AUTO-TICKET ASSIGNMENT USING NATURAL LANGUAGE PROCESSING

Capstone Project - AIML

**Group- 11**



**Group Members**

Kavitha Chetana Didugu

Ayush

Prashant Gautam

Archit Narang

**Introduction**

For small organisations and large alike, customer support is a very important part of customer retention. Tickets raised by customers need to be directed to the right teams for speedy resolution. However, assigning tickets to the right support team is in itself a challenge. Existing systems use manual triage of tickets, which is error prone, and is also a very tedious process. For organisations with clients all over the world, this is even more complicated since it requires understanding tickets raised in various languages. There are issue tracking systems (ITS) which automate the process to a certain extent, but currently they still have to rely on human decision making.

As an attempt to automate the ticket assignment process, we propose machine learning based methods that improve the accuracy of ticket assignment. We take a natural language processing (NLP) approach called Topic Modelling, where we try to extract keywords and the context of each ticket and use them for ticket classification. To the best of our knowledge, this is the first attempt in proposing an automated ticket assignment method with multiple language support.

Keywords: ITS, Topic Modelling, LDA, text classification, embedding, language models, NLP

# **Problem Statement**

One of the core process areas of ITSM is Incident Management that aims to restore a normal service operation as quickly as possible to minimize the impact on business. Incident management works closely with the Service desk, which acts as a single point of contact for all the users to interact with the IT.

Whenever the service is disrupted or fails to deliver the promised performance during normal service hours, an incident is raised by the users, which is to be assigned to the right team. Assigning the Incident to the appropriate team is itself a huge challenge and even today the assignment process is performed manually and around ~25% of Incidents is wrongly assigned to functional teams. Re-assigning to the right team leads to waste of time and requires additional efforts resulting in poor customer service.

Summary of problem statement

The problem we intend to solve is that of assigning IT incident tickets to the right categories. It is quite common in any IT company that customer raised tickets are not always assigned to the right team by the customers. As a result, the tickets are rerouted multiple times before they reach the right team. This results in a higher time to resolution.

Objective

The paper attempts to remove the main bottleneck in the Incident assignment system by predicting the right functional group for an Incident. We are defining the objective as to build a classifier that, given an input of texts from the incidents, outputs a vector of a correct functional group for the incidents.

We predicate that machine learning algorithms can extend a human's capability to consume information through the automated assignment of incidents. A number of approaches were taken including SVM, Naive Bayes, Latent Dirichlet Allocation (LDA), Convolution Neural Networks (CNN), and Long Short Term Memory networks (LSTM).

Related work

We found various papers related using various algorithms for the Incident Assignment, which we are segmenting into three groups. Traditional Approaches, Topic Modeling and neural network approaches.

