

Configuration Manual

MSc Research Project Data Analytics

Sejal Shah Student ID: x18196292

School of Computing National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Sejal Shah
Student ID:	x18196292
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	17 August 2020
Project Title:	Configuration Manual
Word Count:	1031
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	17th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).□Attach a Moodle submission receipt of the online project submission, to
each project (including multiple copies).□You must ensure that you retain a HARD COPY of the project, both for
or□

your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sejal Shah x18196292 MSc in Data Analytics

$17 \ {\rm August} \ 2020$

1 Introduction

This configuration manual would present the system setup, hardware and software requirement. Also, the codes which have been used in programming for implementing below research work:

"Automatic Ticket Assignment using Machine Learning and Deep Learning Techniques"

2 System Configuration

2.1 Hardware

- Processor: Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz 2.30 GHz
- GPU: NVIDIA GeForce GTX 1050Ti
- **RAM:** 8GB
- Storage: 1 TB HDD
- Operting System: Windows 10, 64-bit

2.2 Software

- Google Collaboratory: A free cloud service which provides free service on GPU for running machine learning and deep learning models on its environment. Also, for hyper parameter tuning and evaluation.
- Microsoft Excel 2016: used for data storing and data ploting.
- Jupyter Notebook by Anaconda Distribution: An open source software which has been used for Exploratory Data Analysis and data visualization using python verson 3.6.5 on jupyter notebook in this research.

3 Project Development

Python programming language has been used for implementation of this research study. Firstly, Data Analysis, Data pre-processing stage includes detecting and fixing mojibake, translation, data cleaning, lemmatization. Followed by data preparation and applying classification models. Also, K-fold cross validation has been applied on the machine learning models used in this research study.

In the next section codes develop for this research study has been depicted in detail.

3.1 Data collection

The data has been downloaded from public data repository kaggle 1 and loaded into excel for further exploration.

3.1.1 Importing libraries into colab

0	pip install plotly wordcloud ftfy langdetect iso-language-codes #Installing additional packages
Ċ	Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (4.4.1) Requirement already satisfied: wordcloud in /usr/local/lib/python3.6/dist-packages (1.5.0) Requirement already satisfied: langdetect in /usr/local/lib/python3.6/dist-packages (5.8) Requirement already satisfied: iso-language-codes in /usr/local/lib/python3.6/dist-packages (1.0.0) Requirement already satisfied: iso-language-codes in /usr/local/lib/python3.6/dist-packages (1.0.0) Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (1.1.0) Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3) Requirement already satisfied: numpy=1.6.1 in /usr/local/lib/python3.6/dist-packages (from wordcloud) (1.18.5) Requirement already satisfied: pillow in /usr/local/lib/python3.6/dist-packages (from wordcloud) (7.0.0) Requirement already satisfied: would in /usr/local/lib/python3.6/dist-packages (from ftfy) (0.2.5)

Figure 1: Installation of Libraries

D	import pandas as pd #import pandas
-	import numpy as np #import numpy
	from ftfy import fix encoding, fix text, badness #import ftfy
	from wordcloud import WordCloud, STOPWORDS#import word Colud
	import matplotlib.pyplot as plt #import pyplot
	import seaborn as sns #import seaborn
	import plotly as py #import plotly
	import plotly.graph_objs as go #import plotly graph object
	from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot #import plotly for offline use
	from sklearn.pipeline import Pipeline #import pipeline
	from sklearn.naive_bayes import MultinomialNB #import multinomial Naive bayes
	from sklearn.svm import SVC, LinearSVC #import SVM classes
	from sklearn.tree import DecisionTreeClassifier #import Decision Tree
	from sklearn.ensemble import RandomForestClassifier #import Random Forest
	from sklearn.neighbors import KNeighborsClassifier # import KNN
	from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer # import Countvectorizer
	from sklearn.linear_model import LogisticRegression # import Logistic REgression
	import keras.backend as K # import keras
	from keras.models import Sequential, Model # import models
	from keras.layers import Input, Dropout, Flatten, Dense, Embedding, LSTM, GRU, concatenate, Bidirectional # import layers
	from keras.layers import BatchNormalization, TimeDistributed, Conv1D, MaxPooling1D, GlobalMaxPool1D, GlobalMaxPooling1D # import layers
	from keras.constraints import max_norm, unit_norm # import layers
	<pre>from keras.preprocessing.sequence import pad_sequences # import pad sequences</pre>
	from keras.preprocessing.text import Tokenizer, text_to_word_sequence
	from keras.layers.merge import Concatenate # import concatenate
	from keras.callbacks import EarlyStopping, ModelCheckpoint # import callbacks
	from keras import optimizers, regularizers # import optimizers
	from sklearn.model_selection import train_test_split # import train test split
	from sklearn.metrics import confusion_matrix, classification_report, auc # evalution metrics
	<pre>trom sklearn.metrics import roc_curve, accuracy_score, precision_recall_curve # evalution metrics</pre>
	Import re # regular expression
	import string # string related manupulations
	Trom spacy.lang.en.stop_words import SIOP_workDS # import stop words
	Trom Hitk.tokenize import word_tokenize # import word_tokenizer
	Import nates # Import nates
	miltk.dowiload punkt / # dowiload corpus

