AIML ONLINE CAPSTONE - AUTOMATIC IT TICKET ASSIGNMENT

By,

Deepkumar Aravindkumar Patel

Harika.M

Madhumitha.R

Rachit Vaid

Table of Contents

[Introduction 3](#_heading=h.gjdgxs)

[Business Case 3](#_heading=h.30j0zll)

[Data Collection 4](#_heading=h.1fob9te)

[Challenges to be Expected 4](#_heading=h.3znysh7)

[Exploratory Data Analysis 4](#_heading=h.2et92p0)

[Data Pre-Processing 5](#_heading=h.tyjcwt)

[Creation of train and test sets 11](#_heading=h.3dy6vkm)

[Model Building 12](#_heading=h.1t3h5sf)

[Traditional Machine Learning 12](#_heading=h.4d34og8)

[Deep Learning 12](#_heading=h.2s8eyo1)

[Summary 13](#_heading=h.17dp8vu)

[Fine-Tuning 14](#_heading=h.3rdcrjn)

[Data Centric Approach 14](#_heading=h.26in1rg)

[Model Centric Approach 14](#_heading=h.lnxbz9)

# Introduction

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 the Incident Management process to achieve the above Objective. An incident is something that is an 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 the 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.

# Business Case

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

Currently the 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 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. In case 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. In case 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.

# Data Collection

The Organizational IT Service Management Tool is the primary source for project data related to the incidents. Details from all the resolved incidents are collected to form the data for this project. The incidents have features like Short Description, Description, Caller, Assignment Group, etc.

# Challenges to be Expected

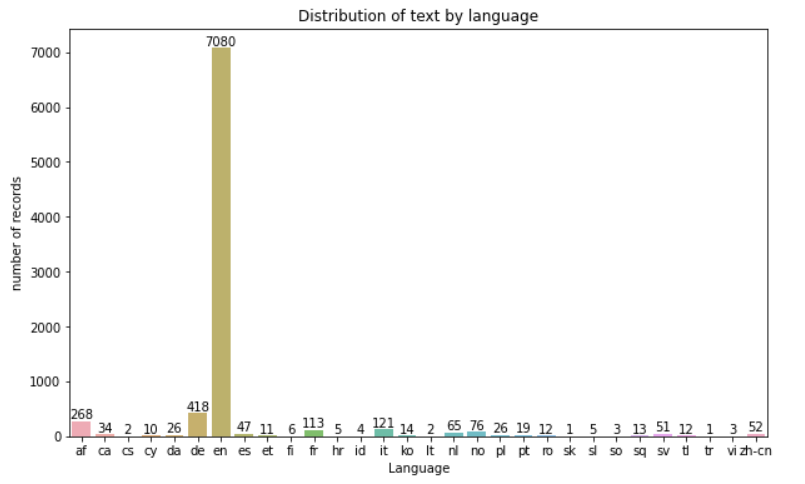
There is no guarantee that the data we receive from the IT Service Management Tool would be clean. There might be several issues with the data and we must be ready to deal with them. Below are some of the issues that can be expected:

* Null or missing values in the tickets. The support teams sometimes fail to fill all the necessary details in the ticket while closing it. We can either remove such tickets from the dataset or perform some imputation to deal with such problems.
* Presence of special characters in the data might be problematic. While building models, we need to ensure that the data fed to the model is clean. Not providing proper data to the model might lead to incorrect results. We can use Regex to remove all the non-word characters which are of no significance to the model.
* Ticket description written in a foreign language. If the IT team is supporting a global application, there are high chances that the description or other details could be written in a different language. In such cases, we either need to manually translate the sentences to a Base language (English in our case) or write additional code to perform the translation based on the volume of data.
* Class imbalance in Multi Class Classification problems might lead to incorrect results. In such cases, we need to use techniques like up-sampling and down-sampling to handle the class imbalance.

