Customer Support Ticket Classification

Team Mentor: Mr. Shaksham Kapoor
Team Coordinator: Snehal Tambe
Team Members: Rohan Kumar
Rohit Kumar Jha
Tayal Devanshu

Abstract

Customers contact companies via multiple channels, from social media platforms and review sites to email and live chats – any time of the day, wherever they are. With the growing number of mobile users, customers can easily access these services, adding to the volume of tickets generated. IT tickets are the generalized term used to refer to a record of work performed (or needing to be performed) by an organization to operate the company's technology environment, fix issues, and resolve user requests. These tickets can be raised through the web, mobile app, emails, calls, or even in customer care centers.

When an issue or support ticket drops into the IT helpdesk, first it needs to be processed and assigned a tag or category. These tickets are then routed to the right agent for resolution, according to the department that would be a perfect fit for the label of that ticket. Hence, the ticket needs to be correctly labeled, so that there is no waste of time and resources in routing the ticket to the right agent, only adding to the resolution time of that ticket. When this process of ticket classification is a manual one, usually what happens is that there are too many tickets to be tagged, and routed. Which delays the entire routing process as a lot of time is spent processing those queries which don't require human intervention at all and can be solved by a bot.

Ticket Classification is an essential task for every organization. However, it is one that is very mundane to allocate manual resources to. This is one of the main reasons why automation of Ticket Classification is so essential today. Also the continuous growth of the eCommerce industry (includes everything from buying clothes to food to furniture), and due to COVID this growth has increased a lot more due to the state-wise restrictions imposed by the governments.

Introduction

This case study shows how to create a model for text analysis and classification and deploy it as a web service in google colab in order to automatically classify support tickets.

Our combined team tried 3 different approaches to tackle this challenge using:

- Logistic Regression
- Random Forest
- Lightgbm

Problem Statement and Objective

Company XYZ is well-known in the space of eCommerce and has been serving its customers happily for the past 30 years. They follow their motto, “Customer is King”, seriously, and ensure that customers' issues are resolved in time. However, it is a time taking task. For each ticket, a team of SMEs read the ticket to understand the issue and assign it to the appropriate support level. This means, many times, for a simple issue, a customer might need to wait for hours, maybe days before it is resolved, which is not good. An unhappy customer could mean a loss of millions of dollars in revenue.

They have come to you with this problem, and see if you can help. What they want is a system that can automatically classify the tickets to either one of the levels (L0 or L1) based on its content.

User Stories

1. As part of management, I would be interested to see how the new ticket classification system impacts the bottom line of the business.

2. As a product owner, I would be interested to see the main areas of concern for our customer (can be a category, sub-category etc). This will help in understanding ‘what customers want’.

3. As a marketing lead, I would be interested to see the customer feedback based on which I can lead product, pricing and marketing discussions.

4. As a data science lead, my focus will be to ensure that the customer tickets are classified correctly for a timely resolution. I will iteratively improve the underwriting algorithm by using suitable means and having checks of data quality.

Expected Solution

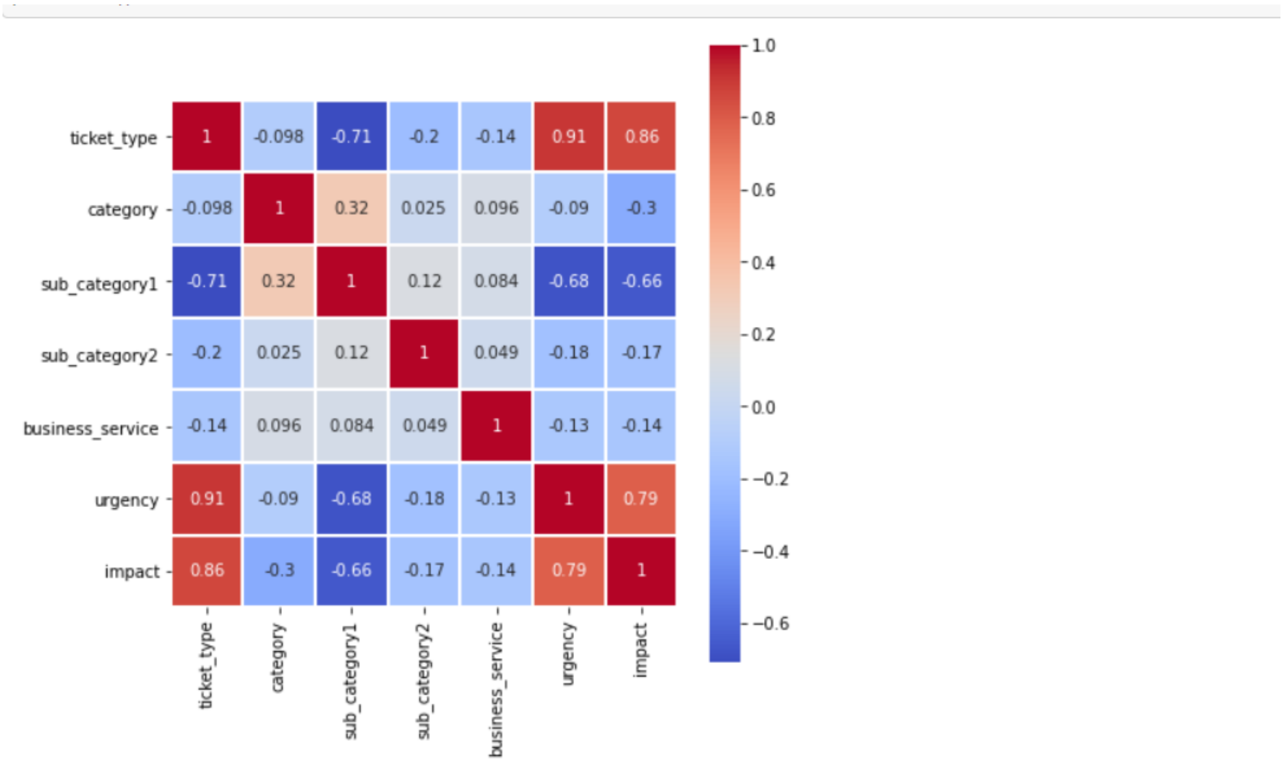
The solution will focus on developing a classification system that is capable of identifying and assigning the appropriate customer-level support to resolve the customer queries in time. A simple UI interface will be developed where a user query is given as input, and the outcome is the assigned level (L0 or L1) along with a probability score.

