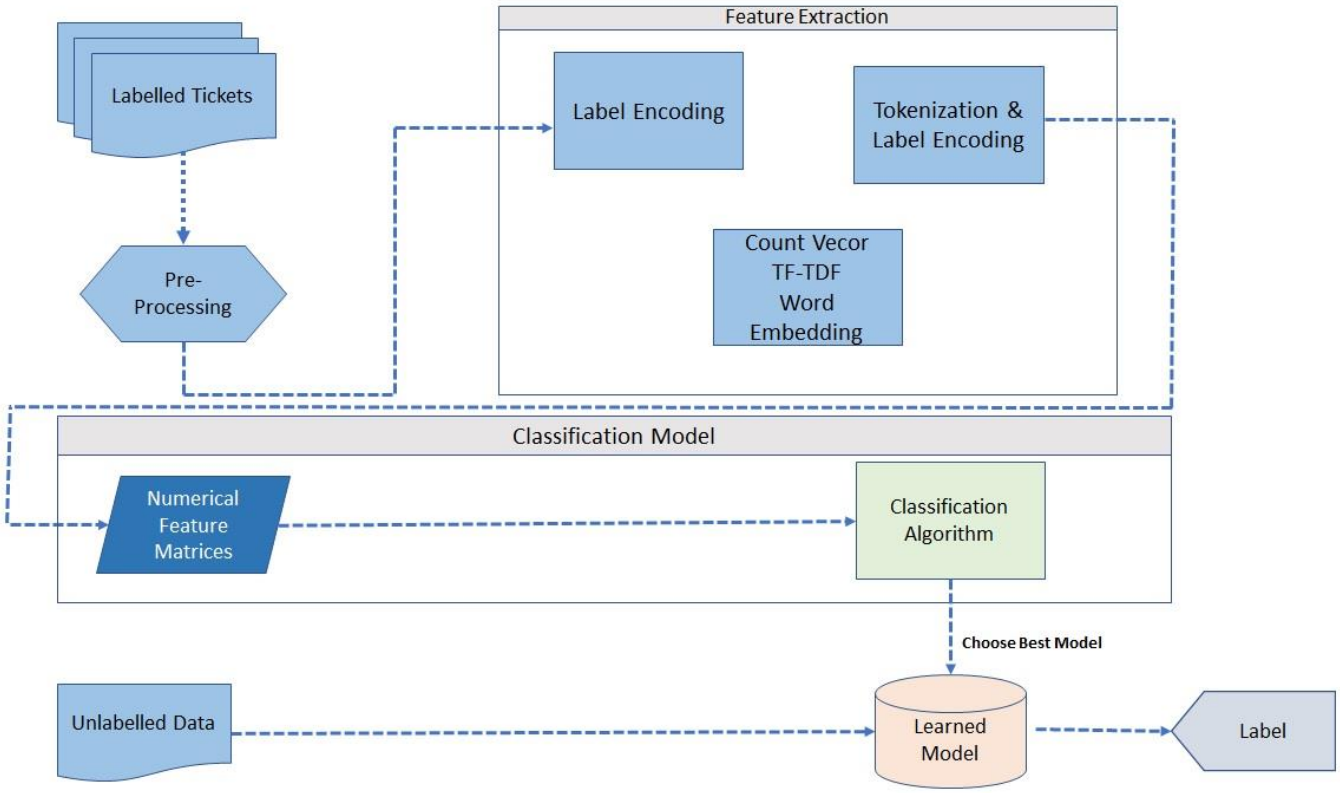



**Fig – 1.2**

## 7. Implementation and Experimental Results

The proposed system will be using the "Input Data Synthesis" database. In this section all the implementation steps have been discussed carefully in details.

We will use terminologies as 'Short Description = 1', 'Description = 2','Caller = 3' & 'Assignment Group = 4'

### 7.1 Dataset

The dataset has 4 columns and 8500 issue tickets in English and German collected from various Business and IT Users, End Users/ Vendors providing details about the incidents being created. Every row (incident) can be called as a TICKET. Each ticket contains the short issue name stating what is issue is about in **Short Description** column, detailed issue & comments by internal executive in **Description** column in customer words or the final resolution by team if the case is resolved, the encrypted caller details in **Caller** column and the **Assignment group** mentioning to which department or group the ticket has been allocated to. Database details and One sample of a ticket has been given below Fig 1.1 & 1.2 respectively for reference.