**1.Traditional Approaches-**

● Naïve Baye

● SVM

● Logistic Regression

● Random Forest

**2.Topic Modeling**

● LDA

**3.Neural Network based-**

● ANN

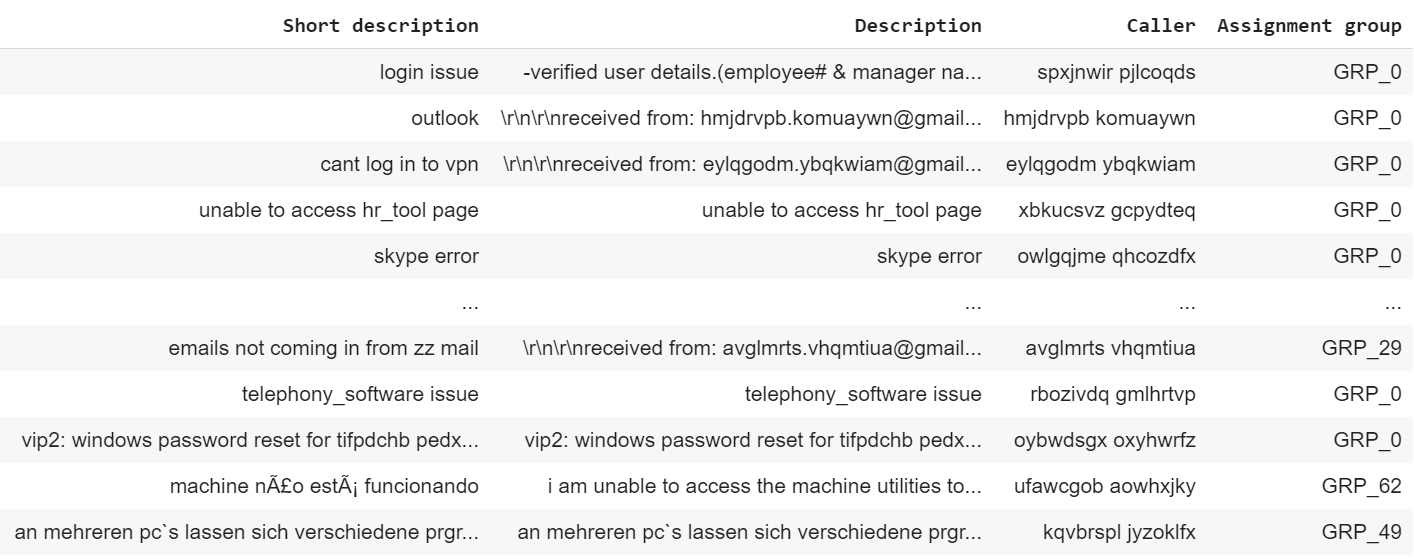
● LSAT

|  |  |  |  |
| --- | --- | --- | --- |
| **Technique** | | **Method** | **Pros/Cons** |
| **Naive Bayes** | | Supervised Learning Method | ● Simple and Robust  ● High speed  ● Ease of training  ● Requires large training data |
| **Support Vector Machine** | | Supervised Learning Method | ● High Predictive Power  ● Avoid Overfitting  ● Require good Kernel  ● Requires large training data |
| **Latent Dirichlet Allocation (LDA)** | | Unsupervised Learning Method | ● Provide Predictive & latent topic representation of corpus  ● Low cost |
| **Convolutional Neural Network (CNN)** | | Semi Supervised learning Method | ● Provide Predictive & latent topic representation of corpus better than LDA  ● Can capture the shape of the Topic  ● Faster and easy to train |
| **Long Short-Term Memory**  **(LSTM)** | | ● Unsupervised Learning Method  ● Also, It trains by itself so Self Supervised | ● Suited to learn from sequential experience when long time lags of unknown size exist between important events.  ● Faster and Robust |

## 

## Data

The dataset contains 8500 tickets in various languages, and belong to 74 different assignment groups. The raw input data consists of 4 columns: Short Description, Description, Caller and Assignment Group.



Below table shows basic statistics about the dataset. There is sufficient text data in order to run the algorithms for Incidents classification.

|  |  |
| --- | --- |
| Number of Columns | 4 |
| Total Incidents | 8501 |
| Total words in the Subject | 62142 |
| Total words in the description | 245456 |
| Average words per subject | 7.3 |
| Average words per description | 28.9 |
| Total No of Assignment Groups | 74 |

## Findings

1. 8500 records with 8 null values in Short Description column, and 1 null value in Description