Work Done

What will you find inside:

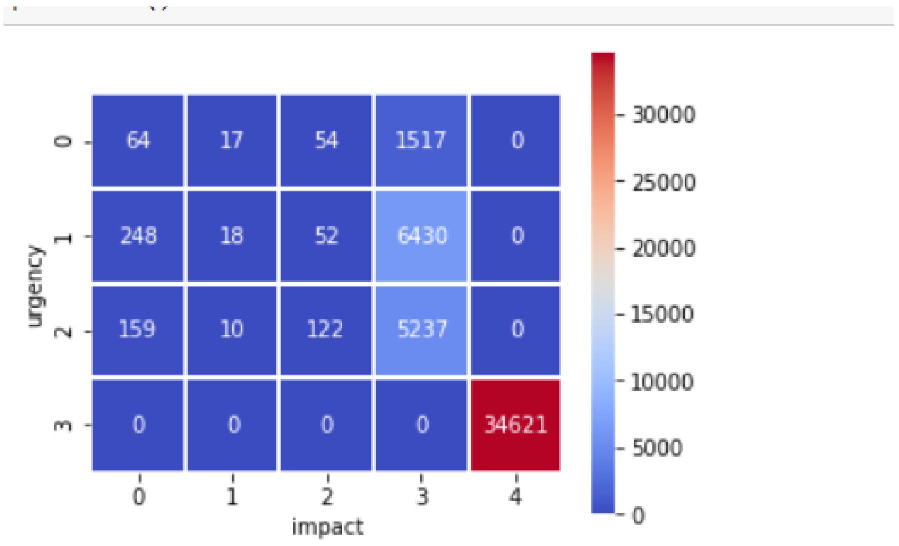
- How to clean and prepare text data and featurize it to make it valuable for machine learning scenarios
- How to strip the data from any sensitive information and also anonymize/encrypt it
- How to create a classification model using Python modules like: sklearn, nltk, matplotlib, pandas
- How to create a web UI using streamlit
- How to deploy a trained model on google colab.

Exploratory Data Analysis (EDA)

