

|  |  |
| --- | --- |
| Capstone Project: Automatic Ticket Classification | |
|  | Team: Naresh, Ganesh, Seresha, Raju  Mentor: Arun |

Text

Description automatically generated with low confidence

Table of Contents

[Capstone Project: Automatic Ticket Classification 1](#_Toc102813915)

[Goal of the project 3](#_Toc102813916)

[1st Milestone: 4](#_Toc102813917)

[1.Problem interpretation 4](#_Toc102813918)

[2. Data analysis and preprocessing 5](#_Toc102813919)

[Step 1: Import, Read, and clean data 5](#_Toc102813920)

[Step 2: Exploratory Analysis 7](#_Toc102813921)

[Step 3: Preprocessing and Feature Engineering 8](#_Toc102813922)

[3.Machine Learning Models 14](#_Toc102813923)

[Step 1: Build Supervised ML models on dataset using TFIDF 15](#_Toc102813924)

[Step 2: Hyper-parameter tuning 16](#_Toc102813925)

[Step 3: Evaluate Hyper-parameter tuning 17](#_Toc102813926)

[Step 4: Pickle model for future use 17](#_Toc102813927)

[2nd Milestone 17](#_Toc102813928)

[4. Deep Learning Models 17](#_Toc102813929)

[Model 1: CNN 19](#_Toc102813930)

[Model 2: Forward LSTM 22](#_Toc102813931)

[Model 3: Bidirectional LSTM 26](#_Toc102813932)

[Model 4: BERT 30](#_Toc102813933)

[5. Model Summary and Overall Evaluation 33](#_Toc102813934)

[Overall, BERT is the best model with both Accuracy and Recall metrics close to 95%. 34](#_Toc102813935)

# Goal of the project

**Objective**:

● To undertake a multi-faceted project that demonstrates our understanding and mastery of the key conceptual and technological aspects of Deep Learning.

● To develop an understanding of how challenging human-level problems can be approached and solved using a combination of tools and techniques.

● To understand current scenarios in deep learning, understand the practicalities and the trade-offs that need to be made when solving a problem in real life.

**Milestones**: This project has 2 milestones. You get 3 weeks for completing each milestone. The 1st milestone should be submitted as part of the interim report.

**1st Milestone:**

1. Problem interpretation

∙ Understand the data

∙ Make an abstract or an overview based on your approach

∙ Break the problem into smaller tasks

∙ Discuss among your teammates and share responsibilities

2. Data analysis and preprocessing: Visual displays are powerful when used well, so think carefully about the information the display.

∙ Include any insightful visualization

∙ Share and explain particularly meaningful features, interactions or summary of data

∙ Display examples to input in your model

∙ Explain changes to be incorporated into data so that it becomes ready for the model

3. Modeling

∙ What kind of neural network you have used and why? ∙ What progress you have made towards your intended solution?

**2nd Milestone:**

4. Model evaluation

∙ Describe how you will proceed with the analysis Proprietary content.

∙ Compare different models and choose which model to use

∙ Do hyper-parameter tuning of your model

∙ How will you build on your initial analysis to increase the accuracy of your model?

5. Presentation and Report

∙ You should start preparing the final report at least 2 weeks prior to the project completion date.

∙ Teams should send a draft of the Last of the project before the last session to the mentor and get the necessary inputs for submission.

∙ The expectations for the final report will be included in your Capstone course page

# 1st Milestone:

## 1.Problem interpretation

Great Learning PGP - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING culminates with a Capstone Project. Automatic Ticket Assignment is final capstone project

This Project involves “Natural Language Processing” and consists of analyzing ticket short description and description and other relevant features to arrive at a model to automatically assign a ticket to one of the Groups

Dataset has one Class Label (Incident / Ticket Group) and three independent Features (Short Description, Description and Caller)

|  |  |
| --- | --- |
| Data Set Feature/Label | Description |
| Short description | Description of the ticket in as fewer number of words as possible, while describing the issue |
| Description | Full description of the incident in as much detail as possible to help on appropriate classification and to help on successful resolution of the issue |
| Caller | Person calling about ticket (Caller identifier or name) |
| Assignment Group | Label of this dataset and denotes the final Assignment group of this ticket. |

Objectives of the project:

* Load and clean files for analysis
* Exploratory analysis on data present in the files
* Build frequency plots and word clouds to pictorially represent the data
* Understand the specifics of each text column and be able to tokenize the words
* Complete Feature Engineering to decide on features required for model
* Complete Hypothesis testing, if required
* Build Models aimed at predicting Assignment Group
* Finalize on one best and accurate model

This problem needs collaborative efforts from all the team members and after multiple discussions and considering strengths/interest of team members, below is the task assignment finalized

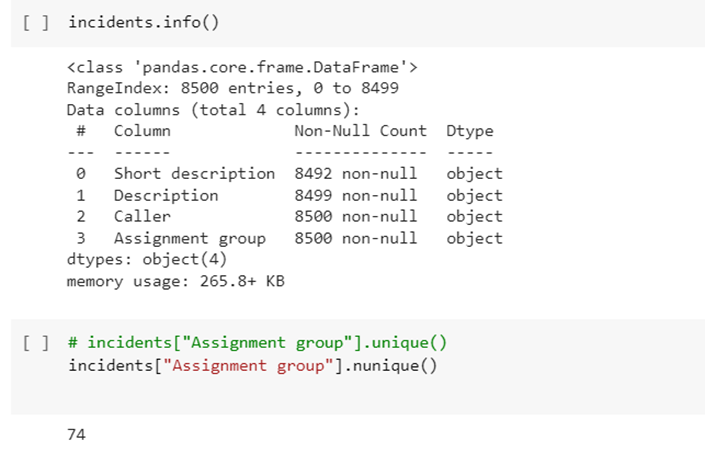
|  |  |
| --- | --- |
| Task | Assignee |
| Load and clean files for analysis | Ganesh |
| Exploratory Data analysis on data present in the files | Ganesh, Naresh |
| Build frequency plots and word clouds to pictorially represent the data | Sireesha |
| Understand the specifics of each text column and be able to tokenize the words | Raju |
| Complete Feature Engineering to decide on features required for model | Ganesh, Naresh |
| Complete Hypothesis testing | Naresh |
| Build Models aimed at predicting Assignment Group | Ganesh, Naresh, Sireesha, Raju |
| Finalize on one best and accurate model | Ganesh, Naresh, Sireesha, Raju |
| Documentation for Interim and Final Report | Ganesh, Naresh, Sireesha, Raju |

## 2. Data analysis and preprocessing

### Step 1: Import, Read, and clean data

We read the source files and sample data to form as base for the Auto Assignment prediction exercise. The sub-steps involved in the exercise are:

* Read the input file, there are 8500 rows and four columns
* There are 74 unique Class labels, this is a multi-class classification problem statement



● Data set has some duplicate entries and can be removed; Data set has some features with null values, and they are in minority and will not impact the model.