| Sr. No | Column Name | Description | Data Type |
|---|---|---|---|
| 1 | Short description | short Issue description | Object |
| 2 | Description | Complete issue provided by user in detail | Object |
| 3 | Caller | Encrypted Caller Address | Object |
| 4 | Assignment group | Allocated Department | Object |

**Fig – 1.3**

| Short description | Description | Caller | Assignment group |
|---|---|---|---|
| Unlock logon password | received from: cnhkypxw.lafncksi@gmail.com<br><br>hello,<br>please help to unlock my company net log on password. urgent. | cnhkypxw lafncksi | GRP_0 |

**Fig – 1.4**

### 7.2 EDA

In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Various EDA techniques were performed on given database in order to understand the structure of database and below conclusions were made.

**Data Structure** - The database has 4 columns and 8500 rows. All columns are of 'Object' datatype. All the incidents (Tickets) were allocated amongst 73 groups in target column i.e. Assignment group

**Missing Values** - Data had 8 null values in Short description (1) column and 1 null value in Description (2).

**Target Column** - Assignment group (4) is the target column which has 73 subgroups for categorization

Data Inconsistency - Database has below mentioned inconstancies.

- Presence of unwanted characters and symbols
- Meaningless keywords (e.g. Jdshfkshdfjh)
- Email address and associated common keywords (e.g. @, .com)
- Presence of punctuation
- Ip addresses
- Imbalance in target column amongst assignment group
- Encrypted caller details in caller (3) group
- Spelling mistakes
- Autogenerated numbers by some source
- Presence of Capital and small letters

### 7.3 Data Visualization

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.We performed some of the regular data visualization techniques like Groupby function, bar graph, word cloud etc to understand the database and our finding visually.
As we have subgroups in our target column i.e. Assignment group we checked the distribution of data among these subgroups with the help of 'groupby()' function

```
data.groupby(['Assignment group']).count()
```

| Assignment group | Short description | Description | Caller |
|---|---|---|---|
| GRP_0 | 3969 | 3975 | 3976 |
| GRP_1 | 31 | 31 | 31 |
| GRP_10 | 140 | 140 | 140 |
| GRP_11 | 30 | 30 | 30 |
| GRP_12 | 257 | 257 | 257 |
| ... | ... | ... | ... |
| GRP_71 | 2 | 2 | 2 |
| GRP_72 | 2 | 2 | 2 |
| GRP_73 | 1 | 1 | 1 |
| GRP_8 | 661 | 661 | 661 |
| GRP_9 | 252 | 252 | 252 |