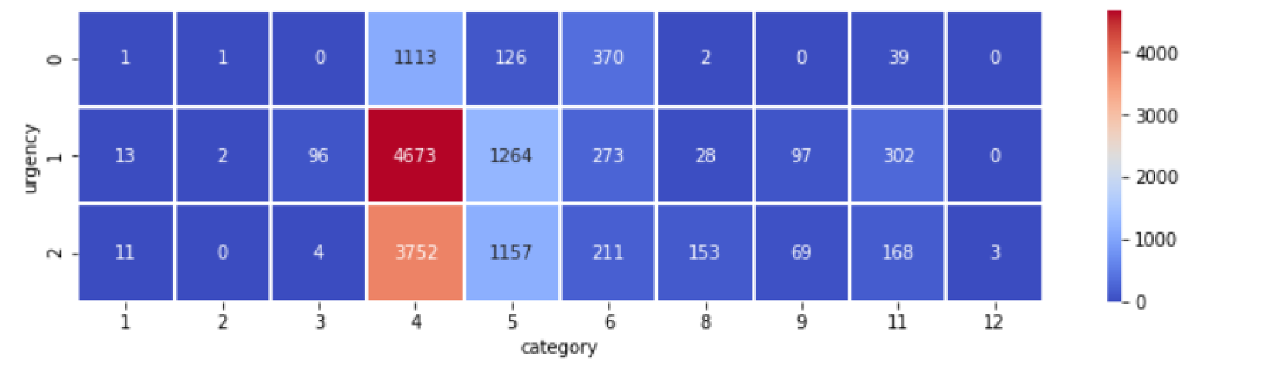


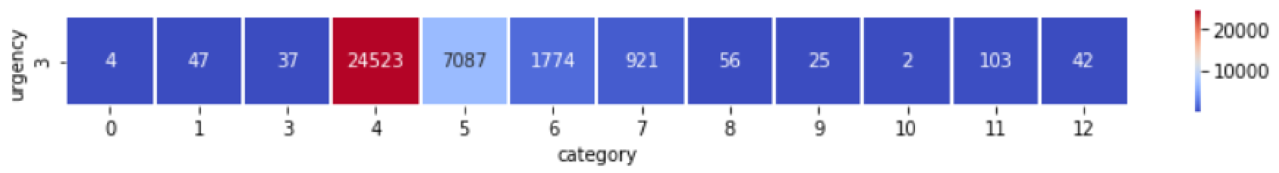
	ticket_type	category	sub_category1	sub_category2	business_service	urgency	impact
ticket_type	1	-0.098	-0.71	-0.2	-0.14	0.91	0.86
category	-0.098	1	0.32	0.025	0.096	-0.09	-0.3
sub_category1	-0.71	0.32	1	0.12	0.084	-0.68	-0.66
sub_category2	-0.2	0.025	0.12	1	0.049	-0.18	-0.17
business_service	-0.14	0.096	0.084	0.049	1	-0.13	-0.14
urgency	0.91	-0.09	-0.68	-0.18	-0.13	1	0.79
impact	0.86	-0.3	-0.66	-0.17	-0.14	0.79	1

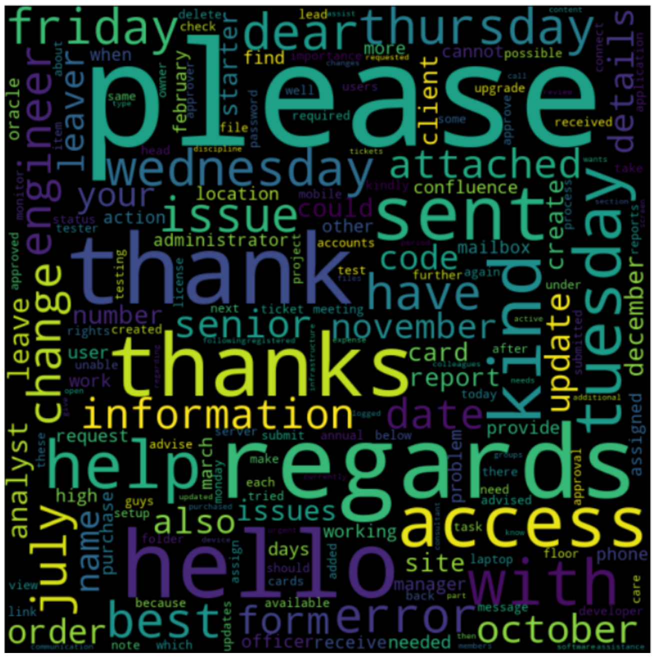
We can see that there are no duplicate values and with the help of heatmap we conclude that urgency, impact and ticket type has highest correlation amongst them.

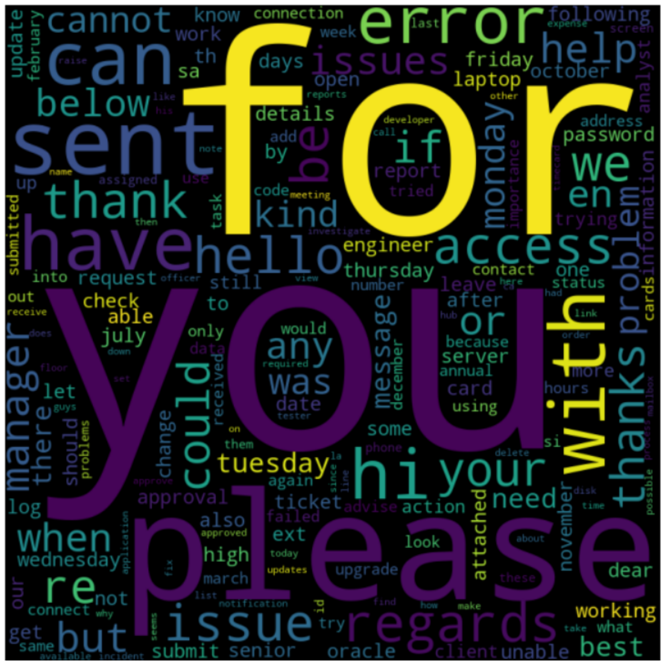


urgency \ impact	0	1	2	3	4
0	64	17	54	1517	0
1	248	18	52	6430	0
2	159	10	122	5237	0
3	0	0	0	0	34621



