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 review 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

Approach taken for Exploratory Data Analysis & Pre-Processing :

- We start exploratory data analysis by getting the basic information on our dataset.

The snapshot of the same is shared below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8500 entries, 0 to 8499
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Short description  8492 non-null   object
 1   Description        8499 non-null   object
 2   Caller             8500 non-null   object
 3   Assignment group   8500 non-null   object
dtypes: object(4)
memory usage: 265.8+ KB
```

- Next, we try to identify the total number of unique assignment groups that are used for classifying the incidents. We come across 74 such unique groups.

- Similarly, we tried to identify the total number of unique entries for the Short Description, Description and Caller columns and we found that there were 7482 unique entries for the Short Description column, 7818 unique entries for the Description column and 2950 unique entries for the Caller column.
- We then tried to check for the text encoding on random columns and we came across samples where the encoding wasn't done properly as the samples were multilingual ,hence we had to encode the text samples correctly before moving further with our pre-processing.

  A snapshot of one such sample is shared below.

```
[ ] dfOriginal.loc[5257,'Short description']

'å^\xa0é™¤ä¸ºäº†ä¸€ä¸ªexcel æ-‡æ¡£ï¼Œæ-‡æ¡£åœ°å\x9d€ï¼š\\\\HostName_17\\teams\\business\\c2 qualitycontrol\\c25 quality proje
ct\\c251 k100\\weekly layered process audi'
```

- We used the FTFY package to deal with the encoding challenge with our dataset and were able to successfully rectify the same.

  A snapshot of the same is shared below.

```
[ ] dfOriginal.loc[5257,'Short description']

