**AIML CAPSTONE PROJECT**

**INTERIM REPORT – AUTOMATIC TICKET ASSIGNMENT**

**AIML JUN GROUP 1A-NLP**

**Supervised by: Divya Teresa**

**Date :17 May 2020**

Contents

[Problem Statement 3](#_Toc40627010)

[Support Process 4](#_Toc40627011)

[Project Description 5](#_Toc40627012)

[Data Set 5](#_Toc40627013)

[Classification Models 5](#_Toc40627014)

[Milestone 1: 5](#_Toc40627015)

[Explore the data files -EDA 6](#_Toc40627016)

[Observations 6](#_Toc40627017)

[Data Sampling 7](#_Toc40627018)

[Raw Word Count before Visualization 8](#_Toc40627019)

[Visualization 8](#_Toc40627020)

[Raw Data Visualization 8](#_Toc40627021)

[Data Distribution before clean-up 10](#_Toc40627022)

[Data Cleanup 10](#_Toc40627023)

[Benefits of removing StopWords: 10](#_Toc40627024)

[TokeniZation 11](#_Toc40627025)

[Lemmatization 11](#_Toc40627026)

[Word Count after Data Pre-Processing 12](#_Toc40627027)

[Word Cloud of Cleaned & Class Balanced Corpus Data after dropping poorly represented classes 13](#_Toc40627028)

[LABEL ENCODING 13](#_Toc40627029)

[Development of ML Models 15](#_Toc40627030)

[Logistic Regression 15](#_Toc40627031)

[Naïve Bayes 15](#_Toc40627032)

[Linear SVM 16](#_Toc40627033)

[Decision tree 16](#_Toc40627034)

[Ensemble Technique applied 17](#_Toc40627035)

[Random Forests 17](#_Toc40627036)

[Bagging 17](#_Toc40627037)

[Compare the model Scores 17](#_Toc40627038)

# Problem Statement

INTRODUCTION

Due to the rise of usage of virtual systems, support ticket systems have come into prominence. Addressing the issue tickets to appropriate person or unit in the support team has critical importance in order to provide improved end user satisfaction while ensuring better allotment of support recourses.

The assignment of help ticket to appropriate group is still manually performed. Especially at large organizations, the manual assignment is not applicable sufficiently. It is time consuming and requires human efforts. There may be mistakes due to human errors. Also, resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response time which result in end user dissatisfaction.

Tickets flow in from Various sources, End users, monitoring systems, Alerts etc. End users do not want to fill long ticket forms which are needed to identify the issue. In this project, an extension to ITS for auto-assigning issue ticket to the relevant person or unit in support team is proposed. Here the approach is to build a classifier that can classify incident to the right functional group using machine learning techniques, NLP and related algorithms which proven performance in text processing are used.

## Support Process

In the support process, incoming incidents are analysed and assessed by organization’s support teams to fulfil the request.

* All tickets are assigned to the L1 /L2 teams who do the initial analysis and go through the Standard operating procedure to assign the tickets to relevant L3 teams or experts
* Few incidents are directly assigned to L3 teams for resolution by users and monitoring systems.
* The L1 and L2 teams need to assign tickets to functional Groups after applying the SOP’s provided to them
* Once the incident is assigned to the functional group , they decide if some Vendor product support or SME help is obtained as per requirement to resolve the issue finally

#### Statistics related to Issue resolution

* Approximately ~54% of the incidents are resolved by L1 / L2 teams
* Around ~56% of incidents are resolved by Functional / L3 teams. In case if vendor support is needed, they will reach out for their support towards incident closure.
* L1 / L2 teams review SOP for a minimum of 25-30% of the tickets before escalation to L3. 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.
* Approximately 25% of the incidents were assigned to wrong functional groups increasing the resolution time for end users

## Project Description

In this project, the goal is to build a classifier that can classify the tickets by analysing text. In this project, the approach is to build a classifier that can classify incident to the right functional group using NLP and related algorithms which proven performance in text processing.

## Data Set

Details about the data and dataset files are given in below link

<https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ>

* The above Data set is a dump from the ticketing system of a company which consists of 8500 records with 4 columns
* Each record has a short description, Description, Caller, Assignment Group. The short description mainly provides a clue on which is the application which has an issue. The Description column provides the next level of details on the exact issue, logs , users, frequency etc. The Caller column provides details of the user name as recorded in the organization systems. Assignment Group provides the details of which Group the issue has been assigned to. Each Group is an indicative of a specific functional department or sub department to which it is mapped
* Prima facie the data seems to have a requirement to be cleaned as there are several cases where there is irrelevant information such as email id in the description column. Ticket no in the Short Description column
* The initial data pre-processing includes cleaning of data (removing duplicates, removing empty rows, removing stop words etc.)

## Classification Model

The Classification model implemented for this Project is based on the Functional group classification, so that each ticket is sent to the best person/group equipped to resolve the issue. A functional group classifier can also be useful for organizations that have more than one product and different support teams responsible for each one. In this instance, you’d use a classification model to categorize tickets by topic issues that fall under each product/Functional Group (e.g. Login issues, Password reset etc). It is expected that if there is a new description and Description that comes in then the model should be able to automatically classify to reduce any manual intervention.

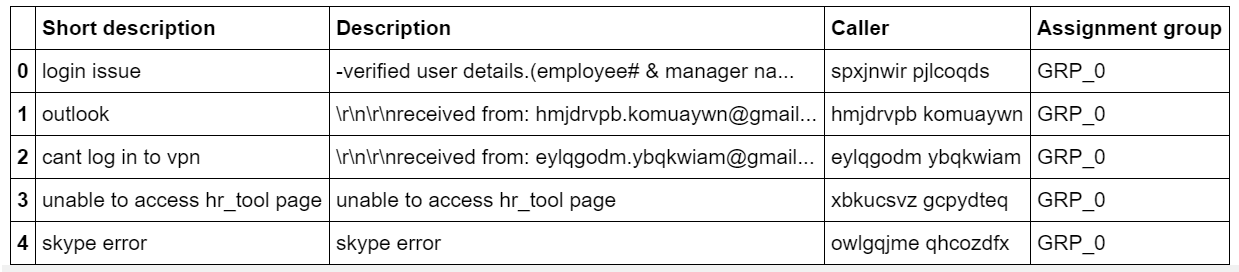
# Milestone 1:

Pre-Processing, Data Visualisation and EDA Overview

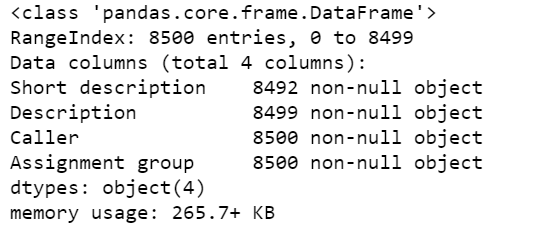
* Exploring the given Data files
* Understanding the structure of data
* Missing points in data
* Finding inconsistencies in the data
* Visualizing different patterns
* Visualizing different text features
* Dealing with missing values

## Explore the data files -EDA

1. It was found that the Given data consists of 8500 records with 4 Features as shown below



1. The concise summary of the data provided looks as below



1. The missing Values were found using the df.isna() of Pandas. The list of missing values is as below

- Short description 8

- Description 1

- Caller 0

- Assignment group 0

- dtype: int64

As you can see above 8 records for the short Description and 1 record for Description are missing values

### Observations

1. Objects containing unique values using the df.Value\_Counts( ) it was found that this data has 74 unique Labels .
2. Highly imbalanced data set and 46% of the data set is represented by a single class GRP\_0. Out of the 74 Classes 30 classes have less than 15 data points of which 19 classes have less than 5 data points
3. The key to building a good machine learning model is the data it is trained on. Therefore, it is imperative that the training data be clean and balanced.