Figure 2: Importing required python libraries

¹https://www.kaggle.com/aviskumar/automatic-ticket-assignment-using-nlp

Refer code from Figure 1 and Figure 2 for all the python libraries which have been used in this research.

4 Data Pre-Processing and EDA

The outcomes of predictive models are dependent on the type of applied data preprocessing technique. Generally, the dataset contains noise (irrelevant data) with information. It is important to minimize the noise for obtaining high quality information from the data. In this research, techniques namely text cleaning, fixing mojibake, translating text, lemmatization and data cleaning have been applied. Furthermore, to transform the data one-hot encoding, TF-IDF vectorization and synthetic attribute construction have been applied which enabled model training.

4.1 Initial analysis of data set

✓ Reading file

[] DataFrame = pd.read_excel('data.xlsx') # read excel file into a pandas dataframe

Figure 3: Data loading from excel

•	Unc	lers	tanding	g the str	ucture	of data				
	[]	pri pri	nt('\033 nt('\033	[1m''Numb [1m''Numb	er of Ro er of Co	ows in the da olumns in the	ataset:',DataFrame e dataset:',DataFr	e.shape[0]) # p mame.shape[1])	rint Number of rows #print Number of co	in the dataset lumns in the dataset
	₽	Numi Numi	ber of R ber of C	ows in th olumns in	e datase the dat	t: 8500 aset: 4				
	[]	Data	aFrame.h	ead() #pr	int head	I				
	C⇒			Short des	cription	1		Description	Caller	Assignment group
		0		ŀ	ogin issue	e -verified u	user details.(employee	# & manager na	spxjnwir pjlcoqds	GRP_0
		1			outlook	<pre>/r/n/r/nrecei</pre>	ved from: hmjdrvpb.ko	omuaywn@gmail	hmjdrvpb komuaywn	GRP_0
		2		cant log	g in to vpr	n \r\n\r\nrecei	ved from: eylqgodm.y	bqkwiam@gmail	eylqgodm ybqkwiam	GRP_0
		3	unable to	access hr	_tool page)	unable to a	ccess hr_tool page	xbkucsvz gcpydteq	GRP_0
		4		S	kype erroi	r		skype error	owlgqjme qhcozdfx	GRP_0
	r 1	Date	aEnamo d	occepiba()	#Datace	t Summany				
	LJ	Data	arrame.u	escribe()	#Datase	et Summarry				
	C⇒		Sh	ort descr	iption	Description	Caller	Assignment gro	up	
		со	ount		8492	8499	8500	85	00	
		uni	ique		7481	7817	2950		74	
		t	ор	passwor	d reset	the	bpctwhsn kzqsbmtp	GRF	_0	
		fr	eq		38	56	810	39	76	

Figure 4: Head and summary of the dataset

Figure 3 depicts the data loading i.e reading of excel file into pandas dataframe whereas, Figure 4 represents top 5 rows and summary of the dataset. From summary it can be seen that 'password reset' issues are majorly faced while an inconsistency in Description column can be observed. Top caller is 'bpctwhsn kzqsbmtp' and 'GRP_0' has been assigned most incidents.

