AUTOMATIC TICKET ASSIGNMENT

Final Report

The project encompasses the process of reading and merging datasets, pre-processing the gathered data and identifying the most relevant Deep Learning model to aid the process of ticket classification with regards to Incident Management.

Shalini Tiwari

AnandaKrishna S

Gourav Saha

Vamsee Krishna

Table of Contents

INTRODUCTION

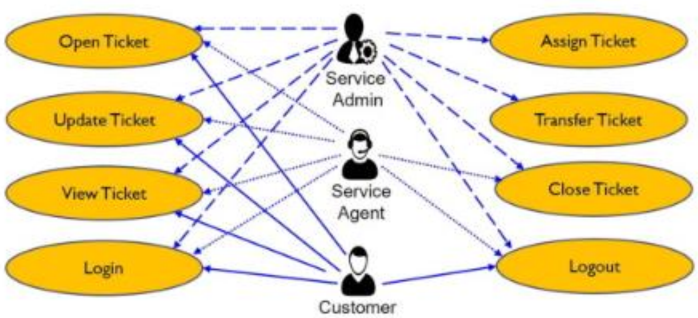
Business Domain Value

In any industry, Incident Management plays an important role in delivering quality support to customers. When a user submits a support ticket, it flows into the platform via email, phone or an embedded portal. Each ticket contains a bit of text about the problem or request, which is quickly reviewed by a support professional and sent on its way. Once the correct assignment group picks up the ticket, some amount of work gets completed and the incident state reverts to closed.

As per one of the client records in 2019, more than 60000 tickets were submitted at their platform with intent to reach nearly 15 business groups. Every ticket cost the organization $13, despite an average accuracy score of only 40%. **Incorrectly** assigned tickets bounced between business groups for an average of 21 days before landing in the right place. Cost, latency and accuracy were a huge concern that raises the urgency of automating this process.

Understanding the Project Problem Statement

An incident ticket is created by various groups of people within the organization to resolve an issue as quickly as possible based on its severity. Whenever an incident is created, it reaches the Service desk team and then it gets assigned to the respective teams to work on the incident. The Service Desk team will perform basic analysis on the user's requirement, identify the issue based on given descriptions and assign it to the respective teams.



The manual assignment of these incidents might have below disadvantages:

- More resource usage and expenses.
- Human errors - Incidents get assigned to the wrong assignment groups
- Delay in assigning the tickets
- More resolution times
- If a particular ticket takes more time in analysis, other productive tasks get affected for the Service Desk

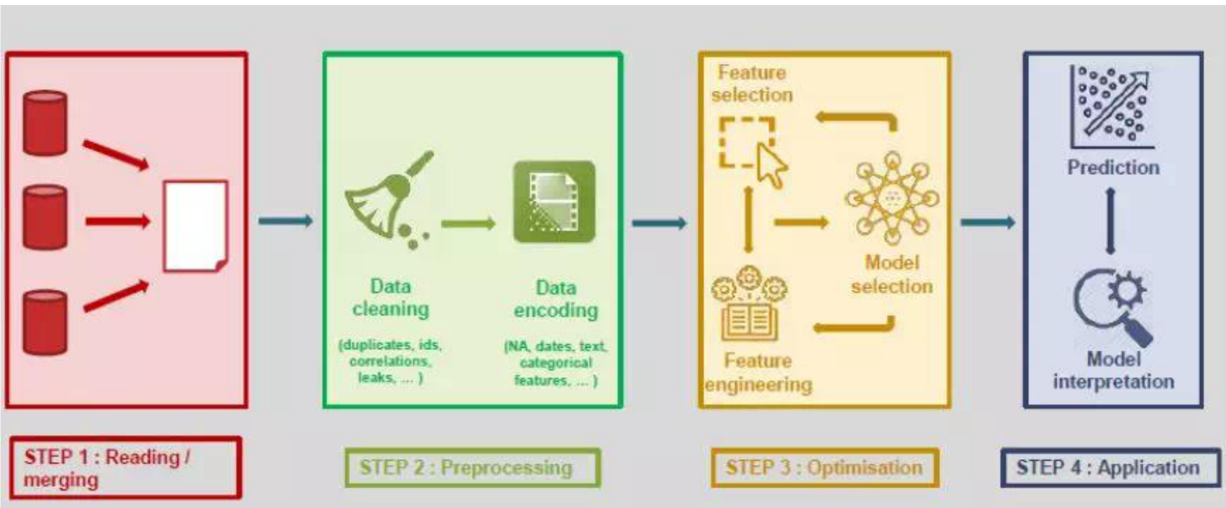
If this ticket assignment is automated, it can be more cost-effective, less resolution time and the Service Desk team can focus on other productive tasks.

Objective

The objective of the project is to build an AI-based classifier model to assign the tickets to right functional groups using:

- Different classification models.
- Transfer learning to use pre-built models
- Set the optimizers, loss functions, epochs, learning rate, batch size, check pointing and early stopping to achieve an accuracy of at least 85%.

Machine Learning Process



The diagram depicts end to end machine learning model building flow that forms the base for every problem statement. From data reading, pre-processing, optimization and result prediction, each step is controlled and manually performed.

In any dataset the data is first read and analyzed whether the problem statement is Supervised or Unsupervised problem. If the problem belongs to Supervised category it is further analyzed for Regression and Classification problem. Next step is data analysis where null and missing values are identified, data is checked if imbalanced and visualizing the relation of independent features with the target variable. Preprocessing step comprised of Data cleaning and Data encoding, which generally comprises dealing with null value, if the data is categorical then encoding[label or one hot] them to machine understandable code. If the input feature is a text then NLP preprocessing is used. For Model building, the accuracy of the model is compared with different classical Machine learning model along with Deep learning model. The performance of each model is compared based on accuracy precision and recall score. The model summary can be visualized using classification report.

In order to improve model performance many techniques like optimizers, loss function (categorical_crossentopy, binary_crossentropy), additional layers, dropouts to reduce overfitting, managing no of epochs, adding weight embedding should be used.

READING AND MERGING DATASET

Observations from the Dataset

- Dataset has three features – Short Description, Description, Caller and one labeled/Target class -Assignment group.

```
data = pd.read_excel("input_data.xlsx")
data.head()
```

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

- Problem statement is a classification problem
- There are total 8500 records.
- There are 74 assignment groups with GRP 0 having the maximum frequency[3976] followed by second most frequent group GRP 8 [661 occurrence]. This huge difference clearly states that the data is highly imbalanced.

```
Shape of the dataset :  (8500, 4)
```

	Short description	Description	Caller	Assignment group
count	8492	8499	8500	8500
unique	7481	7817	2950	74
top	password reset	the	bpctwhsn kzqsbmtp	GRP_0
freq	38	56	810	3976

- Many non-English languages also found in the data. Need to translate them to English.
- Dataset is highly inconsistent as it contains digits, Email/chats, special characters, punctuations, image file format, hyperlinks, urls.

	Combined Description	Assignment group	Language
4878	install kis \ewew8323506 \guvgytniak install k...	GRP_24	Hungarian
4879	install kis \ewew8323504 \zlqfptjx xnklbfua in...	GRP_24	Hungarian
7445	probleme mit EU_tool \obqridjk ugelctsz proble...	GRP_24	Hungarian
8465	vpn è¿žæŽ¥ä□äŠ vpnè¿žä□äŠï¼Œè¯·è½¬ç»™ è´°æ...	GRP_30	Hungarian

- There are 8 records for which short description is missing and for 1 record description is missing. Also few descriptions same as the short description.

```
[162] # Finding the record at which the value
      data.loc[data['Short description'].isnull()]
```

	Short description	Description	Caller	Assignment group
2604	NaN	\r\n\r\nreceived from: ohdrnswl.rezuibdt@gmail...	ohdrnswl rezuibdt	GRP_34
3383	NaN	\r\n-connected to the user system using teamvi...	qftpazns fxpnytmk	GRP_0
3906	NaN	-user unable tologin to vpn.\r\n-connected to...	awpcmsey ctdiuqwe	GRP_0
3910	NaN	-user unable tologin to vpn.\r\n-connected to...	rhwsmefo tvphyura	GRP_0
3915	NaN	-user unable tologin to vpn.\r\n-connected to...	hxripljo efzounig	GRP_0
3921	NaN	-user unable tologin to vpn.\r\n-connected to...	cziadygo veiosxby	GRP_0
3924	NaN	name:wvqgbdhm fwchqjor\nlanguage:\nbrowser:mic...	wvqgbdhm fwchqjor	GRP_0
4341	NaN	\r\n\r\nreceived from: eqmuniov.ehxkcbgj@gmail...	eqmuniov ehxkcbgj	GRP_0

```
[163] data.loc[data['Description'].isnull()]
```

	Short description	Description	Caller	Assignment group
4395	i am locked out of skype	NaN	viyglzfo ajtfzpkb	GRP_0

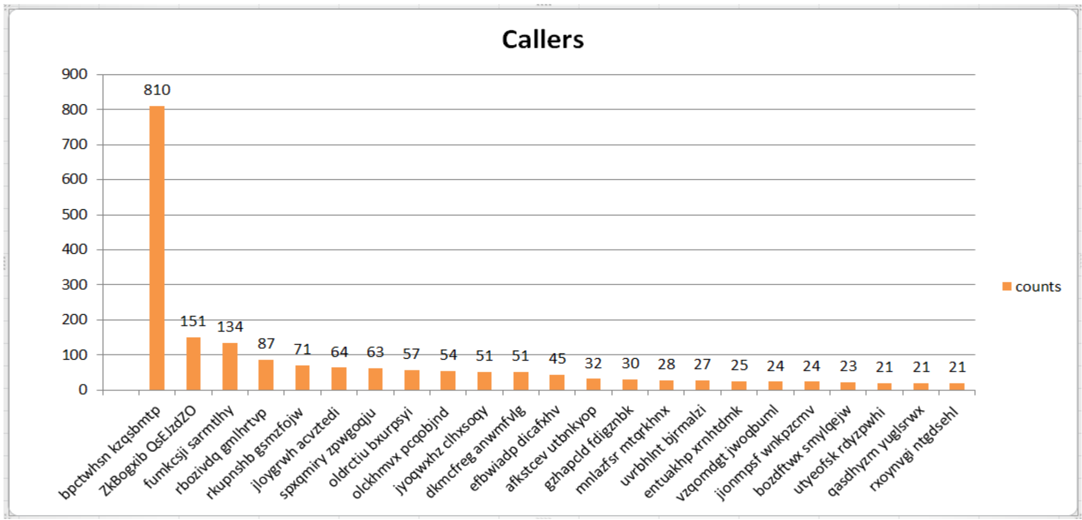
- There are no such records from which both description and short description is missing. Hence in order to deal with null values, both the columns short description and description can be combined and then all the NLP operations can be performed on the combined column.
- Also we can then drop the short description and description column as well.