'删除了一个excel 文档，文档地址：\\\\HostName_17\\teams\\business\\c2 qualitycontrol\\c25 quality project\\c251 k100\\weekly
layered process audi'
```

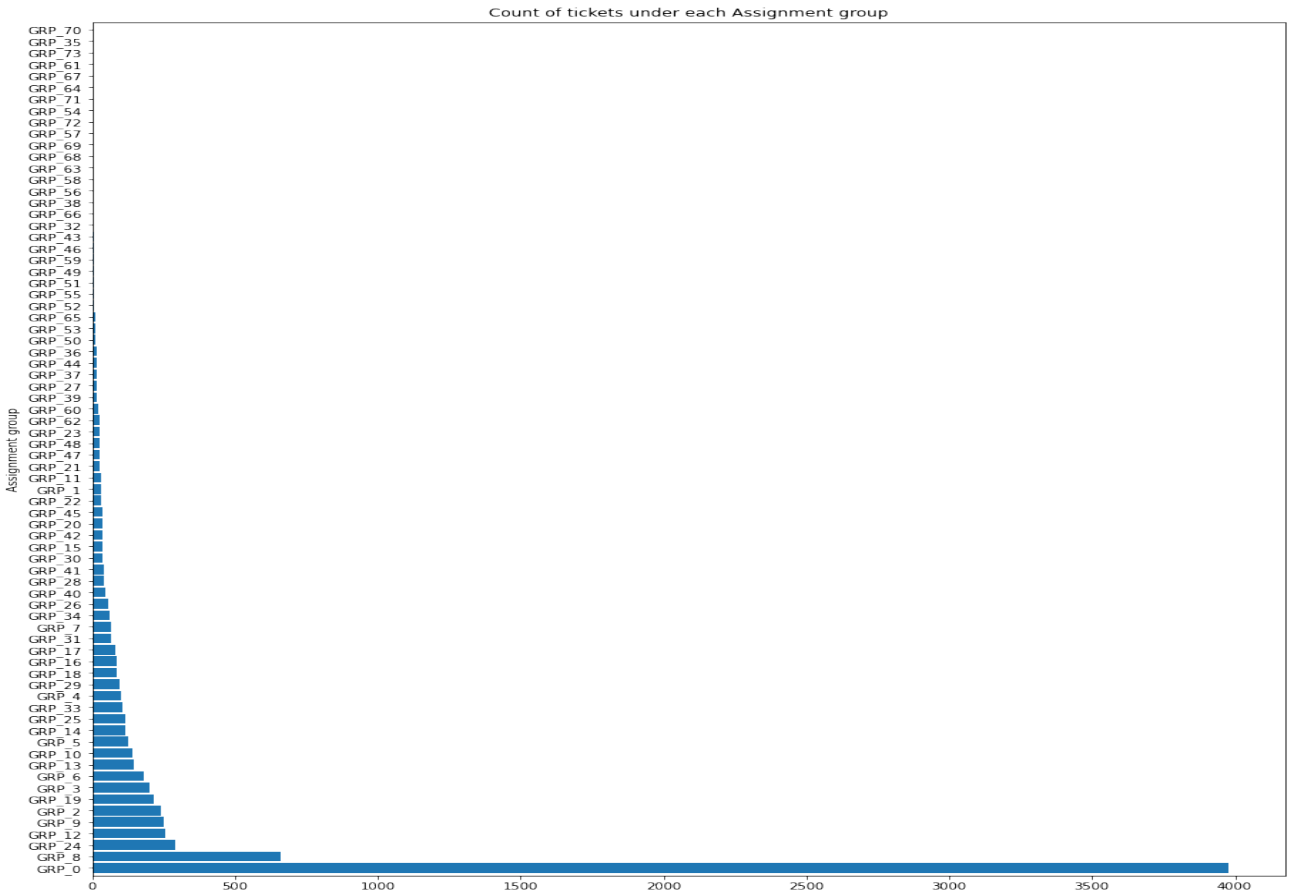
- Next, we used the Langdect package to find out the ceilings of 'Short description' and 'Description' columns. The accuracy of the detection was found to be mixed, as there were a few text samples that contained english text but it was misidentified and there were exceptions thrown while detecting.
- Given the fact that our dataset was multilingual, we decided to use the Google Translator API to convert all textual data to the English language, thereby increasing the vocabulary for our training sample.
- Next, we decided to strip the irrelevant data(for ex: mail id,etc.) from the Description column using regular expressions. We also removed all white spaces from our dataset and converted everything to lower case.

- We now also removed all the punctuation marks from the dataset.
- Next, we stripped all the leading and trailing white spaces.
- Next, we dropped all the unwanted columns.
- We now decided to look at the description of Assignment group column.

  A snapshot of the same is shared below.

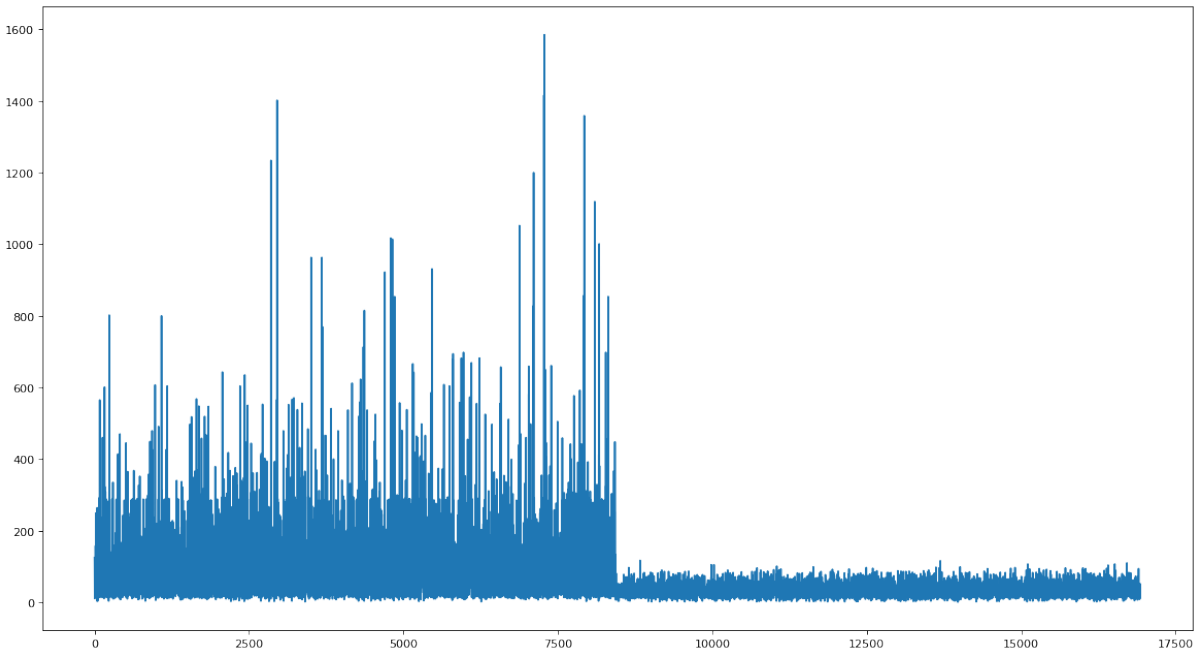
```
count      74.000000
mean      114.864865
std       465.747516
min         1.000000
25%         5.250000
50%        26.000000
75%        84.000000
max      3976.000000
Name: Assignment group, dtype: float64
```

- Next, we plotted a graph to see the count of tickets/incidents under each assignment groups. A snapshot of the same is shared below.



- We now moved on to removing all the stopwords from our dataset and performed Lemmatization on our dataset.
- We tried to create 2 datasets, one where we concatenated the short description and assignment group columns and the other where we concatenated the description and assignment group columns. We did this thinking we could build two different models and train them separately based on the data given as a part of the Short Description and Description columns and combine them together to classify new tickets. But this approach did not work very well.
- Next, we created a word cloud for our dataset, a snapshot of which is shared below.

- We then went ahead and performed tokenization on our vocabulary.



- Next, we plotted a graph to see the text count in the Description column and Short description column per row. A snapshot of the graph is shared below, where we can see the left half of the graph is that of the text count of the Description column and the right half is that of the Short description column.

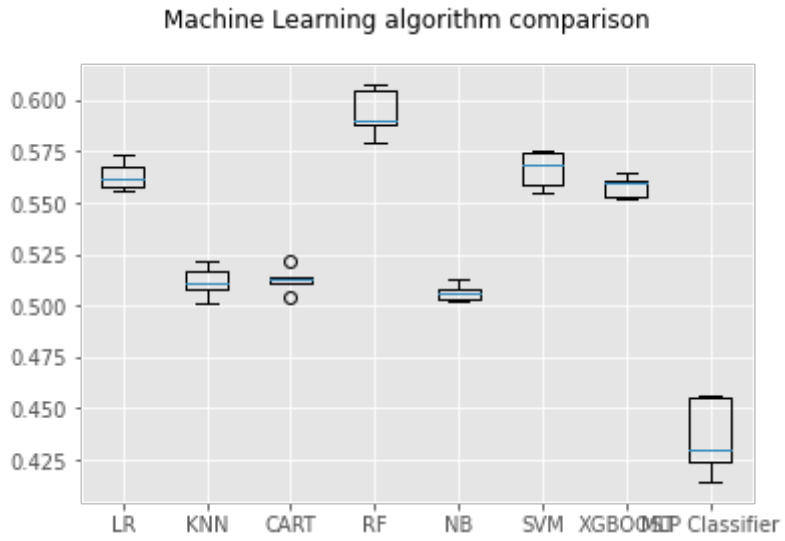
- We used Label Encoding and TF-IDF Vectorizer to pre process our data before feeding it to our models.

Approach taken for Model building :

- **Machine Learning**
- First, we tried the Machine learning algorithms like
  - Logistic Regression
  - KNN
  - CART
  - Random Forest
  - Naive Bayes
  - SVM
  - XGBOOST
  - MLP Classifier

    Our observations for each algorithm is illustrated below.

  - LR: 0.563263 (0.006412)
  - KNN: 0.511620 (0.007172)
  - CART: 0.512676 (0.005624)
  - RF: 0.593897 (0.010706)
  - NB: 0.506455 (0.003980)
  - SVM: 0.566549 (0.008288)
  - XGBOOST: 0.558216 (0.004971)
  - MLP Classifier: 0.435915 (0.016917)\



- Found Random forest to have better results.

Next tried with Deep learning approach

- **ANN - Artificial Neural Networks**

  The architecture used for our ANN is illustrated below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 300)]             0
_________________________________________________________________
embedding_13 (Embedding)     (None, 300, 128)          1552000
_________________________________________________________________
flatten_10 (Flatten)         (None, 38400)             0
_________________________________________________________________
dense_26 (Dense)             (None, 32)                1228832
_________________________________________________________________
dense_27 (Dense)             (None, 74)                2442
=================================================================
Total params: 2,783,274
Trainable params: 2,783,274
Non-trainable params: 0
_________________________________________________________________
```