# Exploratory Data Analysis

After loading the data from the excel sheet, we did some basic EDA to get an idea on the look and feel of the dataset. The below are some observations:

1. The dataset has 8500 rows/tickets and 4 columns/properties.
2. Each ticket has a short description, a description, caller and the assignment group. All the values are of type ‘object’.
3. There are 74 unique assignment groups in the dataset.
4. About 46.8% of the total tickets are assigned to Group\_0. The rest is distributed among the remaining assignment groups. This shows that there is high Class Imbalance in the target.
5. There are 9 rows in the dataset which contain NaN. All the NaN values have been converted to type string. The rows which previously had NaN values now contain the text ‘NaN’.
6. On careful observation, few tickets in the dataset had descriptions written in a different language. On translating the words manually, we found them to be in German. This indicates that the dataset has tickets written in multiple languages.
7. EDA indicated the presence of foreign languages in the dataset. To find out the languages, we have used the detect function from the ‘langdetect’ library. There are a total of 30 languages used in the dataset. A simple bar chart visualization showed that most of the tickets were written in English followed by tickets in German.



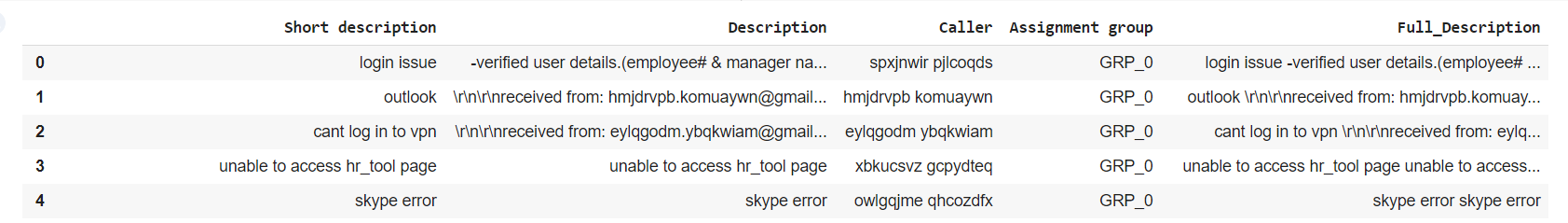
**Figure 1: Distribution of languages in the text**

# Data Pre-Processing

The dataset provided to us has 3 features and 1 target. But we cannot directly feed it to the model as input as the data isn’t clean yet. It contains special characters, stop words and lots of irrelevant text ‘sent’, ‘email’, ‘from’, etc. To get high model accuracy, we need to ensure that the model input is clean.

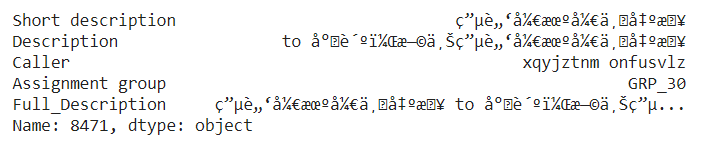
The following are some methods implemented to clean out the text in the dataset:

1. We merged the ‘Short description’ and ‘Description’ columns to create a new feature called ‘Full\_Description’.

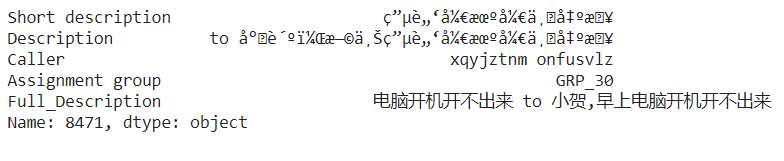


**Figure 2: Creation of new feature**

1. When dealing with text written in multiple languages, there’s a high chance of encountering Mojibakes. Mojibake often occurs when a character coding is incorrectly tagged in a document, or when a document is moved to a system with a different default coding than its preceding location. Using the ‘ftfy’ library, we have identified the mojibakes in the dataset and fixed them.

