AIML Online Capstone - AUTOMATIC TICKET ASSIGNMENT - Interim Report

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.

Problem Statement :

In the support process, incoming incidents are analyzed and assessed by the 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. 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 $\sim$ 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 $\sim$ 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 $\sim$ 25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP reviews for each incident. Minimum of $\sim$ 1 FTE effort needed only for incident assignment to L3 teams.

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 $\sim$ 25% of Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to the right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service. 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.

Summarizing The Data Findings :

We have received unstructured textual data as our dataset. The dataset spans over 8500 rows and 4 columns (**Short description, Description, Caller and Assignment group**).

The 4 columns can be broadly described as below :-

- **Short Description :** This column gives us a quick look at the broad categorization of the complaint. For ex : Login issue, outlook, skype error, etc.
- **Description :** This column gives us a little more detailed insight into the complaint (compared to the Short Description column) that the end user has. It talks about the brief description of the complaint/ issue the end user is facing. For ex : event: critical:HostName_221.company.com the value of mountpoint threshold for /oracle/SID_37/erpdata21/sr3psa1d_7/sr3psa1d.data7,perpsr3psa1d,4524 is 98.
- **Caller :** This column has the end user's name who is filing the complaint or raising the request. For ex : pfmcnahv ofzlusri
- **Assignment Group :** This column talks about the category/group into which the complaint is classified for it to be directed to the right department for the issue to be resolved. For ex : GRP_0, GRP_1, etc.

The key findings for the dataset at hand are as follows :-

- There are a total of 74 different groups for the Assignment Group column.
- Approximately 50% of the dataset comprises complaints that are corresponding to GRP_0.
- There are a few rows of data that have the same text for the Short Description and the Description column.

- The dataset is found to be multilingual, we have come across data in Deutch, German, Latin languages too other than English.
- There are some rows that have missing data either for the Short Description or Description column.
- The total number of incidents reported in the dataset are 8500.
- There are a total number of 8 null records in the Short Description column.
- There is 1 null record in the Description column
- There are no null records in the Caller and Assignment Group columns.
- The minimum occurrence count for the 74 unique Assignment groups is 1 while the maximum occurrence count is 3976. The average occurrence count is 114.86

Overview of the final process

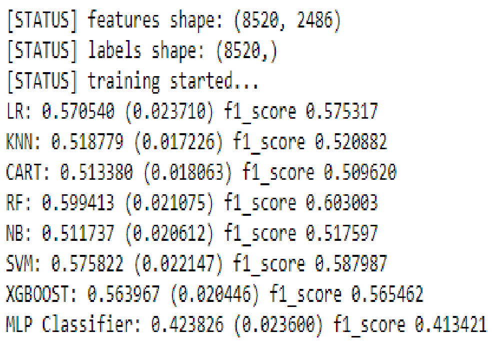
1) We start our project by importing all the necessary libraries.
2) We then do the data analysis for our input dataset, to see the description of the data, encoding of the data, shape and size of the data, null values present in the dataset, duplicate values present in the dataset and so on. (Our data findings are mentioned above)
3) We used the FTFY package to overcome the encoding problem that we were facing as we found our dataset to be multi-lingual.
4) We then used the Langdect package to find the different languages text present in both 'Short Description' column and 'Description' columns.
5) We then translated the entire dataset to English as this would increase the vocabulary of our dataset.
6) We then replaced all null values in our dataset with blank spaces.
7) We then created word clouds for the 'Short Description' and 'Description' column.
8) We then stripped off all unwanted text data, for ex : "received from" , etc, stripped off all unwanted whitespaces, punctuation marks and leading and trailing spaces.

9) We then did lemmatization on our dataset as stemming would trim the text when applied.

10) We then did Label Encoding for the same and used TF-IDF Vectorizer and tokenized the dataset.

11) We then went about splitting our dataset into training and testing samples.