	Combined Description	Caller	Assignment group
0	login issue -verified user details.(employee# ...	spxjnwir pjlcoqds	GRP_0
1	outlook \r\n\r\nreceived from: hmjdrvpb.komuay...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn \r\n\r\nreceived from: eylq...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page unable to access...	xbkucsvz gcpydteq	GRP_0
4	skype error skype error	owlgqjme qhcozdfx	GRP_0

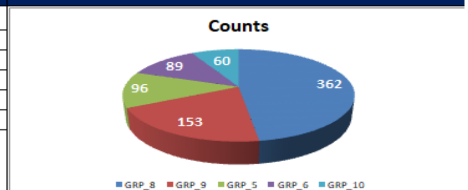
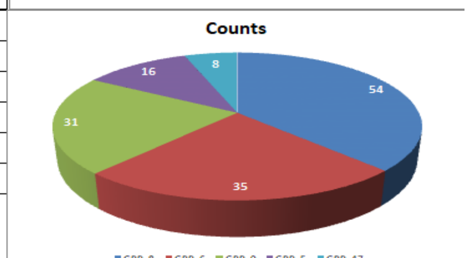
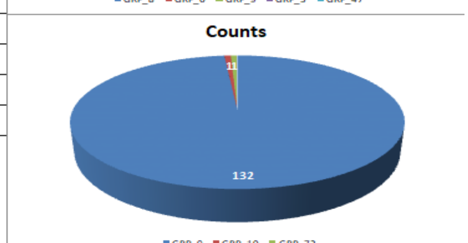
Visualizing Different Patterns

After dealing with the missing values, we visualized the relation of each feature with the labeled output column to derive the final conclusion whether a particular feature is actually contributing to decide the output or not. From the final data collected so far, we did this analysis on "Caller" column and have considered caller records who have locked a ticket more than 20 times in various categories. Below are our findings.

- There are 23 callers who have raised tickets more than 20 times.



- Top 5 groups for each frequent callers will help to identify if maximum users are logging tickets in same group.

Caller	Table		Pie Chart
bpctwhsn kzqsbmtp	**Groupname**	**Counts**	
	GRP 8	**362**	
	GRP 9	**153**	
	GRP 5	**96**	
	GRP 6	**89**	
	GRP 10	**60**	
ZkBogxib QsEJzdZO	**Groupname**	**Counts**	
	GRP_8	**54**	
	GRP_6	**35**	
	GRP_9	**31**	
	GRP_5	**16**	
	GRP_47	**8**	
fumkcsji sarmtlhy	**Groupname**	**Counts**	
	GRP_0	**132**	
	GRP_19	**1**	
	GRP_72	**1**	

- Seeing the results it can be concluded that not all frequent users are logging issues into same group and hence no solid inference can be drawn from Caller column and hence can be dropped.

- After finalizing on Caller column the data looks like

	Combined Description	Assignment group
0	login issue -verified user details.(employee# ...	GRP_0
1	outlook \r\n\r\nreceived from: hmjdrvpb.komuay...	GRP_0
2	cant log in to vpn \r\n\r\nreceived from: eylq...	GRP_0
3	unable to access hr_tool page unable to access...	GRP_0
4	skype error skype error	GRP_0

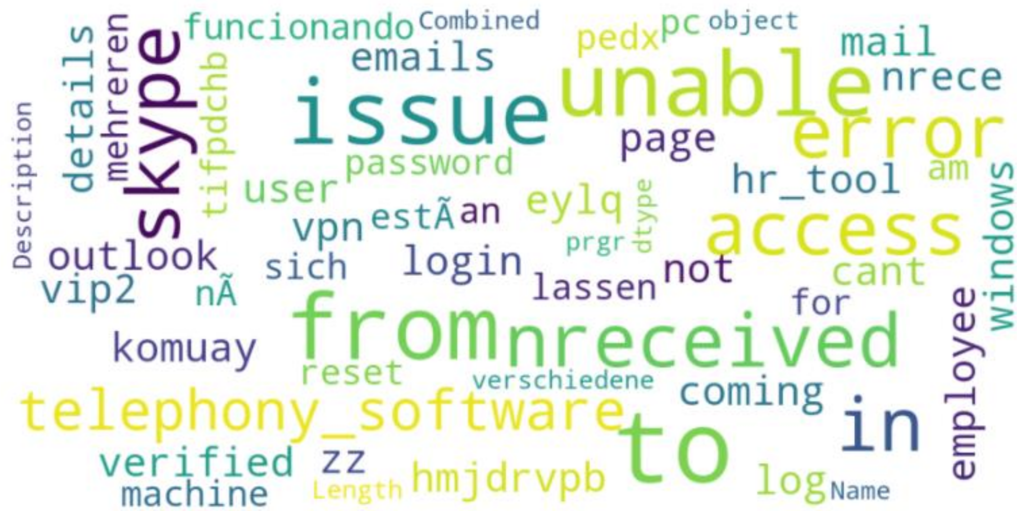
Visualizing different text features

Wordcloud is used to visualize unprocessed description in the data. At this point no preprocessing is done on the data and can be considered as raw data with all abnormalities (stop words, punctuations, email ids, special characters). Below are the most frequent words and their word cloud representation

```
# top 50 frequent words without preprocessing and when no stop words have been applied
Counter(" ".join(data['Combined Description']).split()).most_common(50)
```

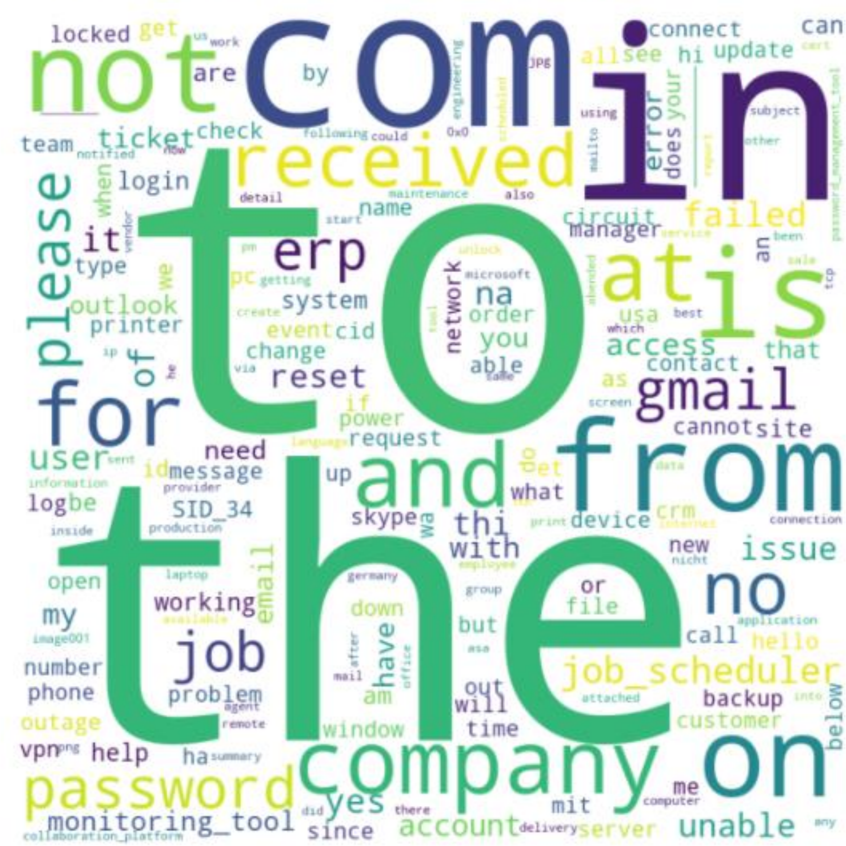
```
[('to', 8446),
 ('the', 7020),
 ('in', 4909),
 ('is', 3525),
 ('on', 2891),
 ('not', 2851),
 ('and', 2758),
 ('for', 2629),
 ('from:', 2499),
 ('i', 2432),
 ('received', 2371),
 ('a', 2012),
 ('job', 1975),
 ('please', 1946),
 ('password', 1865),
 ('of', 1845),
 ('erp', 1827),
 ('failed', 1700),
 ('job_scheduler', 1629),
 ('at:', 1614),
 ('(yes/no/na)', 1566),
 ('?', 1564),
 ('unable', 1483),
 ('this', 1438),
 ('-', 1434),
 ('it', 1401),
 ('reset', 1374),
 ('user', 1223),
 ('my', 1154),
 ('with', 1134),
 ('account', 1117),
 ('access', 1080),
 ('from', 1079),
 (':', 1065),
 ('you', 1056),
 ('have', 1031),
 ('monitoring_tool@company.com', 961),
 ('company', 899),
 ('issue', 898),
 ('are', 890),
 ('at', 885),
 ('error', 822),
 ('be', 817),
 ('am', 812),
 ('that', 803),
 ('outlook', 797),
 ('ticket', 787),
 ('can', 786),
 ('site', 753),
 ('login', 749)]
```

```
1  # printing wordcloud of most frequent words with stop words
2  generateWordCloud(data['Combined Description'])
```



When the word cloud is generated using generate from text function

```
1  #printing wordcloud of most frequent words without stop words
2  generateWordlLoud_FromText(data['Combined Description'])
```



When we do not apply any preprocessing the data contains text in different language, so we can conclude that the data is multilingual and needs language detection and translation.