Datatypes of Features

[]	DataFrame.dtypes.to_fra	me('Datatypes of	attributes').	T # Fin	d datatypes	of attr	ibute
C⇒	s	Short description	Description	Caller	Assignment	group	
	Datatypes of attributes	object	object	object		object	
[]	DataFrame.info() # Prov	ide dataframe inf	o				
Ċ	<class 'pandas.core.fra<br="">RangeIndex: 8500 entrie Data columns (total 4 c # Column</class>	me.DataFrame'> s, 0 to 8499 olumns): Non-Null Count	Dtype				
	0 Short description 1 Description 2 Caller 3 Assignment group dtypes: object(4) memory usage: 265.8+ KB	8492 non-null 8499 non-null 8500 non-null 8500 non-null	object object object object				

Figure 5: Additional information about dataset

Figure 5 shows that The data set has 4 columns named 'short description', 'description', 'Caller' and 'Assignment group'. The datatype of all features is object.

	luei	ntifica	tion and hand	ing of iviss	ing va	lues in	data							
	[]	DataFr	ame.isnull().sum	().to_frame('	Count o	f missin	g values	').T	# Checking co	ount o	of missing va	alues	for eac	ch column
	₽			Short descr	iption	Descrip	tion Cal	ller	Assignment gro	oup				
		Count	t of missing values		8		1	0		0				
	[]	DataFr DataFr	rame.fillna((str(rame.isnull().sum)), inplace=1 ().to_frame('	f <mark>rue)</mark> # Count o	Imputin f missin	ng with e ng values	mpty ').T	string # Verifying p	resen	ce of missir	ng val	ues	
	C→			Short descr	iption	Descrip	tion Cal	ller	Assignment gro	oup				
		Count	t of missing values		0		0	0		0				
•	Finc	ling ir _{DataFr}	nconsistencies	s in the data	'the'].	head()	# Rows w	hich	contain only '	the'	in 'Descript	tion'	column	
•	Finc	ling ir _{DataFr}	nconsistencies rame[DataFrame.De	s in the data scription == Show	'the']. rt descr	head()	# Rows w Descript	hich	contain only ' Call	'the' Ler A	in 'Descript Assignment gr	tion' roup	column	
•	Finc [] D	DataFr 1049	rame[DataFrame.De	s in the data scription == Show	'the']. rt descr	head() head ()	# Rows w Descript	hich ion the	contain only ' Call soldfnbq uhnbsv	'the' Ler A	in 'Descript Assignment gr GRF	tion' roup	column	
•	Finc [] C	ling in DataFr 1049 1054	nconsistencies name[DataFname.De reset passwords for reset passwords for	s in the data scription == Show for soldfnbq uhn or fygrwuna gon	'the']. 'the']. nt descr bsvqd usi ncekzi usi	head() iption ing pa	# Rows w Descript	the	contain only ' Call soldfnbq uhnbsv fygrwuna gomce	the' Ler A vqd ekzi	in 'Descript ssignment gr GRF GRF	tion' roup P_17 P_17	column	
•	Finc	ling in DataFr 1049 1054 1144	reset passwords f reset passwords f reset passwords f	s in the data scription == Shor for soldfnbq uhn or fygrwuna gon s for wydxnkhf ji	'the']. 'the']. rt descr bsvqd usi ncekzi usi recvta usi	head() ription ing pa ing pa	# Rows w Descript	the the the	contain only ' Call soldfnbq uhnbsv fygrwuna gomce wvdxnkhf jirec	the' Ler A vqd ekzi vta	in 'Descript Issignment gr GRF GRF GRF	roup P_17 P_17 P_17	column	
•	Find []	ling in DataFr 1049 1054 1144 1184	reset passwords reset passwords reset passwords reset passwords	s in the data scription == Show for soldfnbq uhn or fygrwuna gon s for wydxnkhf jii Is for pxyjczdt ki	'the']. rt descr bsvqd usi ncekzi usi recvta usi zsjfpq usi	head() iption ing pa ing pa ing pa	# Rows w Descript	the the the the	contain only ' Call soldfnbq uhnbsv fygrwuna gomce wydxnkhf jirec pxvjczdt kizsji	the' Ler A vqd ekzi vta	in 'Descript ssignment gr GRF GRF GRF GRF	roup P_17 P_17 P_17 P_17	column	
	Find []	ling in DataFr 1049 1054 1144 1184 1292	ame [DataFrame . De reset passwords f reset passwords reset passwords reset passwords reset passwords	s in the data scription == Show for soldfnbq uhn or fygrwuna gon s for wydxnkhf ji is for pxyjczdt ki r cubdsrml znev	'the']. 't descr bsvqd usi ncekzi usi recvta usi zsjfpq usi vqgop usi	head() ing pa ing pa ing pa ing pa ing pa	# Rows w	the the the the the	contain only ' Call soldfnbq uhnbsv fygrwuna gomce wvdxnkhf jirec pxvjczdt kizsji cubdsrml znewqg	lthe' Vqd ekzi Vta fpq gop	in 'Descript ssignment gr GRF GRF GRF GRF GRF	roup P_17 P_17 P_17 P_17 P_17 P_17	column	