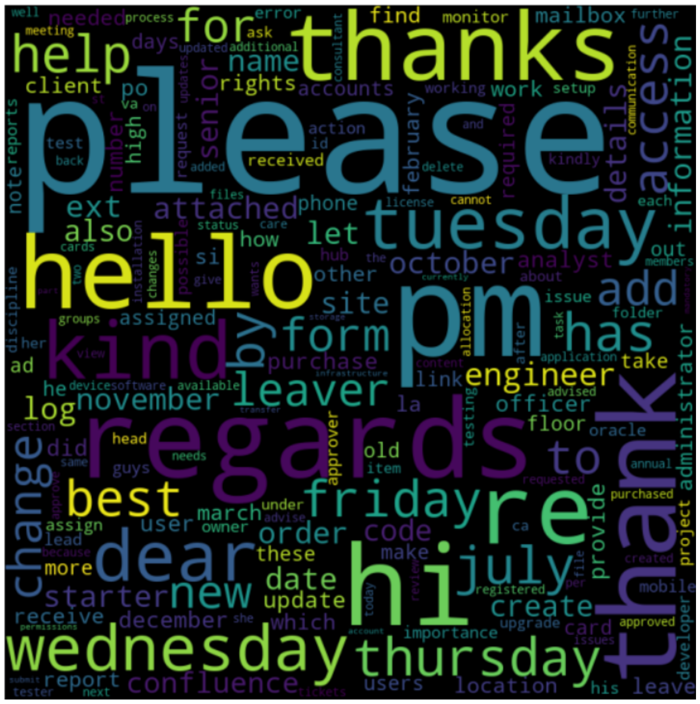






Word Cloud for entire document

Word Cloud for ticket 1



Word Cloud for ticket 0

Data Cleaning

Data cleaning is the art of extracting meaningful portions from data by eliminating unnecessary details.

It is the process of preparing raw text for NLP (Natural Language Processing) so that machines can understand human language. Text cleaning can be performed using simple Python code that eliminates stopwords, removes unicode words, and simplifies complex words to their root form.

Stop words, such as "am," "the," and "are," occur frequently in text data. Although they help us construct sentences properly, we can find the meaning even if we remove them. This means that the meaning of text can be inferred even without them. So, removing stop words from text is one of the preprocessing steps in NLP tasks. In Python, nltk, and textblob, text can be used to remove stop words from text. We have used nltk to remove the stopwords and punctuations.

We performed the two most basic text cleaning techniques on our text data:

1. Normalizing Text

2. Removing Stopwords

Feature Extraction from Texts

Machine learning algorithms do not understand textual data directly. We need to represent the text data in numerical form or vectors. To convert each textual sentence into a vector, we need to represent it as a set of features. This set of features should uniquely represent the text, though, individually, some of the features may be common across many textual sentences. Features can be classified into two different categories:

- **General features**: These features are statistical calculations and do not depend on the content of the text. Some examples of general features could be the number of tokens in the text, the number of characters in the text, and so on.

- **Specific features**: These features are dependent on the inherent meaning of the text and represent the semantics of the text. For example, the frequency of unique words in the text is a specific feature.

TF-IDF:

Term Frequency-Inverse Document Frequency (**TFIDF**) is another method of representing text data in a vector format. In TF-IDF each document is taken as a list whose length is equal to the number of unique words/tokens in all documents (corpus), but the vector here not only represents the presence and absence of a word, but also the frequency of the word—both in the current document and the whole corpus.

This technique is based on the idea that the rarely occurring words are better representatives of the document than frequently occurring words. Hence, this representation gives more weightage to the rarer or less frequent words than frequently occurring words. It does so with the following formula:

$$Tf-idf = term-frequency * inverse\ document\ frequency$$

While calculating the tf-idf vectors we had limited the maximum numbers of features to 5000.

Baseline Model

We have trained our dataset on three different models :

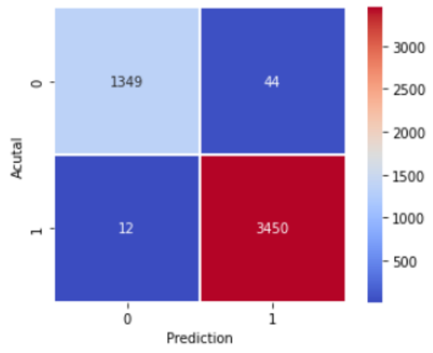
1. Logistic regression model
2. Random Forest model
3. LightGBM

We have selected one basic machine learning model : Logistic Regression. And we have selected two ensemble learning models, with Random Forest as basic ensemble model and LightGBM as advanced ensemble model.

We have only taken text data entered by the end user to calculate the features and trained the model.