Data Preprocessing

Language Detection

For converting multilingual text into common language, the first step is to detect the language of each text. Once the detection is done then we can translate the other languages back to English. For language detection, langdetect and polyglot libraries have been used but it seems polyglot gives the best results.

LangDetect

```
1   data['Language_LangDet'].value_counts()
```

```
en    7063
de     380
af     277
it     137
fr     120
sv      79
da      73
no      72
nl      69
es      50
ca      47
pl      29
pt      24
cy      11
tl      10
sq      10
ro      10
sl       9
et       6
hr       5
id       5
tr       3
fi       3
so       2
cs       2
lt       2
lv       1
sk       1
Name: Language_LangDet, dtype: int64
```

Polyglot

```
1   data['Language_Polyglot'].value_counts()
```

```
English              7935
German                401
Danish                 43
un                     28
Latin                  12
Portuguese             10
Hungarian               8
Waray                   7
Nauru                   6
Scots                   5
Luxembourgish           5
Polish                  5
Uzbek                   3
Dutch                   3
Zhuang                  3
Scottish Gaelic         2
Ganda                   2
Kinyarwanda             2
Spanish                 2
Norwegian               2
Interlingua             2
Tagalog                 1
Maltese                 1
Icelandic               1
Galician                1
Finnish                 1
Irish                   1
Estonian                1
Malay                   1
Norwegian Nynorsk       1
Catalan                 1
Latvian                 1
French                  1
Tsonga                  1
Welsh                   1
Name: Language_Polyglot, dtype: int64
```

Out of 8500 total records, polyglot recognized 7935 records as English whereas langdetect recognized 7063 records. There are total 575 records in with language other than English for which language translation needs to be applied

Language Translation

For language translation the data was splitted into two sets, one in which there will be only English text and in another set there will be records only in different language. Language translation(goggle translation) will be applied on other language dataset and after translation it will be merged with the English dataset and final translation dataset will be created. Form the conversion result it has been observed that goggle translator does not give 100% accuracy.

```
1 OtherLanguagedf["LanguageAfterConversion"].value_counts()
```

```
de    388
en    143
pt      8
pl      4
es      3
ro      3
co      3
tl      2
vi      2
lb      2
af      1
jw      1
is      1
gl      1
da      1
cy      1
ar      1
Name: LanguageAfterConversion, dtype: int64
```

Out of 575 records 143 were successfully converted in English. Now will merge the converted text with English dataset and find the total count of English text.

```
1 DataAfterTranslation['Language'].value_counts()
```

```
English                8338
German                   62
un                       28
Danish                    9
Latin                     9
Nauru                     6
Hungarian                 4
Waray                     4
Scots                     4
Polish                    4
Uzbek                     3
Portuguese                3
Kinyarwanda               2
Ganda                     2
Zhuang                    2
Scottish Gaelic           2
Interlingua               2
Estonian                  1
Maltese                   1
Finnish                   1
Luxembourgish             1
Irish                     1
Spanish                   1
Norwegian                 1
Latvian                   1
French                    1
Dutch                     1
Norwegian Nynorsk         1
Malay                     1
Tagalog                   1
Tsonga                    1
Catalan                   1
Welsh                     1
Name: Language, dtype: int64
```

After all the translation, a total of 8338 records out of 8500 have been identified as English. So far the data after translation is displayed below:

	Combined Description	Language	Assignment group
4	skype error skype error	Latin	GRP_0
146	erp_print_tool install. erp_print_tool install.	Kinyarwanda	GRP_0
148	install acrobat standard install acrobat standard	Malay	GRP_0
223	problems with bluescreen. Hello ,\n\nit happen...	English	GRP_24
251	reset the password for fygrwuna gomcekzi on e-...	English	GRP_0

Text Preprocessing

After translation, the raw text is preprocessed in which all the anomalies have been removed. Below is sequence in which the preprocessing is done:

- Removing Duplicate words,
- Removing Contractions,
- Removing Email,
- Removing Digits,
- Removing Special Characters
- Removing Punctuations
- Lemmatization
- Tokenization[Regex Tokenization].

	Combined Description	Language	Assignment group	CLeanText	Tokenize
4	skype error skype error	Latin	GRP_0	skype error	[skype, error]
146	erp_print_tool install. erp_print_tool install.	Kinyarwanda	GRP_0	erp print tool install	[erp, print, tool, install]
148	install acrobat standard install acrobat standard	Malay	GRP_0	install acrobat standard	[install, acrobat, standard]
223	problems with bluescreen. Hello ,\n\nit happen...	English	GRP_24	problem with bluescreen hello it happen again ...	[problem, with, bluescreen, hello, it, happen,...
251	reset the password for fygrwuna gomcekzi on e-...	English	GRP_0	reset the password for fygrwuna gomcekzi on em...	[reset, the, password, for, fygrwuna, gomcekzi...

Visualizing Processed Text without removing stop words


```
1  # top 50 frequent words with preprocessing and when no stop words have been applied
2  Counter(" ".join(DataAfterTranslation["CLeanText"]).split()).most_common(50)
```

```
[('be', 5022),
 ('to', 3659),
 ('job', 3380),
 ('from', 3211),
 ('the', 2901),
 ('in', 2826),
 ('not', 2595),
 ('receive', 2515),
 ('i', 2193),
 ('please', 1970),
 ('on', 1688),
 ('and', 1645),
 ('have', 1601),
 ('for', 1600),
 ('a', 1525),
 ('password', 1489),
 ('at', 1483),
 ('it', 1331),
 ('tool', 1234),
 ('issue', 1212),
 ('do', 1209),
 ('erp', 1171),
 ('can', 1168),
 ('of', 1106),
 ('user', 1102),
 ('no', 1065),
 ('this', 1043),
 ('with', 1025),
 ('work', 1015),
 ('sid', 996),
 ('reset', 992),
 ('scheduler', 970),
 ('fail', 939),
 ('company', 936),
 ('unable', 882),
 ('you', 874),
 ('error', 854),
 ('my', 838),
 ('ticket', 768),
 ('access', 760),
 ('need', 757),
 ('account', 753),
 ('hello', 691),
 ('system', 683),
 ('help', 678),
 ('that', 646),
 ('when', 636),
 ('get', 633),
 ('we', 621),
 ('email', 601)]
```

Visualizing Processed Text after removing stop words

From the stop words we have removed negative words like no, nor and not, since they might add some value to the text. Hence these words are not removed from the text.


```
1  # top 50 frequent words with preprocessing and when stop words have been removed
2  Counter(" ".join(DataAfterTranslation["CLeanText_WithoutStopWords"]).split()).most_common(50)
```

```
[('job', 3380),
 ('not', 2595),
 ('receive', 2515),
 ('please', 1970),
 ('password', 1489),
 ('tool', 1234),
 ('issue', 1212),
 ('erp', 1171),
 ('user', 1102),
 ('no', 1065),
 ('work', 1015),
 ('sid', 996),
 ('reset', 992),
 ('scheduler', 970),
 ('fail', 939),
 ('company', 936),
 ('unable', 882),
 ('error', 854),
 ('ticket', 768),
 ('access', 760),
 ('need', 757),
 ('account', 753),
 ('hello', 691),
 ('system', 683),
 ('help', 678),
 ('get', 633),
 ('email', 601),
 ('network', 600),
 ('use', 576),
 ('hostname', 575),
 ('vendor', 556),
 ('would', 554),
 ('outlook', 550),
 ('login', 546),
 ('power', 504),
 ('message', 503),
 ('telecom', 477),
 ('see', 470),
 ('update', 469),
 ('change', 451),
 ('check', 441),
 ('start', 436),
 ('problem', 430),
 ('event', 426),
 ('try', 421),
 ('name', 420),
 ('send', 414),
 ('team', 409),
 ('call', 406),
 ('connect', 401)]
```

Visualizing Assignment Group

Model Building

Model Selection

For model building, the accuracy has been calculated both on traditional ML classification algorithm and deep learning algorithm using LSTM. Machine Learning algorithms like Decision tree, Random Forest, Naïve Bayes, KNN and Logistic Regression have been compared.

In deep learning initially the model is build using single layer LSTM and further the performance is checked using glove embedding as well.

Classification ML Algorithms Report

Since our data is textual, the data needs to be encoded and for which below two approaches have been followed

- Count Vectorizer followed by TFID Transform
- TFID Vectorizer

Both the above techniques provide the same output. From the data that we got after preprocessing the input to the vectorizer will be the final preprocessed text and the output will be the assignment group.