Figure 6: Missing value and inconsistency in dataset

There are 8 missing values in 'Short Description', 1 in 'Description' and no missing values are present in 'Caller' and 'Assignment Group' and one of the inconsistencies can be observed from Figure 6.

Detecting and Fixing Mojibake

```
[ ] #False = Mojibake Alert, True = The text is not Mojibake affected
def mojibake_test(data): #Define function for detecting Mojibake
if badness.sequence_weirdness(data): #check if affected
return False #return false if text is affected(Mojibake alert)
try:
    data.encode('windows-1252') #encode if exception is thrown
except UnicodeEncodeError: #Windows-1252 encodable(Mojibake alert)
return False
else: #the text is not affected by Wrong Decoding
return True
```

Figure 7: Function for mojibake identification

For identification of mojibake, function is written in python which can be observed from Figure 7

```
    Translation

   [] from langdetect import detect
           from iso_language_codes import *
          def detect_lang(text):
                tokens = text.split()
                for token in tokens:
if detect(token) != 'en': # Checking for presence of non english text
                   lang = detect(token)
return language(lang)["Name"]
                return language(detect(text))["Name"] # if no other language present returns english
            except:
    return language('en')["Name"]
          DataFrame['Language Before Translation'] = DataFrame["Description"].apply(detect_lang)
         from google.cloud import translate_v2 as translate
    O
          translate_client = translate.Client.from_service_account_json('TicketAssignmentTranslation-4e6e83c85d08.json')
           from time import sleep
          for index , row in DataFrame.iterrows():
    result = translate_client.translate([row['Short description'],row["Description"]], target_language='en')
            Part = translete_file(:translate([fond_short obscription_], fond_best[]
DataFrame["bescription"].iloc[index] = result[0]['translatedText']
DataFrame["Description"].iloc[index] = result[1]['translatedText']
print("Translation completed for row {}".format(index))
sleep(0.5)
```

Figure 8: Function for translation

Figure 8 represents the function written for translation of text. Also, for generation of Google Translation API key refer 2

 $^{^{2}} https://translatepress.com/docs/automatic-translation/generate-google-api-key/$

4.2 Data Cleaning

```
def get_step1_regex_list():
    ''' Removes the mentioned regex text from the document.'''
O
         regexList = []
         # received from: xxxx
         regexList += ['received from:[\ ]?[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}']
regexList += ['^received from:[\ ]?.*']
         # from - mail address or names
         regexList += ['^from:[\ ]?.*']
         # to - mail addresses
         regexList += ['^to:[\
# cc - mail addresses
                                  ]?.*']
         regexList += ['^cc:[\ ]?.*']
         # sent: xxx
         regexList += ['^sent:[\ ]?.*']
         # email: xxxx
         regexList += ['^email:[\ ]?.*']
         # name: xxxx
         regexList += ['^name:[\ ]?.*']
         # customer number: xxxx
         regexList += ['^customer number:[\ ]?.*']
         # browser: xxxx
         regexList += ['^browser:[\ ]?.*']
         # language: xxxx
regexList += ['^language:[\ ]?.*']
         # user: xxxx
         regexList += ['^user:[\ ]?.*']
         # user id: xxxx
         regexList += ['^user id:[\ ]?.*']
         # login id: xxx>
         regexList += ['^login id:[\ ]?.*']
         # computer: xxxx
         regexList += ['^computer:[\ ]?.*']
         # computer name: xxxx
         regexList += ['^computer name:[\ ]?.*']
         # service tag: xxxx
regexList += ['^service tag:[\ ]?.*']
         # service tag: xxxx
regexList += ['^model details:[\ ]?.*']
         # images
         regexList += ['\[cid:(.*)\]']
         # telephone: xxxx
         regexList += ['^telephone[:\ ].*']
         # id: xxxx
regexList += ['^id[:\].*']
         # email
         regexList += ['[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}']
         # timestamp 12/12/2020 12:12:12
regexList += ['\d{2}/\d{2}/\d{4}[\ ]?\d{2}:\d{2}:\d{2}']
# timestamp 10/25/2016 1:47 pm
         regexList += ['\d{2}/\d{2}/\d{4}[\ ]?\d{1,2}:\d{1,2}[\ ]?(am|pm)']
          # date 10/25/2016 or 10-25-2010
         regexList += ['d{1,2}[/-]?d{1,2}[/-]?d{4}']
         # time 01:12 am
         regexList += ['\d{1,2}:\d{1,2}[\ ]?(am|pm)']
         # time 12:11:23
         regexList += ['\d{2}:\d{2}:\d{2}']
         # subject
         regexList += ['subject:']
         #summary
         regexList += ['summary:']
          # forward
         regexList += ['(fwd:|fw:|re:)']
         # <mailto: >
         regexList += ['\<mailto.*\>']
         # hostname_xxxx
         # regexList += ['hostname_\d*']
         # ticket number
         regexList += ['ticket_no\d*']
regexList += ['tck no[\.\d]*']
         return regexList
```

