GREAT LEARNING

Final Report

On

Automatic Ticket Assignment

Submitted By

Akash Bhende
Kishore V
Rassil Krishnan
Pramod
Sneha G K

AIML May Batch - Group 10

2021-2022

Index

1. Summary of problem statement, data and findings

1.1 Problem Statement

The main objective of any organization is to keep their operations running 24*7 without any delay. To achieve this they have to keep a steady monitoring on the operations and if any unplanned interruption occurs they have to rectify it as fast as possible. These interruptions are reported using ticketing systems.When it comes to IT functioning these kinds of incidents occur regularly. For resolving incidents raised by various Business and IT Users, End Users/Vendors they must occupy great man power. Each incident is assigned manually to the appropriate person or unit in the support team. Manual assignments of incidents are time consuming and require human efforts. Due to human error tickets are often misaddressed, all these leads to user dissatisfaction and poor customer service.

Better allocation of tickets and effective usage of valuable support resources will directly result in substantial cost savings. For this we can automate the process by applying traditional machine learning techniques and neural network based NLP. These will analyze each ticket and assign it to the corresponding owner. Reassignment count is reduced because of better assignments in the first place. It saves time and cost of the support team. Faster ticket resolution leads to good customer satisfaction and improves business.

1.2 Data and Findings

Data used: input_data.xlsx

Shape of record: (8500,4) 8500 rows and 4 columns.

Data Fields:

Short description	A short summary on the problem
Description	Detailed statement about the problem
Caller	User Id of the ticket raiser
Assignment group	Various groups to which ticket is assigned

Sample Data:

Short description	Description	Caller	Assignment group
login issue	-verified user details.(employee# & manager name)_	spxjnwir pjlcoqds	GRP_0
outlook	_x000D_\n_x000D_\nreceived from: hmjdrvpb.komu	hmjdrvpb komuaywn	GRP_0
cant log in to vpn	_x000D_\n_x000D_\nreceived from: eylqgodm.ybqk	eylqgodm ybqkwiam	GRP_0

2. Overview of the final process

We have implemented multiple models to classify tickets for their respective assignment groups.
We have got a total 74 unique assignment groups in the dataset. The data is imbalanced because the majority of records are associated with the top few assignment groups and remaining assignment groups are having very less number of records.

The brief approach for the solution is given below :

- Exploratory Data Analysis and Visualization:
    - We started the project with exploratory data analysis where we understood the significance of each field given in the dataset.
      We also did a data visualization using countplot and word clouds. It helped in understanding the imbalances in the dataset.

- Data Cleaning:
  - We checked for null values in our dataset and removed them.

```
df_ticket.isnull().sum()
```

```
Short description    8
Description          1
Caller               0
Assignment group     0
dtype: int64
```

We have 8 null values in short description and 1 in description

Since they are in small amounts it won't affect our dataset even if we remove them.

  - We found that the caller column doesn't provide any meaning to our model so removed it.
  - Found out that there were duplicate records so we dropped them.
  - Combined the columns 'Short description' and 'Description' based on their similarity.
  - We did preprocessing of the text data by creating a user defined function

```
def preprocess_text(txt):
  txt = txt.lower()    #Converting the text into lower case
  txt = re.sub(r'\S+@\S+', ' ', txt)  #Removing email address
  txt = re.sub('[^a-z ]+', ' ', txt)  #Removing specaial characters and numbers
  txt = txt.replace('x d'," ")  #It is observable that many text contains the ch
arcters 'x d' repeatedly, so removing it
  txt = txt.replace('xd'," ")
  txt = txt.replace('zz'," ") #It is also seen that repeated use of 'zz' , so re
moving it.
  txt = ' '.join(txt.split()) #Removing extra spaces
  txt = remove_stopwords(txt)  #Calling a function to remove stopswords
  txt = " ".join(gw for gw in nltk.wordpunct_tokenize(txt) if gw.lower() not in
words_to_remove)   #Removing greetings
  txt = " ".join(w for w in nltk.wordpunct_tokenize(txt) if w.lower() in words)
  txt = ps.stem(txt) #Stemming
  return txt
```

Each record was converted to lowercase, then we removed the email address,special characters and numbers using regex. Replaced some repeated words,greetings etc and removed extra spaces. We used a function to remove stop words,stemming and removing non english records.