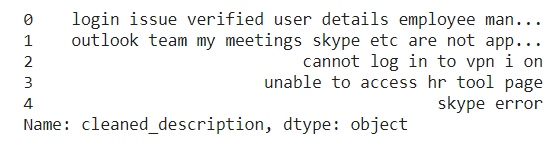


**Figure 3: Original text**



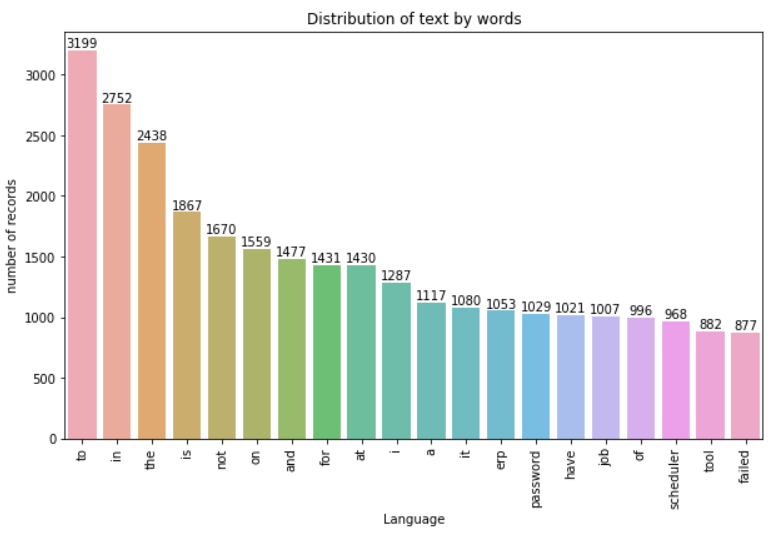
**Figure 4: Text after removing mojibakes**

1. With the help of regex, we removed all the irrelevant content from the new feature ‘Full\_Description’. Irrelevant text includes all the special characters, whitespaces, URLs, email addresses, salutations, etc.



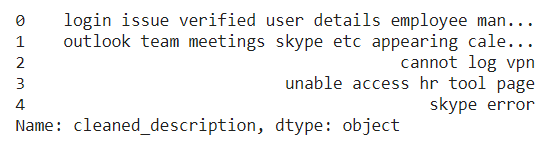
**Figure 5: Clean text**

1. Calculated word count to find out the most frequent words. Most of the words were stop words that do not add any meaning to the content.



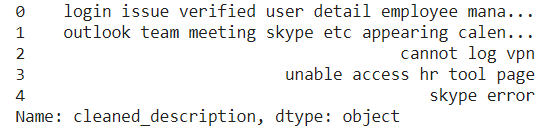
**Figure 6: Bar chart for most frequent words**