Dropped the duplicate rows and rows that had null values

● 75% of the Groups have less than 100 tickets assigned

● Dataset is Highly skewed with ~50% of tickets assigned to the Class label - GRP\_0

● Many Assignment groups have less representation in this Dataset



### Step 2: Exploratory Analysis

* There are close to 3000 unique callers, however, by looking at this feature and the class labels - there is not much of any significance noted. Also, did a chi-square test to validate the null hypothesis and didn’t find any significance. Hence this feature is dropped.

Graphical user interface, text, application, email

Description automatically generated

* Almost 75% of the class label rows have less than 100 rows. This means that there are several class labels with very less sample
* Looking at the distribution of the class labels - GRP\_0 stands out with almost close to 50% of the data in one single class: and the remaining in the other 73 classes. It also shows that there is class imbalance.

### Step 3: Preprocessing and Feature Engineering

* Tried a new idea to create groups based on the number of incidents created against the class labels and grouped them as those with <50 rows, 51 to 100 rows, 101 to 150 rows and 151 to 200 rows.

A picture containing chart

Description automatically generated

Chart, bar chart

Description automatically generated

* With this grouping - we got 11 new groups. We can use these labels to classify first and we can build a cascading model to further classify within these new groups.

Graphical user interface, text, email

Description automatically generated

* Though grouping based on count of tickets may not sound much logical, this is just an initial attempt to break down the problem and see what can be done further.
* Now, leaving the GRP\_0 label, we can see that all the other 10 new class labels have at least some balances in terms of the data.
* It is also noted that the features ‘Short Description’ and ‘Description’ are duplicated in several rows. So, an idea struck us that we can merge these two texts to one feature and ensure there is no duplicated text. Created a Full Description as a new column that can be used further for analysis and modeling.

Graphical user interface, application

Description automatically generated

* At this point, created the data frame that will hold - Full description, original class values and New Class values

Graphical user interface, text, application, email

Description automatically generated

* Did text preprocessing covering the basic processing like removing numbers, punctuations, whitespaces, conversion of all words to lowercase, remove HTML tags, replace contractions, remove stop words, remove mail related words, remove junk characters, remove accented characters and finally word lemmatization

Graphical user interface, text, application, email

Description automatically generated

* Created the word cloud for each group to understand what are the top words that predominantly represent those classes.

Text

Description automatically generated

* Based on the manual analysis using word cloud - was able to identify that some of the classes are having similar keywords and eventually can be grouped as one class.

Text

Description automatically generated

* Also, did the same analysis for the new groups created and cross validated them across groups and created new groups that have keyword correlation. After this step the number of groups / class labels have become 12. example: job failure related incidents are grouped as one; ERP related incidents are grouped as one; there are three groups that had some special chars or different language - they are grouped as one - we can even delete this group later.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

* Now again, leaving the GRP\_0 class, the distribution for the dataset is good, but still need to do some work on up sampling if possible

A picture containing bar chart

Description automatically generated

* Revalidated all the word cloud with the newly formed groups and double checked for any keyword correlation manually. Now, it seems to be fine.
* Based on the word cloud and using TFIDF, we tried to do PCA for feature dimension reduction. Used TFIDF Vectorization method to convert the Words in the description to number vectors. Used clustering mechanisms to see if there are clusters that can be visualized. Used KMeans find out the Optimal number of Clusters and then visualize the same. From the PCA cluster plot and TSNE cluster plot - we can understand that there is good clustering of data seen across the classes. So, as an exploratory exercise, there is a plan to create labels from these clusters and compare them with the given class labels. Also, build models on these new cluster-based Class labels and see how much variation on prediction happens. Based on that, we will do more analysis as part of phase 2 of this project in the upcoming weeks.

Chart, line chart

Description automatically generated

A picture containing text, vector graphics, screenshot

Description automatically generated

* Translated all non-English sentences to English Language for ease of model building using Google Translator
* We need to Up Sample all other Groups except GR0. We thought to reduce amount of up sampling by sub grouping GR0 based on key words and then up sampling all other Groups to incident counts of GR0 Subgroups. However, this approach resulted in reduced accuracy on the models and so we up sampled all other non GR0 groups to the level of GR0. We have done this using nlpaug