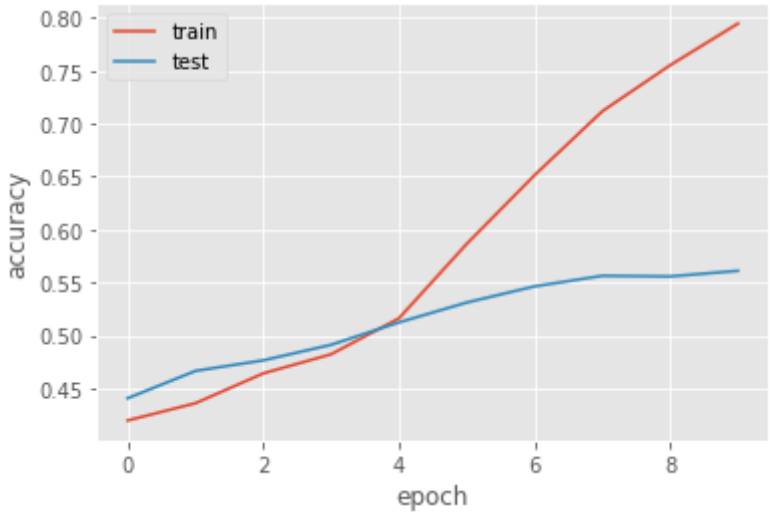
The output for our ANN is illustrated below.

```
Epoch 1/10
69/69 [==============================] - 5s 60ms/step - loss: 3.2010 - acc: 0.4111 - val_loss: 2.7494 - val_acc: 0.4407

Epoch 00001: val_acc improved from -inf to 0.44073, saving model to weights-simple.hdf5
Epoch 2/10
69/69 [==============================] - 4s 58ms/step - loss: 2.6315 - acc: 0.4391 - val_loss: 2.4842 - val_acc: 0.4665

Epoch 00002: val_acc improved from 0.44073 to 0.46655, saving model to weights-simple.hdf5
Epoch 3/10
69/69 [==============================] - 4s 58ms/step - loss: 2.4321 - acc: 0.4531 - val_loss: 2.3504 - val_acc: 0.4765

Epoch 00003: val_acc improved from 0.46655 to 0.47653, saving model to weights-simple.hdf5
Epoch 4/10
69/69 [==============================] - 4s 58ms/step - loss: 2.1807 - acc: 0.4744 - val_loss: 2.2497 - val_acc: 0.4912

Epoch 00004: val_acc improved from 0.47653 to 0.49120, saving model to weights-simple.hdf5
Epoch 5/10
69/69 [==============================] - 4s 58ms/step - loss: 1.9165 - acc: 0.5125 - val_loss: 2.1768 - val_acc: 0.5123

Epoch 00005: val_acc improved from 0.49120 to 0.51232, saving model to weights-simple.hdf5
Epoch 6/10
69/69 [==============================] - 4s 57ms/step - loss: 1.7097 - acc: 0.5704 - val_loss: 2.1083 - val_acc: 0.5311
```