	FinalProcessedData	Assignment group
4	skype error	GRP_0
146	erp print tool install	GRP_0
148	install acrobat standard	GRP_0
223	problem bluescreen hello happen pc hang presen...	GRP_24
251	reset password fygrwuna gomcekzi email please ...	GRP_0

For model building, the training and test split used is 70:30. Also training data is transformed using TFidVectorizer and label encoder.

```
tfidfvectorizer.fit(X_train)
xtrain_tfidf  = tfidfvectorizer.transform(X_train)
xtest_tfidf   = tfidfvectorizer.transform(X_test)
print("Sparse Matrix form of test data : \n")
xtest_tfidf.todense()
```

```
Sparse Matrix form of test data :

matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

```
encoder = preprocessing.LabelEncoder()
Y_train = encoder.fit_transform(Y_train)
Y_test = encoder.fit_transform(Y_test)
```

ML Model

```python
def model_score_df(model_dict):
    model_name, train_ac_list, ac_score_list, p_score_list, r_score_list, f1_score_list = [], [], [], [], [], []

    for k, v in model_dict.items():
        model_name.append(k)
        v.fit(xtrain_tfidf.toarray(), Y_train)
        y_pred = v.predict(xtest_tfidf.toarray())
        train_ac_list.append(v.score(xtrain_tfidf.toarray(), Y_train))
        ac_score_list.append(accuracy_score(Y_test, y_pred))
        p_score_list.append(precision_score(Y_test, y_pred, average='macro'))
        r_score_list.append(recall_score(Y_test, y_pred, average='macro'))
        f1_score_list.append(f1_score(Y_test, y_pred, average='macro'))
        model_comparison_df = pd.DataFrame([model_name, train_ac_list, ac_score_list, p_score_list, r_score_list, f1_score_list]).T
        model_comparison_df.columns = ['model_name', 'train_accuracy', 'test_accuracy_score', 'precision_score', 'recall_score', 'f1_score']
        model_comparison_df = model_comparison_df.sort_values(by='f1_score', ascending=False)

    return model_comparison_df
```

ML modern performance without hyper parameters

```python
model_dict = {'Random Forest': RandomForestClassifier(random_state=3),
              'Decsision Tree': DecisionTreeClassifier(random_state=3),
              'Gaussian Naive Bayes': GaussianNB(),
              'K Nearest Neighbor': KNeighborsClassifier(),
              'Logistic Regression': LogisticRegression()}

```

	model_name	train_accuracy	test_accuracy_score	precision_score	recall_score	f1_score
1	Decsision Tree	0.949412	0.482745	0.119363	0.091028	0.100592
0	Random Forest	0.949412	0.542745	0.161301	0.0784274	0.0955955
2	Gaussian Naive Bayes	0.675798	0.293725	0.102926	0.0960204	0.0934008
4	Logistic Regression	0.672773	0.535686	0.139867	0.0682148	0.0814219
3	K Nearest Neighbor	0.450924	0.305882	0.128847	0.0398201	0.0510187

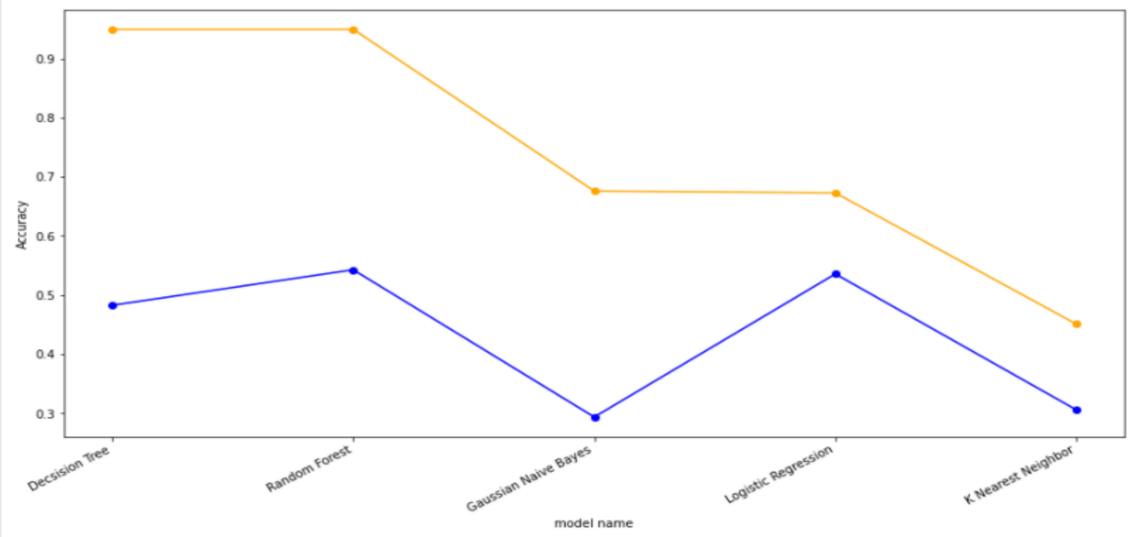
ML model performance with hyper parameters

```python
model_dict_with_hyperparameters = {'Random Forest': RandomForestClassifier(random_state=3,max_depth=20, criterion = "entropy"),
                                   'Decsision Tree': DecisionTreeClassifier(random_state=3, max_depth=20),
                                   'Gaussian Naive Bayes': GaussianNB(),
                                   'K Nearest Neighbor': KNeighborsClassifier(n_neighbors= 3 , weights = 'distance'),
                                   'Logistic Regression': LogisticRegression()}
```

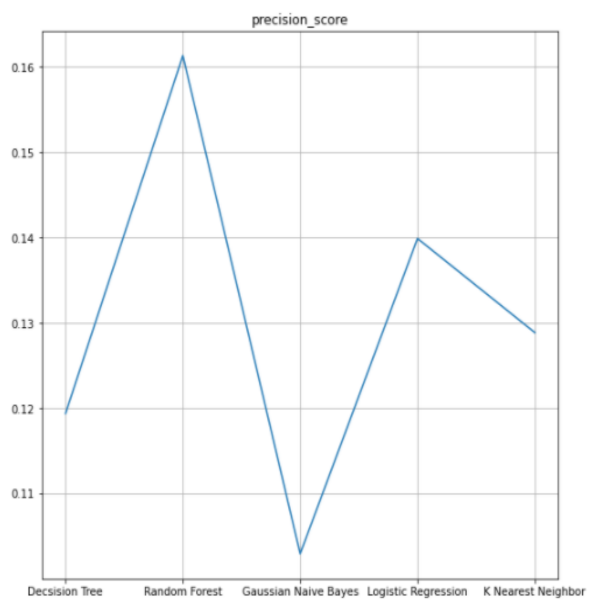
	model_name	train_accuracy	test_accuracy_score	precision_score	recall_score	f1_score
2	Gaussian Naive Bayes	0.675462	0.293725	0.103358	0.0964467	0.0938583
4	Logistic Regression	0.672773	0.534902	0.139883	0.0675988	0.0807159
3	K Nearest Neighbor	0.930756	0.358824	0.152629	0.0672544	0.0747451
1	Decsision Tree	0.675798	0.510588	0.120826	0.0613496	0.0742307
0	Random Forest	0.596975	0.473333	0.0490047	0.0226535	0.0224446

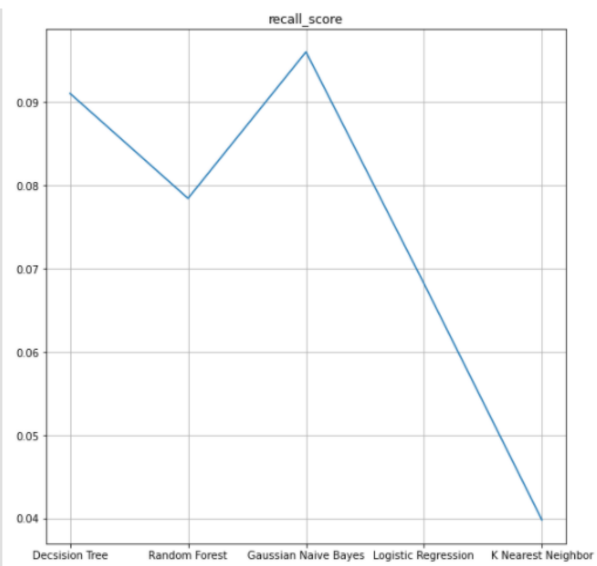
Visualization of ML Performance

- **Training vs. Test accuracy [Training- orange and test- blue]**

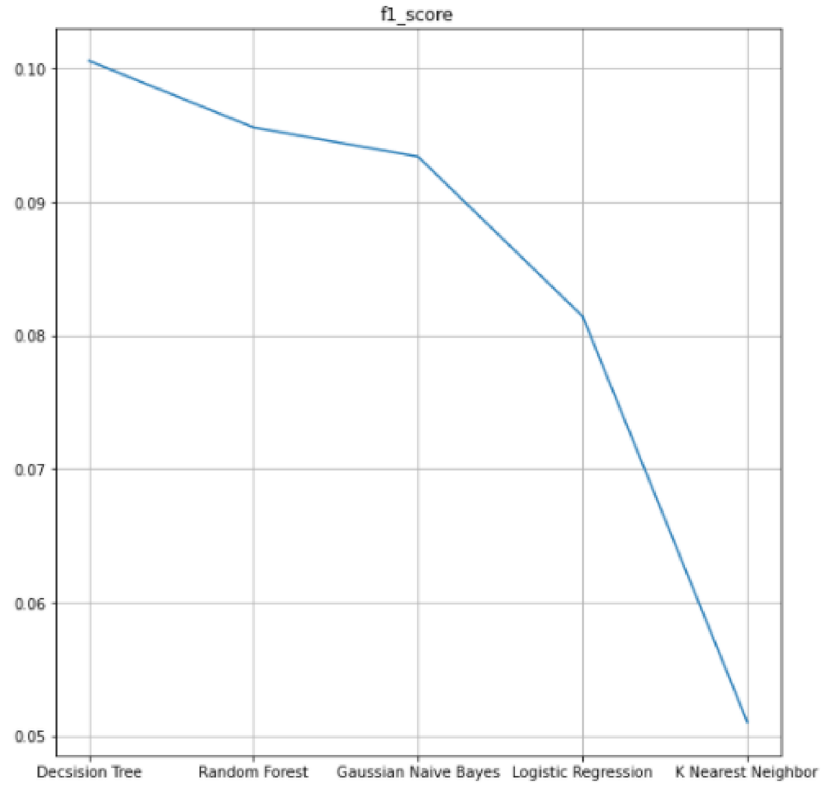


- **Precision-Recall Score**





- **F1-Score**



Conclusion

- None of the models could reach the accuracy of 60 % in test even with hyper parameters.
- Seeing the huge difference in training and test accuracy, it can be concluded that the models are over fitting
- Precision is the ratio of true positive and sum of true positive and false positive. Hence the lower the value o f precision in the result indicates, higher will be the false positive in the model prediction.
- Recall is the ratio of true positive and sum of true positive and false negative. Hence lower the value of reca ll indicates higher false negatives in the model prediction.
- Ideally higher the f1 score, the better the model performs but the data indicated the other way round

Building LSTM model

For LSTM the input textual data is first tokenized, fit to texts and then converted into sequences that can further be given as input to the model.

FinalProcessedData	Assignment group	TextToSequence
skype error	GRP_0	[85, 18]
erp print tool install	GRP_0	[8, 87, 6, 214]
install acrobat standard	GRP_0	[214, 2435, 992]
problem bluescreen hello happen pc hang presen...	GRP_24	[43, 2146, 23, 283, 79, 865, 1079, 762, 113, 1...
reset password fygrwuna gomcekzi email please ...	GRP_0	[13, 5, 2436, 2437, 27, 4, 27, 4, 57, 72]

Model1: Single Layer LSTM

- The input data is first padded with the maximum word length and one hot encoding.