Table

Description automatically generated

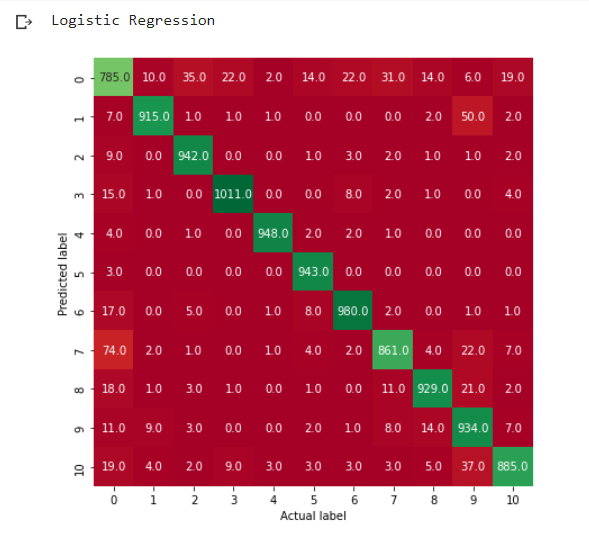
* Chart, bar chart

  Description automatically generated

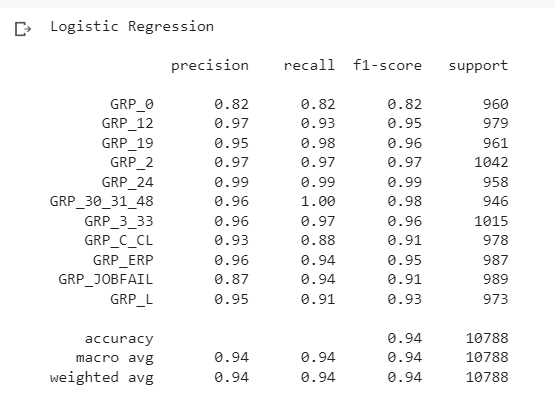
## 3.Machine Learning Models

### Step 1: Build Supervised ML models on dataset using TFIDF

* Create the train & test data using the train\_test\_split
* create the vector for the corpus
* Fit the Train word vector to tf-idf
* Tried few of the Supervised ML models on the train & test data:
  + Logistic Regression
  + Decision Tree Classifier
  + Random Forest Classifier
  + SVM Classifier
  + Naive Bayes Classifier
* Performed hyper parameter tuning for each of the models and picked the best model parameters
* pickled the model for future use
* Loaded the pickled models to predict for the IDF transformed test data
* Created the confusion matrix to validate the output from each model (a sample visualization for Logistic Regression is shown here)



* Also, created the classification report for performance against each class level and at overall level. One of the models LR classification outputs given below.



* Stored the accuracy and F1 metrics score before tuning and after tuning and from the model prediction
* As per the models built so far, we are getting good F1 scores in Logistic regression and SVM classifier supervised ML models.

### Step 2: Hyper-parameter tuning

Hyper tuning was done with Logistic regression, Decision tree classifier, SVM classifier and Naïve Bayes.

A screenshot of a computer

Description automatically generated with medium confidence

### Step 3: Evaluate Hyper-parameter tuning

Table

Description automatically generated

### Step 4: Pickle model for future use

* Pickled the hyper tuned models and loaded the pickled model to check for test data and results.

# 2nd Milestone

## 4. Deep Learning Models

We have tried all possible traditional Machine Learning Classifiers including ensembles in Milestone 1. This being a text classification problem, deep learning and in specific NLP models are generally better choices due to the learning it can bring in.

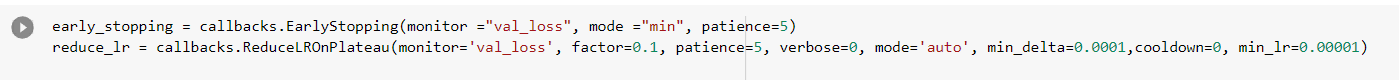
We have tried below different Deep Learning models part of this exercise

1. CNN
2. Forward LSTM
3. Bidirectional LSTM
4. BERT

To successfully train and evaluate these models, we have completed all below necessary steps

1. Using NLTK tokenize, built the Word Corpus and determine Vocabulary Size
2. Using Label Encoder transformed the labels to one-hot encoded labels, so that we can use them effectively for this classification problem in neural network models.
3. Tokenize/Vectorize the words to get Deep Learning Feedable Embeddings
4. Glove Embeddings and Weight Matrix
5. Train Test split of Embeddings/Vectorizations
6. Define architecture, fit, evaluate and hyper tune all these models

Additionally, set up the early stopping and reducing learning rate in epochs by monitoring ‘validation losses. Early stopping is done based on the patience level set, so that we need not run if there is not much reduction on the validation loss; Also, learning rate is reduced periodically based on the constraints set, so that we can surely hit the global minima while looking for the best performance.



Let’s discuss about all these models below

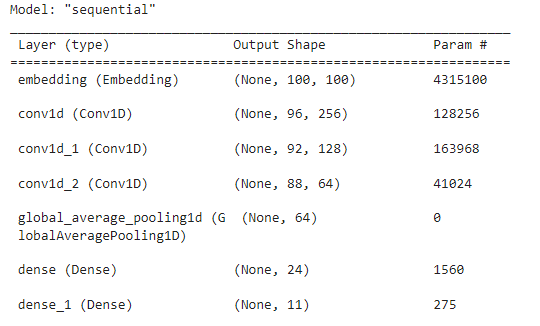
### Model 1: CNN

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most applied to analyze visual imagery. In this case, we tried to adopt this model for text classification as the text is converted to vectors. Convnets are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. The network employs a mathematical operation called Convolution. Convolution is a specialized kind of linear operation.

***Architecture***

Built a Traditional CNN model with below architecture.

Final Layer outputs being number of classes, i.e., 11 which is number of distinct groups