- Feature Building
    - We did TF-IDF Vectorization to convert textual data to numeric format so that machine learning models can be trained on it.
    - We did Word2Vec embedding.

- Model Building
    - After data preprocessing we created multiple machine learning models including traditional supervised learning models and neural networks and compared their performances.

3. Step-by-step walk through the solution

We started the process with EDA and data cleaning. We analyzed the cleaned data in order to be able to fit it on the ML models.
Visualizing the data helped us in cleaning it in a better way because we could notice certain patterns in the text which could have adverse effect on the model performance and we removed / amended those parts (Ex.- numbers, special characters, user's email ids, non english words, etc.) to make our data more suitable for training the models. We checked for similarities between short description and description combined them whenever they had different values.
We have clustered the groups with less than 100 records into a single miscellaneous group to deal with the class imbalance.
Then we vectorised / tokenized the data to bring it into a numeric format from the textual format.
We tried multiple models with the above approach but results were not upto the benchmark. It was happening due to the imbalance in the data.

We resampled the data and created records for the assignment groups having very few number of records and matched their record count with the second largest assignment group in the dataset.
We created the word embedding and used it as a weight in the LSTM model.

Following are the models we have implemented:

- Support Vector Classifier:
  This is a traditional supervised learning model used for classification problems. We used the grid search algorithm to try out different hyperparameters and find the best ones.

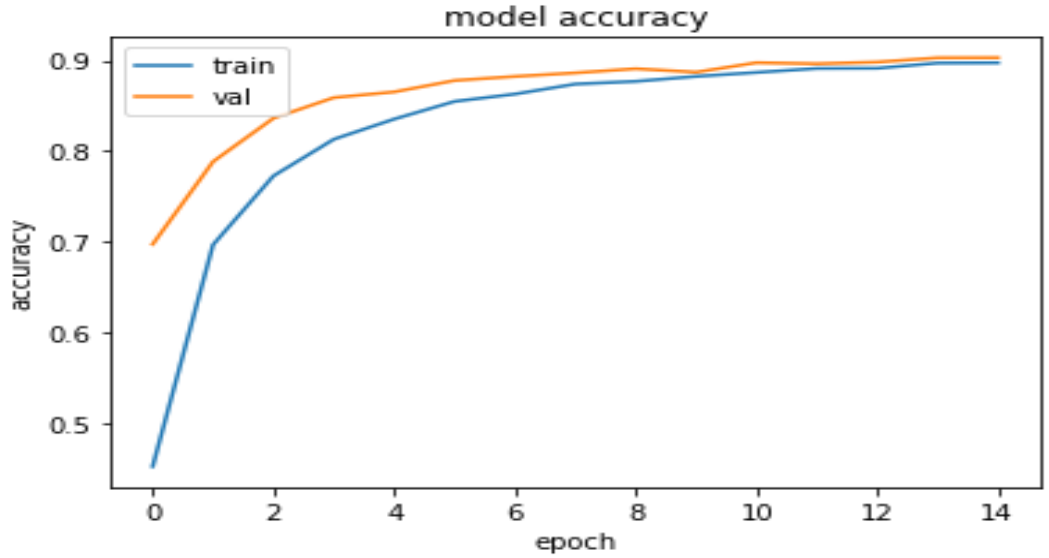
- Artificial Neural Network:
  We implemented a simple neural network with a single hidden layer to classify the tickets.