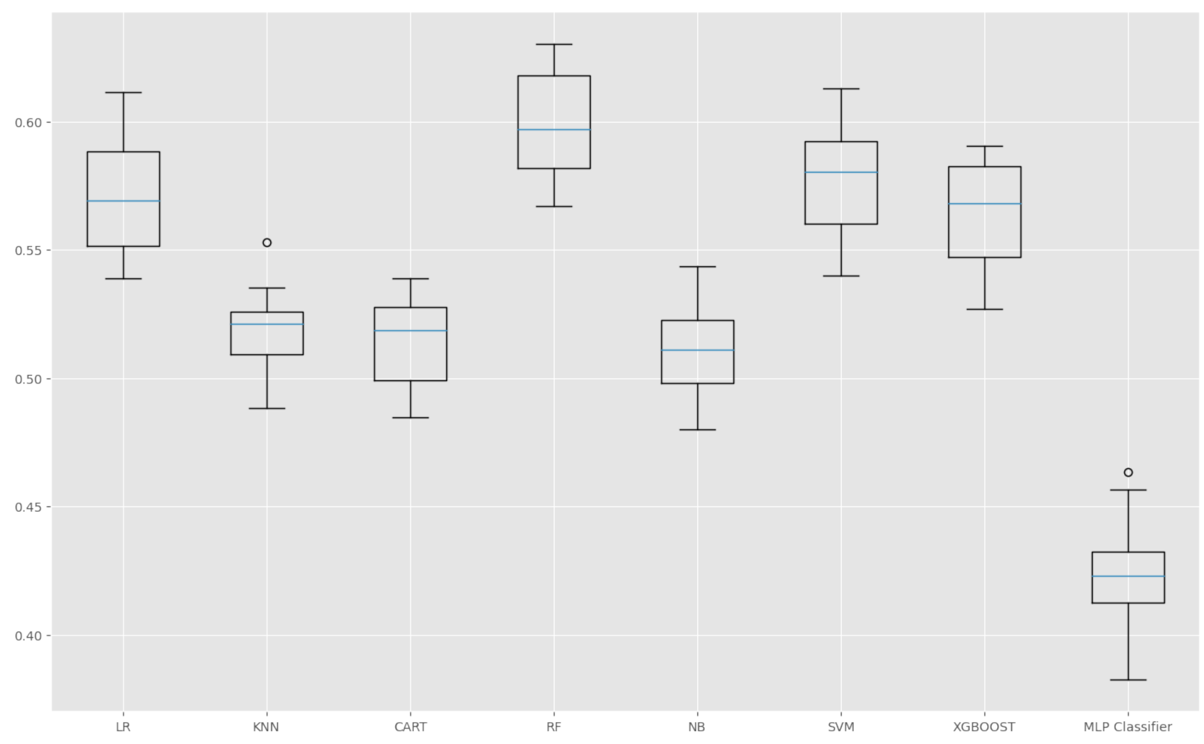
12) We then used machine learning algorithms like Logistic Regression, KNN classifier, Decision tree classifier, Random Forest classifier, Multinomial Naive Bayes classifier, SVM, XGBClassifier and MLP classifier. We found the Random Forest Classifier to work the best. The snapshot of our observations are as follows.



```
[STATUS] features shape: (8520, 2486)
[STATUS] labels shape: (8520,)
[STATUS] training started...
LR: 0.570540 (0.023710) f1_score 0.575317
KNN: 0.518779 (0.017226) f1_score 0.520882
CART: 0.513380 (0.018063) f1_score 0.509620
RF: 0.599413 (0.021075) f1_score 0.603003
NB: 0.511737 (0.020612) f1_score 0.517597
SVM: 0.575822 (0.022147) f1_score 0.587987
XGBOOST: 0.563967 (0.020446) f1_score 0.565462
MLP Classifier: 0.423826 (0.023600) f1_score 0.413421
```

Machine Learning algorithm comparison with average = micro

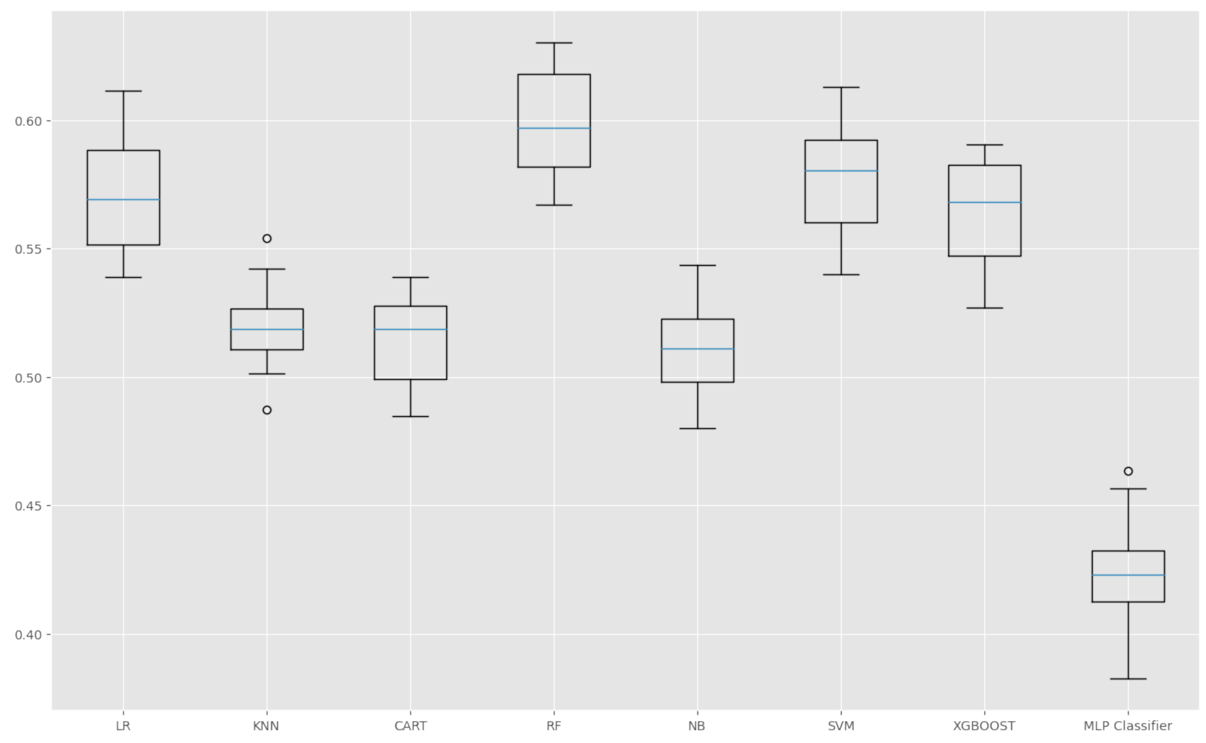
Machine Learning algorithm comparison with average = micro