### Data Sampling

A widely adopted technique for dealing with highly unbalanced datasets is called resampling.

* 1. Resampling is done after the data is split into training, test and validation sets. Resampling is done only on the training set or the performance measures could get skewed. Resampling can be of two types: Over-sampling and Under-sampling. Which is the approach adopted in this case
  2. Under sampling involves removing samples from the majority class and over-sampling involves adding more examples from the minority class . The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfitting. In under-sampling, the simplest technique involves removing random records from the majority class, which can cause loss of information.

DATA SAMPLING

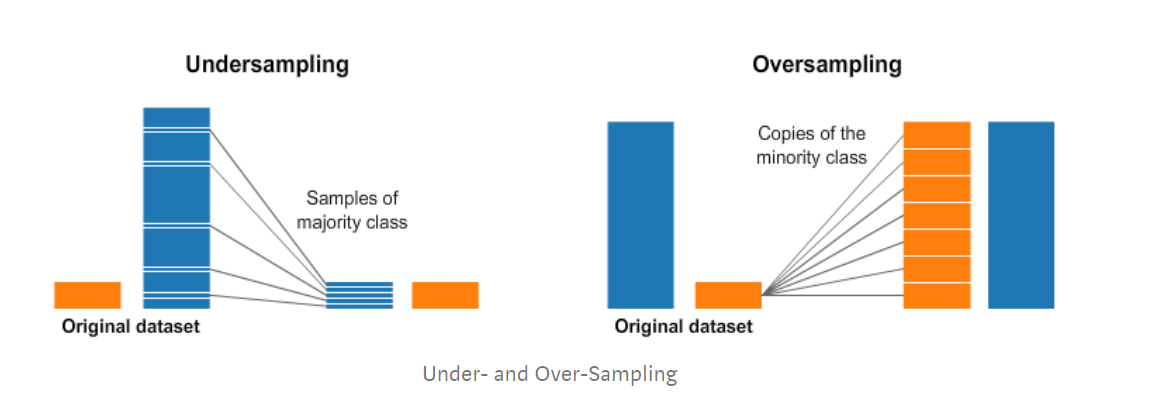


Figure 1

In order to further proceed and simplify the analysis the Column “Short Description” has been changed to “Title” and our data looks like below after this

First 4 Rows of the Data Set after Column Change

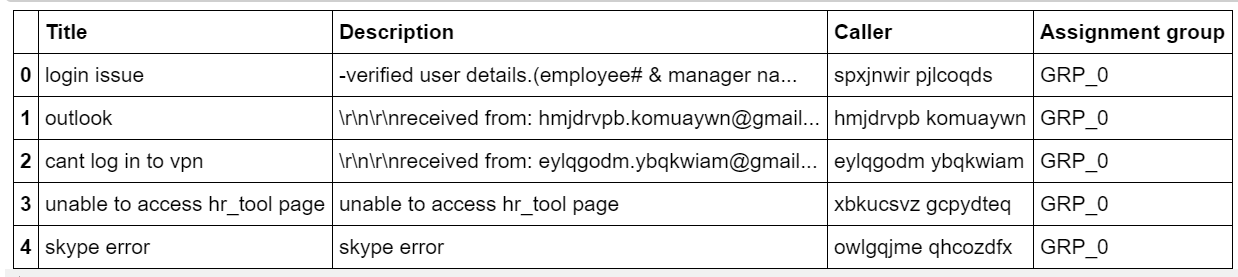


Figure 2

* Pre-processing of text is done, wherein the text is to be cleaned up to bring it to a required format for the information extraction models. This includes normalizing different tenses of words, normalizing synonyms, spell correction etc.
* Mails are Segregated to identify if a block of text is header or signature, email address, user name or the body of the mail
* Spell correction and normalization of abbreviations are done using regular expressions.
* This processed text then will to be tokenized, which is to split the raw text to a list of words, using popular open source libraries like nltk and spaCy.

### Raw Word Count before Visualization

Let us look at the Raw word count across before pre-processing

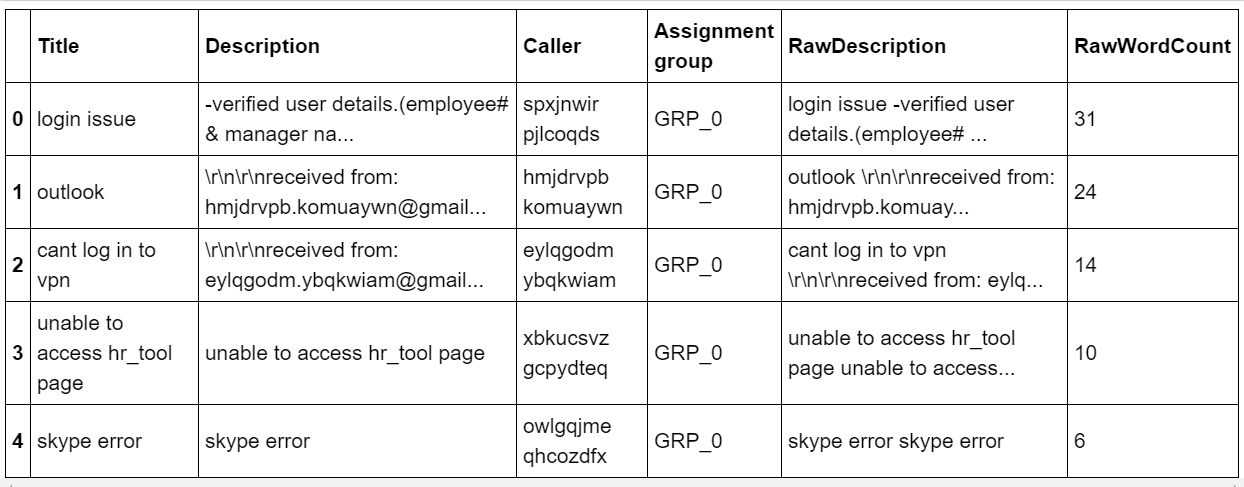


Figure 3

As we can see above are the Raw word counts before pre-processing

The total Word count before clean-up is as below

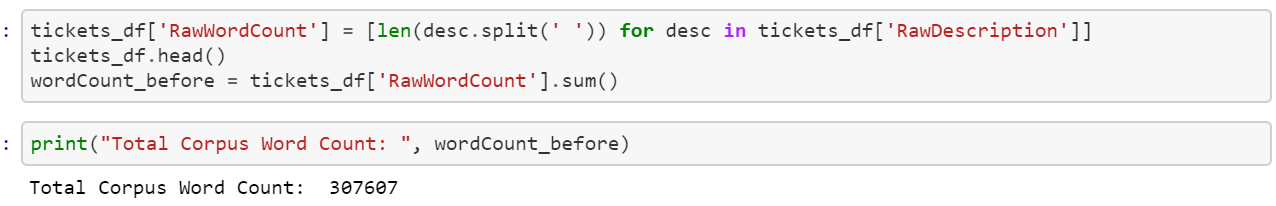


Figure 4

## Visualization

A TAG Cloud or Word Cloud is used to represent the Frequency of each word and hence we are checking how the Word cloud for each of the column looks before the data clean up