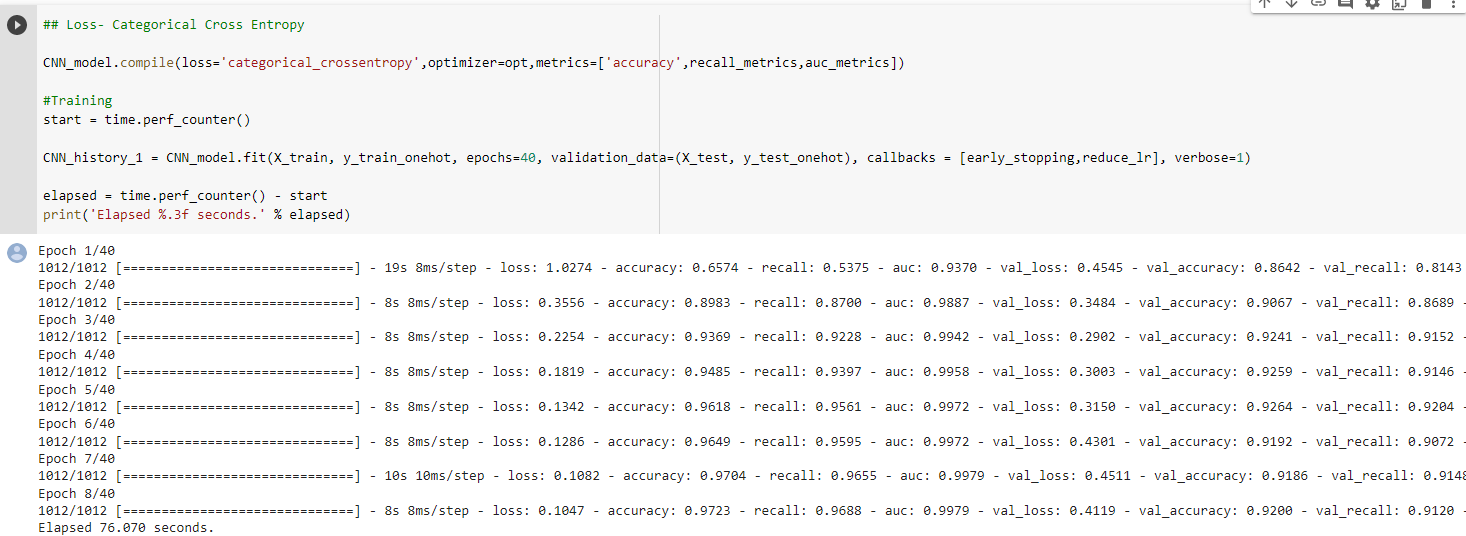


***Training and Validation***

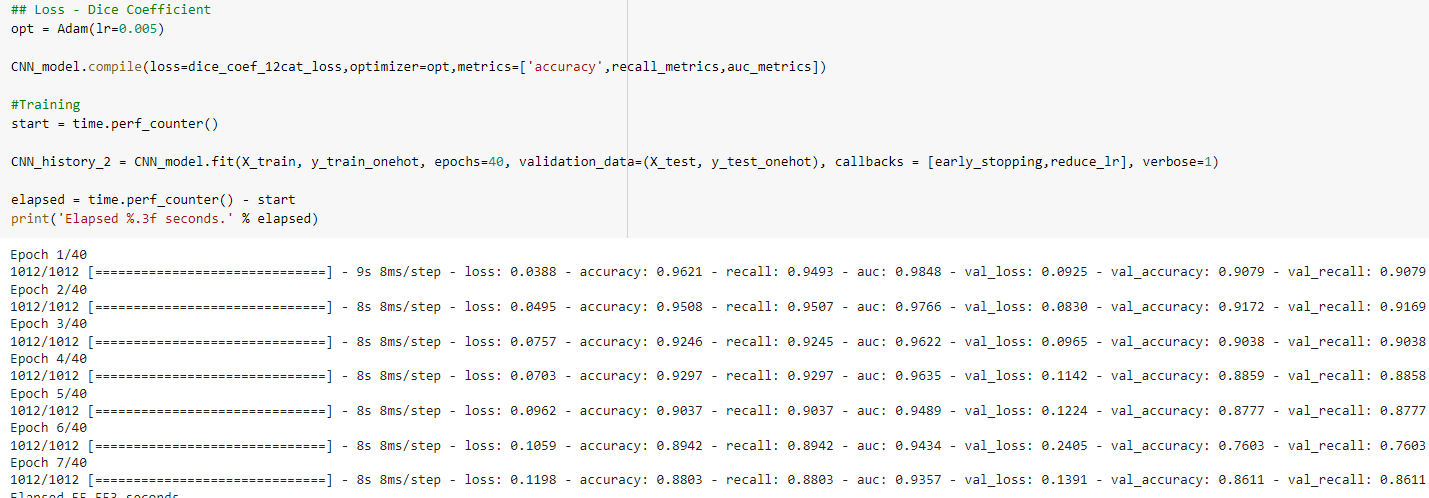
We have tried to fit the model with different loss functions and tried to see which loss function is performing better. We have tired below three loss functions