1. With the help of ‘stopwords’ corpus from nltk, we remove all the stop words and create ‘cleaned\_description’.



**Figure 7: Text after removing stopwords**

1. We performed Lemmatization on the ‘cleaned\_description’ to find the root form or the lemma of the words.



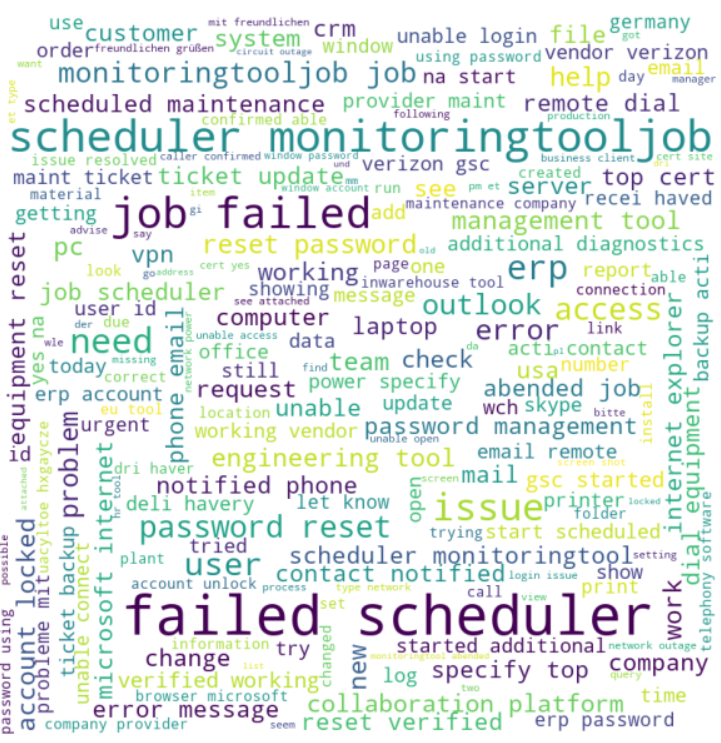
**Figure 8: Lemmatised text**

1. After lemmatizing the text, we dropped those rows whose description contained only one word.
2. A lot of visualizations were done like bar chart, pie chart and pareto chart to check the distribution of the tickets among various groups. Some of the observations are listed below:

* Top 5 groups with highest tickets are - 0, 8, 24, 12 and 9.
* Bottom 5 groups with lowest tickets are - 70, 64, 67, 73 and 35.
* There are 40 groups with just 30 or less tickets assigned, amongst which 6 groups happen to be assigned with just 1 ticket, 4 groups with just 2 tickets each and 5 groups with 3 tickets each.
* There are 15 assignment groups that have more than 100 tickets which accounts to only 20.3% of the overall dataset.

1. WordClouds were created for the whole dataset and a few individual groups. The findings are as follows:

* Group 0 seems to have issues related to password reset, access, login, etc.
* Groups that have words similar to Group\_0 are 2, 7,12,14,17, etc.
* Group 8 has issues related to monitoring tool, scheduled maintenance, outage, job failure, etc.
* Groups that have similar words as that of Group\_8 are 9, 6, 5, 10, 4, 47, 0, 45, 12, 1, 13, 14, 29, 18, etc.
* Group\_12 tickets mostly revolve around server, asa deny, dst outside, outside access.
* Group\_24 consists of German language which needs translation.



**Figure 9: WordCloud created for cleaned description**

1. Analysis was done on the caller column and WordClouds were created. The findings are as follows:

* Top 10 frequent callers were identified from the database.



**Figure 10: Top 10 frequent callers**

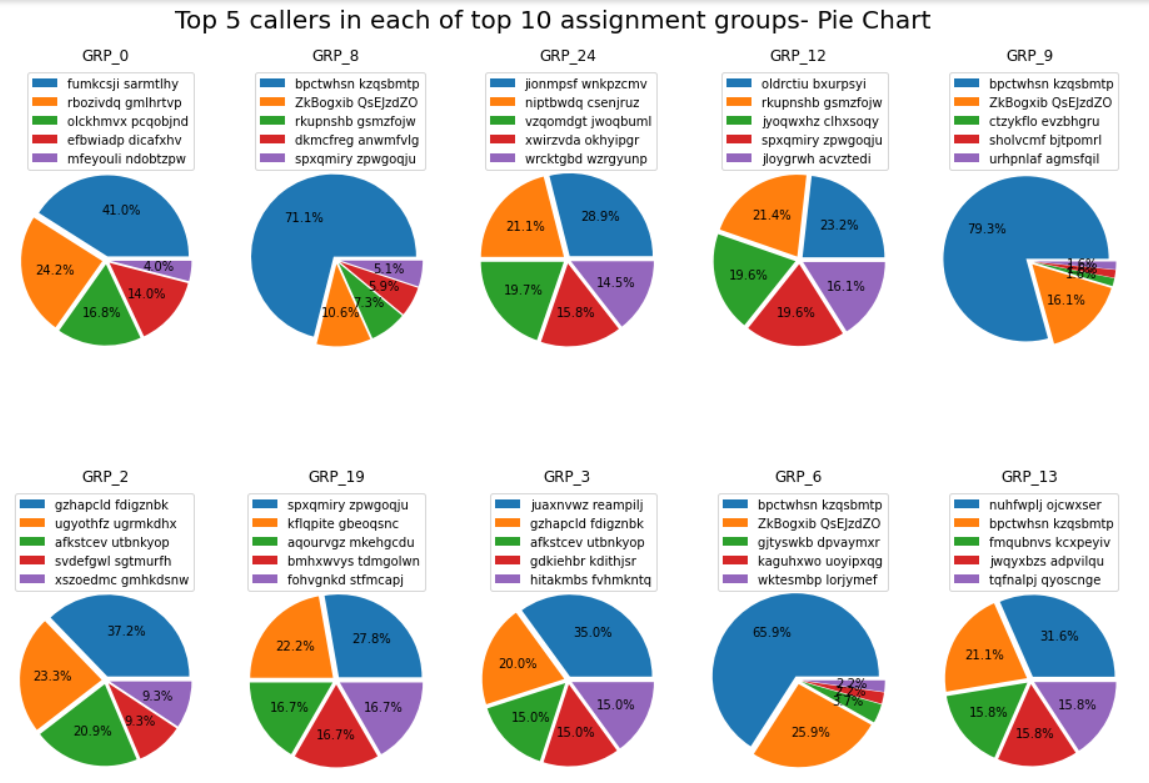
* The top callers ‘pctwhsn kzqsbmtp’ & ‘ZkBogxib QsEJzdZO’ mostly addressed issues related to job scheduler and monitoring tool.
* Caller ‘fumkcsji sarmtlhy’ addressed issues related to ticket updates and specific ticket related issues.
* Tickets created by caller ‘rbozivdq gmlhrtvp’ are mostly related to basic front desk resolvable issues such as loud noise, general enquiry, loud noise, etc.