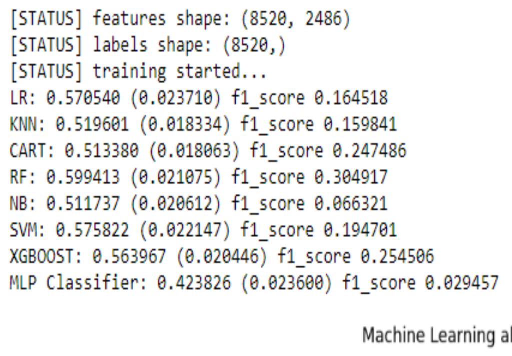


```
[STATUS] features shape: (8520, 2486)
[STATUS] labels shape: (8520,)
[STATUS] training started...
LR: 0.570540 (0.023710) f1_score 0.497977
KNN: 0.519601 (0.018334) f1_score 0.431637
CART: 0.513380 (0.018063) f1_score 0.496143
RF: 0.599413 (0.021075) f1_score 0.551804
NB: 0.511737 (0.020612) f1_score 0.390689
SVM: 0.575822 (0.022147) f1_score 0.509079
XGBOOST: 0.563967 (0.020446) f1_score 0.501995
MLP Classifier: 0.423826 (0.023600) f1_score 0.358920
```

Machine Learning algorithm comparison with average = weighted

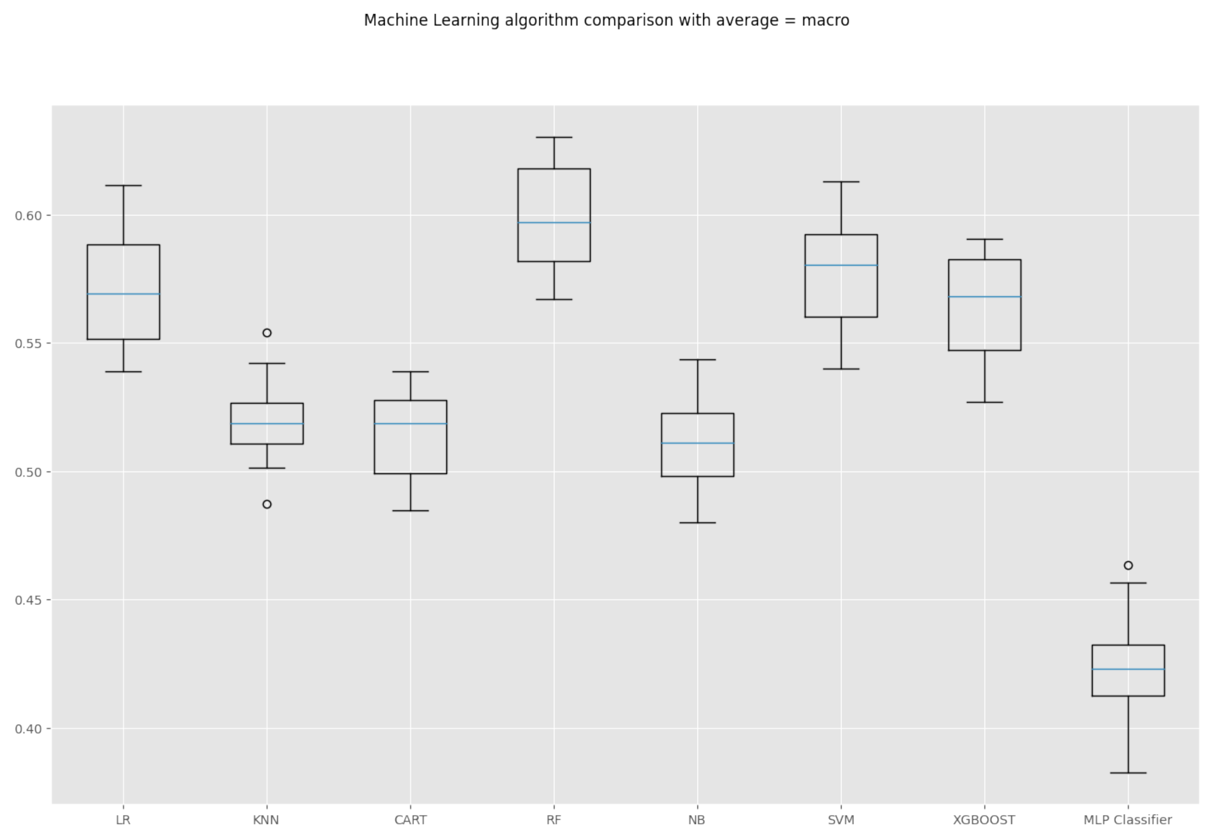
Machine Learning algorithm comparison with average = weighted





```
[STATUS] features shape: (8520, 2486)
[STATUS] labels shape: (8520,)
[STATUS] training started...
LR: 0.570540 (0.023710) f1_score 0.164518
KNN: 0.519601 (0.018334) f1_score 0.159841
CART: 0.513380 (0.018063) f1_score 0.247486
RF: 0.599413 (0.021075) f1_score 0.304917
NB: 0.511737 (0.020612) f1_score 0.066321
SVM: 0.575822 (0.022147) f1_score 0.194701
XGBOOST: 0.563967 (0.020446) f1_score 0.254506
MLP Classifier: 0.423826 (0.023600) f1_score 0.029457
```