### Raw Data Visualization

Word cloud for Title column

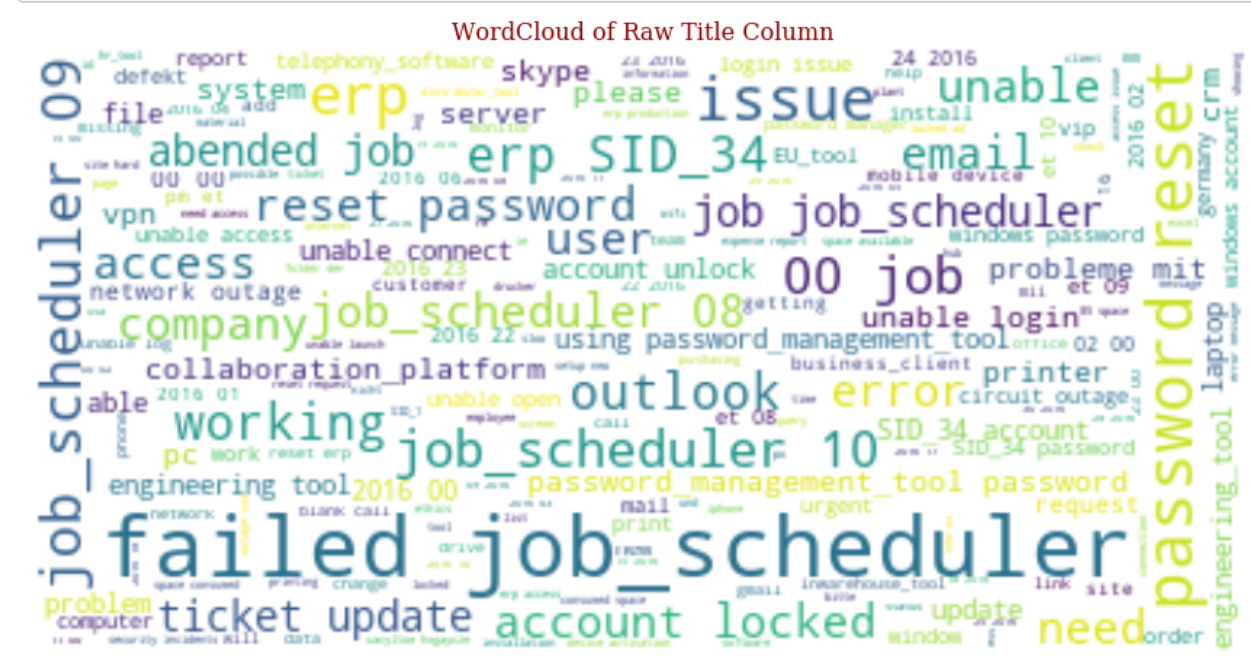


Figure 5

Word Cloud for “Description” Column

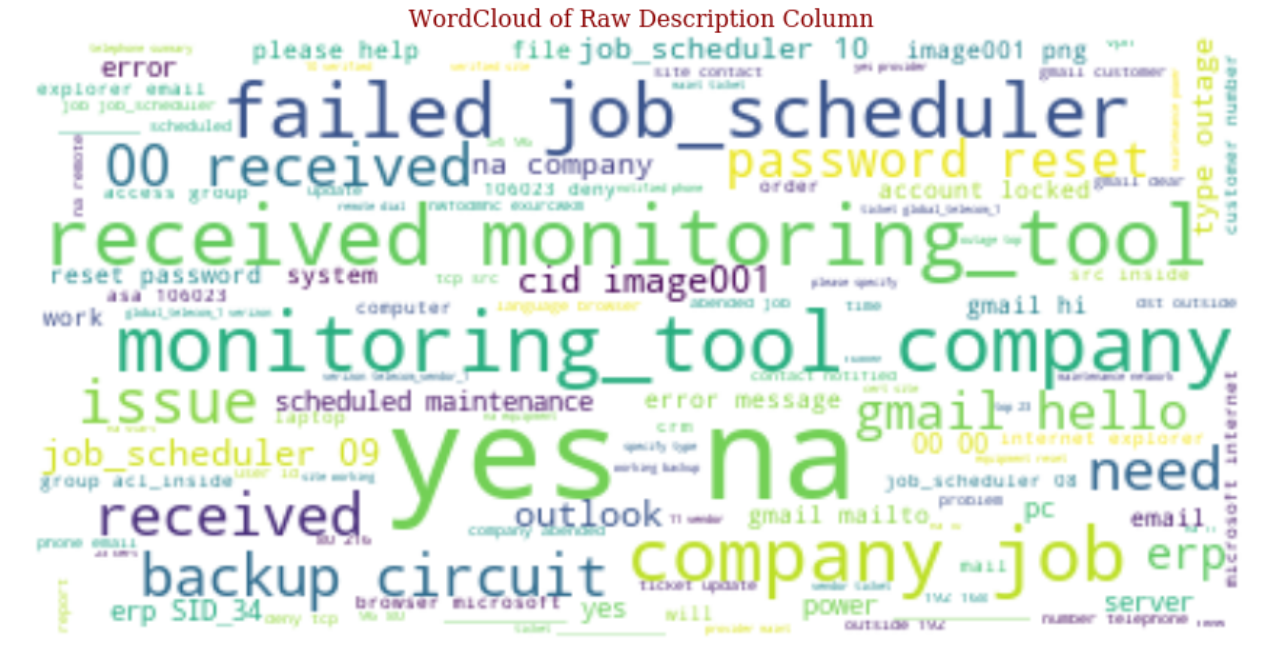


Figure 6

Word Cloud of Raw Data for the corpus - "FullDescription"