**Figure 11: WordCloud created for caller**

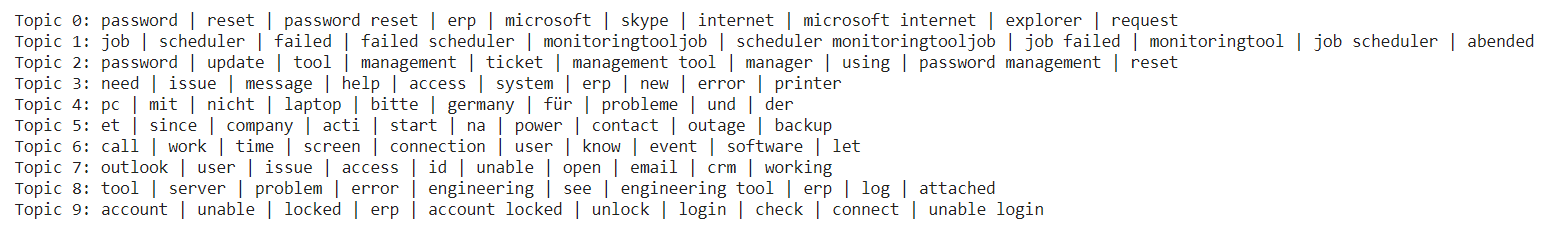
1. Performed some analysis on the relation between caller and assignment group and found the below insights:

* There are overall 2950 callers in the dataset.
* 15 callers are involved in raising tickets for multiple assignment groups, overall contributing to 281 tickets in the dataset.
* The top 5 caller ticket distribution in every group is widely distributed, almost having a balanced distribution. This indicated that most of the group resolves issues widely over various fields impacting business.



**Figure 12: Distribution of top 5 callers in top 10 assignment groups**

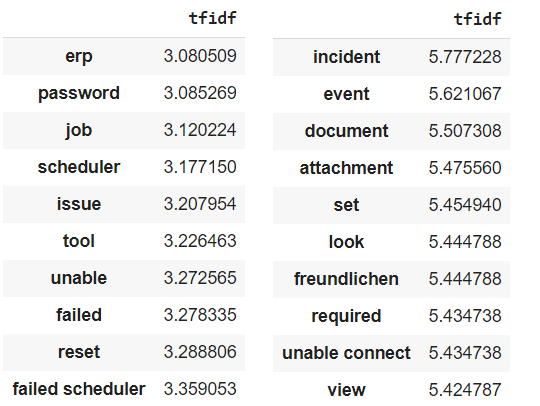
1. Unigram, bigram and trigram models are used to get the most frequent words or phrases from the cleaned description.
2. Used LDA modelling (Latent Dirichlet Allocation) to discover topics that are hidden in the cleaned description. The below are few topics and the relevant words in each topic:



**Figure 13: Topics created using LDA model**

1. Performed TF\_IDF i.e., the term frequency – inverse document frequency to find out the relevancy of the words in the corpus. Few observations are listed below:

* Words that have low significance in the dataset are ‘Erp, job, passwords, jobscheduler, failed’ etc.
* Words that have high significance in the dataset are ‘outside accessgroup, dst outside, src inside, src, accessgroup’, etc.