Figure 9: Regular Expression

Data Cleaning

```
[ ] DataFrame_copy = DataFrame #copy of original dataframe
  def apply_regex(text, regexList): # Apply regex to the text using regexList
    ''' Apply list of regex expressions on to text '''
    for regex_ in regexList:
        text = re.sub(regex_, '', text)
    return text # Return clean text
```

Figure 10: Applied regular expression

In this research study of incident reporting there is no need to concern about emails, or the sender, the main cognizance of language processing here is to concentrate on content and subject of incident ticket, the message it is trying to deliver, that being stated email id's, to, cc, bcc, problem, footers, computer names, ip addresses, numbers all are inappropriate and may be removed using regular expressions expressions which is represented by Figure 9 and Figure 10.

Removal of stop words

Removal of Stop Words

"Stop Words" is a technical term to represent high frequency words which are widely used in a language and corpus which add little or no value in vocabulary. We need to remove one's words before training any of our models. Most of the NLP libraries obtainable has a well-known set of stop words defined, in our case we are going to use "NLTK's" stop words.



Figure 11: Removing of stop words

Stop words removal is necessary before any of the model Figure 11 explains the function return for stop words.

Data Normalization



Figure 12: Lemmatization code

Data Normalization has been performed using lemmatization to get root word, code has shown in Figure 12. 3

Inserting word count and length of sentences [] sdlen = DataFrame['Clean Short Description'].apply(len) #Calculate length of Short Description DataFrame.insert(5,'short_desc_length',sdlen) #insert length into column sdword_c = DataFrame['Clean Short Description'].apply(lambda x: len(str(x).split())) #Calculate Word Count DataFrame.insert(6,'short_desc_word_c',sdword_c) #insert word count into DataFrame desclen = DataFrame['Clean Description'].apply(len) #Calculate length of Description DataFrame.insert(8,'Description_length',desclen)#insert length into column descword_c = DataFrame['Clean Description'].apply(lambda x: len(str(x).split())) #Calculate Word Count DataFrame.insert(9, 'Description_word_c', descword_c) #insert word count into DataFrameDataFrame.head()

Figure 13: Word count and length

Merging cleaned short description and description columns.

[] DataFrame.insert(loc=10,column='Training_data', allow_duplicates=True, value=list(DataFrame['Clean Short Description'].str.strip() + ' ' + DataFrame['Clean Description']

Figure 14: Short description and description merged

³https://www.tutorialspoint.com/natural_language_toolkit/natural_language_ toolkit_stemming_lemmatization.htm#:~:text=Difference%20between%20Stemming%20%26% 20Lemmatization&text=In%20simple%20words%2C%20stemming%20technique,always%20get%20a% 20valid%20word

Creating a Target column from 'Assignment group'

💿 DataFrame['Target'] = DataFrame['Assignment group'].astype('category').cat.codes # Converting assignment groups to their corresponding numerical codes

Figure 15: Target column for training

Adding word count and length, merging of short description and description column and creating target column from 'Assignment_ group' has been explained in Figure 13, Figure 14 and Figure 15. Also, Dataset summary after pre-processing has been shown in Figure 16.