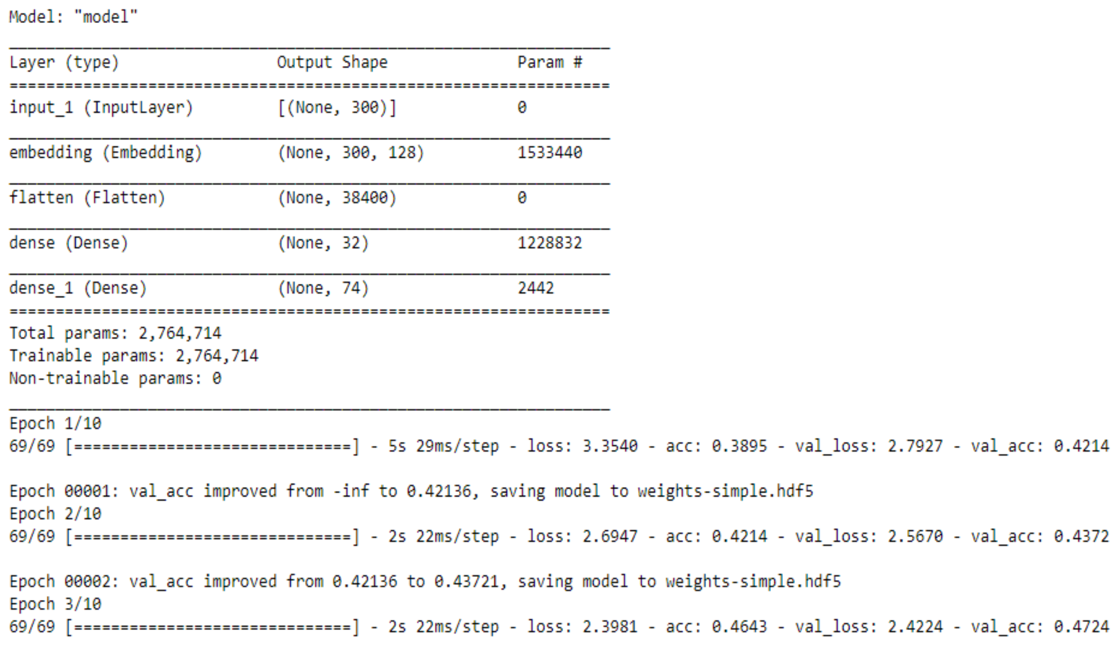
Machine Learning algorithm comparison with average = macro



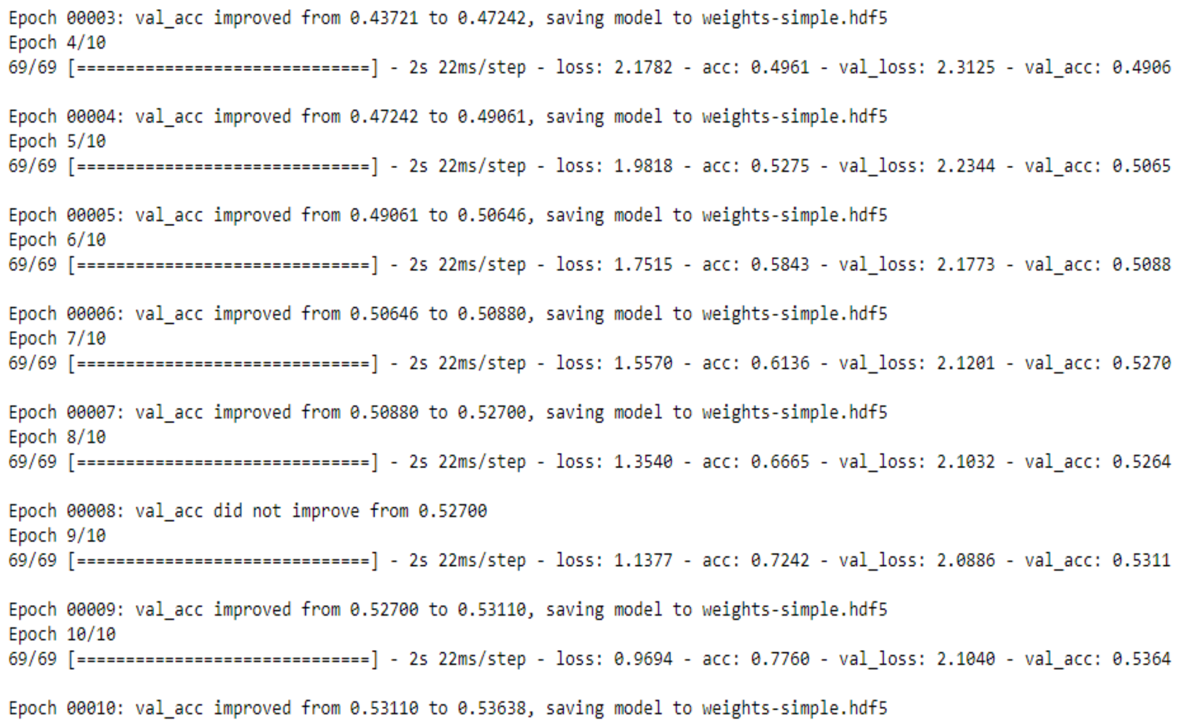
13) We calculated the F1 Score for these as well.