74 rows × 3 columns

**Fig – 1.5**

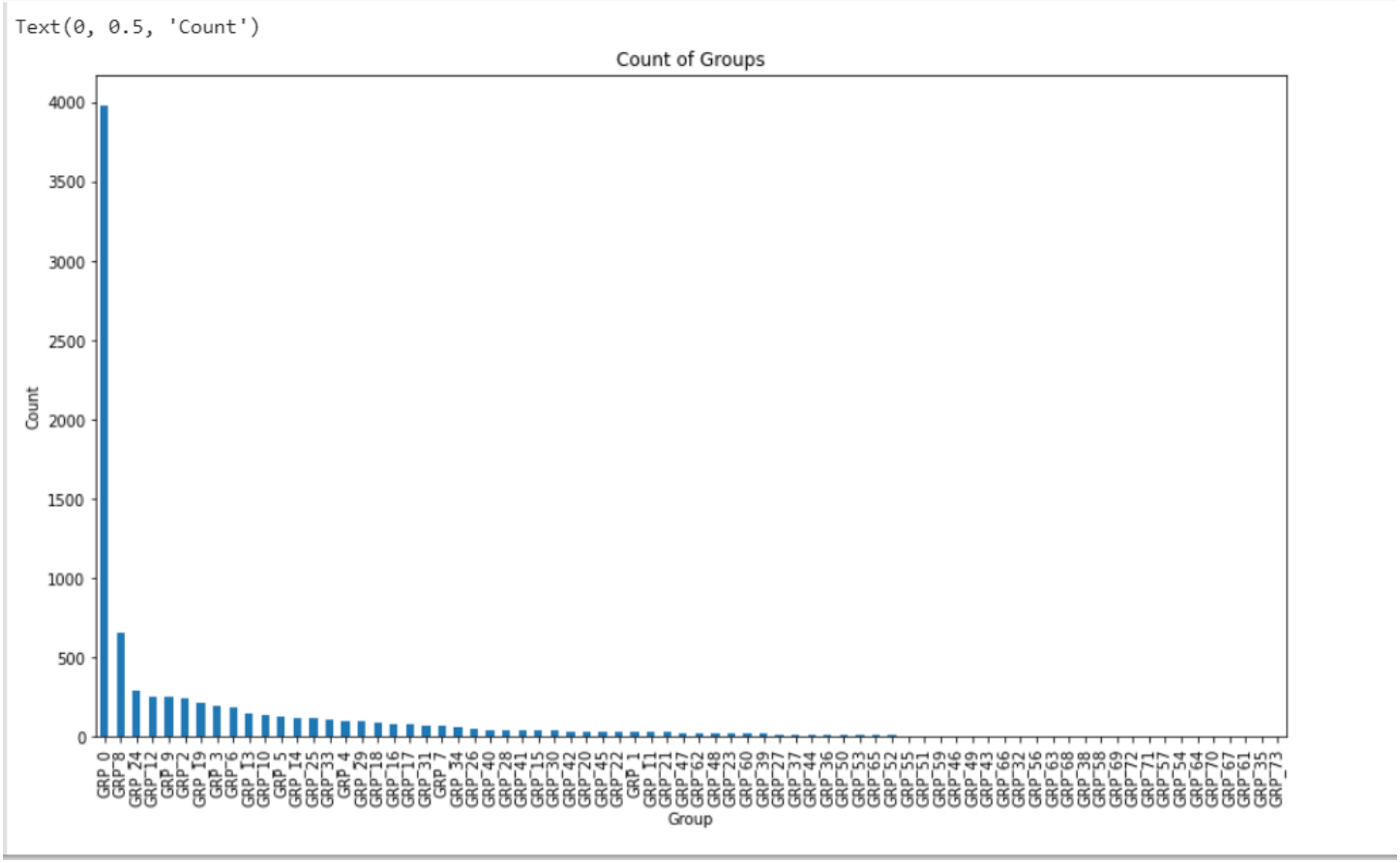



**Fig – 1.6**

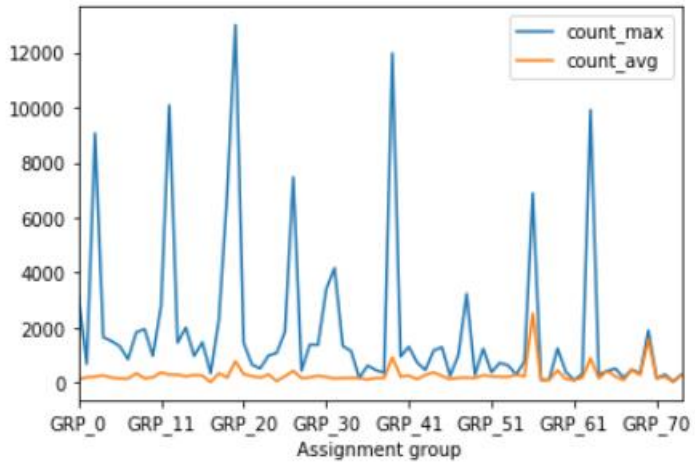
We can see from output and graph that the distribution of tickets is not evenly distributed. Means there are some incidents which are very frequent while some are just 1 or 2. Instead of dealing with so much of subgroups and to make the prediction clear we can merge or drop the subgroups with numbers less than 10.

```
df_groupby = data.groupby(['Assignment group']).count()
df_insuff = df_groupby.query('Description<10')
df_insuff
```

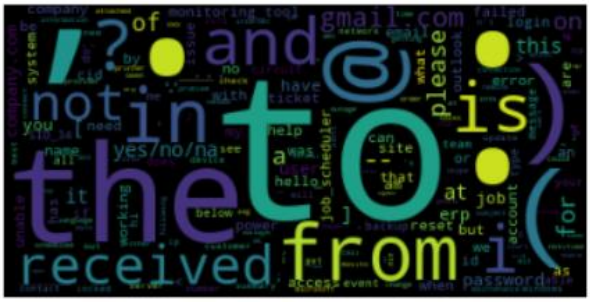
| Assignment group | Short description | Description | Caller |
|---|---|---|---|
| GRP_32 | 4 | 4 | 4 |
| GRP_35 | 1 | 1 | 1 |
| GRP_38 | 3 | 3 | 3 |
| GRP_43 | 5 | 5 | 5 |
| GRP_46 | 6 | 6 | 6 |
| GRP_49 | 6 | 6 | 6 |
| GRP_51 | 8 | 8 | 8 |
| GRP_52 | 9 | 9 | 9 |
| GRP_54 | 2 | 2 | 2 |
| GRP_55 | 8 | 8 | 8 |
| GRP_56 | 3 | 3 | 3 |
| GRP_57 | 2 | 2 | 2 |
| GRP_58 | 3 | 3 | 3 |
| GRP_59 | 6 | 6 | 6 |