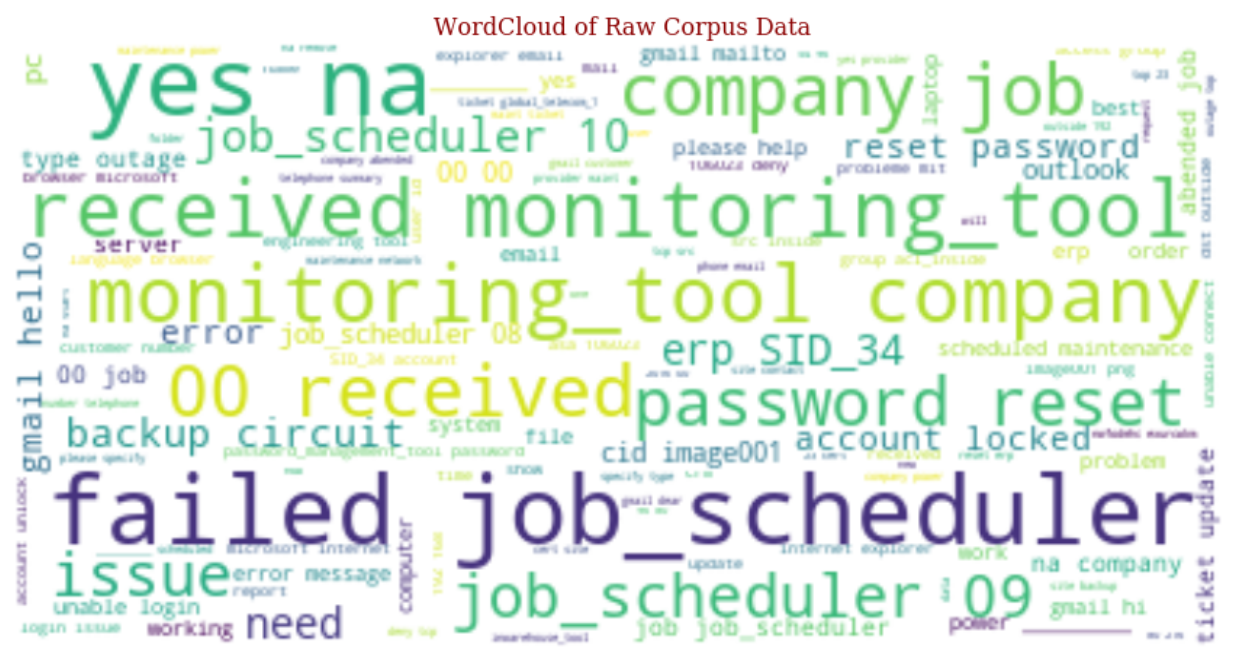
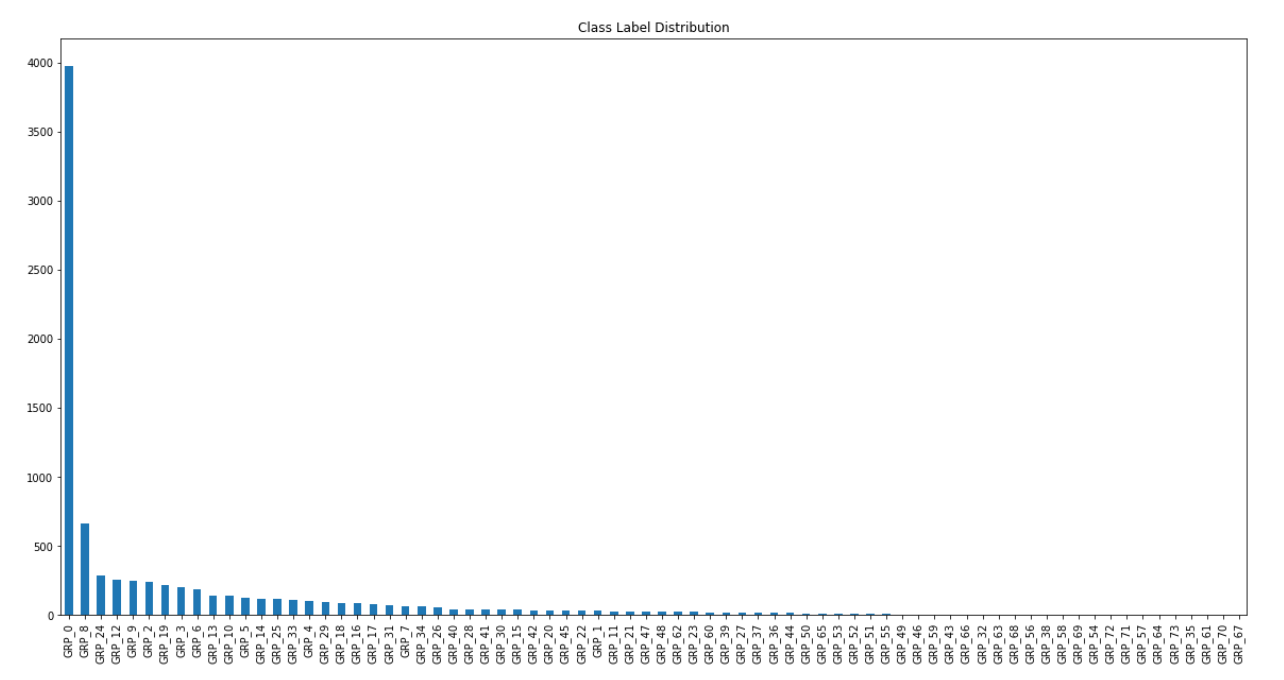


Figure 7

### Data Distribution before clean-up



### Data Cleanup

* The data is now cleaned by converting all data into lower case and adding regex list as required to remove unnecessary text such as https:? , numbers, special characters, single alphabets, email tags and email ID’s , user names etc and store the cleaned text in 3 new columns called Rawdescription ,RawWordCount and CleanDescription
* StopWords are the most common words in NLP for ex is, as the, a, an etc. For the purpose of analysing text data and building NLP models, these StopWords might not add much value to the meaning of the document.
* For tasks like text classification, where the text is to be classified into different categories, StopWords are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text.

|  |  |
| --- | --- |
| **Sample Text with Stop words** | **Sample text without Stop words** |
| There is a book on the table | There book Table |

### Benefits of removing StopWords:

* Up on removing StopWords, dataset size decreases and the time to train the model also decreases
* Removing StopWords helps improve the performance as there are fewer and only meaningful tokens left. Thus, it increases classification accuracy

#### StopWords have been removed in the following steps

1. Text Classification
   * Spam Filtering
   * Language Classification
   * Genre Classification
2. Caption Generation
3. Auto-Tag Generation

#### Avoid removing StopWords in the below steps

1. Machine Translation
2. Language Modelling
3. Text Summarization
4. Question-Answering problems

NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text pre-processing. It’s one of my favourite Python libraries. **NLTK has a list of StopWords stored in 16 different languages.**

### TokeniZation

**Tokenization is a process of splitting up large body of texts into more smaller sentences or words. We tokenize the words using word\_tokenize function available as part of nltk.**

**Code Snippet for Tokenization**

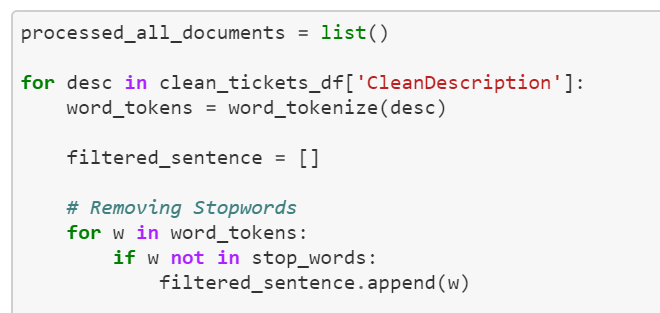


Figure 8

### Lemmatization

**Lemmatization** is the process of grouping together the different inflected forms of a word so they can be analysed as a single item.  lemmatization does morphological analysis of the words.