14) We now tried out different Deep Learning models for our dataset as well, including ANN, RNN, LSTM, Bidirectional GRU.

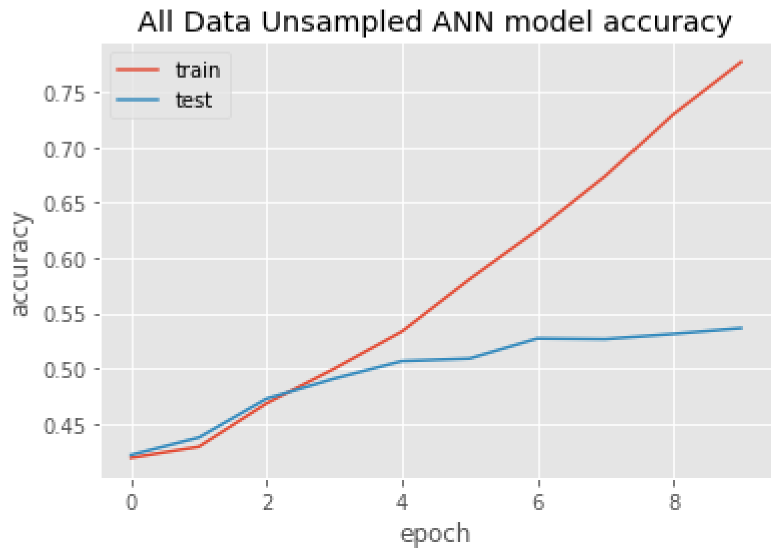
    The snapshots of our observations have been shared below. **ANN**

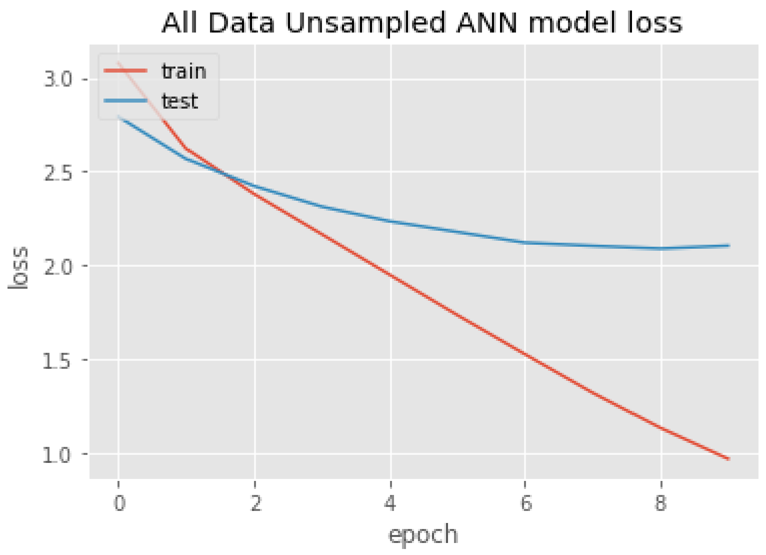


```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding (Embedding)        (None, 300, 128)          1533440
_________________________________________________________________
flatten (Flatten)            (None, 38400)             0
_________________________________________________________________
dense (Dense)                (None, 32)                1228832
_________________________________________________________________
dense_1 (Dense)              (None, 74)                2442
=================================================================
Total params: 2,764,714
Trainable params: 2,764,714
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
69/69 [==============================] - 5s 29ms/step - loss: 3.3540 - acc: 0.3895 - val_loss: 2.7927 - val_acc: 0.4214

Epoch 00001: val_acc improved from -inf to 0.42136, saving model to weights-simple.hdf5
Epoch 2/10
69/69 [==============================] - 2s 22ms/step - loss: 2.6947 - acc: 0.4214 - val_loss: 2.5670 - val_acc: 0.4372

Epoch 00002: val_acc improved from 0.42136 to 0.43721, saving model to weights-simple.hdf5
Epoch 3/10
69/69 [==============================] - 2s 22ms/step - loss: 2.3981 - acc: 0.4643 - val_loss: 2.4224 - val_acc: 0.4724
```

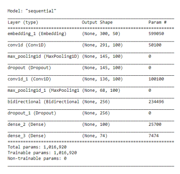


```
Epoch 00003: val_acc improved from 0.43721 to 0.47242, saving model to weights-simple.hdf5
Epoch 4/10
69/69 [==============================] - 2s 22ms/step - loss: 2.1782 - acc: 0.4961 - val_loss: 2.3125 - val_acc: 0.4906

Epoch 00004: val_acc improved from 0.47242 to 0.49061, saving model to weights-simple.hdf5
Epoch 5/10
69/69 [==============================] - 2s 22ms/step - loss: 1.9818 - acc: 0.5275 - val_loss: 2.2344 - val_acc: 0.5065

Epoch 00005: val_acc improved from 0.49061 to 0.50646, saving model to weights-simple.hdf5
Epoch 6/10
69/69 [==============================] - 2s 22ms/step - loss: 1.7515 - acc: 0.5843 - val_loss: 2.1773 - val_acc: 0.5088

Epoch 00006: val_acc improved from 0.50646 to 0.50880, saving model to weights-simple.hdf5
Epoch 7/10
69/69 [==============================] - 2s 22ms/step - loss: 1.5570 - acc: 0.6136 - val_loss: 2.1201 - val_acc: 0.5270

Epoch 00007: val_acc improved from 0.50880 to 0.52700, saving model to weights-simple.hdf5
Epoch 8/10
69/69 [==============================] - 2s 22ms/step - loss: 1.3540 - acc: 0.6665 - val_loss: 2.1032 - val_acc: 0.5264

Epoch 00008: val_acc did not improve from 0.52700
Epoch 9/10
69/69 [==============================] - 2s 22ms/step - loss: 1.1377 - acc: 0.7242 - val_loss: 2.0886 - val_acc: 0.5311

Epoch 00009: val_acc improved from 0.52700 to 0.53110, saving model to weights-simple.hdf5
Epoch 10/10
69/69 [==============================] - 2s 22ms/step - loss: 0.9694 - acc: 0.7760 - val_loss: 2.1040 - val_acc: 0.5364

Epoch 00010: val_acc improved from 0.53110 to 0.53638, saving model to weights-simple.hdf5
```