```
[302]
X = pad_sequences(DataframeForModelling["TextToSequence"], maxlen = max(DataframeForModelling["WordLength"]))
y = pd.get_dummies(DataframeForModelling['Assignment group']).values
print(X.shape)
print(y.shape)
```

```
(8500, 694)
(8500, 74)
```

- The parameters used for compiling and fitting the model are:
  - Loss: categorical_crossentropy
  - Optimizer: Adam
  - Metrics: accuracy
  - Early stopping : monitor='val_loss', mode='min', patience=6

```
model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=max(DataframeForModelling["WordLength"])))
model.add(LSTM(128, return_sequences=False))
model.add(Dense(74, activation= 'sigmoid'))
```

```
Epoch 1/10
60/60 [==============================] - 146s 2s/step - loss: 1.4670 - accuracy: 0.6311 - val_loss: 1.9907 - val_accuracy: 0.5651
Epoch 2/10
60/60 [==============================] - 144s 2s/step - loss: 1.3277 - accuracy: 0.6578 - val_loss: 1.9685 - val_accuracy: 0.5631
Epoch 3/10
60/60 [==============================] - 143s 2s/step - loss: 1.2061 - accuracy: 0.6845 - val_loss: 1.9782 - val_accuracy: 0.5478
Epoch 4/10
60/60 [==============================] - 143s 2s/step - loss: 1.0980 - accuracy: 0.7101 - val_loss: 1.9706 - val_accuracy: 0.5494
Epoch 5/10
60/60 [==============================] - 145s 2s/step - loss: 0.9848 - accuracy: 0.7373 - val_loss: 1.9672 - val_accuracy: 0.5533
Epoch 6/10
60/60 [==============================] - 143s 2s/step - loss: 0.9275 - accuracy: 0.7561 - val_loss: 1.9627 - val_accuracy: 0.5541
Epoch 7/10
60/60 [==============================] - 143s 2s/step - loss: 0.7923 - accuracy: 0.7928 - val_loss: 1.9649 - val_accuracy: 0.5694
```

Conclusion

- Maximum accuracy that could be achieved in training set is 80 but is making the model very over fit as the validation accuracy is not even reaching 60%.
- In the model we have passed complete X and y variables
- Early stopping is used to work make sure if the accuracy does not improves for more than 6 epochs then the execution must stop.

Model2: Single Layer LSTM with glove embedding

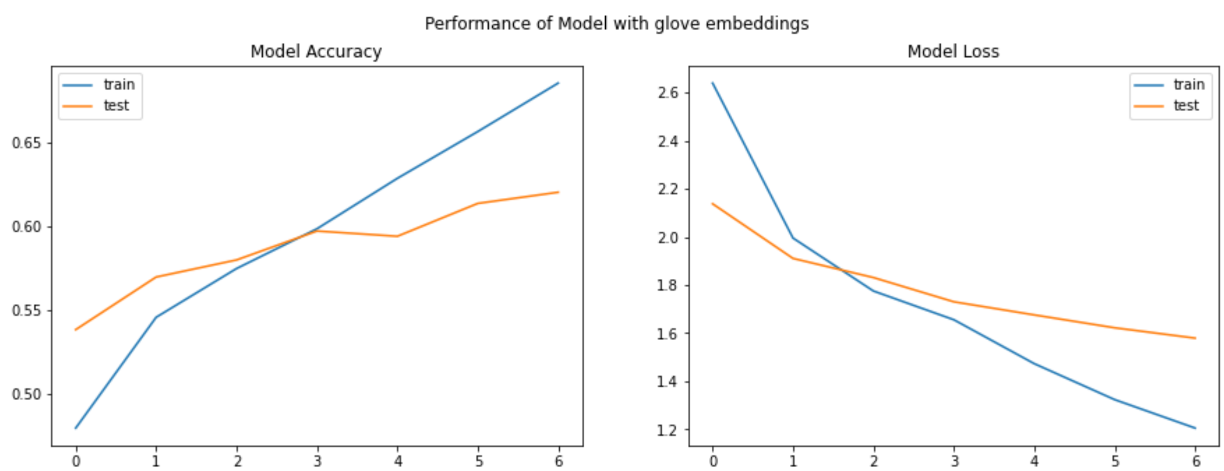
- The model is the same just updated the weight matrix with glove.6B.100d.txt

```
model1 = Sequential()
model1.add(Embedding(vocabulary, embedding_dim, weights = [embedding_matrix]))
model1.add(LSTM(128, return_sequences=False))
model1.add(Dense(74, activation='sigmoid'))
```

- The model is compiled with same parameters.

```
Epoch 1/10
60/60 [==============================] - 143s 2s/step - loss: 2.6394 - accuracy: 0.4792 - val_loss: 2.1376 - val_accuracy: 0.5382
Epoch 2/10
60/60 [==============================] - 143s 2s/step - loss: 1.9953 - accuracy: 0.5456 - val_loss: 1.9107 - val_accuracy: 0.5697
Epoch 3/10
60/60 [==============================] - 143s 2s/step - loss: 1.7752 - accuracy: 0.5748 - val_loss: 1.8311 - val_accuracy: 0.5799
Epoch 4/10
60/60 [==============================] - 143s 2s/step - loss: 1.6558 - accuracy: 0.5986 - val_loss: 1.7304 - val_accuracy: 0.5972
Epoch 5/10
60/60 [==============================] - 146s 2s/step - loss: 1.4733 - accuracy: 0.6288 - val_loss: 1.6758 - val_accuracy: 0.5941
Epoch 6/10
60/60 [==============================] - 145s 2s/step - loss: 1.3238 - accuracy: 0.6568 - val_loss: 1.6223 - val_accuracy: 0.6138
Epoch 7/10
60/60 [==============================] - 146s 2s/step - loss: 1.2056 - accuracy: 0.6859 - val_loss: 1.5794 - val_accuracy: 0.6205
Epoch 00007: early stopping
```

Visualizing LSTM Performance



Conclusion

- LSTM with single layer gives maximum of 56% accuracy in test while LSTM with glove improves it to 62% but the model still is overfit.

Model Optimization

Below are few of the techniques that can be followed to improve model accuracy:

- By adding more dense layer
- By adding dropouts to reduce over fitting
- By using pretrained model like BERT, ULMFIT and Fasttext
- By using glove.6B.300d.txt as updated weight matrix
- By adding Time distributed layer
- By using BidirectionalLSTM
- By using different regularizes while compiling the model.

Observations from Deep Learning Models

1. Below is the base model on which all the modifications have been done