**Code Snippet for Lemmatization**

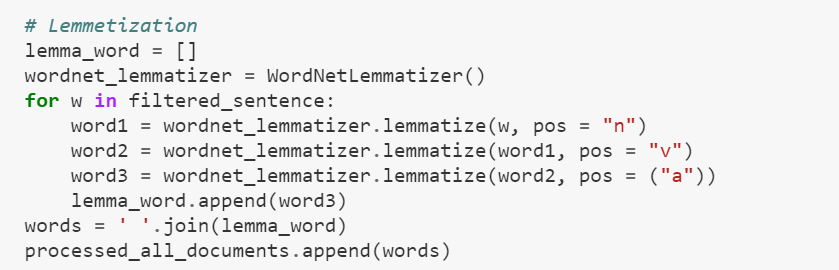


Figure 9

**After Tokenization and Lemmatization, it was found that**

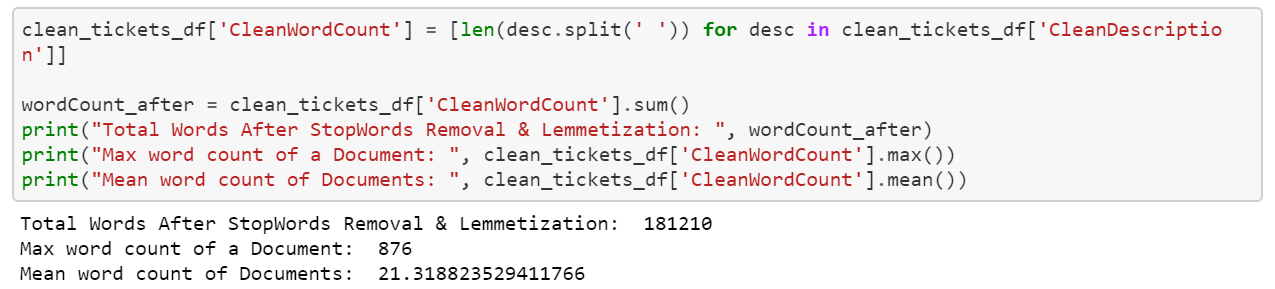


Figure 10

we can clearly see that the word count for each record has significantly decreased after the above process. Before the StopWord removal the Total word Count was 307607 and after removal the total word count is 181210

### Word Count after Data Pre-Processing

We have compared the data after StopWord removal and Lemmatization to the data before it .



Figure 11

Difference in the total word count before and after the StopWord removal process

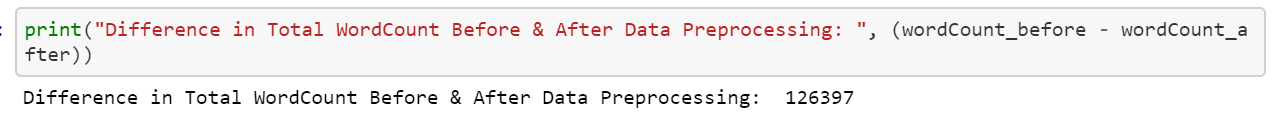
****

Figure 12

**Observations after Data Clean-up and class imbalance**

* In order to remove the class imbalance where there were 74 Groups all Groups less that 20 representations have been dropped and the finals
* Upon checking the Group Frequency it is found that for the 95th Percentile, 3 1 is our magic number

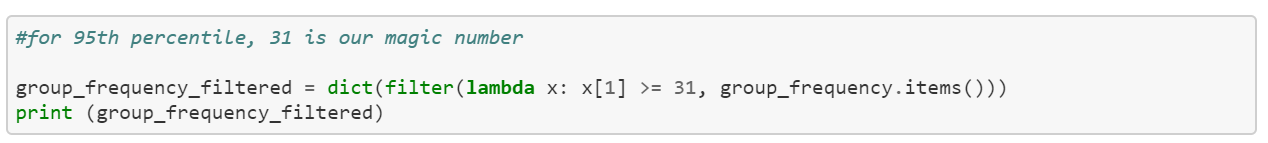


Figure 13

* We have filtered the data frame and retained only necessary groups post this it Is found that the data frame with clean tickets is 8113 and their length is 34

### Word Cloud of Cleaned & Class Balanced Corpus Data after dropping poorly represented classes

Below is the Word Cloud of the cleaned data and observations as said above

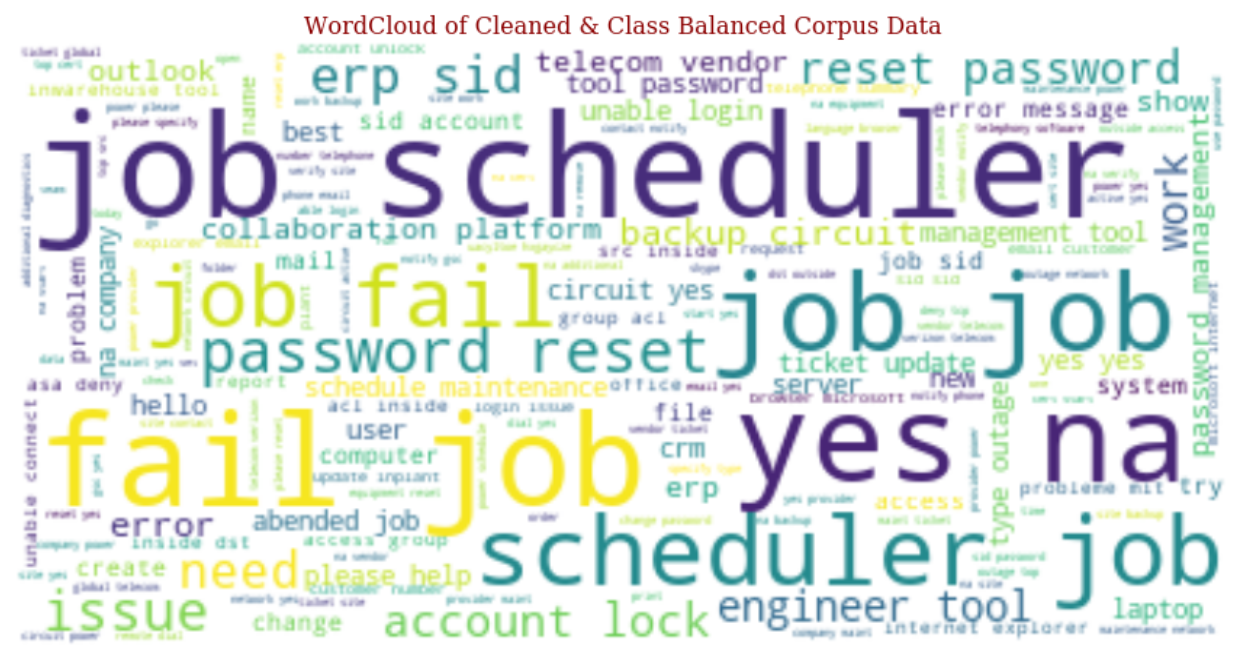


Figure 14

**Inference :**No of Classes reduced from 74 to 34 and no of records reduces from 8500 to 8113

## LABEL ENCODING

**Label Encoding** is an important pre-processing step for the structured dataset in supervised learning. It refers to converting the labels into numeric form so as to convert it into the machine-readable form.