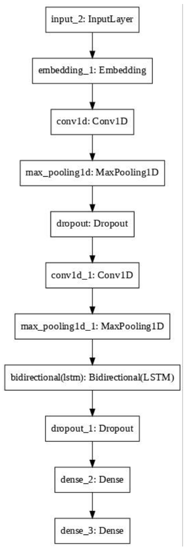

All Data Unsampled ANN model accuracy
train
test
accuracy
epoch
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8

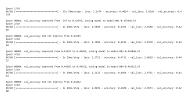

All Data Unsampled ANN model loss
train
test
loss
epoch
3.0
2.5
2.0
1.5
1.0
0
2
4
6
8

RNN

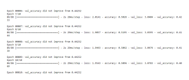


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 50)           599050
_________________________________________________________________
conv1d (Conv1D)              (None, 291, 100)          50100
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 145, 100)          0
_________________________________________________________________
dropout (Dropout)            (None, 145, 100)          0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 136, 100)          100100
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 68, 100)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 256)               234496
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               25700
_________________________________________________________________
dense_3 (Dense)              (None, 74)                7474
=================================================================
Total params: 1,016,920
Trainable params: 1,016,920
Non-trainable params: 0
_________________________________________________________________
```

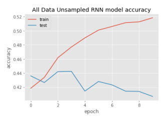


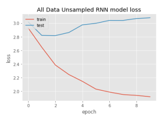


```
Epoch 1/10
86/86 [==============================] - 35s 30ms/step - loss: 3.1674 - accuracy: 0.4024 - val_loss: 3.0166 - val_accuracy: 0.4
359

Epoch 00001: val_accuracy improved from -inf to 0.43595, saving model to model-001-0.435946.h5
Epoch 2/10
86/86 [==============================] - 2s 29ms/step - loss: 2.6849 - accuracy: 0.4351 - val_loss: 2.8198 - val_accuracy: 0.42
66

Epoch 00002: val_accuracy did not improve from 0.43595
Epoch 3/10
86/86 [==============================] - 2s 29ms/step - loss: 2.3904 - accuracy: 0.4612 - val_loss: 2.8170 - val_accuracy: 0.44
20

Epoch 00003: val_accuracy improved from 0.43595 to 0.44205, saving model to model-003-0.442046.h5
Epoch 4/10
86/86 [==============================] - 2s 29ms/step - loss: 2.2752 - accuracy: 0.4713 - val_loss: 2.8628 - val_accuracy: 0.44
25

Epoch 00004: val_accuracy improved from 0.44205 to 0.44252, saving model to model-004-0.442515.h5
Epoch 5/10
86/86 [==============================] - 3s 29ms/step - loss: 2.1132 - accuracy: 0.4964 - val_loss: 2.9751 - val_accuracy: 0.41
44

Epoch 00005: val_accuracy did not improve from 0.44252
Epoch 6/10
86/86 [==============================] - 2s 29ms/step - loss: 2.0499 - accuracy: 0.4960 - val_loss: 2.9973 - val_accuracy: 0.42
80
```

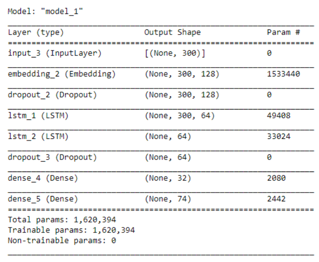


```
Epoch 00006: val_accuracy did not improve from 0.44252
Epoch 7/10
86/86 [==============================] - 2s 28ms/step - loss: 2.0141 - accuracy: 0.5026 - val_loss: 3.0404 - val_accuracy: 0.42
33

Epoch 00007: val_accuracy did not improve from 0.44252
Epoch 8/10
86/86 [==============================] - 2s 29ms/step - loss: 1.9057 - accuracy: 0.5195 - val_loss: 3.0395 - val_accuracy: 0.41
44

Epoch 00008: val_accuracy did not improve from 0.44252
Epoch 9/10
86/86 [==============================] - 2s 29ms/step - loss: 1.8463 - accuracy: 0.5062 - val_loss: 3.0676 - val_accuracy: 0.41
39

Epoch 00009: val_accuracy did not improve from 0.44252
Epoch 10/10
86/86 [==============================] - 2s 29ms/step - loss: 1.9543 - accuracy: 0.5096 - val_loss: 3.0783 - val_accuracy: 0.40
69