**Fig – 1.7**

For all subgroup we plotted the graph of average word count and the maximum words in the description column and the outputs can be seen very interestingly with the difference in avg count and max count. By performing various EDA and other techniques we can clean the database for effectively for further processing.




The concept of word cloud was also performed on the database to check the top 50 words visually, and we can see that there are lots of unwanted data to be removed



### 7.4 Pre-Processing

In Preprocessing steps below mentioned steps were carried out step by step.

- Removed the rows with missing 1 and 2.
- Removed unwanted characters from 1 and 2.
- Removed all the email addresses from the 1 and 2.
- Removed all stop words from 1 & 2.
- Removed punctuation from 1 & 2.
- Spelling correction and removed common words.
- We tokenized the words and checked the word cloud.
- Lemmatization of the token words

Note – During preprocessing we have not performed any type of stemming and lemmatization.

### 7.5 Feature Engineering

Based on BoW (Bag of Words) approach the 'Short Description' and 'Description' columns were preprocessed as per preprocessing mentioned in 7.4 after preprocessing both the columns, we created 2 different BoW. Both the preprocessed and BoWs' we merged to one and performed label encoding.

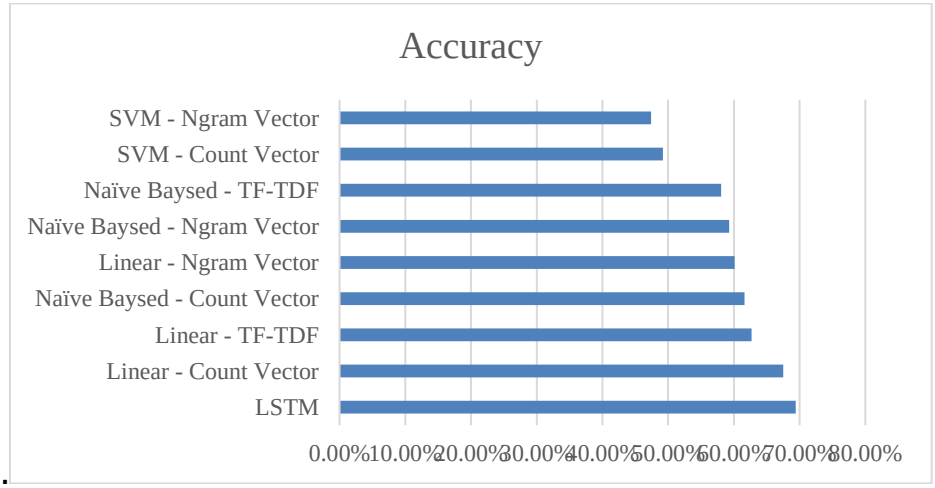
We performed train test split on the database and created 3 approaches for the classification.

1. Created count vectorizer object
2. TF-IDF (N-gram, word level and character level)
3. Word embedding

### 7.6 Classification

Based on above feature engineering techniques we chose Linear regression, Naïve Baysed, SVM method of supervised learning and LSTM neural network. The results are mentioned below.

| Model | Accuracy |
|---|---|
| Linear - Count Vector | 67.51% |
| Linear – N-gram Vector | 60.10% |
| Linear - TF-TDF | 62.67% |
| Naïve Baysed - Count Vector | 61.60% |
| Naïve Baysed – N-gram Vector | 59.27% |
| Naïve Baysed - TF-TDF | 58.04% |
| SVM - Count Vector | 49.20% |
| SVM – N-gram Vector | 47.42% |
| SVM - TF-TDF | 47.42% |
| LSTM | 69.40% |



## 8. Model Improvement

B above mentioned models are trained on the merged short description and description columns where as in order to improve the accuracy we performed below mentioned steps on short descriptions only and All the preprocessed data is then saved in a new CSV file to avoid repetition of same tasks.