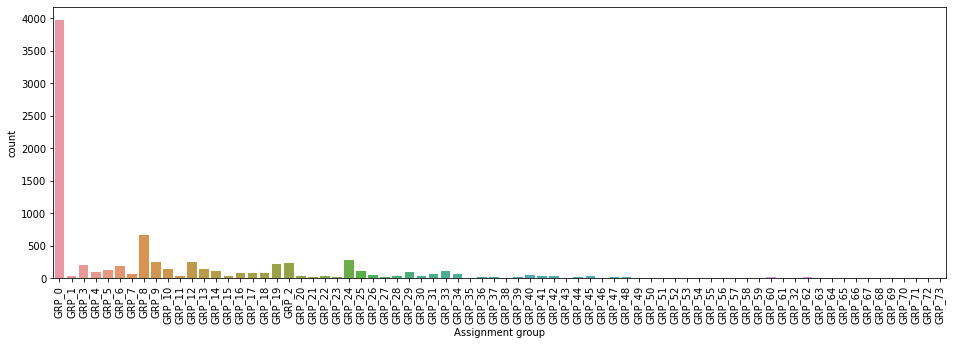
Short description 8

Description 1

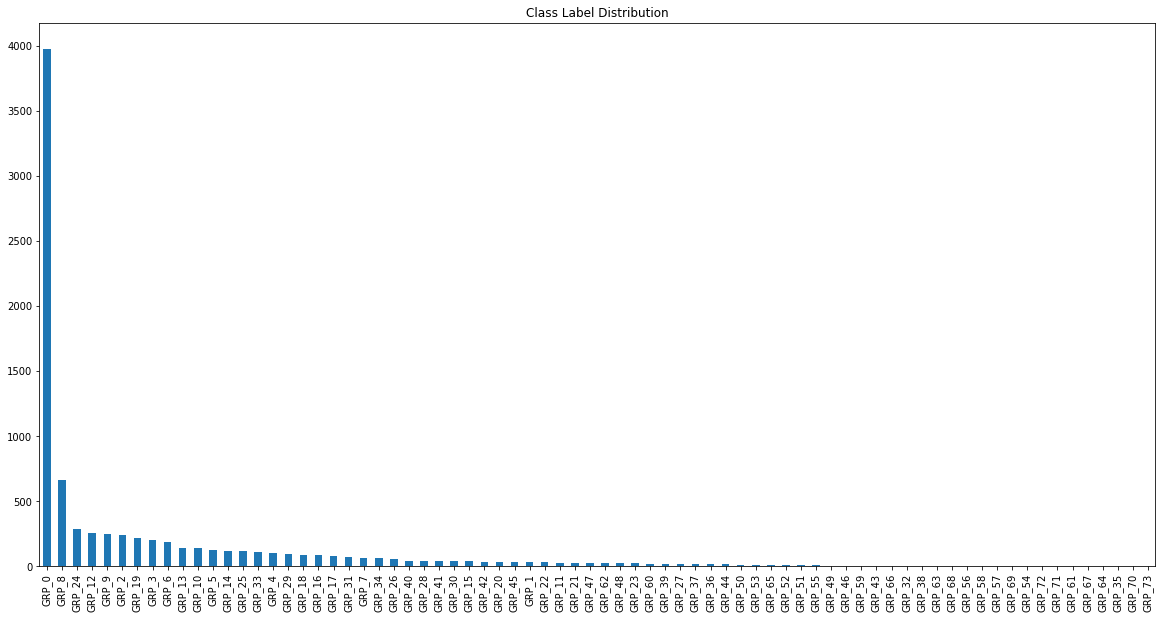
Caller 0

Assignment group 0

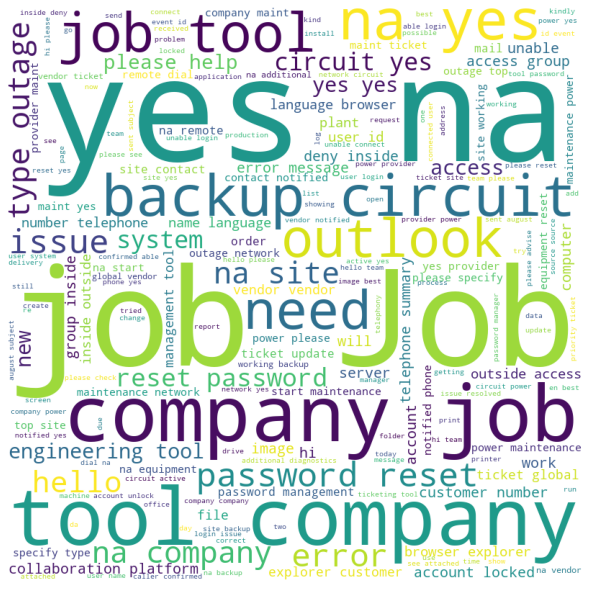
1. In a majority of the records, the content of the Short Description column is repeated in the Description column. This makes Short Description redundant. Hence it was dropped
2. Caller names appear in almost all Descriptions. Seems redundant, and also not influential in determining the assignment category. Hence it was dropped.
3. Distribution of tickets across assignment groups
   1. Ordered by group