1. Categorical Cross Entropy
2. Dice loss
3. IOU loss

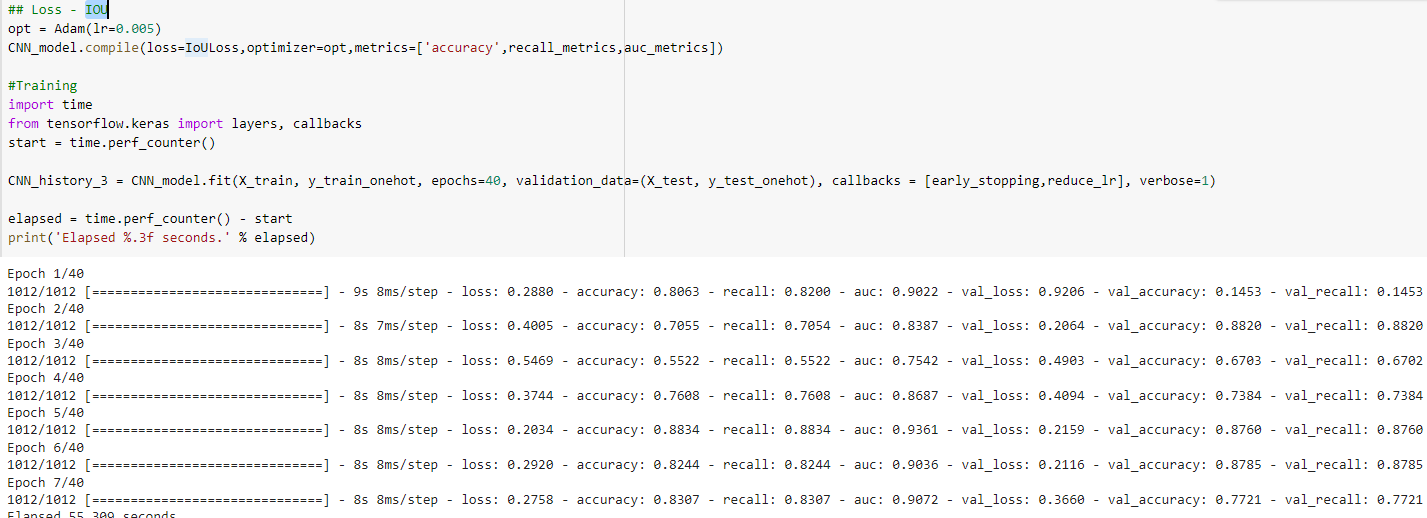
Categorical Cross Entropy



Dice Coefficient



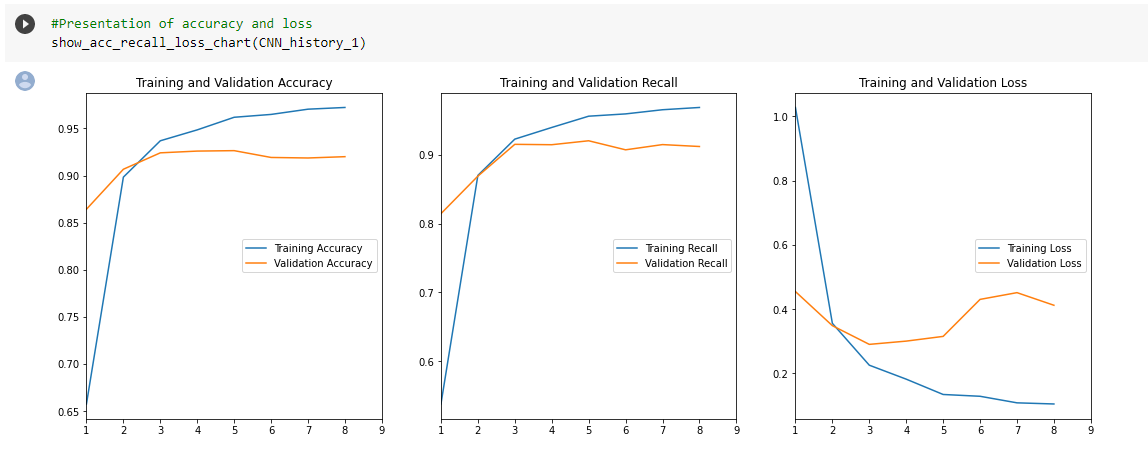
IOU



***Summary and Observations***

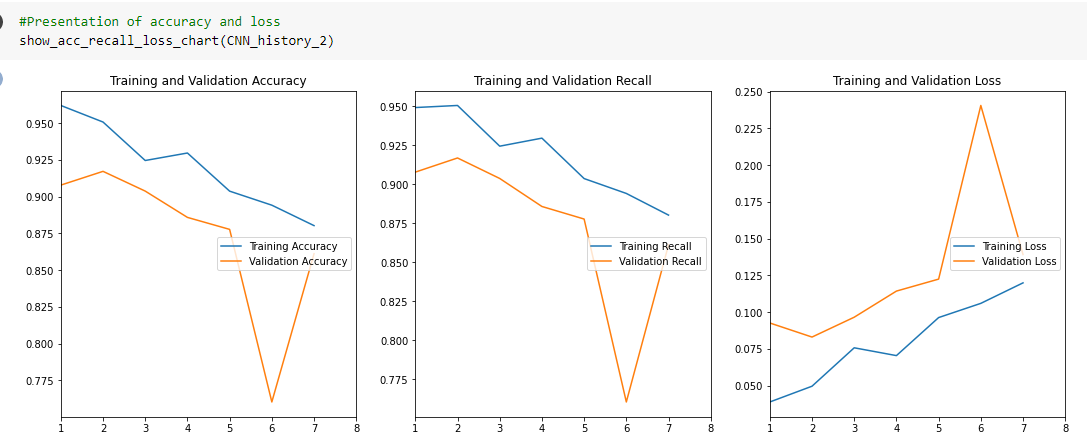
Find below summary and observations, after training defined models with different loss functions and their checking on the model’s validation accuracy.

Categorical Cross Entropy, we have seen validation accuracy increasing and Validation loss decreasing as epochs increase. The validation accuracy and recall has stagnated and not increasing much after a point.



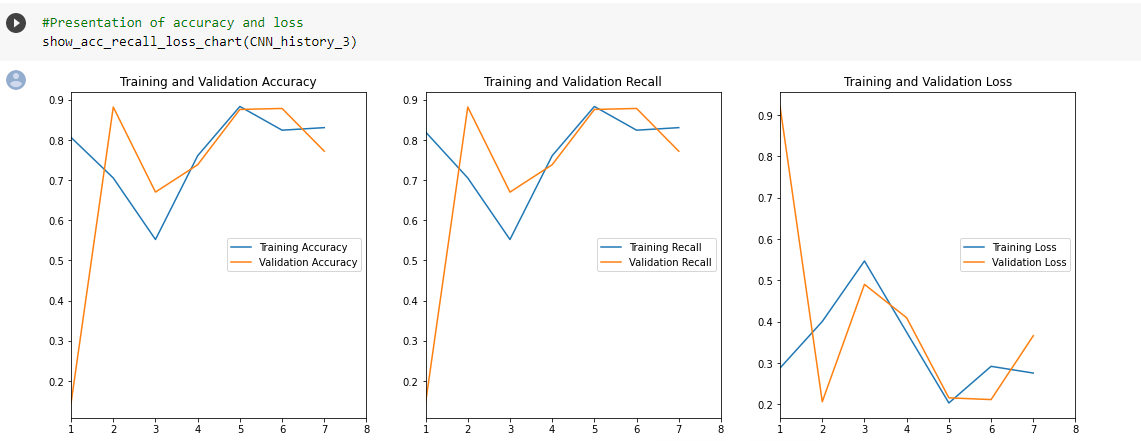
Dice Coefficient

We have seen validation accuracy decreasing and Validation loss increasing with the number of epochs. This is not what we are expecting and at this moment, Categorical cross entropy seems to be better. Let’s conclude after checking on the final Loss function as well.



IOU

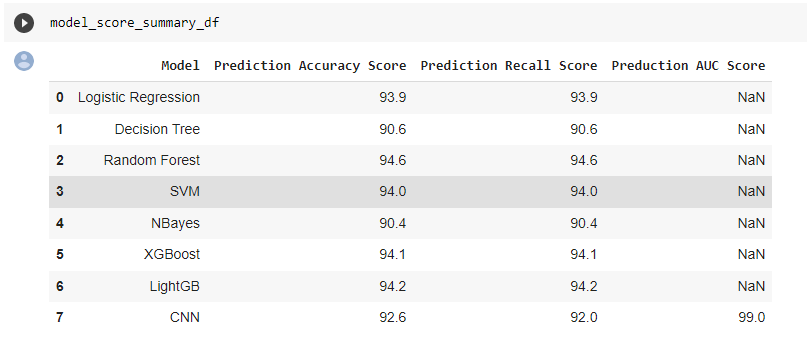
We have not seen a consistent pattern and the training accuracy and recall is not increasing beyond 90%, so we can conclude Categorical cross entropy is best of all these options for given classification problem



Also, find below comparison of all models, we have tried up to now.