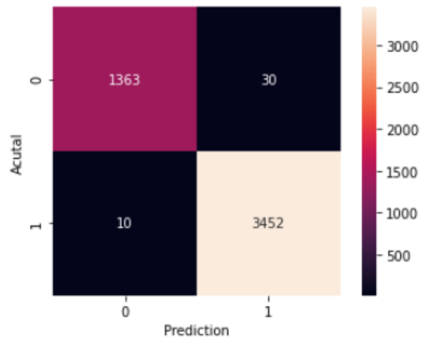
Logistic Regression:

Logistic regression, also known as logit regression or logit model, is a mathematical model used in statistics to estimate (guess) the probability of an event occurring having been given some previous data. Logistic regression works with binary data, where either the event happens (1) or the event does not happen (0).
After training our baseline model on Logistic Regression, the confusion matrix obtained is:
We got a score of 0.99031 and accuracy of 98.85 %.



Random Forest:

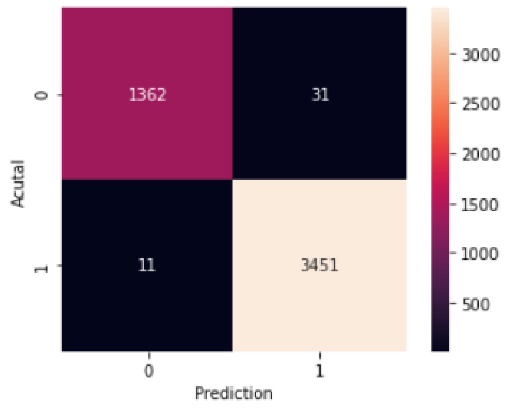
Random forests or random decision forests are an **ensemble learning method** for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.
After training our baseline model on Random Forest, the confusion matrix obtained is:
We got an accuracy of 99.18 %.



LightGBM:

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

1. Faster training speed and higher efficiency.
2. Lower memory usage.
3. Better accuracy.
4. Support of parallel, distributed, and GPU learning.
5. Capable of handling large-scale data.



After training our baseline model on LightGBM, the confusion matrix obtained is:
We got an accuracy of 99.13 %.

Vector Space / Embeddings

The vector representations discussed in the preceding section have some serious disadvantages, as discussed here:

Sparsity and large size: The sizes of frequency-based vectors depend upon the number of unique words in the corpus. This means that when the size of the corpus increases, the number of unique words increases, thereby increasing the size of the vectors in turn.

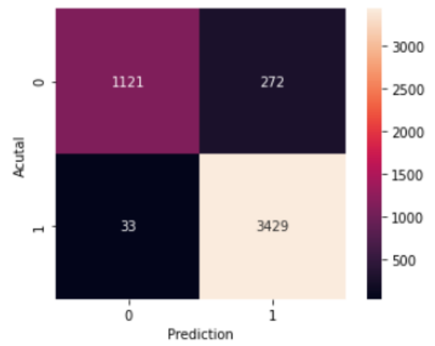
Context: None of these vector representations consider the words with respect to its context while representing it as a vector. However, the meaning of a word in any language depends upon the context it is used in. Not taking the context into account can often lead to inaccurate results.

Prediction-based word embeddings or Vector Space try to address both problems. These methods represent words with a fixed number of dimensions. Moreover, these representations are actually learned from the different contexts in which the word has been used at different places. **Learned word embeddings** is actually a collective name given to a set of language models that represent words in such a way that words with similar meanings have somewhat similar representations. There are different techniques for creating learned word embeddings, such as Word2Vec, Fasttext and GloVe.

So due to the above mentioned disadvantages of TF-IDF, we have further trained our model using 3 different Vector Space models. Those are Word2Vec, Fasttext and Glove.

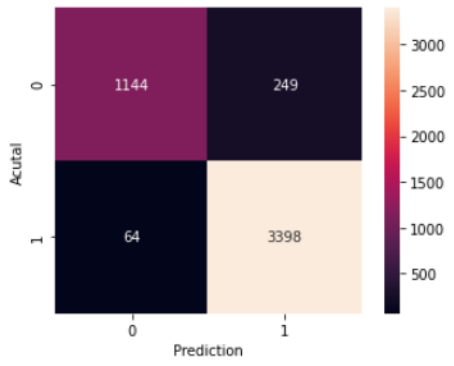
Word2Vec:

Word2vec is a two-layer neural net that processes text by “vectorizing” words. Its input is a text corpus and its output is a set of vectors: feature vectors that represent words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep neural networks can understand.
The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words.
After encoding the text data using Word2Vec and training it on Logistic regression, the confusion matrix obtained is:
And we got an accuracy of 93.72% on our model.