* 1. Ordered by frequency



1. Most frequent words - Wordcloud



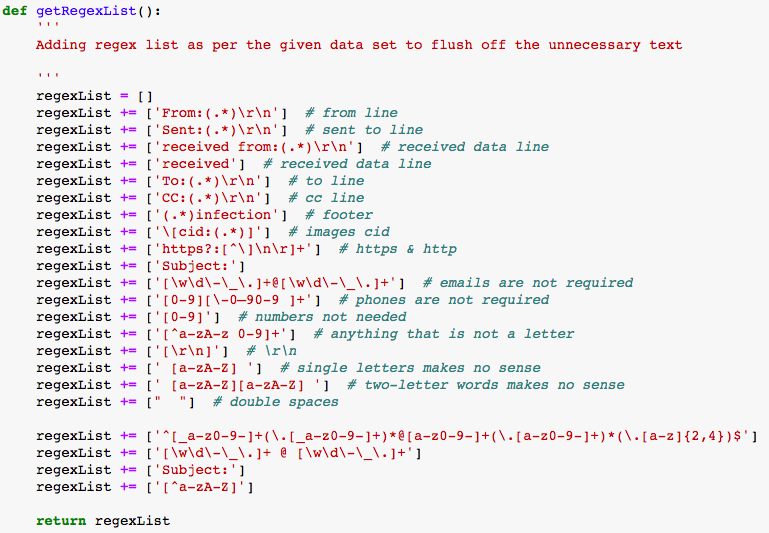
We see that some words such as Job, Company, Yes, Na, are most common, but do not indicate any particular incident or assignment group. However, relatively frequent are the phrases “reset password”, “account locked”, “collaboration platform”. We need a word vectorizer that can account for the high frequency of non-keywords and relative frequency of keywords. This led us to select the TfIDf Vectorizer.

# Overview of the final process

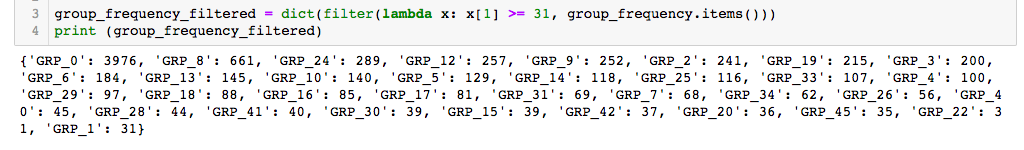
In our experiments, we used two different data pipelines. The first one involved reduction of the number of output classes, while the other retained the original number of output classes. The details of the two are given below.

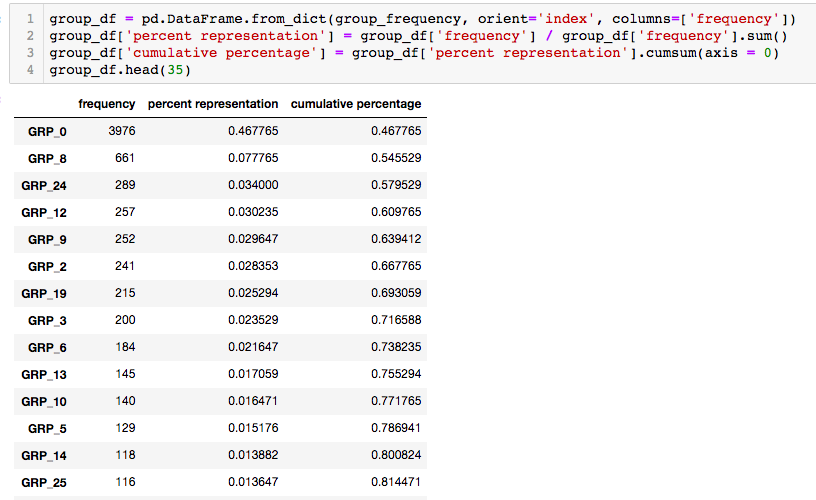
Data Pipeline 1:

1. Data Preprocessing using NLTK (regex based cleaning, stopword removal)

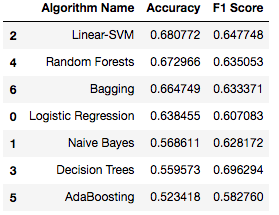


1. Limiting the dataset to only those GROUPS that have a minimum of 20 entries, to reduce noise. After removing statistically insignificant Groups from the dataset, the number of Groups reduced from 74 to 34. Number of rows in dataset reduce to 8113





1. Using CountVectorizer() and TfidfTransformer() on cleaned ticket description to create train and test sets
2. Evaluating models with a test set size of 30%, based on accuracy and F1 score



Data Pipeline 1 was used with the machine learning models while Data Pipeline 2 was used with deep learning models.

Step-by-step code walkthrough

Though we have built several classification models on our data, we broadly follow the same structured approach. We list below the steps of the approach as it appears in our code:

1. Data import, Library import, Data pre-processing
2. EDA - class distribution graphs, word clouds
3. Data pre-processing using NLTK library
   1. removing special characters
   2. converting to lower case
   3. tokenisation
   4. removing stopwords from various languages
   5. removing any sentences that contain less than 2 words
   6. stemming the tokens
   7. rejoining processed sentences
4. Creating Word Vector using TfIDf vectorizer
5. Building the classification model
6. Model fitting
7. Test prediction
8. Model accuracy metrics

Model evaluation

1. Our data is a multi-class classification data, hence we eliminated the use or AUC-ROC.
2. Since the classes were unbalanced, though we extracted the overall accuracy, we did not use it for model evaluation.
3. We used Precision, Recall and F1 score primarily, as evaluation metrics for our models.
4. For aiding in visualisation, we printed out the confusion matrices for the deep learning models.

Comparison to benchmark

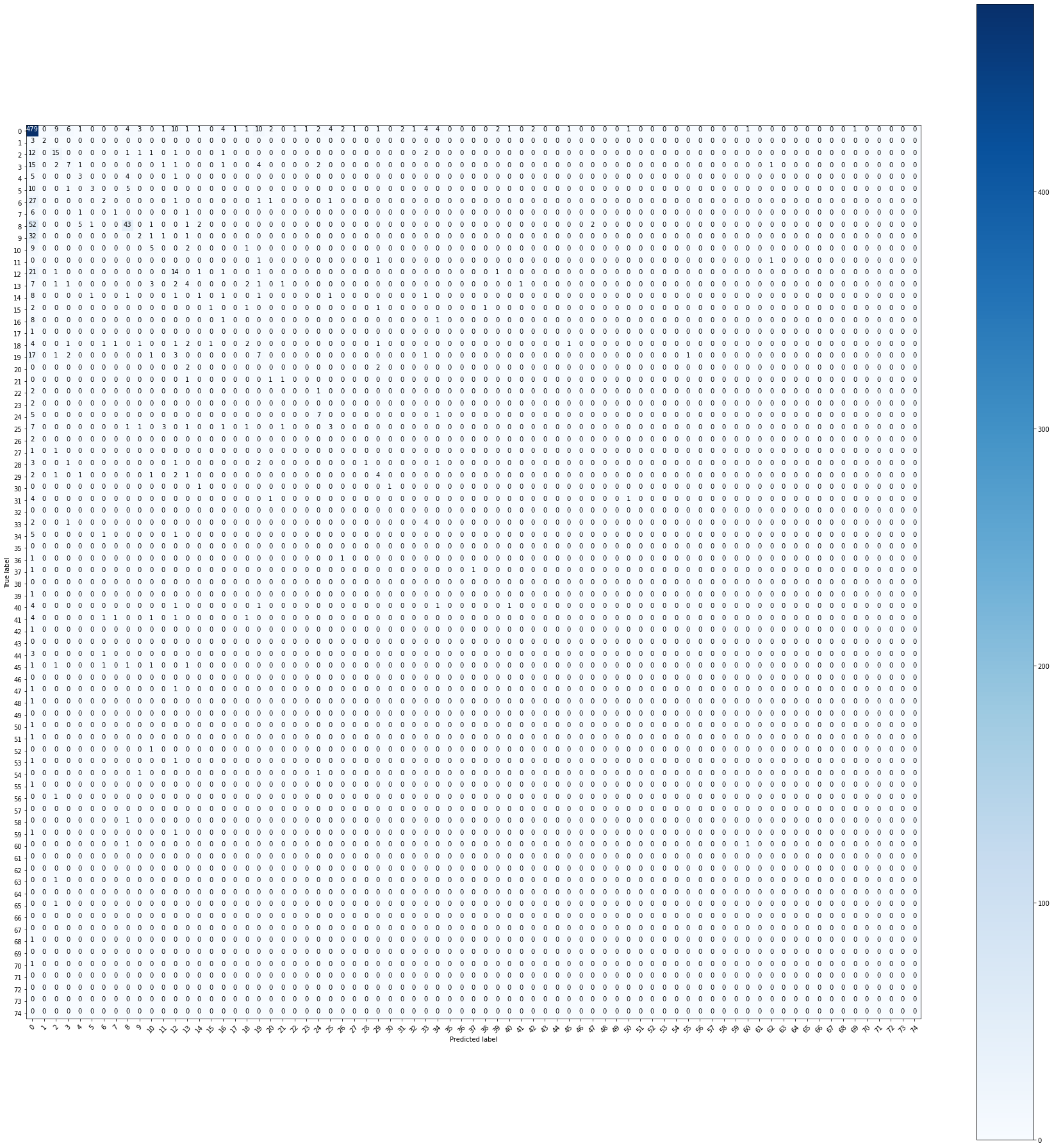
Eventhough the problem in question is extensively researched, the dataset we used is not a benchmark dataset. Hence, it seems infeasible to compare our results with a benchmark. Hence we choose to provide a comparison of all the models we built. We show the Precision, Recall and F1 score of the deep learning models.