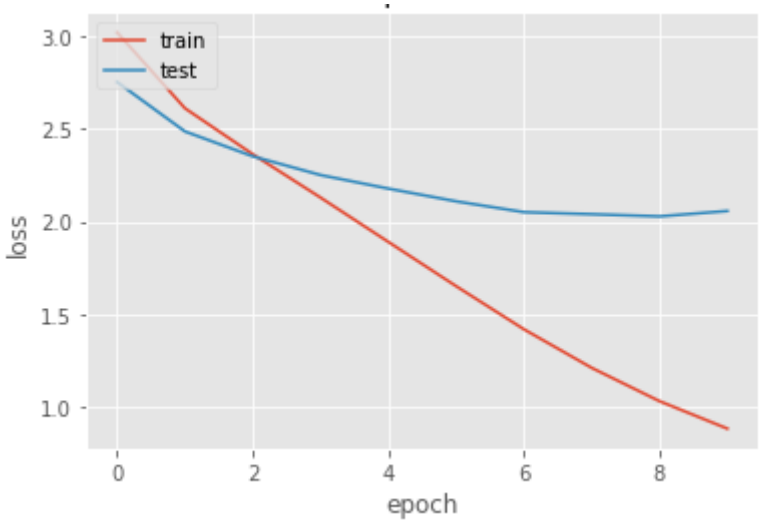
```
Epoch 00006: val_acc improved from 0.51232 to 0.53110, saving model to weights-simple.hdf5
Epoch 7/10
69/69 [==============================] - 4s 58ms/step - loss: 1.4588 - acc: 0.6463 - val_loss: 2.0502 - val_acc: 0.5464

Epoch 00007: val_acc improved from 0.53110 to 0.54636, saving model to weights-simple.hdf5
Epoch 8/10
69/69 [==============================] - 4s 58ms/step - loss: 1.2314 - acc: 0.7046 - val_loss: 2.0392 - val_acc: 0.5563

Epoch 00008: val_acc improved from 0.54636 to 0.55634, saving model to weights-simple.hdf5
Epoch 9/10
69/69 [==============================] - 4s 57ms/step - loss: 1.0338 - acc: 0.7549 - val_loss: 2.0275 - val_acc: 0.5558

Epoch 00009: val_acc did not improve from 0.55634
Epoch 10/10
69/69 [==============================] - 4s 57ms/step - loss: 0.9123 - acc: 0.7857 - val_loss: 2.0564 - val_acc: 0.5610

Epoch 00010: val_acc improved from 0.55634 to 0.56103, saving model to weights-simple.hdf5
```

We also plotted the graphs for All Data Unsampled ANN model accuracy and All Data Unsampled ANN model loss, both are illustrated below.