DataFrame.info()

C→	<clas< th=""><th>ss 'pandas.core.frame.DataFra</th><th>me'></th><th></th><th></th></clas<>	ss 'pandas.core.frame.DataFra	me'>		
	Range	eIndex: 8500 entries, 0 to 84	99		
	Data	columns (total 13 columns):			
	#	Column	Non-I	Null Count	Dtype
	0	Short description	8500	non-null	object
	1	Description	8500	non-null	object
	2	Caller	8500	non-null	object
	3	Assignment group	8500	non-null	object
	4	Language Before Translation	8500	non-null	object
	5	short_desc_length	8500	non-null	int64
	6	short_desc_word_c	8500	non-null	int64
	7	Clean Description	8500	non-null	object
	8	Description_length	8500	non-null	int64
	9	Description_word_c	8500	non-null	int64
	10	Training_data	8500	non-null	object
	11	Clean Short Description	8500	non-null	object
	12	Target	8500	non-null	int8
	dtype	es: int64(4), int8(1), object	(8)		
	memor	ry usage: 805.3+ KB			

Figure 16: Dataset summary after pre-processing

Feature selection

PCA and Resampling of dataset



Figure 17: Resampling and PCA

Principal Component Analysis (PCA)



Figure 18: Principal component Analysis

The use of resampling techniques has been depicted in Figure 17 which at first undersamples and then oversamples the data set to handle the class imbalance. Figure 18 depicts the use of PCA with 95% variance which has reduced the dimensions of data set from 10736 to 3468.

5 Code used for Machine Learning and Deep Learning models

This section consist of code used for modelling which has been done using scikit libraries. Models such as SVM, Random Forest classifier, K-Nearest Neighbours, ANN and BLSTM have been applied.

5.1 Data Splitting

Data has been split into train and test i.e 80:20 ratio. Also, K- fold cross validation has been used for generating efficient and better results (K. S. Manjunatha and Guru; 2019).

•	Mo	odel Building
		Building a classifier model/network architecture & training Use of Machine Learning as well as Deep Learning for classification
-	Tra	in test split
	[]	X_train, X_test, y_train, y_test = train_test_split(X_over_sample, y_over_sample, test_size=0.2, random_state=4)
	[]	print('Training data shape :',X_train.shape, y_train.shape) #print shape of training data print('Test data shape :',X_test.shape, y_test.shape)#print shape of test data
	₽	Training data shape : (39131, 3468) (39131,) Test data shape : (9783, 3468) (9783,)

Figure 19: Train- test split

Train and test split for model training has been described by Figure 19.

- Fu	inction for training and evalution of models
-	
C	%matplotlib inline
	from scipy.stats import sem, t
	import matplotib.pyplot as plt
	def classify(model, X_train, X_test, y_train, y_test, NeuralNet-False, callback - None,
	val_data=None, pipelined = True, epochs=10, ConfidenceintervalNN = False):
	if pipelined == True:
	<pre>model = Pipeline([('model',model)])</pre>
	if NeuralNet True:
	<pre>model_H = model.fit(X_train, y_train, epochs = epochs, batch_size = 500, verbose = 1, validation_data-val_data, callbi v_train_ended_ta</pre>
	y train pred = np.argmax(y train pred, axis=1)
	<pre>y_test_pred = model.predict(X_test)</pre>
	<pre>y_test_pred = np.argmax(y_test_pred, axis=1)</pre>
	import matplotlib.pyplot as plt #import pyplot
	# Plot the loss function
	<pre>fig, ax = plt.subplots(1, 1, figsize=(6,4)) #Create subplots</pre>
	ax.plot(np.sqrt(model_H.history['loss']), 'b', label='train') #Plot training loss
	ax, set xlabel(r'Epoch', fontsize=14) #set xlabel
	ax.set_ylabel(r'Loss', fontsize=14) #set ylabel
	ax.legend() #set legend
	# Plot the accuracy
	fig, ax = plt.subplots(1, 1, figsize=(6,4)) #Create subplots
	ax.plot(np.sqrt(model_H.history['accuracy']), 'b', label='train') #Plot training accuracy
	ax.plot(np.sqrt(model_H.history['val_accuracy']), 'r',label='val') #Plot validation accuracy
	ax.set_Nlabel(P'Epoch', fontsize=14) #set vlabel
	ax.legend() #set legend
	ax.tick_params(labelsize=20) #set label size
	if ConfidenceintervalNN == True:
	<pre>n = len(model_H.history['val_accuracy'])</pre>
	<pre>m = mean(model_H.history['val_accuracy'])</pre>
	std_err = sem(model_H.history['val_accuracy'])
	$n = s(0_1 + n - h)$
	end = m + h
	# plot scores
	<pre>#pit.nist(model_H.nistory[val_accuracy]) #pit.show()</pre>
	print('\033[1m''%.1f confidence interval %.1f%% and %.1f%%''\033[1m' % (0.95*100, start*100, end*100))
	<pre>#print('\033[1m''The range of confidence interval is:''\033[1m',start,end)</pre>
	else:
	model.fit(X_train,y_train)
	y_train_pred = model.predict(X_train)
	<pre>y_test_pred = model.predict(X_test) print(')033[im'')Used (lassifier'')033[im' model)</pre>
	print('->'*63)
	<pre>training_acc = accuracy_score(y_train,y_train_pred)</pre>
	<pre>print('\033[im''Training Accuracy of Classifier :{:.2f}'.format(training_acc)) #for accuracy score tots acc accuracy score to tots accuracy score</pre>
	<pre>print('\033im''Test Accuracy of Classifier :{:.2f}'.format(test acc)) #for accuracy score</pre>
	print('-'*30)
	<pre>print('\033[Im''Confusion Matrix\n',confusion_matrix(y_test,y_test_pred)) #for confusion matrix print('\033[Im''Confusion Matrix\n',confusion_matrix(y_test,y_test_pred))</pre>
	print(' 1.30) print(' 1.03311m''\n Classification Report\n', classification report(y test, y test pred,zero division=1)) #for classification
	print('->'*63)