**Example of Label Encoding after applying this on the Assignment Group column**

|  |  |
| --- | --- |
| Before applying Label Encoding | After applying Label Encoding |
| Grp\_0 | 0 |
| Grp\_1 | 1 |
| Grp\_2 | 2 |
| Grp\_3 | 3 |

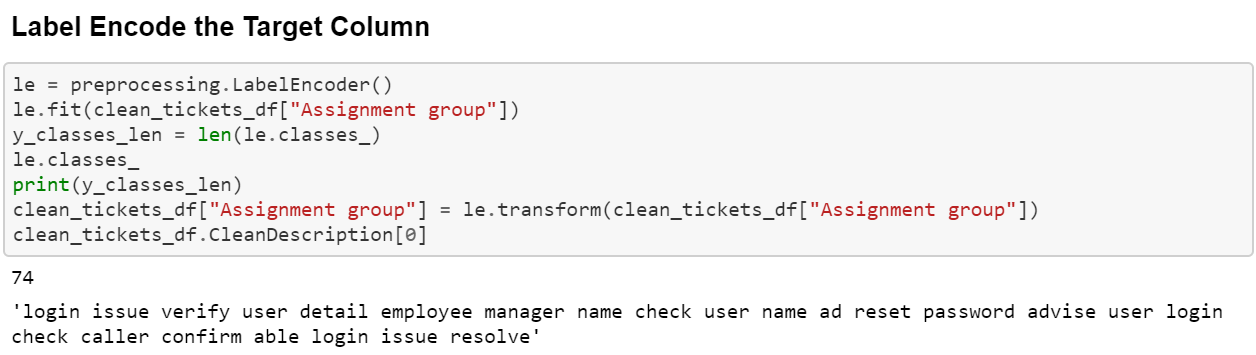


Figure 15

## Create Tf-idf Vector

Increase the weight given to more occurrences of a term in Ticket Description by creating below TF-idf vectors, code snippet below

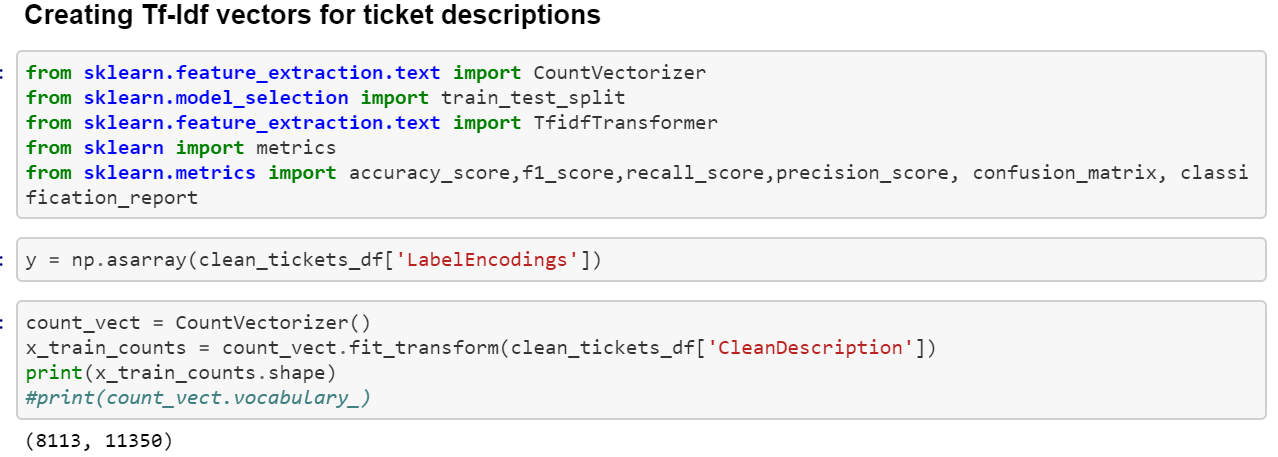


Figure 16

In Label encoding a count Vectorizer is created to count the term frequency and Vector word count shape is found as above

We will compute the IDF Values by using the below code snippet

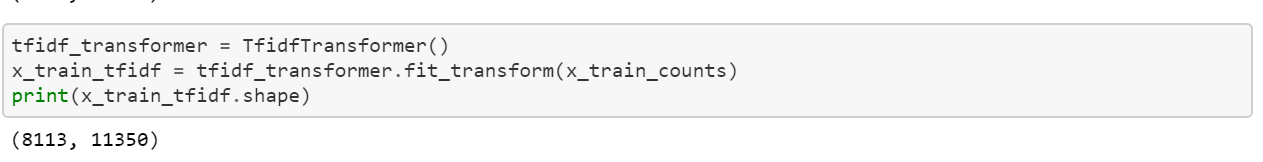


Figure 17

With Tfidftransformer you will systematically compute word counts using CountVectorizer and then compute the Inverse Document Frequency (IDF) values

Splitting of the Test and Train data and its Shape

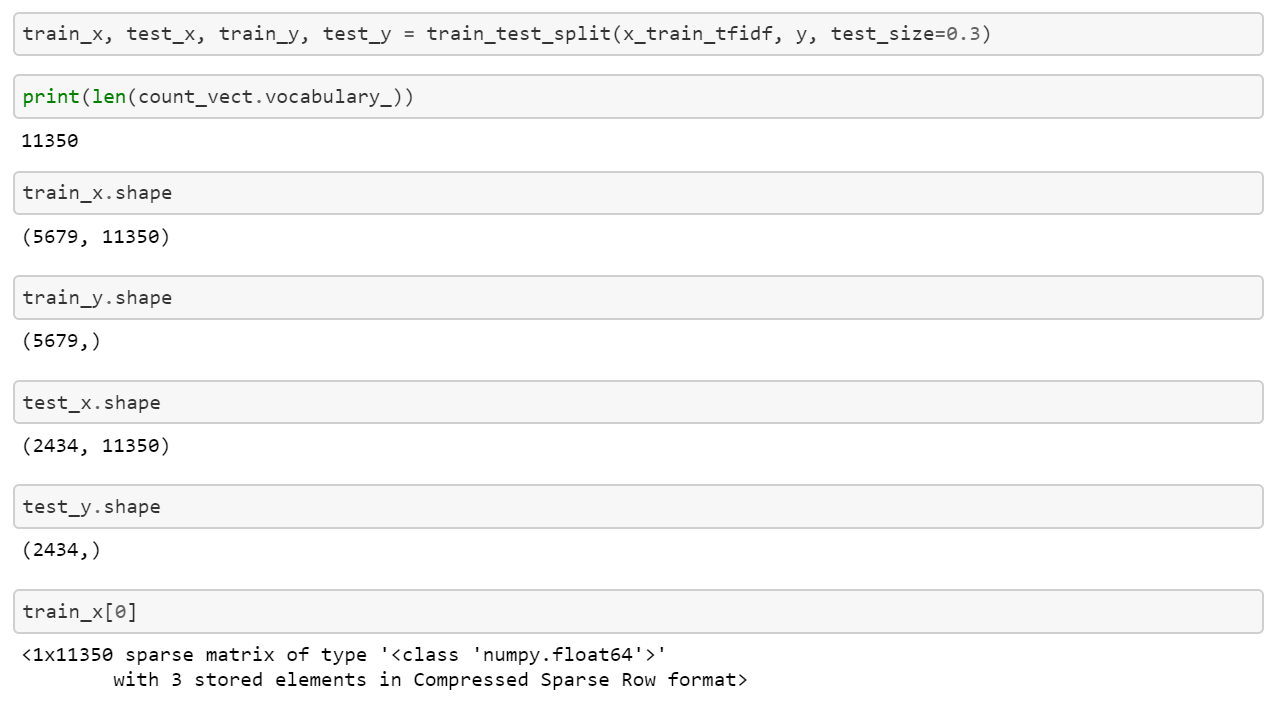


Figure 18

## Development of ML Models

Classification

Logistic Regression is known for its versatility and explain-ability. It is very simple to train and the results are interpretable as you can easily extract the most important coefficients from the model.