Graph for All Data Unsampled ANN Model Accuracy

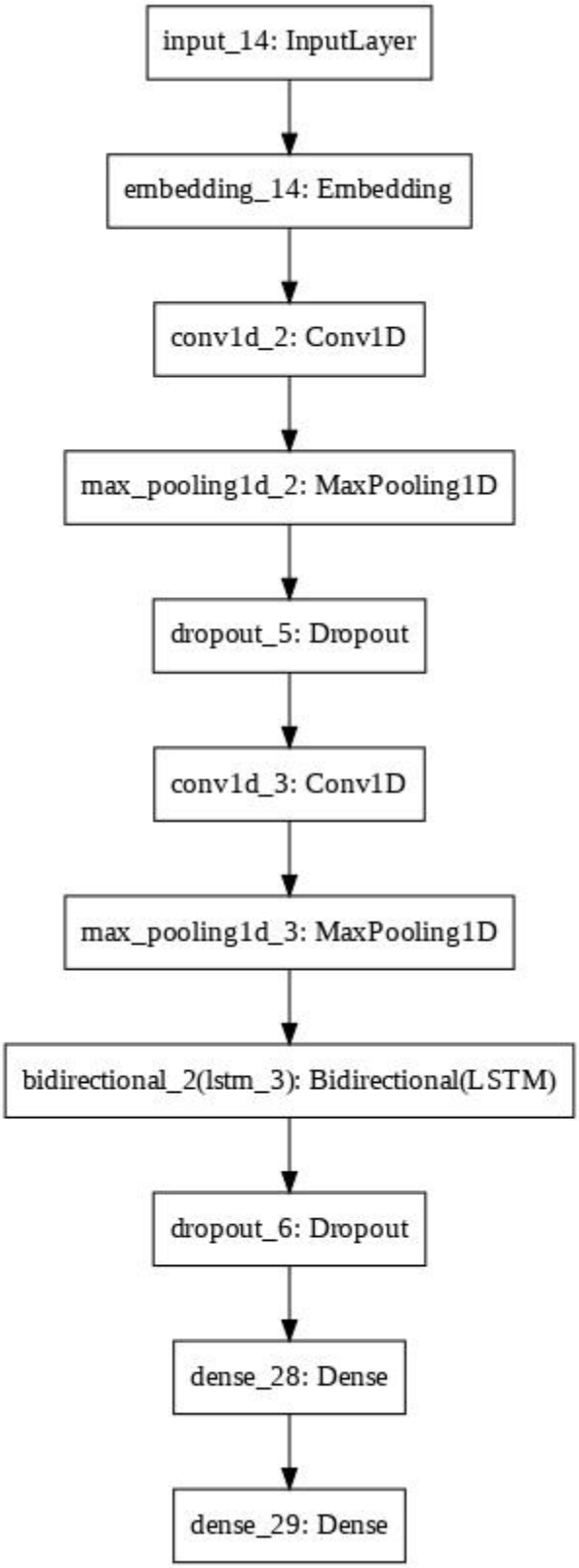


Graph for All Data Unsampled ANN Model Loss



- **RNN - Recurrent Neural Network**

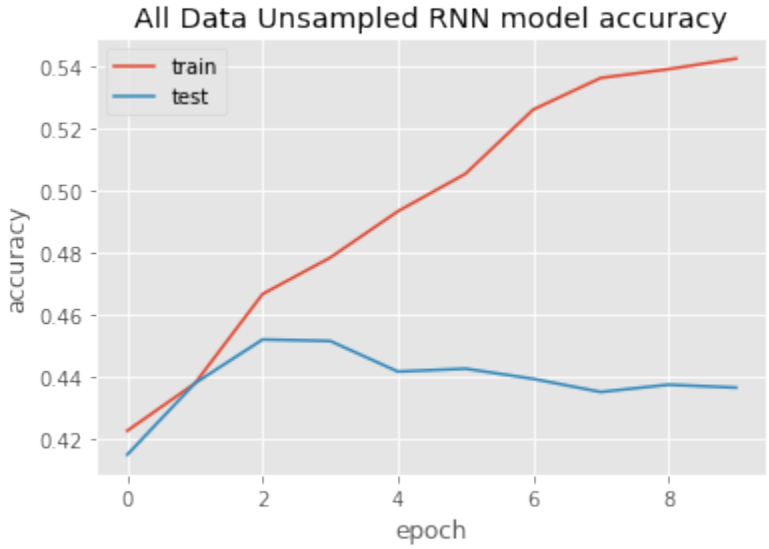
  The diagrammatic representation of the model built is illustrated below.