- LSTM based Neural Networks:
  We implemented the multiple LSTM based models including bidirectional LSTM in order to get better performance in the classification.
  Our final model is a bidirectional LSTM model.

4. Model evaluation

We are getting 89.73% test accuracy with our final model. It is a bidirectional LSTM model.

Below graph shows the changes in training and validation accuracies with number of epochs.



We trained this model for 15 epochs and batch size of 100. We used 20% of training data for validation. Following line of code is added for reference.

```
model.fit(X_train, y_train, epochs=15, batch_size=100, validation_split=0.2)
```

Below is the summary of the final model:

Model: "sequential_12"

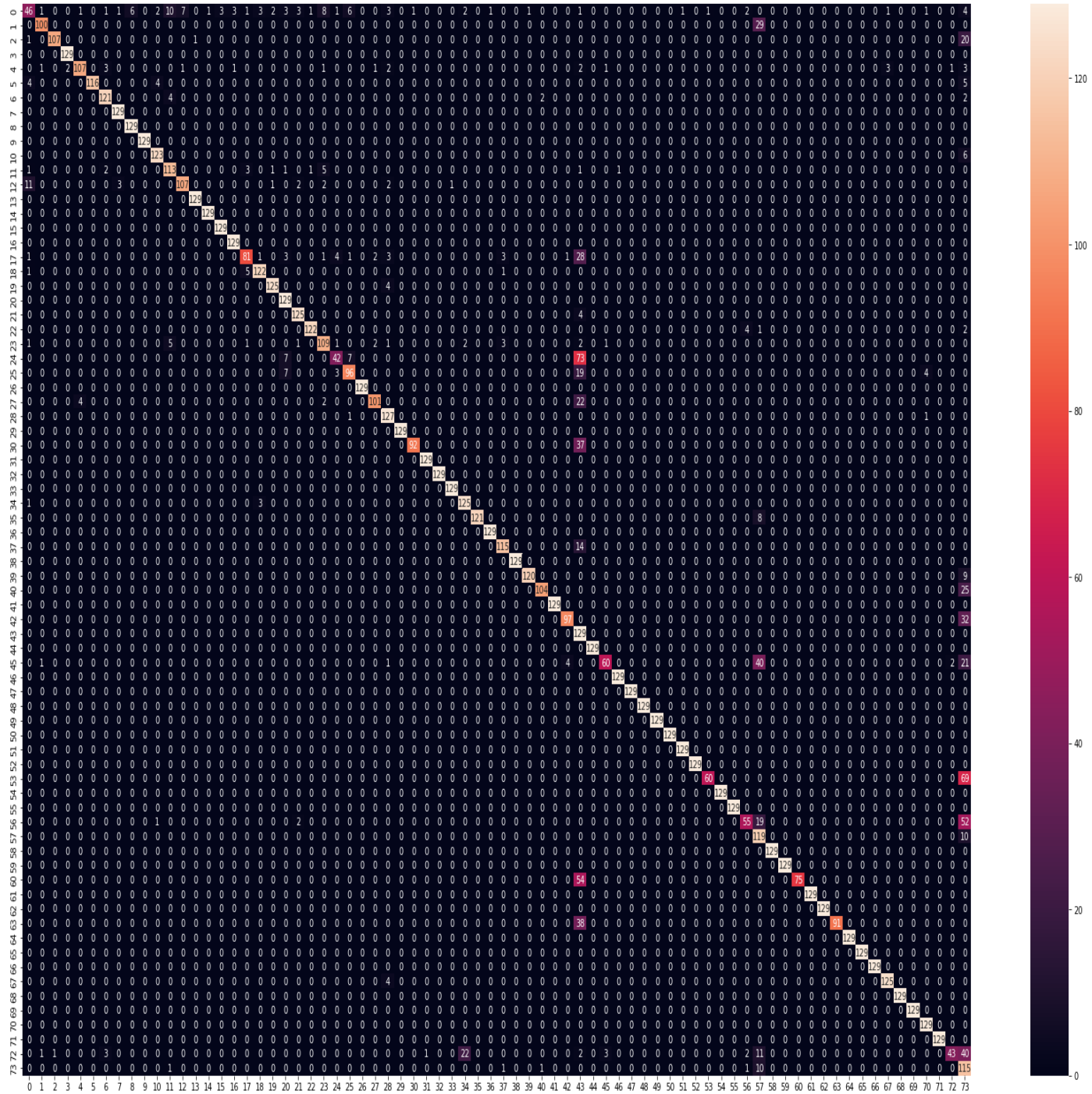
Layer (type)	Output Shape	Param #
embedding_11 (Embedding)	(None, None, 100)	309700
bidirectional_3 (Bidirectional)	(None, 256)	234496
dropout_30 (Dropout)	(None, 256)	0
dense_34 (Dense)	(None, 100)	25700
dropout_31 (Dropout)	(None, 100)	0
dense_35 (Dense)	(None, 100)	10100
dense_36 (Dense)	(None, 74)	7474