Epoch 00010: val_accuracy did not improve from 0.44252
```

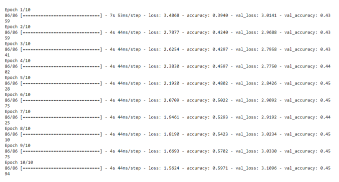




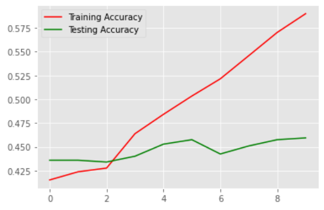
LSTM

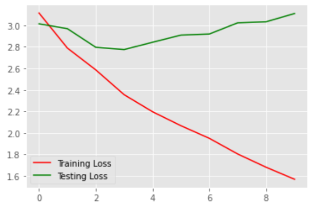


```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding_2 (Embedding)      (None, 300, 128)          1533440
_________________________________________________________________
dropout_2 (Dropout)          (None, 300, 128)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 300, 64)           49408
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_5 (Dense)              (None, 74)                2442
=================================================================
Total params: 1,620,394
Trainable params: 1,620,394
Non-trainable params: 0
_________________________________________________________________
```

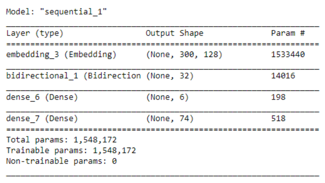


```
Epoch 1/10
86/86 [==============================] - 7s 53ms/step - loss: 3.4868 - accuracy: 0.3940 - val_loss: 3.0141 - val_accuracy: 0.43
59
Epoch 2/10
86/86 [==============================] - 4s 44ms/step - loss: 2.7877 - accuracy: 0.4240 - val_loss: 2.9688 - val_accuracy: 0.43
59
Epoch 3/10
86/86 [==============================] - 4s 44ms/step - loss: 2.6254 - accuracy: 0.4297 - val_loss: 2.7958 - val_accuracy: 0.43
41
Epoch 4/10
86/86 [==============================] - 4s 44ms/step - loss: 2.3830 - accuracy: 0.4597 - val_loss: 2.7750 - val_accuracy: 0.44
02
Epoch 5/10
86/86 [==============================] - 4s 44ms/step - loss: 2.1920 - accuracy: 0.4802 - val_loss: 2.8426 - val_accuracy: 0.45
28
Epoch 6/10
86/86 [==============================] - 4s 44ms/step - loss: 2.0709 - accuracy: 0.5022 - val_loss: 2.9092 - val_accuracy: 0.45
75
Epoch 7/10
86/86 [==============================] - 4s 44ms/step - loss: 1.9461 - accuracy: 0.5293 - val_loss: 2.9192 - val_accuracy: 0.44
25
Epoch 8/10
86/86 [==============================] - 4s 44ms/step - loss: 1.8190 - accuracy: 0.5423 - val_loss: 3.0234 - val_accuracy: 0.45
10
Epoch 9/10
86/86 [==============================] - 4s 44ms/step - loss: 1.6693 - accuracy: 0.5702 - val_loss: 3.0330 - val_accuracy: 0.45
75
Epoch 10/10
86/86 [==============================] - 4s 44ms/step - loss: 1.5624 - accuracy: 0.5971 - val_loss: 3.1096 - val_accuracy: 0.45
94
```

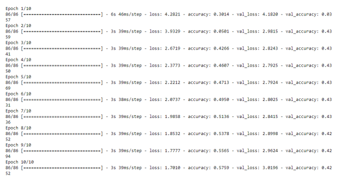




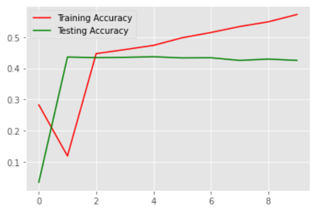
Bidirectional GRU

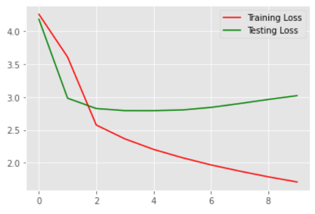


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 300, 128)          1533440
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                14016
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 198
_________________________________________________________________
dense_7 (Dense)              (None, 74)                518
=================================================================
Total params: 1,548,172
Trainable params: 1,548,172
Non-trainable params: 0
_________________________________________________________________
```



```
Epoch 1/10
86/86 [==============================] - 6s 46ms/step - loss: 4.2821 - accuracy: 0.3014 - val_loss: 4.1820 - val_accuracy: 0.03
57
Epoch 2/10
86/86 [==============================] - 3s 39ms/step - loss: 3.9329 - accuracy: 0.0501 - val_loss: 2.9815 - val_accuracy: 0.43
59
Epoch 3/10
86/86 [==============================] - 3s 39ms/step - loss: 2.6719 - accuracy: 0.4266 - val_loss: 2.8243 - val_accuracy: 0.43
41
Epoch 4/10
86/86 [==============================] - 3s 39ms/step - loss: 2.3773 - accuracy: 0.4607 - val_loss: 2.7925 - val_accuracy: 0.43
50
Epoch 5/10
86/86 [==============================] - 3s 39ms/step - loss: 2.2212 - accuracy: 0.4713 - val_loss: 2.7924 - val_accuracy: 0.43
69
Epoch 6/10
86/86 [==============================] - 3s 38ms/step - loss: 2.0737 - accuracy: 0.4950 - val_loss: 2.8025 - val_accuracy: 0.43
31
Epoch 7/10
86/86 [==============================] - 3s 39ms/step - loss: 1.9858 - accuracy: 0.5136 - val_loss: 2.8415 - val_accuracy: 0.43
36
Epoch 8/10
86/86 [==============================] - 3s 39ms/step - loss: 1.8532 - accuracy: 0.5378 - val_loss: 2.8998 - val_accuracy: 0.42
52
Epoch 9/10
86/86 [==============================] - 3s 39ms/step - loss: 1.7777 - accuracy: 0.5565 - val_loss: 2.9624 - val_accuracy: 0.42
94
Epoch 10/10
86/86 [==============================] - 3s 39ms/step - loss: 1.7010 - accuracy: 0.5759 - val_loss: 3.0196 - val_accuracy: 0.42
52
```