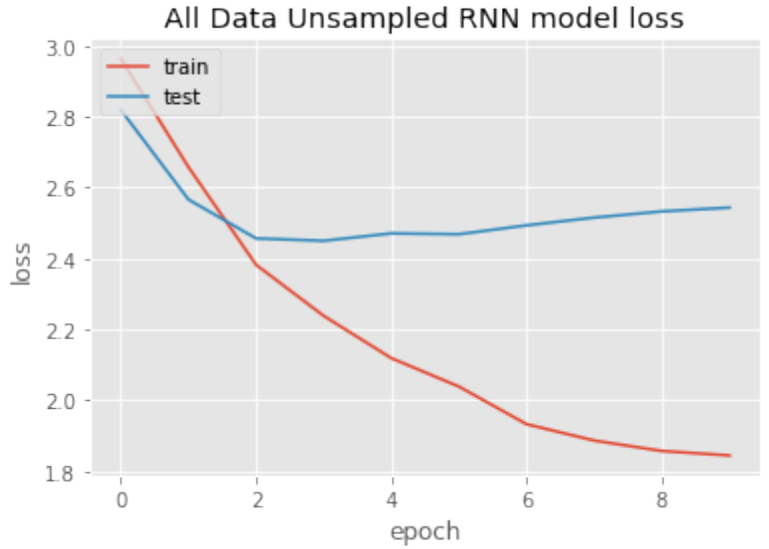

input_14: InputLayer
embedding_14: Embedding
conv1d_2: Conv1D
max_pooling1d_2: MaxPooling1D
dropout_5: Dropout
conv1d_3: Conv1D
max_pooling1d_3: MaxPooling1D
bidirectional_2(lstm_3): Bidirectional(LSTM)
dropout_6: Dropout
dense_28: Dense
dense_29: Dense

The snapshot of the output for the RNN model is illustrated below.

```
Epoch 1/10
86/86 [==============================] - 86s 937ms/step - loss: 3.2535 - accuracy: 0.4064 - val_loss: 2.8184 - val_accuracy: 0.4148

Epoch 00001: val_accuracy improved from -inf to 0.41483, saving model to model-001-0.414829.h5
Epoch 2/10
86/86 [==============================] - 88s 1s/step - loss: 2.7246 - accuracy: 0.4301 - val_loss: 2.5657 - val_accuracy: 0.4378

Epoch 00002: val_accuracy improved from 0.41483 to 0.43782, saving model to model-002-0.437823.h5
Epoch 3/10
86/86 [==============================] - 79s 919ms/step - loss: 2.4205 - accuracy: 0.4599 - val_loss: 2.4560 - val_accuracy: 0.4519

Epoch 00003: val_accuracy improved from 0.43782 to 0.45190, saving model to model-003-0.451901.h5
Epoch 4/10
86/86 [==============================] - 79s 920ms/step - loss: 2.2419 - accuracy: 0.4790 - val_loss: 2.4493 - val_accuracy: 0.4514

Epoch 00004: val_accuracy did not improve from 0.45190
Epoch 5/10
86/86 [==============================] - 79s 921ms/step - loss: 2.1035 - accuracy: 0.4951 - val_loss: 2.4702 - val_accuracy: 0.4416

Epoch 00005: val_accuracy did not improve from 0.45190
Epoch 6/10
86/86 [==============================] - 79s 922ms/step - loss: 2.0451 - accuracy: 0.5052 - val_loss: 2.4678 - val_accuracy: 0.4425

Epoch 7/10
86/86 [==============================] - 79s 919ms/step - loss: 1.9490 - accuracy: 0.5240 - val_loss: 2.4928 - val_accuracy: 0.4392

Epoch 00007: val_accuracy did not improve from 0.45190
Epoch 8/10
86/86 [==============================] - 79s 918ms/step - loss: 1.8780 - accuracy: 0.5359 - val_loss: 2.5148 - val_accuracy: 0.4350

Epoch 00008: val_accuracy did not improve from 0.45190
Epoch 9/10
86/86 [==============================] - 79s 919ms/step - loss: 1.8129 - accuracy: 0.5473 - val_loss: 2.5322 - val_accuracy: 0.4374

Epoch 00009: val_accuracy did not improve from 0.45190
Epoch 10/10
86/86 [==============================] - 79s 917ms/step - loss: 1.8503 - accuracy: 0.5363 - val_loss: 2.5429 - val_accuracy: 0.4364

Epoch 00010: val_accuracy did not improve from 0.45190
```

The graph plots for Unsampled RNN model accuracy and model loss are illustrated below.