- All preprocessing steps mentioned in 7.4 are performed on short description only.
- Lexical analysis
- Spacy language detector to deal with words other than English.
- Stemming of short description
- Lemmatization of short description

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

We loaded saved CSV file, run through the deep learning model, performed glove embedding, padding and which was trained and has saved weights on the model and checked for the final accuracy. Below are the final Results.

### 8.1 Using Deep learning for text classification –

Text classification is a typical use case where neural networks are a good fit. For the purpose of this academic project , we tried out different models. We started with a basic single layer LSTM model and then progressively tried out different combinations since the initial accuracy levels were in the low sixties.

I have highlighted the key models that gave us optimal results among all the different trials. We have also depended on publicly available technical papers and documentation to push our model's accuracy.

For all models, we followed a systematic approach whose key elements are detailed below -

1. For all models, we did the training and validation with 80% of data. Of this the validation split was 20% of initial 80% of data. So that's 16% of original data.
2. We used the below approach for ensuring some amount of cross-validation in the initial training & test data.
   - **a.** The Full data of 8500 rows was split into training and test data with 80-20% split.
   - **b.** Then we looped through the 80% for 5 times. Each time, we randomly split and trained the training data (into training(80%) and validation(20%). The objective was to do some amount of cross-validation with the training and validation. We explored using kfold cross validation from scilearn but was not able to get it to work for a multi-label indicator.
   - **c.** We experimented with the learning rate for Adam optimizer a little and finally settled on .002 since that was giving a good balance of learning time and reducing overfitting.
   - **d.** We used the earlystopping and model checkpoints to prevent overfitting and save the best model. We also used a patience value of 3 to 5 to ensure that the execution doesn't immediately stop in case the validation accuracy drops.
   - **e.** The test data was exposed to the model only for final testing after all of the above processing was completed.

| Model Details | Embedding | Data Column considered | Training/Validation/Test Split | Overall Accuracy |
|---|---|---|---|---|
| Conv1d + Bidirectional LSTM | Glove | Short Description | 64%/16%/20% | 71.47% |
| Conv1d + Bidirectional LSTM | Glove | Description | 64%/16%/20% | 78.50% |
| Convolutional Neural Network | Char based embedding. | Short Description | 64%/16%/20% | 77.80% |
| Convolutional Neural Network | Char based embedding | Description | 64%/16%/20% | 72.41% |

**Detail of Models -**

Conv1d + Bidirectional LSTM Model With Glove embedding -
Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding (Embedding) | (None, 300, 300) | 6000000 |
| conv1d (Conv1D) | (None, 296, 128) | 192128 |
| max_pooling1d (MaxPooling1D) | (None, 148, 128) | 0 |
| dropout (Dropout) | (None, 148, 128) | 0 |
| bidirectional (Bidirectional | (None, 148, 256) | 263168 |
| flatten (Flatten) | (None, 37888) | 0 |
| dense (Dense) | (None, 100) | 3788900 |
| dropout_1 (Dropout) | (None, 100) | 0 |
| dense_1 (Dense) | (None, 74) | 7474 |

Total params: 10,251,670
Trainable params: 4,251,670
Non-trainable params: 6,000,000

Char CNN Based Model with Alphabel embedding -
Full Alphabtet -
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

CharCNN model built:
Model: "model_2"

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| sent_input (InputLayer) | [(None, 128)] | 0 | |
| embedding_2 (Embedding) | (None, 128, 128) | 10880 | sent_input[0][0] |
| Conv1D_256_10 (Conv1D) | (None, 119, 256) | 327936 | embedding_2[0][0] |
| Conv1D_256_7 (Conv1D) | (None, 122, 256) | 229632 | embedding_2[0][0] |
| Conv1D_256_5 (Conv1D) | (None, 124, 256) | 164096 | embedding_2[0][0] |
| Conv1D_256_3 (Conv1D) | (None, 126, 256) | 98560 | embedding_2[0][0] |
| MaxPoolingOverTime_256_10 (Glob | (None, 256) | 0 | Conv1D_256_10[0][0] |
| MaxPoolingOverTime_256_7 (Globa | (None, 256) | 0 | Conv1D_256_7[0][0] |
| MaxPoolingOverTime_256_5 (Globa | (None, 256) | 0 | Conv1D_256_5[0][0] |
| MaxPoolingOverTime_256_3 (Globa | (None, 256) | 0 | Conv1D_256_3[0][0] |
| concatenate_2 (Concatenate) | (None, 1024) | 0 | MaxPoolingOverTime_256_10[0][0] |