15) We then evaluated the accuracy and loss of all these models too.

16) **Manual Grouping** : We now tried with grouping different assignment groups together too as our dataset was highly imbalanced with Group 0 contributing to over 50% of our dataset. Hence to avoid overfitting and improve the accuracy and loss, we tried to group different assignment groups. We grouped together all assignment groups having an occurrence count of <= 20 in our dataset and addressed it as GRP_100.

17) Used Machine Learning models again to see how they perform with grouping of targets. Also tried the grouped data on Deep Learning Models.

18) We noticed the models accuracy/ f1-score didn't improve even after grouping data.

19) **Undersampling** : Tried using undersampling and used the reduced data on ML models but it didn't improve the accuracy. Didn't try this for deep learning data was not sufficient for deep learning.

20) **Oversampling : Random Oversampling / SMOTE** : Tried to oversample the data using Random Sampling and SMOTE. While using SMOTE we removed those target groups which have group count < 8. Resampling increased the data size significantly.

21) Tried using oversampled data with ML models. We found the Random Forest gave better accuracy and F1-score with oversampled data.

22) Tried using oversampled data with Deep Learning models. Here the accuracy increased but there was increased validation loss and overfitting.

Observations

The data set is of a highly imbalanced where 2 classes were on high majority and many other classes were very low compared to them , one might say they were kind of negligible as compared to highly imbalanced classes. The overall information for data that we have was very less for a highly imbalanced data, there were around 8500 of raw data which had a lot of uncleaned raw with poor Unicode format and with different languages. Considering the data set we have the following approaches:-

1) Cleaned the columns short description and long description and then further joined them to get a combined information so that their increase is data counts.

2) Post data preprocessing the cleaned data set was feed to ML and Deep learning algorithms by passing the data set as

   a) Cleaned raw data
   b) Creating a new group by grouping all Assigned Groups having ticket count less than 20.
   c) Over sampling(SMOTE and Random over sampling)
   d) Under sampling(TomekLinks).

   Cleaned raw data :- kfold was executed with several ML algorithms and multi layer perceptron and got the best accuracy and F1 score for random forest which came around 60% for both.

   _
   Data set with New Group:- With this data set accuracy was slightly better with ML algorithms as there is less imbalanced ratio compared from previous data it came around

   Over sampling :-

- With SMOTE and Randomoversampler the accuracy score was less for every ML algorithm except Decision Tree , SVM and Random Forest but still it was less as compared to processed original data in case of Randomoversampler we got high accuracy for Decision Tree and Randomforest around 85% but still F1 score remained the same which means that with oversampling data was highly overfitting and performance remained same or less for testing set.

- For deeplearning also we were getting a bit increase in accuracy but a high validation loss.

- For oversampling the execution time for deep learning algorithms were not much high as compared to machine learning algorithms. Machine learning models on oversampled data the execution took a lot of time.

  Under sampling:- With this approach the data was trimmed down to around 7600 and hence there was information loss , but in case of performance it was the same as metrics of original data. We didn't try DL as with under sampling there would be less data for it to train.

Conclusions:
Several approached were tried to overcome less amount of data with highly imbalanced distribution which included preprocessing and combining the 2 attributes to make it one and made the rows inputs as sensible and sound as it could , for decoding some other encoded formats we used libraries such as ftfy , for that we used google translator , remover punctuations , stopwords etc. and then finally processed and combined data. After that we applied different methods and techniques for imbalanced data which gave less satisfactory results and for deep learning approaches we need to have a lot of data for the algorithm to "learn" something , unfortunately we had less data , and in case of oversampled data we had high validation loss and not highly improved accuracy hence for avoiding the overfitting and information loss we came up with the conclusion of 2 approaches;

1) Use Random forest on original processed data because on all the approaches we noticed the result for random forest was better as compared to others because it itself is an ensemble method and here we are dealing with highly imbalanced data and a lot number of classes(Y values) so random forest fits fine for this situation.

2) Other solution we can pivot for is using the grouping approach where we group classes < 20 , we can use that model , for a result which will fall in that group , the limitation of this model is that for reclassification of the exact group there we will need to do manual coding and use libraries such as regular expression for clarifying further classification of that combined group.

The sequences in the data are very less hence the DL models which are dependent on long sequences do not work successfully. Oversampling with ML works better because of the presence of the words to classify into a category which are learnt better with models like RF