```
model1 = Sequential()
model1.add(Embedding(vocabulary, embedding_dim, weights = [embedding_matrix]))
model1.add(LSTM(128, return_sequences=False))
model1.add(Dense(74, activation='sigmoid'))
```

2. Adding one dense and dropout layer with .5 dropouts certainly reduced the overfitting problem but the accuracy of the model is still to be improved. Result: Train Accuracy 64, Test Accuracy 61.
3. Adding a learning rate in Adam optimizer hardly impacted the accuracy.
4. Adding Glove 300d weight matrix and increasing the number of neurons in the second dense layer, could not improve the accuracy much but kept the overfitting minimal. Result Train Accuracy 66, Test Accuracy 62.
5. Adding a Time Distributed Dense layer and making return_sequence of LSTM layer true, spikes the training accuracy to 82% and test to 65%, thereby making the model extremely overfit.
6. Adding BidirectionalLSTM layer gives the best accuracy, although LSTM takes double the execution time as compared to single LSTM model. Result Train Accuracy 75, Test Accuracy 64.
7. Adding more neurons to LSTM layer, along with multiple dense and dropout layers does not improve the accuracy. Result Train Accuracy 61 Train Accuracy 60.

Modification Scenario	Training Accuracy	Testing Accuracy
Adding one dense and dropout layer with .5 dropouts	64	61
Adding a learning rate in Adam optimizer	63	60
Adding Glove 300d weight matrix and increasing the number of neurons in the second dense layer	66	62
Adding a Time Distributed Dense layer and making return_sequence of LSTM layer true	82	65
Adding BidirectionalLSTM layer gives the best accuracy	75	64
Adding more neurons to LSTM layer, along with multiple dense and dropout layers	61	60

Insights from Pre-Trained Models

ULMFIT

1. Universal Language Model Fine-Tuning(**ULMFIT**) is a transfer learning technique used for NLP tasks.
2. **ULMFIT** incorporates several fine-tuning techniques that could boost performance and that is the major advantage of this model.
3. **ULMFIT** is based on **Inductive Transfer Learning**. In the traditional approach two models are trained separately without either retaining or transferring knowledge from one to the other. An example for transfer learning on the other hand would be to retain knowledge (e.g. weights or features) from training a model 1 and to then utilize this knowledge to train a model 2. In this case, model 1 would be called the source task and model 2 the target task.
4. **ULMFIT** model training and prediction is divided into 3 stage
    - **General-Domain LM Pretraining**
    - **Target Task LM Fine-Tuning**
    - **Target Task Classifier**
5. In a first step, a LM is pretrained on the dataset and the model is able to predict the next word in a sequence. In second step, the knowledge gained in the first step should be utilized for the target task and the Language model is consequently fine-tuned on the data of the target task. In third step the pretrained LM is expanded by two linear blocks so that the final output is a probability distribution.

Fasttext

1. Fasttext is a library for efficient learning of word representations and sentence classification
2. Fasttext supports training continuous bag of words (CBOW) or **Skip-gram models** using negative sampling, softmax or hierarchical softmax loss functions.
3. Fasttext differs in the sense thatword2vec 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.

BERT

1. BERT stands for **Bidirectional Encoder Representations from Transformers** and is based on Transformer architecture. BERT model is generated by stacking encoders from transformers on top of each other.
2. The major advantage of BERT is that it is trained on large corpus of unlabeled text because of which when we start training a model, it starts to pick the deeper and intimate understanding of the langue.
3. One of the main reasons for the good performance of BERT on different NLP tasks was the use of **Semi-Supervised Learning**. This means the model is trained for a specific task that enables it to understand the patterns of the language. After training, the model (BERT) has language processing capabilities that can be used to empower other models that we build and train using supervised learning.
4. BERT is deeply **bidirectional model** as it learns from both left and right side of token's context.
5. BERT is released in two sizes $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$.
    - The BASE model is used to measure the performance of the architecture comparable to architecture. It has 110M parameters
    - The LARGE model produces state-of-the-art results that were reported in the research paper. It has 340M parameters.
6. It was also used in Google search, as of December 2019.It was used in 70 languages.

Observations from Pre-Trained Models

1. The dataset have been tested on 3 Pretrained Models namely ULMFiT, Fasttext and BERT
2. With ULMFiT, we achieved maximum training accuracy as 77% and test accuracy as 67%.
3. With BERT, the maximum training accuracy achieved is 55 and test accuracy as 46%
4. With Fasttext, the maximum training accuracy achieved is 91% and test accuracy as 66%.

Modification Scenario	Training Accuracy	Testing Accuracy
Pretrained UMLFit Model	77	67
Pretrained BERT Model	67	63
BERT with Ktrain model	51	46
Fastext	91	66

Modifications in Dataset

1. Since the data is very imbalanced, even after trying multiple Machine Learning, Deep learning and pretrained models we could not achieve the desired accuracy.
2. So far Bidirectional LSTM and ULMFiT have given the best accuracy.
3. The data has been modified and the accuracy has been measured for **top 5 most frequent groups**.

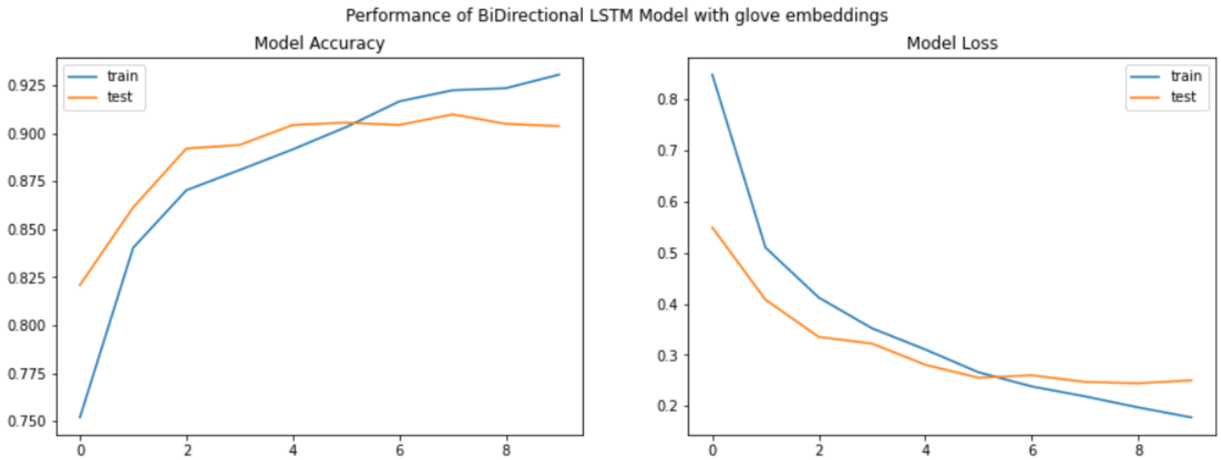
Bidirectional LSTM Performance with Top 5 Groups

Model

```
bi_LSTMmodel = Sequential()
bi_LSTMmodel.add(Embedding(vocabulary_updated, 100, weights = [embedding_matrix_final]))
bi_LSTMmodel.add(Bidirectional(LSTM(128, return_sequences=False, dropout= 0.5)))
bi_LSTMmodel.add(Dense(100, activation="relu"))
bi_LSTMmodel.add(Dropout(0.5))
bi_LSTMmodel.add(Dense(5, activation='softmax'))
```

```
Epoch 1/10
39/39 [==============================] - 91s 2s/step - loss: 0.8486 - accuracy: 0.7520 - val_loss: 0.5494 - val_accuracy: 0.8210
Epoch 2/10
39/39 [==============================] - 89s 2s/step - loss: 0.5099 - accuracy: 0.8404 - val_loss: 0.4079 - val_accuracy: 0.8614
Epoch 3/10
39/39 [==============================] - 90s 2s/step - loss: 0.4122 - accuracy: 0.8704 - val_loss: 0.3350 - val_accuracy: 0.8921
Epoch 4/10
39/39 [==============================] - 90s 2s/step - loss: 0.3520 - accuracy: 0.8809 - val_loss: 0.3220 - val_accuracy: 0.8939
Epoch 5/10
39/39 [==============================] - 89s 2s/step - loss: 0.3106 - accuracy: 0.8917 - val_loss: 0.2806 - val_accuracy: 0.9044
Epoch 6/10
39/39 [==============================] - 90s 2s/step - loss: 0.2659 - accuracy: 0.9032 - val_loss: 0.2551 - val_accuracy: 0.9056
Epoch 7/10
39/39 [==============================] - 89s 2s/step - loss: 0.2382 - accuracy: 0.9166 - val_loss: 0.2600 - val_accuracy: 0.9044
Epoch 8/10
39/39 [==============================] - 90s 2s/step - loss: 0.2187 - accuracy: 0.9224 - val_loss: 0.2470 - val_accuracy: 0.9099
Epoch 9/10
39/39 [==============================] - 89s 2s/step - loss: 0.1971 - accuracy: 0.9235 - val_loss: 0.2442 - val_accuracy: 0.9050
Epoch 10/10
39/39 [==============================] - 89s 2s/step - loss: 0.1775 - accuracy: 0.9306 - val_loss: 0.2502 - val_accuracy: 0.9037
```

Graph



Classification Report

	precision	recall	f1-score	support
0	0.97	0.97	0.97	1180
1	0.68	0.65	0.67	77
2	0.79	0.90	0.84	105
3	0.74	0.92	0.82	196
4	0.42	0.14	0.21	73
accuracy			0.90	1631
macro avg	0.72	0.71	0.70	1631
weighted avg	0.89	0.90	0.89	1631

Prediction

1. For Prediction , randomly few texts have been picked from each group and the same have been tested against the model prediction

```
text_GRP0 = ["skype error"]
text_GRP24 = ["probleme mit bluescreen"]
text_GRP8 = ["abended job scheduler bk hana sid erp dly dp receive"]
text_GRP12 = ["logon server hostname not possible"]
text_GRP9 = ["customer group enhance field receive"]

print("Skype error -- belongs to ", get_Padded_text(text_GRP0))
print("probleme mit bluescreen -- belongs to ", get_Padded_text(text_GRP24))
print("abended job scheduler bk hana sid erp dly dp receive -- belongs to ", get_Padded_text(text_GRP8))
print("logon server hostname not possible -- belongs to ", get_Padded_text(text_GRP12))
print("ustomer group enhance field receive -- belongs to ", get_Padded_text(text_GRP9))
```