Total params: 587,470
Trainable params: 587,470
Non-trainable params: 0

Below is the classification report for our final model.

	precision	recall	f1-score	support
0	0.69	0.36	0.47	129
1	0.96	0.78	0.86	129
2	0.99	0.83	0.90	129
3	0.98	1.00	0.99	129
4	0.96	0.83	0.89	129
5	1.00	0.90	0.95	129
6	0.93	0.94	0.93	129
7	0.97	1.00	0.98	129
8	0.96	1.00	0.98	129
9	1.00	1.00	1.00	129
10	0.95	0.95	0.95	129
11	0.85	0.88	0.86	129
12	0.93	0.83	0.88	129
13	0.99	1.00	1.00	129
14	0.99	1.00	1.00	129
15	0.98	1.00	0.99	129
16	0.97	1.00	0.98	129
17	0.89	0.63	0.74	129
18	0.95	0.95	0.95	129
19	0.97	0.97	0.97	129
20	0.86	1.00	0.92	129
21	0.94	0.97	0.95	129
22	0.98	0.95	0.96	129
23	0.85	0.84	0.85	129
24	0.82	0.33	0.47	129
25	0.86	0.74	0.80	129
26	1.00	1.00	1.00	129
27	0.95	0.78	0.86	129
28	0.86	0.98	0.92	129
29	1.00	1.00	1.00	129
30	0.99	0.71	0.83	129
31	0.99	1.00	1.00	129
32	1.00	1.00	1.00	129
33	0.99	1.00	1.00	129
34	0.83	0.97	0.89	129
35	1.00	0.94	0.97	129
36	0.99	1.00	1.00	129
37	0.93	0.89	0.91	129
37	0.93	0.89	0.91	129
38	1.00	1.00	1.00	129
39	0.99	0.93	0.96	129
40	0.97	0.81	0.88	129
41	1.00	1.00	1.00	129
42	0.95	0.75	0.84	129
43	0.30	1.00	0.46	129
44	1.00	1.00	1.00	129
45	0.92	0.47	0.62	129
46	0.99	1.00	1.00	129
47	1.00	1.00	1.00	129
48	1.00	1.00	1.00	129
49	1.00	1.00	1.00	129
50	1.00	1.00	1.00	129
51	0.99	1.00	1.00	129
52	1.00	1.00	1.00	129
53	0.98	0.47	0.63	129
54	1.00	1.00	1.00	129
55	1.00	1.00	1.00	129
56	0.86	0.43	0.57	129
57	0.50	0.92	0.65	129
58	1.00	1.00	1.00	129
59	1.00	1.00	1.00	129
60	1.00	0.58	0.74	129
61	1.00	1.00	1.00	129
62	1.00	1.00	1.00	129
63	1.00	0.71	0.83	129
64	1.00	1.00	1.00	129
65	1.00	1.00	1.00	129
66	1.00	1.00	1.00	129
67	0.97	0.97	0.97	129
68	1.00	1.00	1.00	129
69	1.00	1.00	1.00	129
70	0.96	1.00	0.98	129
71	1.00	1.00	1.00	129
72	0.93	0.33	0.49	129
73	0.28	0.89	0.42	129
accuracy			0.89	9546
macro avg	0.94	0.89	0.90	9546
weighted avg	0.94	0.89	0.90	9546

Below is the heatmap of confusion matrix for our final model.