Impressive, as we are getting close to 92% accuracy and recall in traditional CNN.

But it is less than some of fine-tuned traditional classifiers and ensemble models.



As there is always a room for improvement, lets try other deep learning models

### Model 2: Forward LSTM

Long short-term memory (LSTM) is an artificial neural network used in deep learning, and is a Gated Recurrent Neural Network. Unlike standard feedforward neural networks / CNN, **LSTM has feedback connections**.

The key feature is that those networks can store information that can be used for future cell processing. We can think of LSTM as an RNN with some memory pool that has two key vectors:

(1) Short-term state: keeps the output at the current time step.

(2) Long-term state: stores, reads, and rejects items meant for the long-term while passing through the network.

The forget and output gates decide whether to keep the incoming new information or throw them away. The memory of the LSTM block and the condition at the output gate produces the model decision. The output then is passed to the network again as an input making a recurrent sequence.

This will help to improve the accuracy based on the word sequence and would slightly have better predictions.

***Architecture***

Built a Sequential very simple forward LSTM model with the architecture below.

Graphical user interface, text

Description automatically generated

***Training and validation***

We have tried to fit the model with different loss functions and tried to see which loss function is performing better. We have tired below three loss functions

1. Categorical Cross Entropy
2. Dice Coefficient
3. IOU

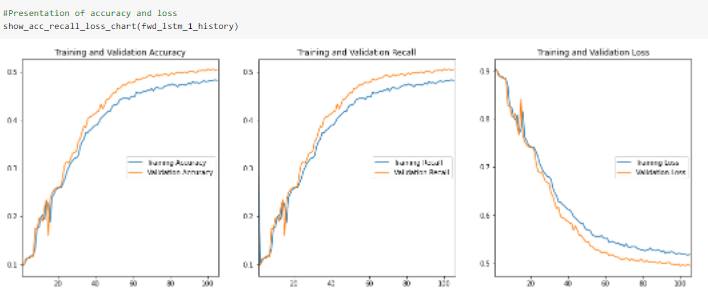
Categorical Cross Entropy

Table

Description automatically generated with medium confidence

Dice Coefficient

We have seen validation accuracy decreasing and Validation loss increasing with the number of epochs. Categorical cross entropy seems to have better results.



IOU

We have seen a consistent pattern, but the metrics are not improving beyond 50% in the validation cases.3 So, we can conclude Categorical cross entropy is best for given classification problem.

We will drop this IOU loss for validation as it is not helping much to the model performance tuning.

Graphical user interface, chart

Description automatically generated

***Summary and Observations***

Find below summary and observations, after training defined models with different loss functions and their validation accuracy.

Impressive, as we are getting close to 93% compared to traditional CNN.

Chart

Description automatically generated

Table

Description automatically generated

### Model 3: Bidirectional LSTM

Bidirectional LSTMs are an extension of LSTMs that can improve model performance on sequence classification problems. In situations where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem. In the context of this problem, this model will validate the reverse sequences of the words to classify them under a given label.

***Architecture***

Different bidirectional cross LSTM models with different activation models were implemented and checked. This model has close to 4.7Mn parameters across the different layers out of which close to 10% is the trainable parameters(400k). Built three BiLSTM models: 1) with activation functions tanh and followed by sigmoid -

2) with activation function tanh and followed by sofmax -

3) with activation function Relu and followed by sofmax -

Although, all these models gave validation accuracy as close to 94%, the Relu\_Softmax combination had the best performance

Table

Description automatically generated

***Training and Validation***

We have tried to fit the model with different loss functions and tried to see which loss function is performing better. We have tired below three loss functions

1. Categorical Cross Entropy
2. Dice Coefficient
3. IOU

Categorical Cross Entropy

Table

Description automatically generated

Dice Coefficient

Graphical user interface, table

Description automatically generated

IOU

Graphical user interface, table

Description automatically generated

***Summary and Observations***

Find below summary and observations, after training defined model with different loss functions and their validation accuracy

Categorical Cross Entropy, we have seen validation accuracy increasing and Validation loss decreasing as epochs increase.

Graphical user interface, chart

Description automatically generated

Dice Coefficient

We have seen validation accuracy decreasing and Validation loss increasing with the number of epochs.

Chart, line chart

Description automatically generated

IOU

We have not seen a consistent pattern and so we can conclude Categorical cross entropy is best of all these options for given classification problem

Chart, line chart

Description automatically generated

Also find below comparison of all models, we have tried up to now.

Improvement is shown, as we are getting to 94%. This is on par with the initial ML models that were built and trained.

Table

Description automatically generated

Relu activation followed by the Softmax activation function for the output layer is overall better in BiLSTM. As again, Softmax is the correct activation to be used for the output layer for multi classification problems.

### Model 4: BERT

BERT stands for **Bidirectional Encoder Representations from Transformers**. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks.

Built the encoder and transformed the class labels as applicable to be used in BERT model.

***Architecture***

Built the BERT model based on the pre-trained model ‘bert-case-uncased’

PRE\_TRAINED\_MODEL\_NAME = 'bert-base-uncased'.

Final Layer outputs being number of classes, i.e., 11 which is number of distinct groups

based

Graphical user interface, text

Description automatically generated

***Training and Validation***

We have tried to fit the model with different loss functions and tried to see which loss function is performing better. We have tired below two loss functions