Visualizations

A good way to compare the performance of the models is to visualise the Confusion Matrices for those models. We provide below the confusion matrices for the ANN model, the Naive Bayes Model and the LSTM model.

1. **Simple ANN**

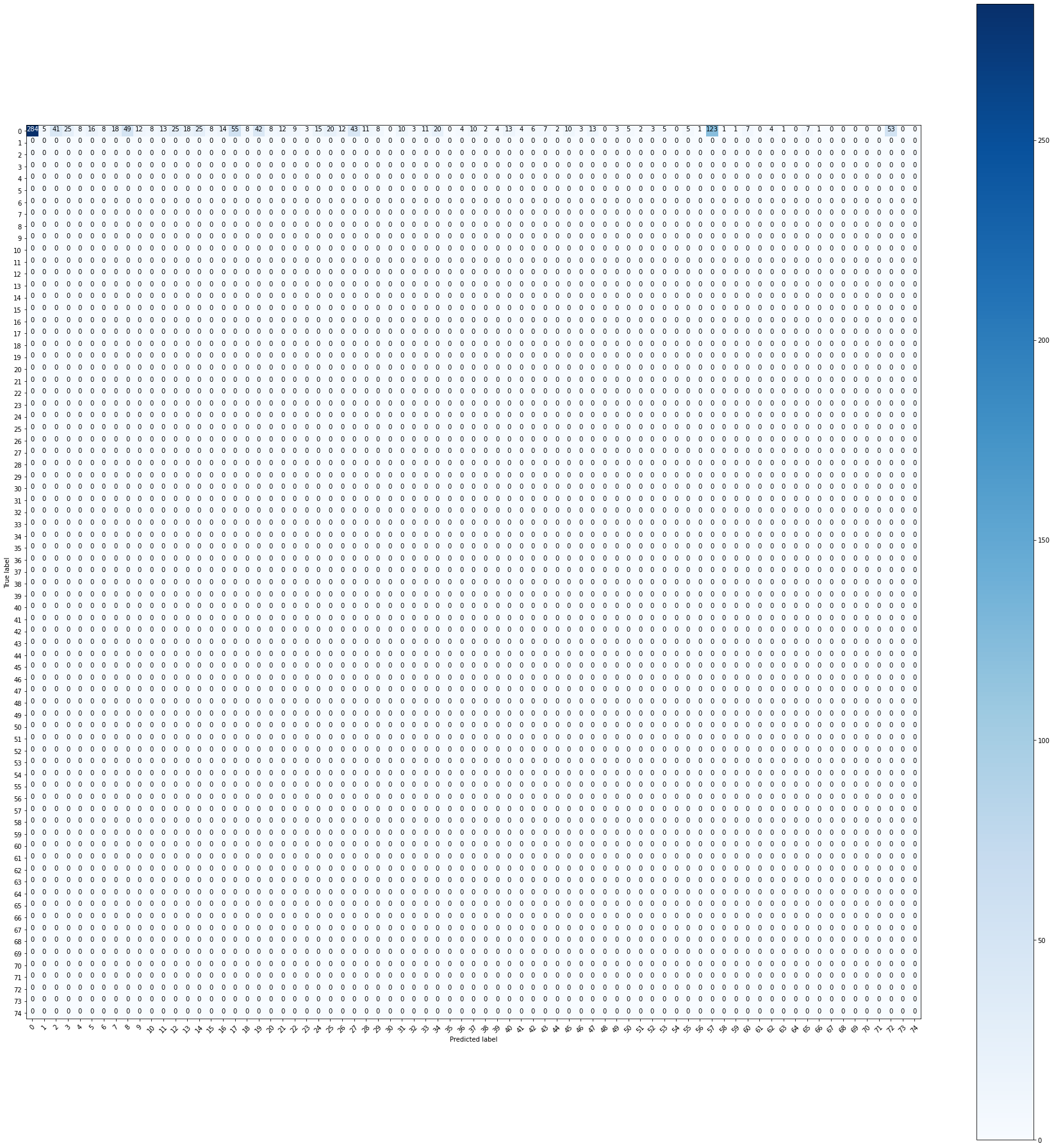
**Accuracy: 0.5141388174807198**

****

As can be inferred from the confusion matrix above, the simple ANN does quite a good job of classifying the majority class tickets. However, the precision drops to close to 50% for the rest of the classes, which pulls the overall accuracy down to close to 52%. The precision for Class 0 alone is close to 85%.