FastText:

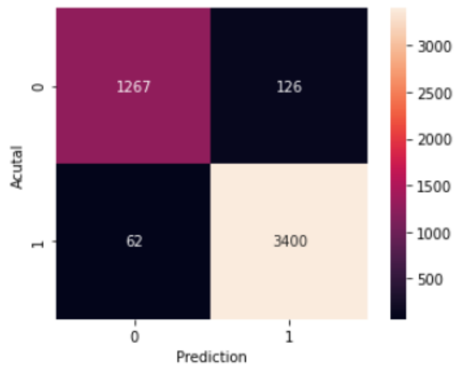
FastText differs in the sense that word vectors a.k.a word2vec treats every single word as the smallest unit whose vector representation is to be found but FastText assumes a word to be formed by a n-grams of character, for example, sunny is composed of [sun, sunn,sunny],[sunny,unny,nny] etc, where n could range from 1 to the length of the word.



After encoding the text data using FastText and training it on Random Forest, the confusion matrix obtained is:
And we got an accuracy of 93.55% on our model.

Glove:

Glove is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.



After encoding the text data using Glove and training it on LightGBM, the confusion matrix obtained is:
And we got an accuracy of 96.13% on our model.

Hyperparameter Tuning

A hypermeter is a parameter whose value is used to control the learning process. In machine learning. Hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. Hyperparameter optimization finds a tuple of hypermeters that yields an optimal model which minimizes a predefined loss function on given independent variables. There are two methods to achieve this goal.

1.GridSearchCV : It implements the most obvious way of finding an optimal value for anything. It simply tries all the possible values (that you pass) one at a time and returns which one yielded the best model results, based on the scoring that you want.

2.RandomizedSearchCV : RandomSearchCV has the same purpose as GridSearchCV: they both were designed to find the best parameters to improve your model. However, here not all parameters are tested.

Rather, the search is randomized, and all the other parameters are held constant while the parameters we are testing are variable.

Model	Parameters	Range
Logistic Regression	Penalty	{ ‘l2’ }
	C	{ 0.1, 1, 10, 100, 1000 }
Random Forest	N_estimators	{ 10, 20, 30,…, 90, 100}
	Max_depth	{ None, 3, 5, 10 }
	Criterion	{ ‘gini’, ‘ entropy’ }
LightGBM	N_estimators	{ 10, 20, 30,…, 90, 100}
	Max_depth	{ None, 3, 5, 10 }
	Criterion	{ ‘gini’, ‘ entropy’ }

Comparison

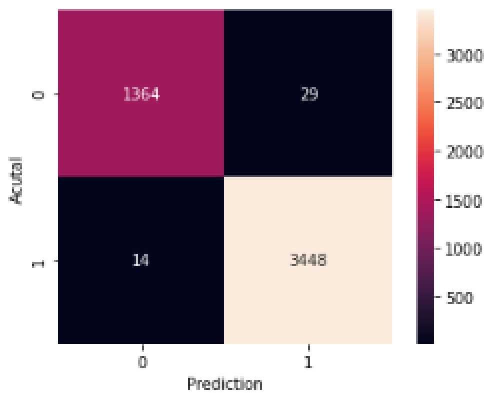
Model		Accuracy	Precision	Recall	F1-Score
Logistic Regression(baseline)	0	98.85	0.99	0.97	0.98
	1		0.99	1	0.99
Random Forest(baseline)	0	99.18	0.99	0.98	0.99
	1		0.99	1	0.99
LightGBM(baseline)	0	99.13	0.99	0.98	0.98
	1		0.99	1	0.99
Logistic Regression(word2vec)	0	95.24	0.92	0.92	0.92
	1		0.97	0.97	0.97
Random Forest(fasttext)	0	93.39	0.96	0.8	0.87
	1		0.93	0.99	0.96
LightGBM(glove)	0	96.17	0.92	0.95	0.93
	1		0.98	0.97	0.97
Logistic Regression(final)	0	58.38	0.28	0.29	0.28
	1		0.7	0.71	0.7
Random Forest(final)	0	93.92	0.95	0.83	0.89
	1		0.93	0.98	0.96
LightGBM(final)	0	95.04	0.89	0.94	0.92
	1		0.98	0.95	0.96

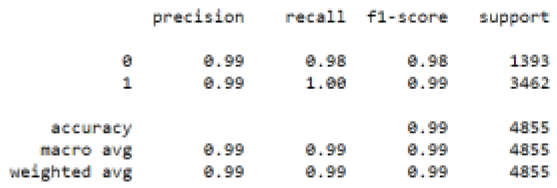
Final Model

In this project, after judging the performance of baseline models, we decided to hyper tune the Random Forest Classifier model. So, starting with making a grid of three of the hyperparameters of random forest: N_estimators, Criterion and Max_depth, with some range. We used RandomizedSearchCV with 10 iterations each for cross-validation of 3. And the best parameters came out to be, n_estimators = 80, max_depth = None and criterion = entropy, with the training score of 99.1%.

There are 4855 records in our testing dataset. So, evaluation of our final model on the testing set is as follows.