Figure 20: Code for training and evaluation of models

Figure 20 shows the all the function used in this research for training and evaluation of the model.

•	Imp	porting Libraries
	[]	from sklearn.pipeline import Pipeline #import pipeline from sklearn.naive_bayes import MultinomialNB #import multinomial Naive bayes from sklearn.svm import SVC, LinearSVC #import SVM classes from sklearn.tree import DecisionTreeClassifier #import Decision Tree from sklearn.neighbors import RANdomForestClassifier #import RANdom Forest from sklearn.neighbors import KNeighborsClassifier #import KNN from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer #import Countvectorizer from sklearn.linear_model import LogisticRegression #import Logistic Regression

Figure 21: importing Libraries for models

Libraries which been imported for running the classification models are represented in Figure 21.

5.2 K-Nearest Neighbours

K-Nearest Neighbour has been applied using default as well as optimized parameters.

```
    KNeighborsClassifier

   [ ] knn=KNeighborsClassifier() #K-neighbors classifier
       classify(knn, X_train, X_test, y_train, y_test) #classify using KNN
   □→ Used Classifier: Pipeline(memory=None,
               Test Accuracy of Classifier :0.93
                       _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

      Confusion Matrix

      [[ 43 0 2 ... 0 0

      [ 0 128 0 ... 0 0

      2 99 ... 0 0

       Confusion Matrix
                          0 0
                                   0]
                                  0]
                                  0
        [
           0
              0
                  0 ... 131
                              0
                                  0]
              2 6 ... 0 61 3]
0 3 ... 0 6 56]]
          0
2
        Γ
```

Figure 22: KNN with default parameters

The code has been shown in Figure 22 for KNN with default parameters. Also, confusion matrix has been described in same.

- Machine Learning Models
- Experiment 1 KNeighborsClassifier

```
[ ] #List Hyperparameters that we want to tune.
n_neighbors = list(range(3,7,2))
#Convert to dictionary
param_grid = dict(n_neighbors=n_neighbors)
#Create new KNN object
knn_2 = KNeighborsClassifier()
kfold=KFold(n_splits=10, shuffle=True, random_state=1)#LOOCV Kfold
grid_knn= GridSearchCV(estimator=knn_2, param_grid=param_grid, cv=kfold, n_jobs=3)
#Fit the model
best_model = grid_knn.fit(X_train, y_train)
#Print The value of best Hyperparameters
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
```

Figure 23: KNN - GridSearchCV and 10 fold cross validation

Refer Figure 23 for KNN with GridSearchCV optimization and 10 fold cross validation.