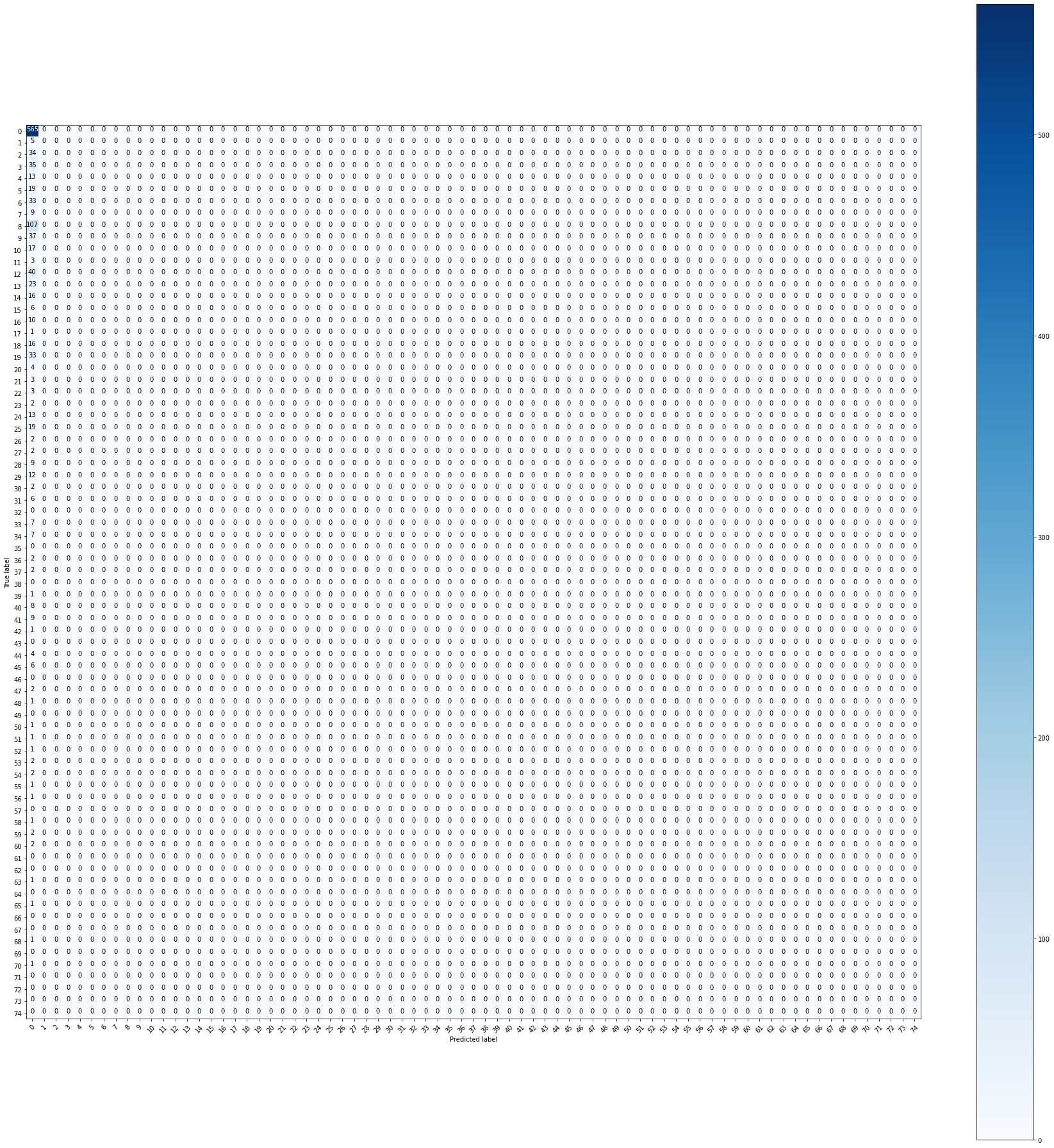
1. **Naive Bayes**

**Accuracy: 0.23393316195372751**

****

From the confusion matrix for Naive Bayes we see that the Naive Bayes algorithm is inefficient in classifying not only the minority classes, but also the majority class. Even the majority classes have been misclassified into the minority classes.

1. **LSTM**



From the confusion matrix of the LSTM model, we see that the precision of the LSTM classifier dips slightly as compared to the ANN models, as it is classifying all tickets into the majority class. The same problem encountered with the ANN classifier is being encountered here. Hence the precision and recall numbers look better than the overall accuracy, owing to the highly unbalanced dataset.

Implications

Limitations

We have listed below the limitation in our models:

1. The data is unbalanced. In this project, no measures are taken to balance it. Future efforts can be towards considering up-sampling or down-sampling of data.
2. Since the input dataset contained tickets from multiple languages, language translation was attempted but we could not find APIs that could accept a large number of translation requests at once. Future research could be in the direction of language detection and translation while data processing.