```
Skype error -- belongs to  GRP_0
probleme mit bluescreen -- belongs to  GRP_24
bended job scheduler bk hana sid erp dly dp receive -- belongs to  GRP_8
logon server hostname not possible -- belongs to  GRP_12
ustomer group enhance field receive -- belongs to  GRP_0
```

Conclusion

1. From the classification report it can be seen that the group 9 is having least f1 score, and precision. Hence the model is predicting GRP 9 text wrongly as GRP_0
2. The maximum accuracy that can be achieved 93% in training and 90 % in test with slight overfitting.

ULMFit Performance with Top 5 Groups

Model

```
filtermodel_classifier.fit_one_cycle(5, slice(1e-3/(2.6**4),1e-3), moms=(0.8,0.7))
filtermodel_classifier.save('Final_classification')
```

epoch	train_loss	valid_loss	accuracy	time
0	0.228723	0.248876	0.911656	04:00
1	0.170322	0.222499	0.921472	04:10
2	0.177571	0.233571	0.913497	04:09
3	0.146296	0.227278	0.919018	04:25
4	0.127185	0.231056	0.919018	04:30

Graph

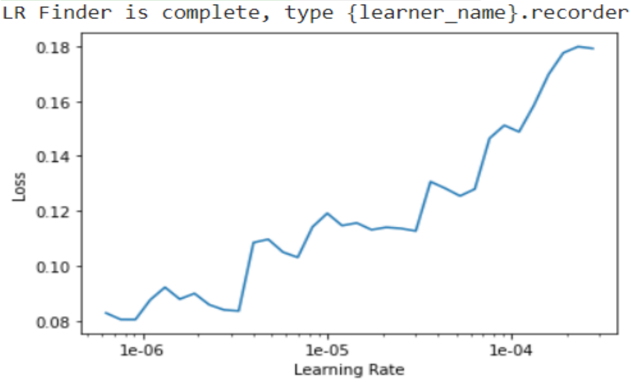
```
filtermodel_classifier.unfreeze()
filtermodel_classifier.lr_find()
filtermodel_classifier.recorder.plot()
```

0.00% [0/2 00:00<00:00]

epoch	train_loss	valid_loss	accuracy	time

60.76% [48/79 02:44<01:46 0.1738]

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.



Classification Report

```
ULMFit_report = metrics.classification_report(targets, predictions)
print(ULMFit_report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1205
           1       0.79      0.67      0.73        79
           2       0.95      0.85      0.90        81
           3       0.94      0.67      0.78       183
           4       0.54      0.76      0.63        82

    accuracy                           0.92      1630
   macro avg       0.84      0.79      0.80      1630
weighted avg       0.93      0.92      0.92      1630
```

Prediction

```
# prediction
text_GRP0 = ["skype error"]
text_GRP24 = ["probleme mit bluescreen"]
text_GRP8 = ["abended job scheduler bk hana sid erp dly dp receive"]
text_GRP12 = ["logon server hostname not possible"]
text_GRP9 = ["customer group enhance field receive"]

print("Skype error -- belongs to ", get_ULMfit_result(text_GRP0))
print("probleme mit bluescreen -- belongs to ", get_ULMfit_result(text_GRP24))
print("abended job scheduler bk hana sid erp dly dp receive -- belongs to ", get_ULMfit_result(text_GRP8))
print("logon server hostname not possible -- belongs to ", get_ULMfit_result(text_GRP12))
print("ustomer group enhance field receive -- belongs to ", get_ULMfit_result(text_GRP9))
```

```
Skype error -- belongs to  GRP_0
probleme mit bluescreen -- belongs to  GRP_24
abended job scheduler bk hana sid erp dly dp receive -- belongs to  GRP_8
logon server hostname not possible -- belongs to  GRP_12
ustomer group enhance field receive -- belongs to  GRP_0
```

Conclusion

1. Just like BiLSTM model, the classification report of ULMFit states group 9 is having least f1 score, and precision. Hence the model is predicting GRP 9 text wrongly as GRP_0

Fasttext Performance with Top 5 Groups

Model

```
fasttext_params_top5 = {
    'input': train_path_top5,
    'lr': .2,
    'lrUpdateRate': 100,
    'thread': 8,
    'epoch': 10,
    'wordNgrams': 2,
    'dim': 1000,
    'loss': 'ova',
    'bucket': 20000,
    'label': "__label__",
    'pretrainedVectors': ""
}
```

```
fasttexttop5model = fasttext.train_supervised(**fasttext_params_top5)
```

Classification Report

```
# Classification report
Fasttext_report = metrics.classification_report(test['Label'], test['prediction'])
print(Fasttext_report)
```

```
               precision    recall  f1-score   support

 __label__GRP_0       0.94      0.99      0.97      1180
__label__GRP_12       0.71      0.48      0.57        77
__label__GRP_24       0.99      0.74      0.85       105
 __label__GRP_8       0.74      0.90      0.81       196
 __label__GRP_9       0.53      0.11      0.18        73

       accuracy                           0.90      1631
      macro avg       0.78      0.64      0.68      1631
   weighted avg       0.89      0.90      0.89      1631
```

Prediction

```
# prediction
text_GRP0 = ["skype error"]
text_GRP24 = ["probleme mit bluescreen"]
text_GRP8 = ["abended job scheduler bk hana sid erp dly dp receive"]
text_GRP12 = ["logon server hostname not possible"]
text_GRP9 = ["customer group enhance field receive"]

print("Skype error -- belongs to ", get_fasttextresult(text_GRP0))
print("probleme mit bluescreen -- belongs to ", get_fasttextresult(text_GRP24))
print("abended job scheduler bk hana sid erp dly dp receive -- belongs to ", get_fasttextresult(text_GRP8))
print("logon server hostname not possible -- belongs to ", get_fasttextresult(text_GRP12))
print("ustomer group enhance field receive -- belongs to ", get_fasttextresult(text_GRP9))
```

```
Skype error -- belongs to  ['__label__GRP_0']
probleme mit bluescreen -- belongs to  ['__label__GRP_24']
abended job scheduler bk hana sid erp dly dp receive -- belongs to  ['__label__GRP_8']
logon server hostname not possible -- belongs to  ['__label__GRP_12']
ustomer group enhance field receive -- belongs to  ['__label__GRP_0']
```

Conclusion

1. From the classification report it can be seen that the group 9 is having least f1 score, and precisi on. Hence the model is predicting GRP 9 text wrongly as GRP_0

BERT Performance with Top 5 Groups

Model

In [26]:
```
model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased",num_labels=len(label_dict))
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when initializing TFBertForSequenceClassif
ication: ['mlm___cls', 'nsp___cls']
- This IS expected if you are initializing TFBertForSequenceClassification from the checkpoint of a model trained
on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertF
orPretraining model).
- This IS NOT expected if you are initializing TFBertForSequenceClassification from the checkpoint of a model that
you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClass
ification model).
Some weights of TFBertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncase
d and are newly initialized: ['classifier', 'dropout_37']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

In [27]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=3e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,name='sparse_categorical_crossentropy')
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
```

In [28]:
```
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
model.summary()
```

```
Model: "tf_bert_for_sequence_classification"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bert (TFBertMainLayer)       multiple                  109482240
_________________________________________________________________
dropout_37 (Dropout)         multiple                  0
_________________________________________________________________
classifier (Dense)           multiple                  3845
=================================================================
Total params: 109,486,085
Trainable params: 109,486,085
Non-trainable params: 0
_________________________________________________________________
```

Classification Report

Classification Report of the model

In [85]:
```
from sklearn.metrics import classification_report
target_names = list(label_dict)
print(classification_report(y_test, y_pred,target_names=target_names))
```

```
              precision    recall  f1-score   support

       GRP_0       0.96      0.98      0.97       249
      GRP_24       0.81      0.87      0.84        15
      GRP_12       0.89      0.57      0.70        14
       GRP_8       0.71      1.00      0.83        30
       GRP_9       0.67      0.21      0.32        19