As we can observe in the classification report of our final model, recall is bit less for a few assignment groups like group 0, group 45, group 53. So there is a chance of classifying false negatives in those groups. Also, precision is less for group 57 and 73 which means their true positives may get misclassified. Apart from this, the majority of the tickets are classified correctly.

Below are the accuracy results of some of the other models we tried out:

Model	**Accuracy**
Support Vector Classifier (SVC)	45.41%
SVC (post grid search)	64%
First LSTM based model	80%
Second LSTM based model	56.92%
Third LSTM based model	57%
Fourth LSTM based model	80.94%
Bidirectional LSTM based model	58.97%
Fifth LSTM based model (Post resampling)	45.49%

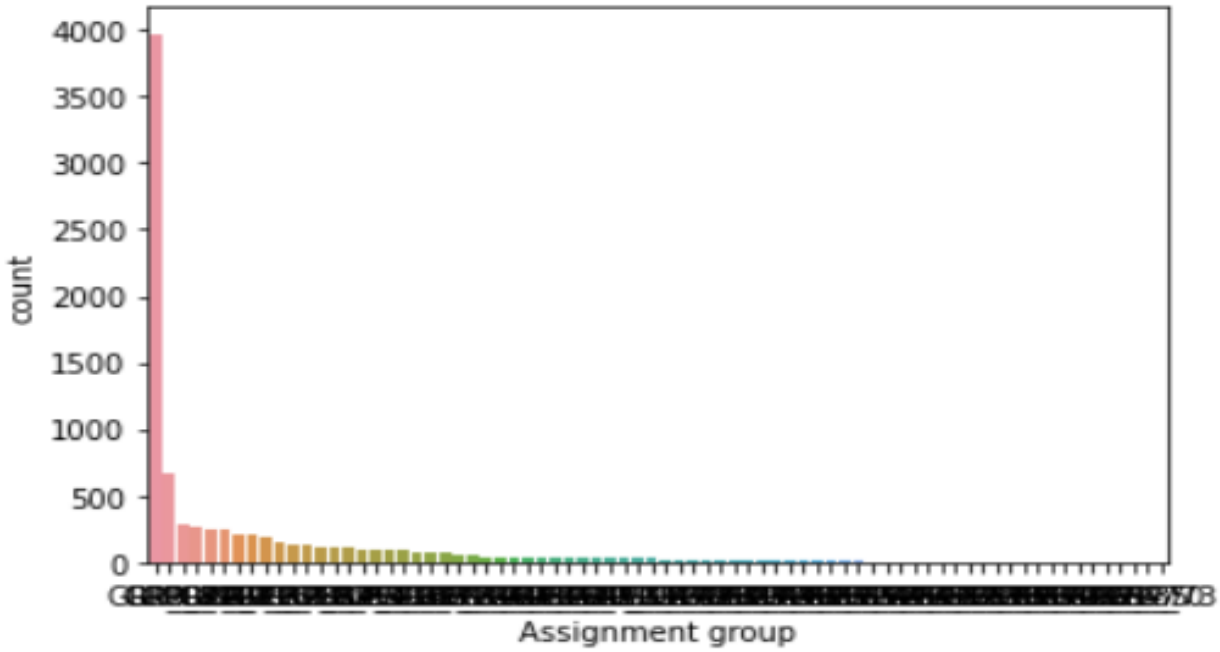
5. Comparison to benchmark

As per the details given in the problem statement around 25% tickets are assigned to the incorrect assignment group. This means that manual work is performed with 75% accuracy.
We can consider the 75% as a benchmark while creating machine learning models. That means that the ML model should give at least 75% accuracy in order to be considered for automating the ticket assignment.