### Logistic Regression

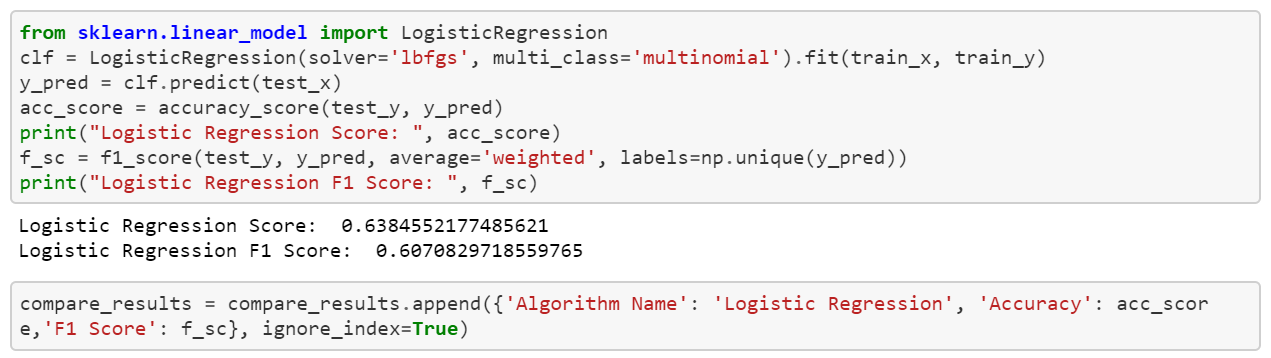


Figure 19-Logistic Regression

Naïve Bayes is a Set of classification algorithms based on **Bayes’ Theorem**.

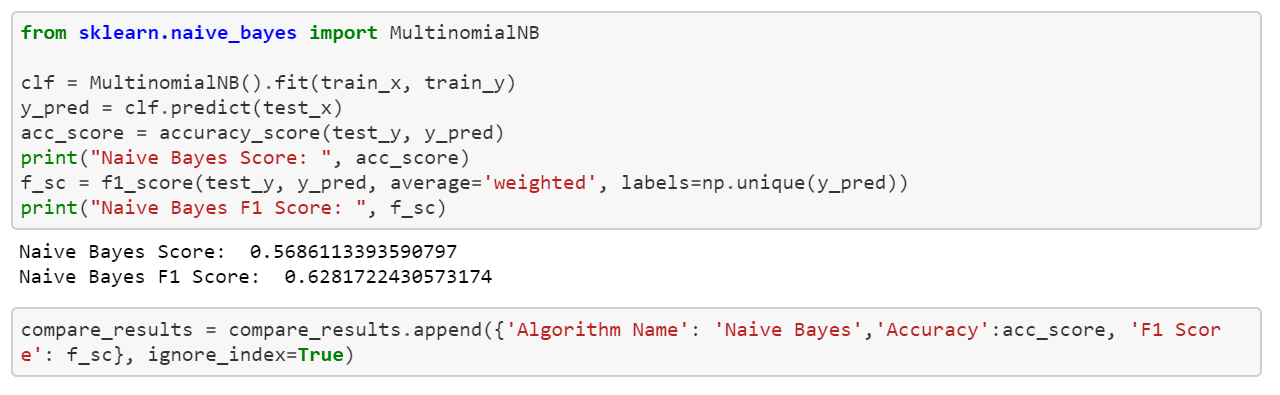
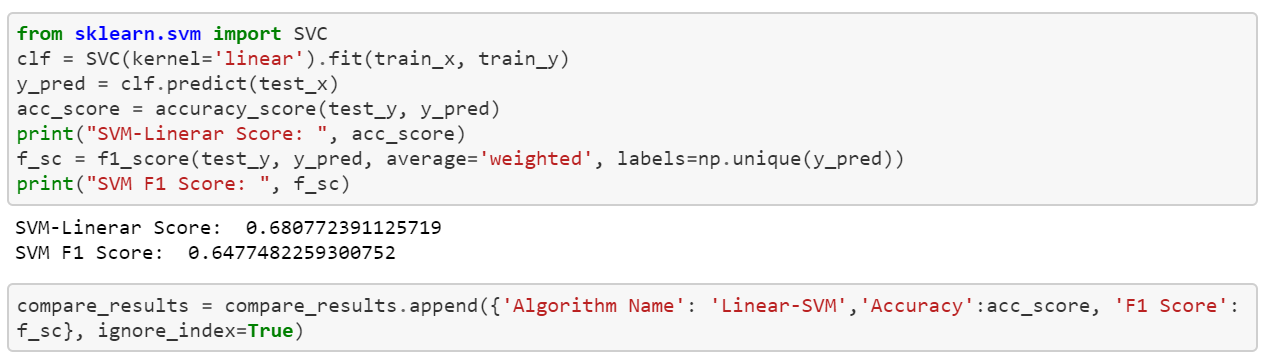