**Figure 14: Tf-Idf values**

# Creation of train and test sets

To create the train and test sets, we first had to create the X and Y variable. X contains the features and Y contains the targets. The creation of X and Y variables depend on the type of model used. The input given to a model depends on the underlying algorithm and other factors. Hence, we followed the below steps to create the train and test sets:

1. We took the tf-idf vectors created previously in the Data Pre-Processing step and converted it to an array. The values were stored in a variable called X.
2. Performed Label Encoding on the Assignment Group column which is the original target. On performing label encoding, we got a list of numbers for the targets. These label-encoded values were put in a new column in the DataFrame and also in a new variable called Y.
3. Using the X and Y variables and the train\_test\_split function, we got our train and test sets.
4. For the traditional Machine Learning models, we used the train set created above for training the model.
5. For the deep learning model, we had to do further processing on the data. Word embedding was created for the ‘cleaned\_description’ column using techniques like bag of words, word2vec, GloVe embeddings, etc. One-hot encoding was performed on the Y variable.

# Model Building

The targets present in the dataset are categorical values indicating that it is a classification problem. Preliminary analysis done on the data indicated that we had multiple classes in the target. Thus, we need to implement algorithms that support multiclass classification.

There are several models/algorithms available that could help us with the given problem. We could use the simple models which involve traditional Machine Learning algorithms. And we could also implement deep learning models which are highly used in the field of Artificial Intelligence. Our approach here is to divide the Modelling into two phases, one using Machine Learning and other using Deep Learning models.

## Traditional Machine Learning

There are several algorithms in Supervised Machine Learning that support multi class classification problems. Below are some algorithms that we have implemented in this project:

1. Multinomial Logistic Regression
2. Multinomial Naïve Bayes
3. K Nearest Neighbor
4. Stochastic Gradient Descent
5. Support Vector Machine
6. Decision Tree
7. Random Forest
8. XGBoost

Most of the models performed moderately in the test set. The accuracy of all the models were in the range of 37-65%. Random forest, XGBoost and Decision Tree seem to be overfitting due to high difference in training and test accuracy. K-NN, XGBoost and Random Forest seem to handle imbalance better than SGD. Meanwhile, SGD, SVM and Naïve Bayes might perform better if the class imbalance is taken care of.