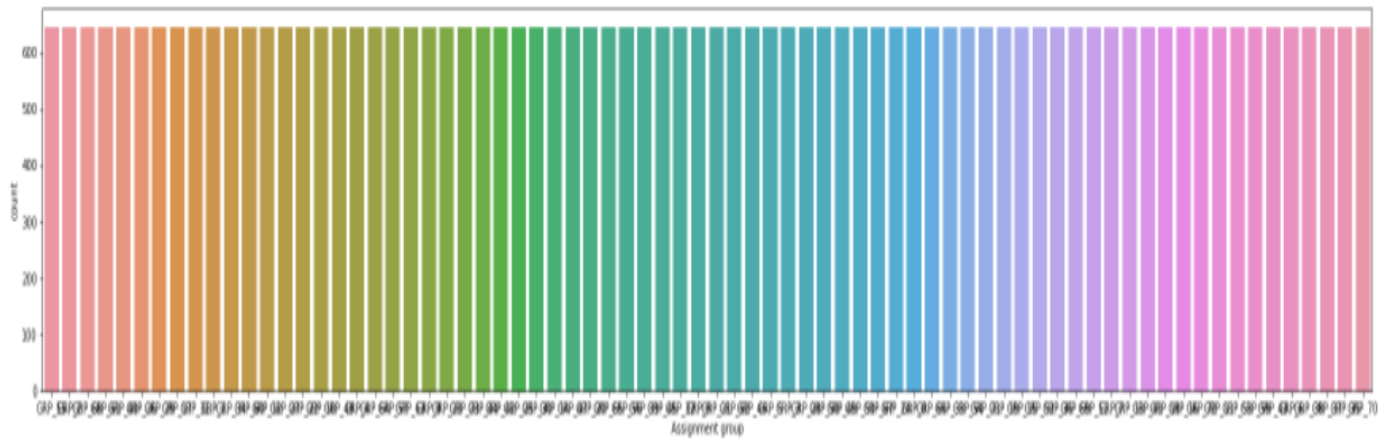
Our final model is giving 89.73% accuracy which is much above the benchmark we have decided. It can be a good fit to automate ticket assignment work.

6. Visualizations

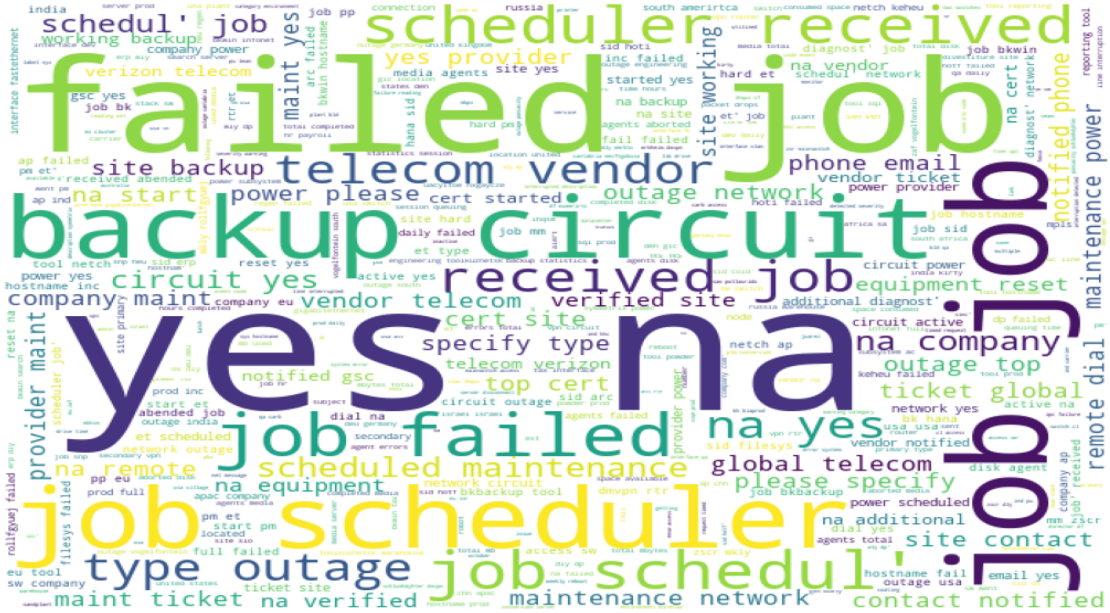
We used the countplot to visualize the record count for each class in the dataset.