    accuracy                           0.91       327
   macro avg       0.81      0.72      0.73       327
weighted avg       0.91      0.91      0.90       327
```

Prediction

	Issue Discription	Actual Group	Predicted Group
0	skype error	GRP0	GRP_0
1	probleme mit bluescreen	GRP24	GRP_24
2	abended job scheduler bk hana sid erp dly dp r...	GRP8	GRP_8
3	logon server hostname not possible	GRP12	GRP_12
4	customer group enhance field receive	GRP9	GRP_0

Conclusion

1. From the above implementation of the BERT model we see that performance on top 5 groups of the data-set is close to 96.5% in training and 91.5% in validation & test.
2. We see very less over-fitting this time while using the BERT model.
3. We see prediction of individual groups from random samples that we have picked, has been done correctly. Only prediction related to GROUP 9 is not correct which is consistent with other models.
4. In the BERT model we have used pre-trained "BERT-base-uncased" which contains 110M parameters. We can also try “BERT -large-uncased" pre-trained model and see the performance. Due to lack of hardware that would be necessary to run “BERT -large-uncased" pre-trained model, we did try it.

Final Performance Report

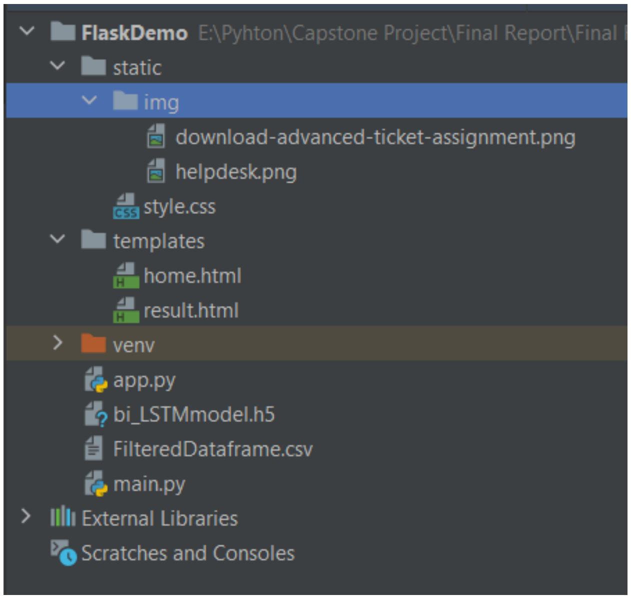
1. Overall all the models have achieved a decent accuracy with minimal overfitting

Model Name	Training Accuracy	Testing Accuracy
BiDirectional LSTM	93	90
ULMFit	94	91
BERT	95	92
FastText	94	90

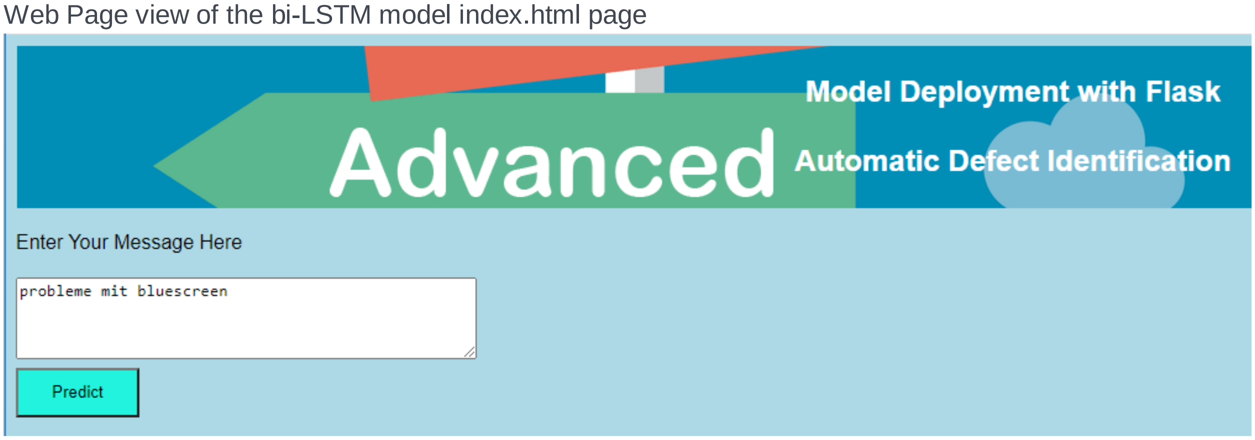
Deployment

1. There are different ways of deploying the model
   - Using Flask as web application
   - Using Heroku
2. The same can also be done from google collab using flask but one cannot see the actual web page, we can only retrieve the results from the hosted environment.
3. There are different IDE for model execution. When we want to run line by line and check the output, it’s idle to go with Jupyter notebooks. When we want to run a chunk of code altogether, Spyder is the better choice and if you want your whole project to look organized, has a lot of files and want to make it look in a structured way, it’s good to go with PyCharm IDE.
4. We have used Flask to deploy the model as web Service.
5. There are three steps to follow while deploying model using Flask:
   - Loading of the saved model(either using pickle or load_method of tensor flow)
   - Redirecting the API to the home page index.html
   - Redirecting the API to predict the result(Assignment group)

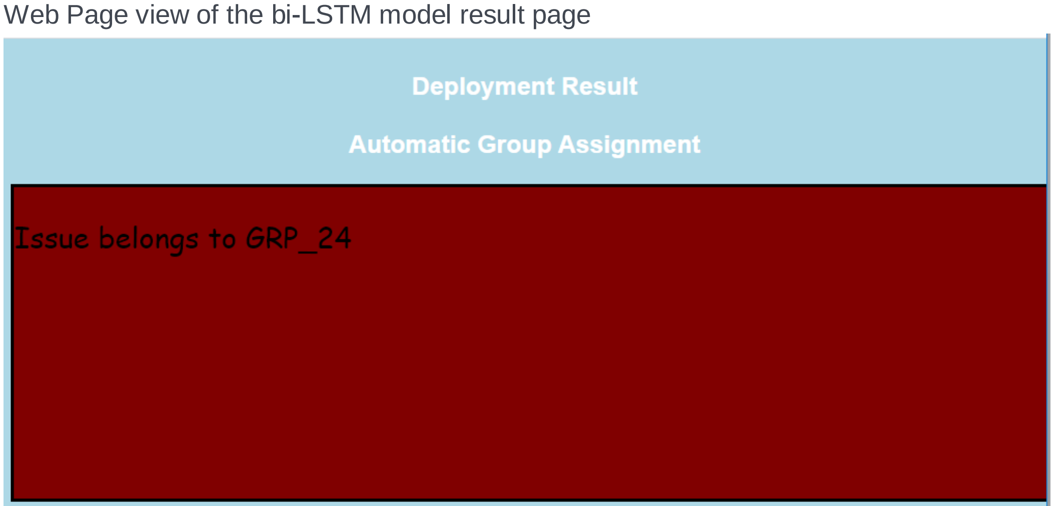
6. Below is the folder structure that is used for deployment



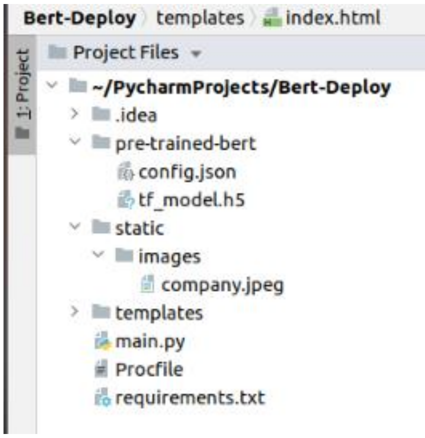
7. Web Page view of the bi-LSTM model index.html page



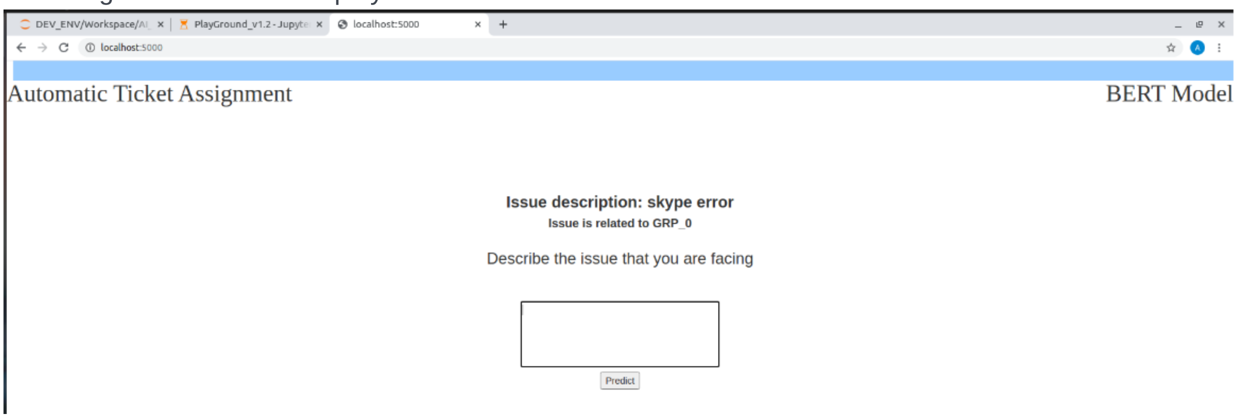
8. Web Page view of the bi-LSTM model result page



9. Folder structure of BERT deployment mode



10. Web Page view of BERT deployment model



Limitations

1. From the preprocessing, it is found that the data is highly imbalanced and contains non-English Text.
2. Google translation was not able to convert all non-English text to English language and hence there remains some discrepancy in the dataset.
3. Due to data being imbalanced, even after trying different ML models, deep learning models, pretrained models, different embeddings, none of the models could achieve a minimum accuracy of 70% in test data. Hence the model performance was tested on top 5 frequent groups.
4. While implementing ULMFit, when the language and classification model was created using TextLMDataBunch and TextClassDataBunch then the model was giving very low accuracy within range of 0-10%.
5. While implementing the BERT model we found that the model was very heavy and took lot of time to complete training. We could run it only on Linux machine that too in local Jupyter instance.
6. When we ran BERT model with ktrain in google collab, the RAM crashes even for 200 records. Hence we could only manage to run BERT model with ktrain only for 100 records.

Closing Reflections

Below are few improvement points that could be taken as further improvement points:

1. After language translation, there should be separate preprocessing done on the non-English texts.
2. There is one frequent word “SID_24” that was holding some meaningful information, could have been restored differently. In current preprocessing we are removing all the digits.
3. Sampling of the data should have been experimented considering the data being highly imbalanced.
4. Different techniques of model deployment can be explored for better results and graphics.
5. Building Machine learning models on android.