**Conclusion**

1. Linear SVM & Random Forest perform better than other algorithms. By doing a thorough hyperparameter tuning(using GridSearchCV), we get better results. We should also consider Decision Trees as this model gives the highest F1 score.
2. Deep learning models ANN also gives better precision and recall numbers look better than the overall accuracy

**References:**

1. Asmussen, C.B., Møller, C. Smart literature review: a practical topic modelling approach to exploratory literature review. *J Big Data* 6, 93 (2019).<https://doi.org/10.1186/s40537-019-0255-7>
2. Qamili, Ruanda & Shabani, Shaban & Schneider, Johannes. (2018). An Intelligent Framework for Issue Ticketing System Based on Machine Learning. 79-86. 10.1109/EDOCW.2018.00022. DOI: [10.1109/EDOCW.2018.00022](https://www.researchgate.net/deref/http%3A%2F%2Fdx.doi.org%2F10.1109%2FEDOCW.2018.00022)
3. Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
4. Altintas, M. and Tantung, A.C. (2014). Machine learning based ticket classification in issue tracking systems. Proceeding of the International Conference on Artificial Intelligence and Computer Science (AICS 2014), 15 - 16 September 2014, Bandung, INDONESIA. (e-ISBN978-967-11768-8-7). Organized by http://WorldConferences.net