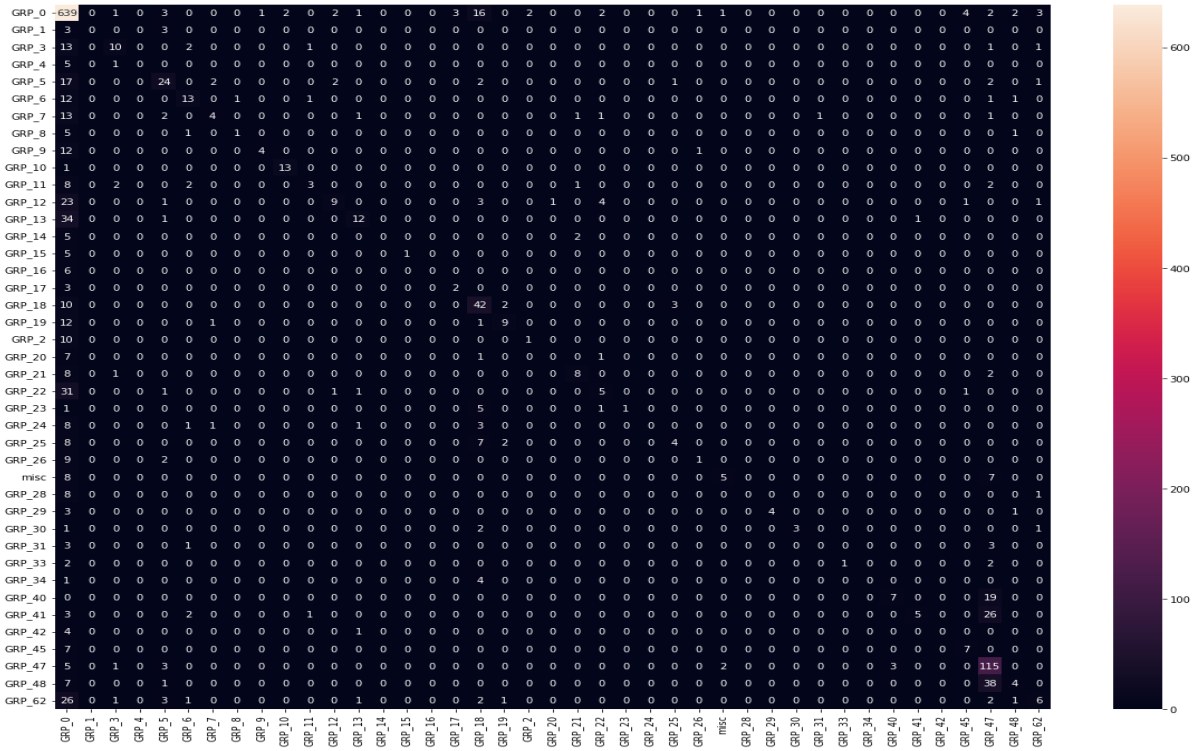
Following countplot is plotted after resampling the dataset