## Deep Learning

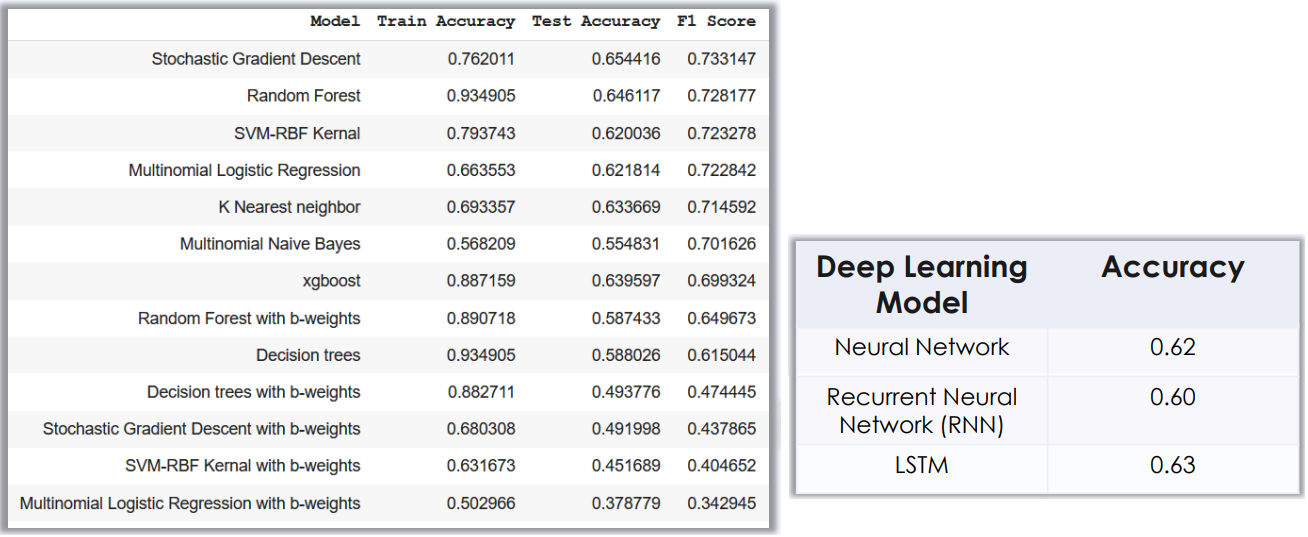
As for the deep learning, we have selected the below algorithms considering the type and the amount of data present:

1. Neural Network
2. Recurrent Neural Network
3. LSTM

For the above models, we used ‘ReLu’ and ‘Softmax’ as the activation functions, ‘SGD’ as the optimizer and ‘categorical\_crossentropy’ as the loss. All the models gave an accuracy of 60% approximately. LSTM/RNN improvement can be achieved with further word embeddings such as bag of Words, word2vec,

## Summary

Both the traditional Machine Learning models and Deep Learning models have given similar accuracies of approximately 62%. The below tables list out all the models used and their accuracies.



**Figure 15: Traditional ML vs AI models**

All the models have given pretty good accuracy in the training. But during testing, the accuracies dropped by a considerable amount. There are several reasons for this.

* High class imbalance in the dataset. This is one of the main reasons why all the models gave moderate accuracy. The model was highly biased towards groups that had a high number of tickets.
* Presence of noise in the dataset. Even though we cleaned the text and implemented several feature engineering techniques, there’s a high possibility that noise was fed to the model during training. This led to a low performance on the test set.
* Overfitting on the train set might have led to poor performance in the test set.

# Fine-Tuning

As mentioned above, there are several reasons that can lead to poor model performance. The below two approaches can be used to improve the model performance.

## Data Centric Approach

The performance of any model depends on the input that we feed to it. As the saying goes, “Garbage In, Garbage Out”. This means the quality of the output depends on the quality of the input. Having good data is especially important in machine learning and deep learning, which gain greater capabilities over time by analysing large sets of data, learning from them and ultimately making adjustments that make the applications more intelligent.

Even though we performed pre-processing on the dataset, there were several shortcomings which reduced the model performance. The below are some steps that can be used to overcome these shortcomings.

* Remove class imbalance in the dataset
  + Up-sampling – Increase the count of less frequent through up-sampling.
  + Down-sampling – Decrease the count of the top frequent groups via down-sampling.
  + Use a threshold like removing groups with less than 30 tickets each.
  + Merging the less frequent groups into one group.
  + Remove or group tickets with foreign languages.
* Add more features such as word count, average, LDA\_topics, etc.

## Model Centric Approach

To improve the performance, we can tweak the existing models or create new ones using advanced algorithms. The below are a few steps that can be implemented.

* Hyper parameter tuning. Adjust the hyper-parameters of the model to get the best performance. GridSearchCV and RandomSearch CV are two algorithms that can help us find the best parameters for the model.
* Using Ensemble models like Stacking to get a better performance. We could stack all the top performing models and create an ensemble model that would give a better performance instead of a plain model.
* Using transfer learning to leverage the capabilities of highly advanced pre-trained models and achieve a better performance.
* As this problem is related to NLP, we can use advanced NLP models like BERT, XLNet, etc.