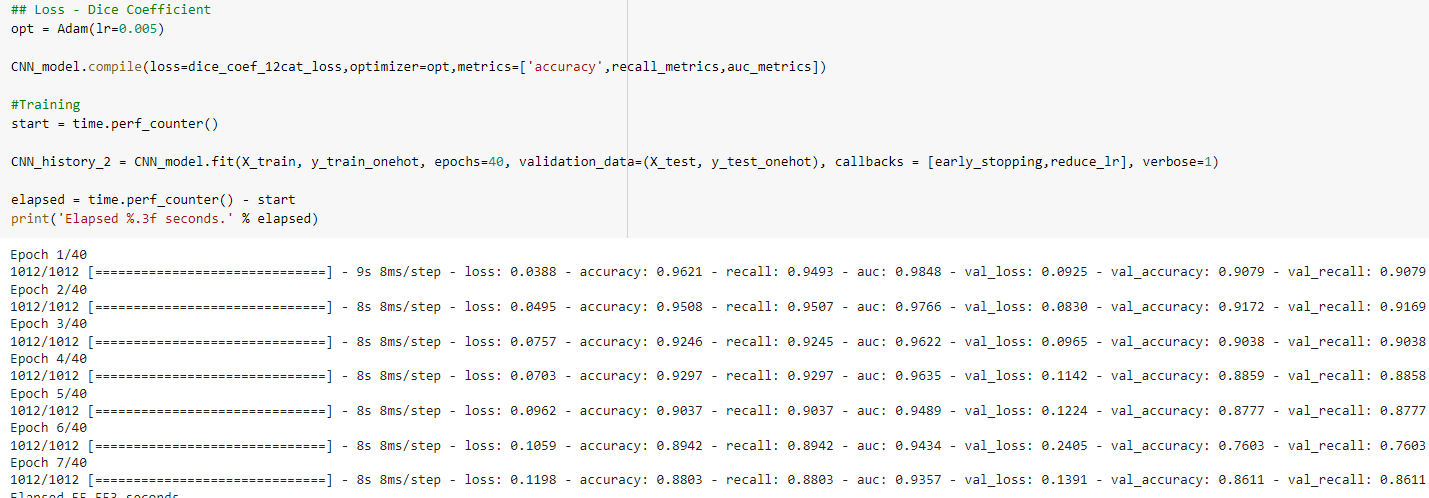
1. Categorical Cross Entropy
2. Dice Coefficient

Categorical Cross Entropy

A picture containing text

Description automatically generated

Dice Coefficient



***Summary and Observations***

Find below summary and observations, after training defined model with different loss functions and their validation accuracy

Here again, the model with Categorical Cross Entropy loss validation gave the best result.

Chart, line chart

Description automatically generated

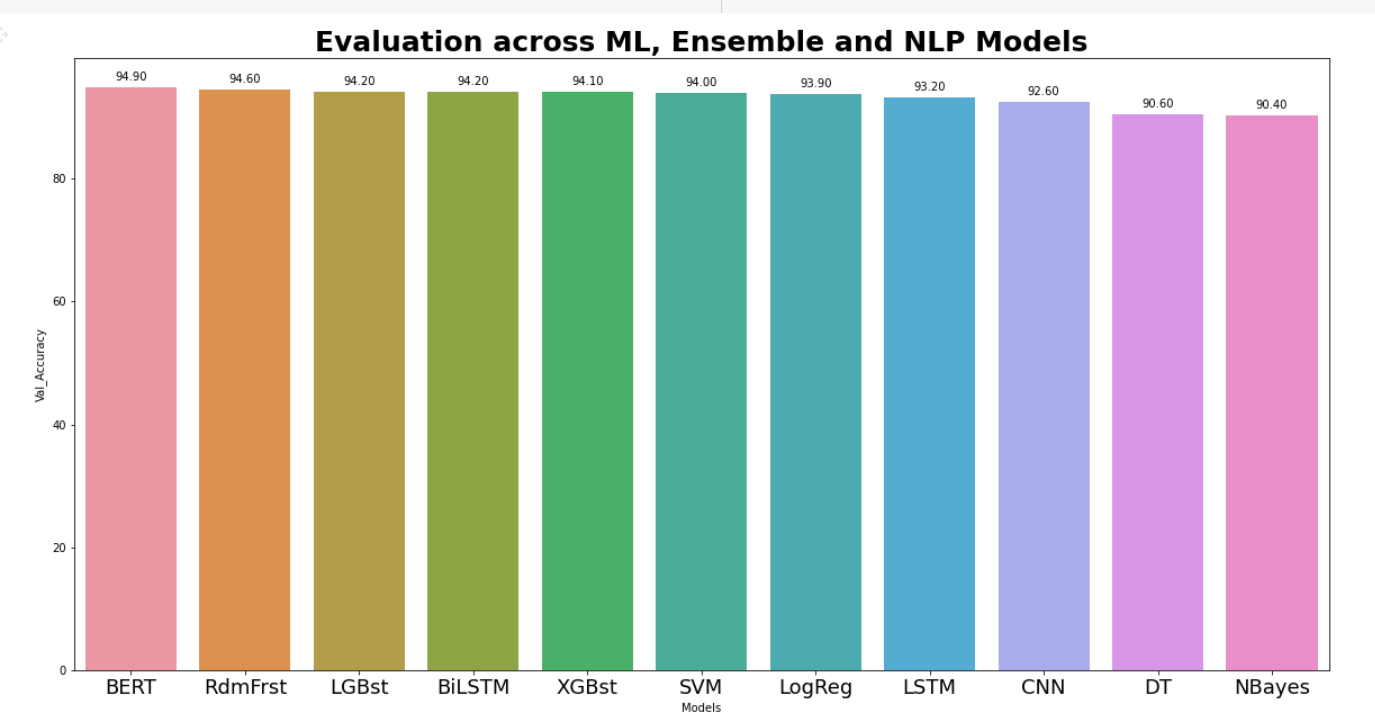
## 5. Model Summary and Overall Evaluation

Table

Description automatically generated

### 

### 



### Overall, BERT is the best model with both Accuracy and Recall metrics close to 95%.

In Deep learning models, we have seen from the performance standpoint BiLSTM> LSTM> CNN - which is quite understandable, and the metrics difference is roughly 1% across the models.

The ML models and the ensemble models have also performed equally well as this is text classification problem and pretty much with the vectorization approach, we see that there is good clustering achieved based on the EDA that was done in the initial part. Particularly, Random Forest, Lightweight Gradient Boost, and XGB models have achieved as close to Deep learning models 94%.

The Logistic Regression and SVM models as well have achieved close to 94% which states that the EDA has been done thoroughly for even the linear models to achieve well.

Apart from the EDA, the text translation and text augmentation has clearly helped all these ML models to achieve close to Deep learning models performance.