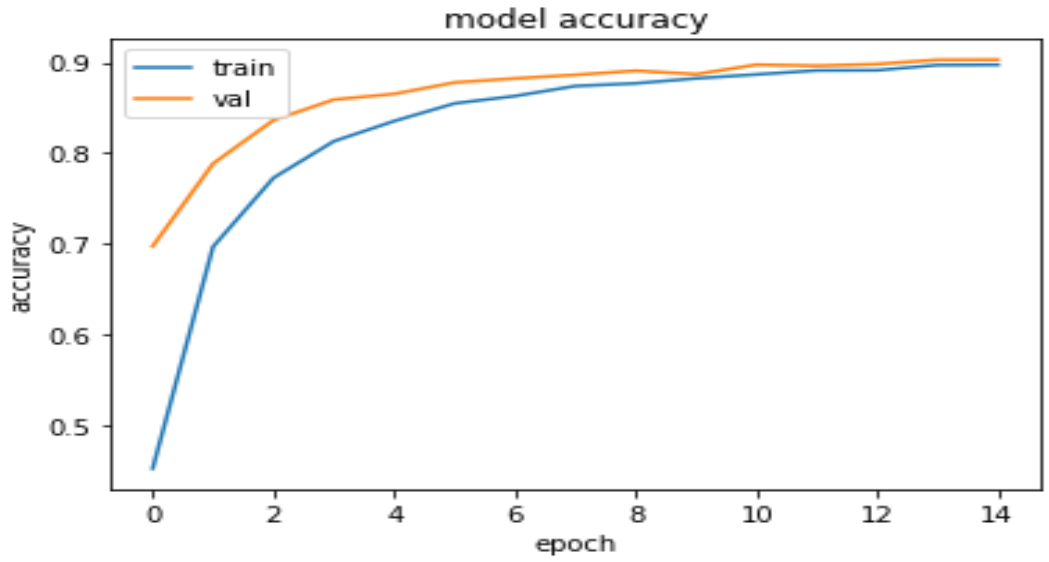
We used wordcloud to visualize the frequent words in the ticket description for the assignment groups.



We used a heatmap to display the performance matrix for the support vector classifier. It shows the actual vs predicted values.



We are using the below graph to compare the training and validation accuracies with number of epochs:



7. Implications

Our solution gives better accuracy results than manual assignments. Also, it doesn't have to look into all the documentations and knowledge base in order to identify the appropriate assignment group for the ticket, which makes it much faster than the humans.
If we combine this solution with RPA (Robotic Process Automation) it will be able to assign the tickets to appropriate support groups in a much more precise and faster manner. It will save the FTEs and cost to the company as well.

8. Limitations

Initially we had clustered the assignment groups with very few records available into a single group, which means tickets belonging to those groups won't be assigned to any particular group. This raised the need for human intervention and our solution does not offer the 100% automation of the problem.

To overcome this limitation we resampled the data and generated more samples for the assignment groups having very few records and then fitted our models on top of that data. It not only helped in overcoming this limitation but also improved the model accuracy significantly.

Since our model is not 100% accurate there is still a small chance of ticket assignment to the incorrect group. In that case manual action might be required.

9. Closing Reflections

We learnt following things from this project:

- To handle the noise in the data
- Multiple ways of handling textual data in the context of machine learning.
- Combining multiple fields to create a single more meaningful field.
- Clustering the multiple classes into a single class to handle the imbalance in the data
- Treating the imbalance in data and to make it more balanced so it is suitable for fitting the machine learning models.
- Developing and tuning multiple ML models to get better results.

Things we could do differently next time:

- Data Collection:
  We will try to get more data specially for the assignment groups having very few records. This will reduce the imbalance and improve the model performance.

- Text Preprocessing:
  We could translate some of the non english words into english to see if they bring more meaning to sentences. This might reduce the loss of information and improve the model performance even further.