MaxPoolingOverTime_256_7[0][0]
MaxPoolingOverTime_256_5[0][0]
MaxPoolingOverTime_256_3[0][0]

| | | | |
|---|---|---|---|
| dense_6 (Dense) | (None, 1024) | 1049600 | concatenate_2[0][0] |
| alpha_dropout_4 (AlphaDropout) | (None, 1024) | 0 | dense_6[0][0] |
| dense_7 (Dense) | (None, 1024) | 1049600 | alpha_dropout_4[0][0] |
| alpha_dropout_5 (AlphaDropout) | (None, 1024) | 0 | dense_7[0][0] |
| dense_8 (Dense) | (None, 74) | 75850 | alpha_dropout_5[0][0] |

Total params: 3,006,154
Trainable params: 3,006,154
Non-trainable params: 0

### 8.2 Observations

The best accuracy was given by the CNN + Bidirectional LSTM for description. This is probably because the description has a lot of text and also some semantic meaning for the entire text. Using the glove embedding, the above model is able to use this text for enhanced accuracy. However it's interesting to see that the accuracy is not so high for the short description. I had referred some examples here to come up with this network.

https://keras.io/examples/imdb_cnn_lstm/

The CNN based model with alphabet based embedding actually gives better accuracy for the short description. The way to understand this model is that the short description has less text and hence the alphabet based embedding works well. There is no semantic information captured here but since the text is less it probably doesn't matter. I have given the link to the paper on this topic and also the github location where there was some code that i was able to reuse.

https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

https://github.com/blues-lin/Char-level-CNN-for-Text-Classification-in-Keras

It's interesting to see the use of SELU as an activation function by this network.

## 9. Conclusion

The manual assignment of issue tickets to appropriate unit or person in support team is not feasible sufficiently for large organizations. It is time consuming and there may be mistakes due to human errors. In this study, to assign tickets automatically, a model based on Deep Learning algorithms is proposed. Dataset consisting of previously categorized tickets are used to train classification algorithms. Bag of words approach is utilized to extract features vectors. Morphological analysis of terms is performed to avoid data sparseness problem and decrease the vector size. Three different supervised and one deep learning 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. Consequently, the proposed approach reduces manual efforts and human errors while ensuring high service levels and improved end-user satisfaction. Also, the proposed system provides to a large organization better allocation and effective usage of the valuable support resources.

## 10. Deliverables

**A. Final Python Program & Data set - Model Based On "Description" Column**

- NLP_Using_pretrained_model_V3.1. ipynb (Requires Below Items)
    - Have tested results for all models
    - Dependent Deliverables to Test the Code
        - capstne_ticket_data.xlsx **(Original Dataset)**
        - desc_corrected_all.xlsx (Preprocessed dataset - "Description")
        - Best_Model_LSTM_Glove_Description.JSON **(Final Model Used)**
        - Best_Model_LSTM_Glove_Description.h5 **(Final Weights Used)**

Model: - CapstoneProject_with_Preprocessing_LSTM_description.ipynb

**B. Other Python files & Tested Options - Model Based on "Short description" Column**

- NLP_Using_pretrained_model_V2.1. ipynb (Requires below items
    - Have tested results for all models
    - Dependent Deliverables to Test the Code
        - capstne_ticket_data.xlsx (Original Dataset)
        - short_desc_corrected_all.xlsx (Pre-processed dataset - "Short description")
        - Best_Model_LSTM_Glove.JSON **(Model Used)**
        - Best_Model_LSTM_Glove.h5 **(Weights Used)**

Model: - CapstoneProject_with_Preprocessing_LSTM_Short_description.ipynb

**C. Other Python files & Tested Options - Deep-learning Model Based on "Short description" Column**

If in future, we decide to move to "Short Description", this model should be considered.

- CapstoneProject_with_Preprocessing_cnn_Pretrained.ipynb
- weights_best_Preprocesseddata_cnn. JSON
- weights_best_Preprocesseddata_cnn. hdf5

Model: - CapstoneProject_with_Preprocessing_cnn.ipynb

[The CNN based model with alphabet-based embedding actually gives better accuracy for the short description. The way to understand this model is that the short description has less text and hence the alphabet-based embedding works well. There is no semantic information captured here but since the text is less it probably doesn't matter.]