### 

***Data Enrichments vs Model Performance***

|  |  |  |
| --- | --- | --- |
| **Data/Feature Engineering** | **Model** | **Prediction Accuracy Score** |
| After Augmentation | Logistic Regression | 93.9 |
| After Augmentation | Decision Tree | 90.6 |
| After Augmentation | Random Forest | 94.6 |
| After Augmentation | SVM | 94 |
| After Augmentation | NBayes | 90.4 |
| After Augmentation | XGBoost | 94.1 |
| After Augmentation | LightGB | 94.2 |
| After Augmentation | CNN | 92.6 |
| After Augmentation | Forwd LSTM | 93.2 |
| After Augmentation | BiDirectional LSTM | 94.2 |
| After Augmentation | BERT model | 94.9 |
| After Translation | Logistic Regression |  |
| After Translation | Decision Tree |  |
| After Translation | Random Forest |  |
| After Translation | SVM |  |
| After Translation | NBayes |  |
| After Translation | XGBoost |  |
| After Translation | LightGB |  |
| After Translation | CNN |  |
| After Translation | Forwd LSTM |  |
| After Translation | BiDirectional LSTM |  |
| After Translation | BERT model |  |
| After EDA and Preprocessing | Logistic Regression |  |
| After EDA and Preprocessing | Decision Tree |  |
| After EDA and Preprocessing | Random Forest |  |
| After EDA and Preprocessing | SVM |  |
| After EDA and Preprocessing | NBayes |  |
| After EDA and Preprocessing | XGBoost |  |
| After EDA and Preprocessing | LightGB |  |
| After EDA and Preprocessing | CNN |  |
| After EDA and Preprocessing | Forwd LSTM |  |
| After EDA and Preprocessing | BiDirectional LSTM |  |
| After EDA and Preprocessing | BERT model |  |
| RAW Data | Logistic Regression |  |
| RAW Data | Decision Tree |  |
| RAW Data | Random Forest |  |
| RAW Data | SVM |  |
| RAW Data | NBayes |  |
| RAW Data | XGBoost |  |
| RAW Data | LightGB |  |
| RAW Data | CNN |  |
| RAW Data | Forwd LSTM |  |
| RAW Data | BiDirectional LSTM |  |
| RAW Data | BERT model |  |

***Implications and Recommendations***

* As per the given problem statement roughly ~25% of the cases the tickets were wrongly classified, and this further increases the manual assignment which takes additional effort of 1 FTE.
* With this BERT model as the final model, we can categorize the incoming incidents / tickets (to the help desk) to automatically get assigned to the correct class / group with 95% confidence across the cases. This translates to,
  + Reducing the wrong classifications from 25% to 5% - this is a 400% reduction in wrong assignments resulting in manual effort savings (0.8 FTE)
  + Enabling faster resolutions by the corresponding teams and eventually customer satisfaction. The value for this is sky bound.
  + The effort savings achieved (0.8 FTE mentioned above) can now be used for other productive purposes. Hard dollars saved in going in position.
* There are few more improvements possible (currently not achieved due to data limitations and incompleteness), and that would make this exercise a fully automated ticket assignment.
  + If we create a solution for the end users to automatically select the issue they face from a set of general issue statements and further gather additional information that are specific based on country, language, business process, system involved and few other parameters (like end user self learning and interactive UI - Chatbot).
  + With this approach, we are cent percent confident that we can have a very solid ticket information management that is end to end automated with high level of clarity and high level of accuracy (as close to 3 sigma confidence level - 99.7%+) to classify to the correct group, eventually helping in faster resolution and customer satisfaction.

***Limitations of the current solution***

* We have used Text augmentation techniques and Text translations which helped to boost the model performance. In real world scenarios, not all groups will have equal amount or tickets and this bias is beaten in the model. However, we cannot ignore reality. If the smaller groups do not have clear demarcation from the class group definitions or assignment reason, then there will be a lot of ambiguity from the data which will not help the model. - Lack of volume in the data to be classified
* Incompleteness in the input / ticket information - Lack of good data creates more noise
* Clarity missing in the ticket information - Lack of good data creates more noise

***Future enhancements to the solution***

* BERT Translation and BERT Augmentation to improvise the current solution. Currently, we have used NLPaug for text augmentation.
* Create a cascading model that broadly classifies big groups like Grp\_0, job fails, ERP,..., other etc.; then have a sub model or a cascade model that will further predict which is the correct sub-group.
* Built a text filter to remove all the caller ids in the incident text - it may not help to improve performance, but it will help to keep the data clean.
* Create a new set of clustering to re-define the entire classification for once and build the model as one of the reasons for 5% wrong classification is the data itself from the real world that has either a lot of noise or incomplete information or general information not helping to classify them under one group. This we noticed in our approach to view how the class clusters were.

***Closing Reflections***

Our Learnings:

Apart from the technical learnings we have had in the project, below are the other key learnings that we want to keep in mind for any project that we will do…

* Understand the problem statement in detail and create a complete perspective of what the problem is at different stakeholder levels (Support Personnel, IT stakeholder, Business stakeholder and End users). the same problem has different definitions at each of these levels.
* Analyze and understand the data in a very detailed manner and associate that with the detailed problem statement
* Define the problem at hand (more like a limitation or frame within which we must operate) we cannot boil the entire ocean to start with. So, start small but with the end goal in mind.
* Explore more on the data to see what is hidden besides the general information or text. Deep analysis and this help us to come up with creative approaches or methods to look at the problem and figure out possible solutions options.
* Based on the problem, try both the ML models/ ensemble models before jumping on to NLP or transformer kind of architecture. Sometimes, the basic ML models provide higher results with not much effort. That can be used as a solution for the high-level problem statement at hand providing a temporary solution while the deep dive final solution is being arrived at in parallel.
* Based on the input, if it is text, build the strong text preprocessing to help the models achieve better results.
* Try all different hyper parameter tuning combinations while building the model, in terms of which libraries to use, which loss functions to use, which metrics are more applicable for the given problem statement.
* Create better visualizations for the model performance and results to help understand what data is doing with the models.
* Compare and contrast on each model performance at different levels(train, validation, test) and see which ones the to choose as top model
* List down all the ideas that we get during this entire process and keep trying them.
* Finally, create a solid solution from all the best combinations at each step
* Last but not the least, DO NOT, at any point of time think, that we have achieved the best result, there is always something more to do to improve (keep that curiosity awake)
* If given a chance, next time, I would like to reclassify and redefine all the ticket data into proper groups based on the new definition and ensure that the data clarity is achieved, and noise is reduced first instead of assuming to go with the given data as it is. Also as recommended, will share these improvements to the upstream process as well to ensure data quality is improved in the first place.