Confusion Matrix Classification Report



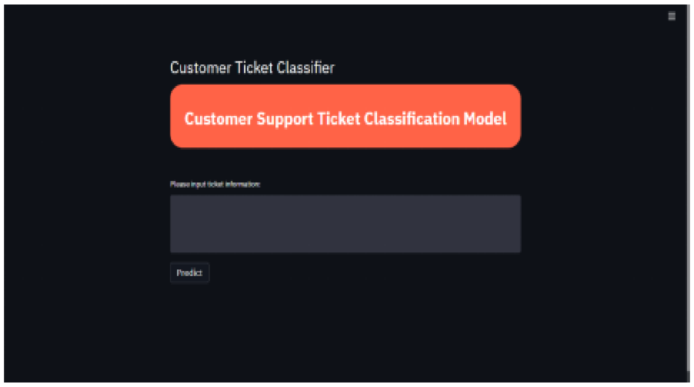

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1393
           1       0.99      1.00      0.99      3462

    accuracy                           0.99      4855
   macro avg       0.99      0.99      0.99      4855
weighted avg       0.99      0.99      0.99      4855
```

Stream lit

Streamlit is an open-source python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes, you can build and deploy powerful data apps.

In this project, we have created the application user interface using the streamlit module of python. So, there are primarily two functions in this application, one is to preprocess the ticket information input by user and convert it into the tf-idf vector. And the other is to predict the maturity of the ticket.

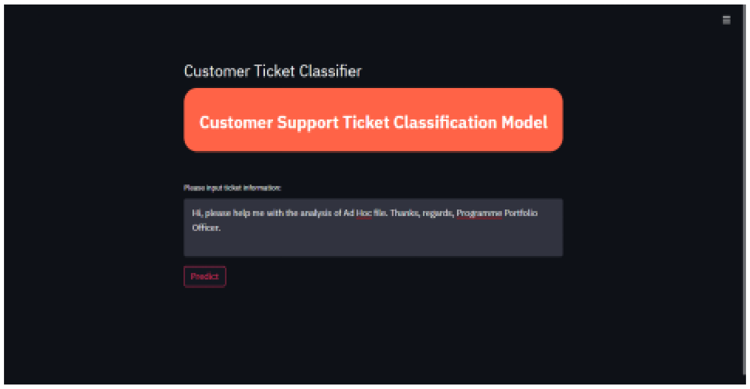
APP UI:


Customer Ticket Classifier

Customer Support Ticket Classification Model

Please input ticket information:

Predict

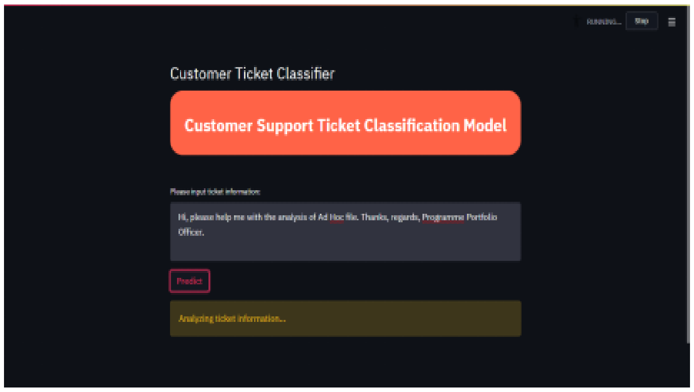

Customer Ticket Classifier

Customer Support Ticket Classification Model

Please input ticket information:

Hi, please help me with the analysis of Ad Hoc file. Thanks, regards, Programme Portfolio Officer.

Predict


Customer Ticket Classifier

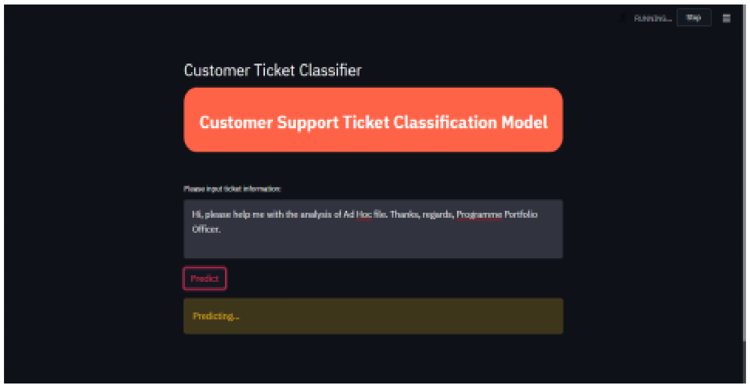
Customer Support Ticket Classification Model

Please input ticket information:

Hi, please help me with the analysis of Ad Hoc file. Thanks, regards, Programme Portfolio Officer.

Predict

Analyzing ticket information...


Customer Ticket Classifier

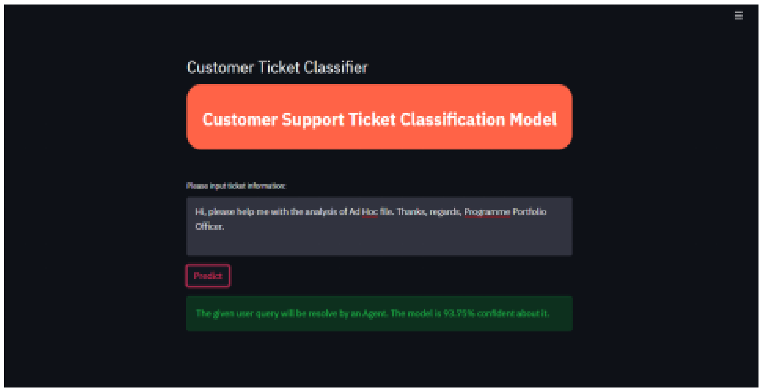
Customer Support Ticket Classification Model

Please input ticket information:

Hi, please help me with the analysis of Ad Hoc file. Thanks, regards, Programme Portfolio Officer.

Predict

Predicting...


Customer Ticket Classifier

Customer Support Ticket Classification Model

Please input ticket information:

Hi, please help me with the analysis of Ad Hoc file. Thanks, regards, Programme Portfolio Officer.

Predict

The given user query will be resolve by an Agent. The model is 93.75% confident about it.

Testing

Ticket Information	Category	Confidence
Hi, please help me with the analysis of Ad Hoc file. Thanks, regards, Programme Portfolio Officer.	1	93.75%
Dear, kindly ask the admin to change the ownership of my outlook account. Thanks 9 November 2020	1	85%
13 March 2021 Hello, please provide me with access to the main application of the project. Thank you Leader	1	98.75%
Hello, please set up new phone numbers in the engineering area. Cheers, Application Engineer	1	100%
Hello, What happened to my computer? Unable to login anymore into the server. Please see to it. Thank you, Ramesh Kumar	1	80%
Hola, facing problems in installing software for ticket classification. Direct me to Senior Engineer. Adios	0	52.5%
Windows upgrade error: I am getting the following error.	0	57.5%
Please send the manager to my desk.	0	91.25%
Analyst is facing a problem in accessing the October update. Kindly try to help her. Regards, Boss	0	61.25%
I have sent you an error log of the manager. Need it solved by Monday.	0	92.5%

Checks

Text	Output
‘ ’ (blank space)	Warning