- **LSTM - Long Short Term Memory networks**

  We created a sequential model with LSTM. The model structure of the same is illustrated below.

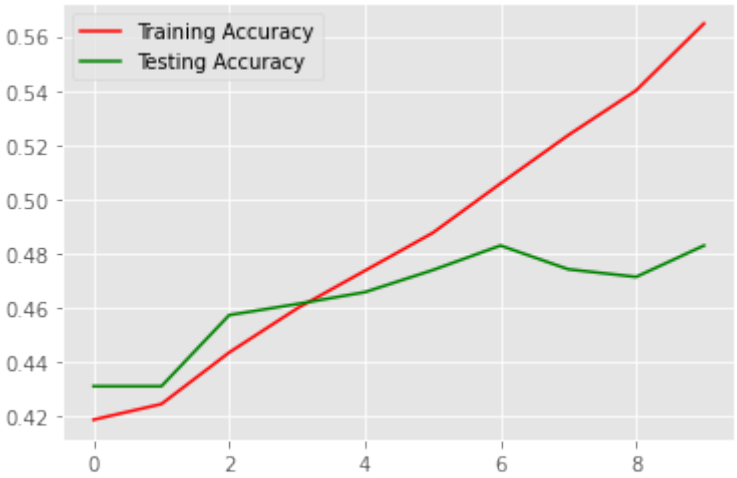
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_15 (InputLayer)        [(None, 500)]             0
_________________________________________________________________
embedding_15 (Embedding)     (None, 500, 128)          1552000
_________________________________________________________________
dropout_7 (Dropout)          (None, 500, 128)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 500, 64)           49408
_________________________________________________________________
lstm_5 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_30 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_31 (Dense)             (None, 74)                2442
=================================================================
Total params: 1,638,954
Trainable params: 1,638,954
Non-trainable params: 0
_________________________________________________________________
```

The output for the sequential model built using LSTM is shown below.

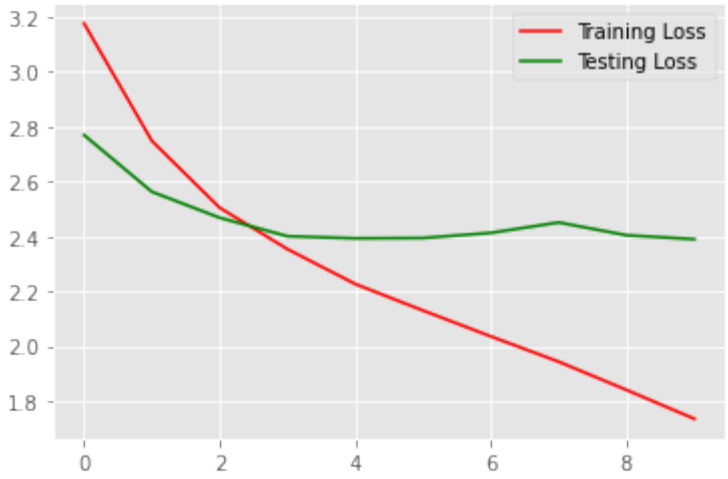
```
Epoch 1/10
75/75 [==============================] - 119s 2s/step - loss: 3.5784 - accuracy: 0.3987 - val_loss: 2.7685 - val_accuracy: 0.4309
Epoch 2/10
75/75 [==============================] - 115s 2s/step - loss: 2.8346 - accuracy: 0.4110 - val_loss: 2.5621 - val_accuracy: 0.4309
Epoch 3/10
75/75 [==============================] - 114s 2s/step - loss: 2.5675 - accuracy: 0.4280 - val_loss: 2.4678 - val_accuracy: 0.4571
Epoch 4/10
75/75 [==============================] - 114s 2s/step - loss: 2.3490 - accuracy: 0.4627 - val_loss: 2.4004 - val_accuracy: 0.4612
Epoch 5/10
75/75 [==============================] - 114s 2s/step - loss: 2.2666 - accuracy: 0.4586 - val_loss: 2.3928 - val_accuracy: 0.4656
Epoch 6/10
75/75 [==============================] - 114s 2s/step - loss: 2.1274 - accuracy: 0.4855 - val_loss: 2.3940 - val_accuracy: 0.4737
Epoch 7/10
75/75 [==============================] - 114s 2s/step - loss: 2.0585 - accuracy: 0.4943 - val_loss: 2.4127 - val_accuracy: 0.4828
Epoch 8/10
75/75 [==============================] - 114s 2s/step - loss: 1.9550 - accuracy: 0.5196 - val_loss: 2.4506 - val_accuracy: 0.4740
Epoch 9/10
75/75 [==============================] - 114s 2s/step - loss: 1.8550 - accuracy: 0.5374 - val_loss: 2.4040 - val_accuracy: 0.4712
Epoch 10/10
75/75 [==============================] - 114s 2s/step - loss: 1.7213 - accuracy: 0.5695 - val_loss: 2.3896 - val_accuracy: 0.4828
```

We had plotted graphs for training and validation accuracy per epoch and training and validation loss per epoch. The two graphs are illustrated below.

Graph for training and validation accuracy per epoch



Graph for training and validation loss per epoch



- **Bidirectional GATED RECURRENT UNIT :**

  The architecture of the bidirectional GRU is illustrated below.

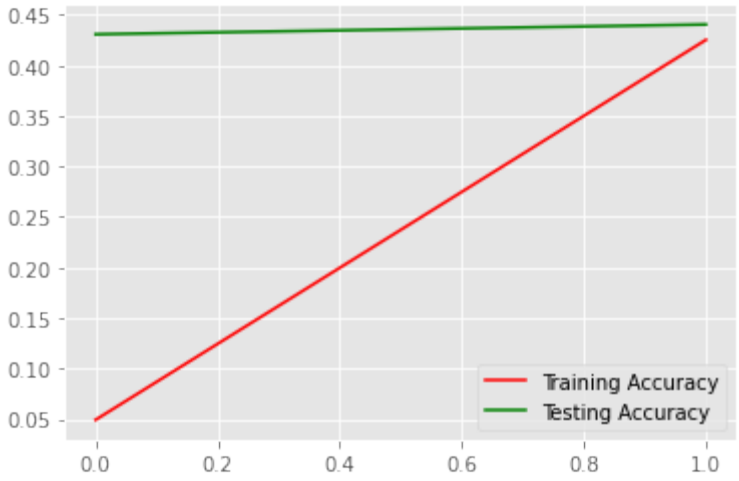
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_16 (Embedding)     (None, 500, 128)          1552000
_________________________________________________________________
bidirectional_3 (Bidirection (None, 32)                14016
_________________________________________________________________
dense_32 (Dense)             (None, 6)                 198
_________________________________________________________________
dense_33 (Dense)             (None, 74)                518
=================================================================
Total params: 1,566,732
Trainable params: 1,566,732
Non-trainable params: 0
_________________________________________________________________
```