Figure 20- Naive Bayes

### Linear SVM

Applying Linear Support Vector Machine



### Decision tree

Applying Decision tree as per below code snippet



Figure 21-- Decision Tree

## Ensemble Technique applied

### Random Forests

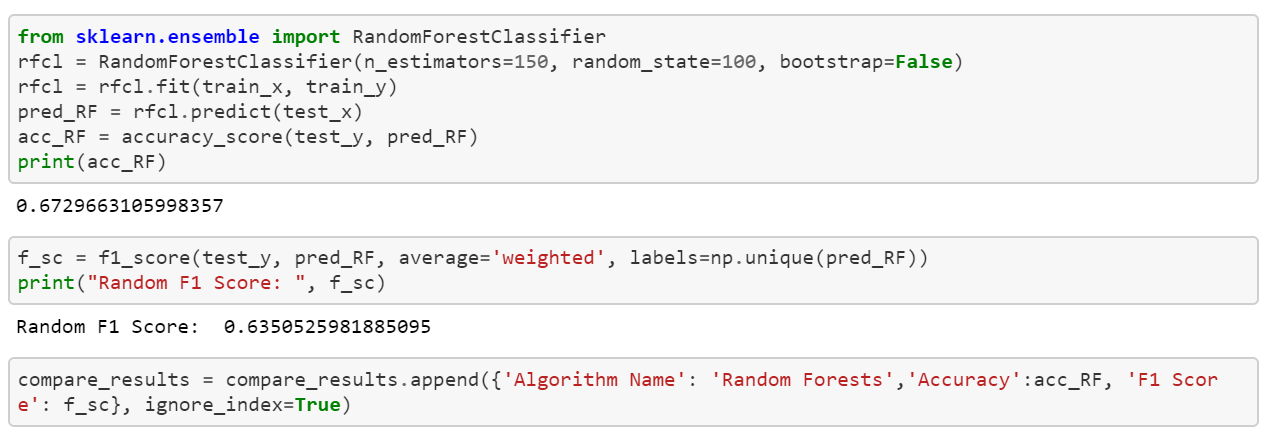


Figure 22 Random Forest

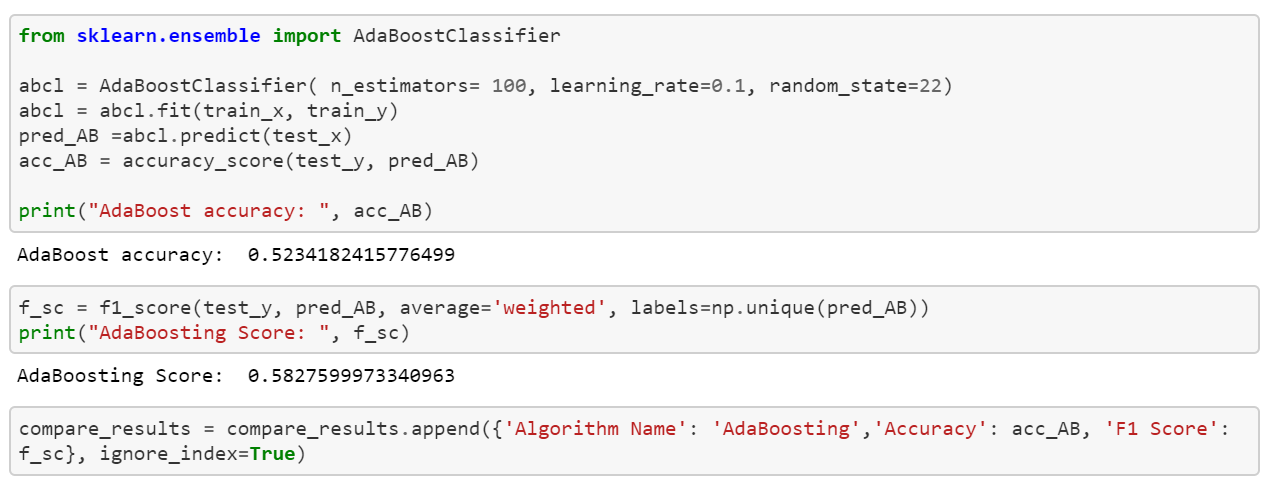
Boosting 

Figure 23 Boosting

### Bagging

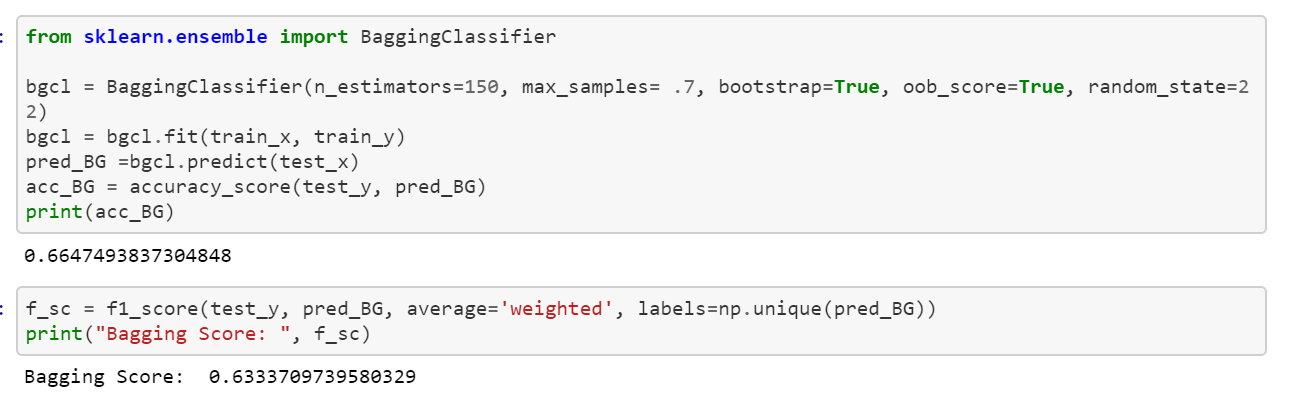


Figure 24 Bagging

Compare the model Scores of all above Models

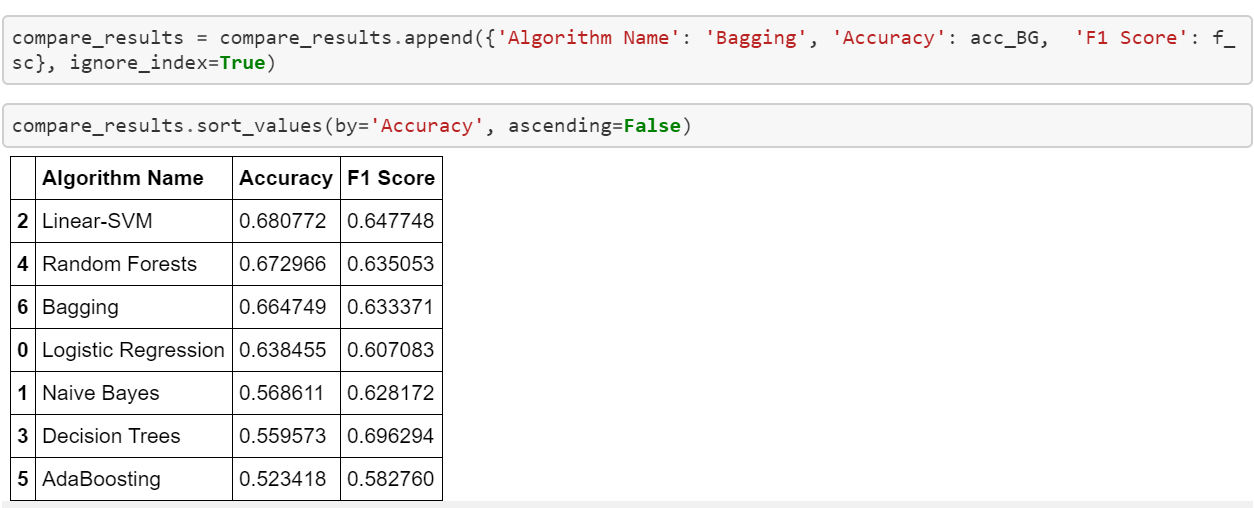


Figure 25 Model Score Comparison

**Inference:**

* From the above we can see that the LSVM model and Random Forests yield better scores than other Models
* Hyperparameter Turning using GridSearchCV is required to get a better insight
* Deep Learning models must be used to evaluate further and enhance the performance

Planned Activities

1. Building DL based models
2. Hyperparameters tuning
3. Any benchmarking if there
4. consolidation