‘ ./’;.,? ’ (punctuations only)	
‘122345’ (numbers only)	
‘ one two three four’ (less than five character)	
‘ and his her hasn’t didn’t you’ (contains stop words only but no information regarding ticket)	

Warnings = {'No information has been written! Kindly write your query again.',

'You have written punctuations only. Kindly write a proper query again.',

'You have written numbers only. Kindly write a proper query again.',

'Ticket information provided is too low. Kindly write at least five words in the query.',

'The information you have entered does not contain any appropriate knowledge. Kindly reframe your query.'}

Technology Stack

Element	Description
Pandas	Python library used for data manipulation & analysis.
Numpy	Used to perform a wide variety of mathematical operations on arrays.
Matplotlib	used for data visualization and exploratory data analysis.
Seaborn	used for data visualization and exploratory data analysis.
Wordcloud	It is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.
NLTK	Python package used for Natural Language Processing (NLP).
Sklearn	Used for model building and evaluation using various metrics.
Pickle	Used to dump/load a model or any python object.
Streamlit	Used to create and share beautiful, custom web apps for machine learning and data science.
Embedding as service	Used for vector space models.

Conclusion:

This problem statement clarifies and throws light upon the fact that IT ticket routing/tagging and any such similar task can be easily automated through NLP and Machine Learning integration within our systems or on Cloud with very high precision and accuracy.
The best classification model for each tier in our case was LightGBM which is a gradient-based model and does well to handle a class imbalance in the data.
Ticket classification is an essential part for Ticket Routing and here are the key advantages that will largely help in implementing a more efficient Customer Care Service:

- It will save hours of manpower, especially for large B2C organizations as they have a huge volume of tickets generated each day.
- Will largely help in taking data-backed decisions for Resource Allocation to optimize customer care.
- Will significantly reduce the overhead spent on routing tickets to the wrong departments.
- It can help detect patterns and seasonal trends and hence, estimate the load beforehand, or identify a larger underlying issue.

Future Scope:

- We can experiment with advanced vector space models like ELMO, BERT. These advanced models resolve the limitations present in existing techniques like word2vec.
- We can try Bayesian techniques for hyper-parameter tuning.
- We can experiment with deep learning models. Since we have text data, experimenting with RNN (recurrent neural networks) might be more useful.
- We can extend the functionality by automating the process of assigning the category and subcategory for a given query.

References

Embedding as service:

https://github.com/amansrivastava17/embedding-as-service

Streamlit:

Introduction To Streamlit. What is Streamlit? Streamlit Tutorial Pt.1

Deploy Machine Learning Models Using StreamLit Library- Data Science

https://docs.streamlit.io/en/stable/api.html