The output for the bidirectional GRU model is illustrated below.

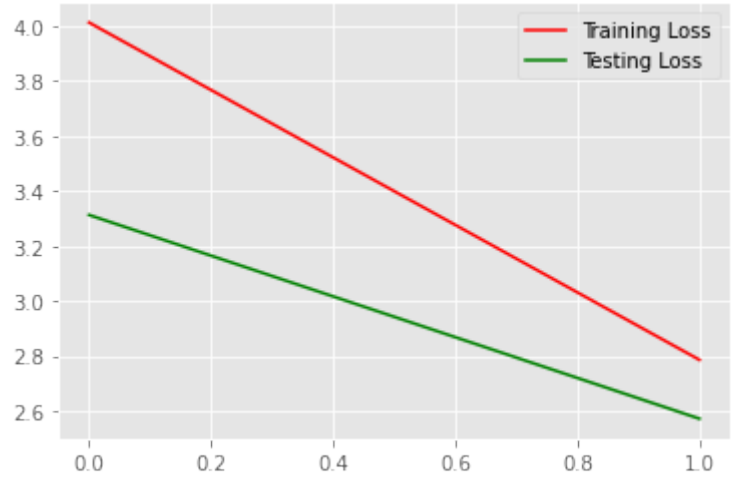
```
Epoch 1/2
75/75 [==============================] - 52s 646ms/step - loss: 4.2028 - accuracy: 0.0220 - val_loss: 3.3125 - val_accuracy: 0.4309
Epoch 2/2
75/75 [==============================] - 48s 637ms/step - loss: 2.9673 - accuracy: 0.4249 - val_loss: 2.5711 - val_accuracy: 0.4406
```

We also plotted the graphs for training and validation accuracy per epoch and training and validation loss per epoch, both of which have been illustrated below.

Graph for training and validation accuracy per epoch



Graph for training and validation loss per epoch



Our observation :

We have tried all the best possible machine learning and deep learning approaches for this NLP project.

We have observed the machine learning algorithms working better for us and our understanding for the same is that our dataset was very limited. Machine learning algorithms look to work better on smaller data sets as compared to Deep Learning models because Deep Learning networks train better over larger datasets.