Figure 24: Model training using tuned hyper parameters

5.3 Random Forest Classifier

```
Random Forest Classifier
   r f = RandomForestClassifier() #random forest classifier
     classify(r_f, X_train, X_test, y_train, y_test) #classify
□→ Used Classifier: Pipeline(memory=None,
             steps=[('model',
                     RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                           class weight=None, criterion='gini',
                                           max_depth=None, max_features='auto',
                                           max_leaf_nodes=None, max_samples=None,
                                           min_impurity_decrease=0.0,
                                           min_impurity_split=None,
                                           min samples leaf=1, min samples split=2,
                                           min_weight_fraction_leaf=0.0,
                                           n_estimators=100, n_jobs=None,
                                           oob_score=False, random_state=None,
                                           verbose=0, warm_start=False))],
             verbose=False)
     Training Accuracy of Classifier :0.96
    Test Accuracy of Classifier :0.95
    Confusion Matrix
     \begin{bmatrix} [101 & 0 & 1 & \dots & 0 & 0 \\ [ & 0 & 128 & 0 & \dots & 0 & 0 \end{bmatrix}
                        0 0 0]
            0 95 ... 0 0 16]
     [ 0
        0
            Ø
               0 ... 131
                           0
     [
                               01
        2
            3
                       0
                           66 40]
                1 ...
     ſ
                           0 104]]
               0 ...
            0
                        0
     [ 3
```

Figure 25: Random Forest with default parameters

The code has been describe in Figure 25 for Random Forest with default parameters and Figure 26 for Random Forest using GridSearchCV with 10 fold cross validation.

```
[] RFR = RandomForestClassifier()#Random Forest Regressor
kfold=KFold(n_splits=10, shuffle=True, random_state=1)#LOOCV Kfold
grid = RandomizedSearchCV(RFR, param_distributions=random_grid, n_iter = 2, cv =
grid_result = grid.fit(X_train, y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
[. Fitting 2 folds for each of 2 candidates, totalling 4 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done 2 out of 4 | elapsed: 14.9min remaining: 14.9min
[Parallel(n_jobs=3)]: Done 4 out of 4 | elapsed: 27.0min finished
Best: 0.921469 using {'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 40, 'bootstrap': False}
Test Score using best parameters 0.940100173770827
```

Figure 26: Random Forest- RandomizedSearchCV and 10 fold cross validation



Figure 27: Model training using tuned hyper parameters

Figure 27 shows code used for model training using the tuned hyper parameters.

5.4 Support Vector Machine

SVM has been applied with default refer Figure 28, GridSearchCV and 10 fold cross validation refer Figure 29 and finally model training with obtained hyper parameter Figure 30.

•	SVM Classifier				
	[]	sv = SVC() #SVM with RBF kernel classify(sv, X_train, X_test, y_train, y_test) #classify			
	¢	<pre>Used Classifier: Pipeline(memory=None, steps=[('model', SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False))], verbose=False) ->->->>->>->>>>>>>>>>>>>>>>>>>>>>>>>></pre>			
		Confusion Matrix [[96 0 0 0 0 0] [0 116 0 0 0 0] [4 0 90 0 0 19] [0 0 0 131 0 0] [5 3 1 0 44 59] [3 0 0 0 0 129]]			

Figure 28: Support Vector Machine with default parameters

Experiment 3 Support Vector Classifier

```
lsv = SVC()
param_grid = {'C': [1, 100, 1000]}
kfold=KFold(n_splits=10, shuffle=True, random_state=1)
grid = GridSearchCV(estimator=lsv, param_grid=param_grid, cv=kfold, n_jobs=3)
grid_result = grid.fit(X_train, y_train)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

Figure 29: SVM - GridSearchCV and 10 fold cross validation



Figure 30: Model training using tuned hyper parameter

Importing libraries for Neural networks

•	NEURAL NETWORK MODELS			
•	Importing libraries			
	[] import keras.backed as K #Import keras from keras.backed as K #Import keras from keras.layers import Input, Dropout, Flatten, Dense, Embedding, LSTM, GRU, concatenate, Bidirectional #import layers from keras.layers import BactNormalization, TimeDistributed, ComVD, MaxPolingID, GlobalMaxPoolingID #import layers from keras.preprocessing.sequence import pad.sequences #import pad sequences from keras.layers.import EarlyStopping, ModelCheckGuine #import callbacks from keras.claback import EarlyStopping, ModelCheckGuine #import callbacks from keras.claback import EarlyStopping, ModelCheckGuine #import callbacks from keras.import EarlyStopping, ModelCheckGuine #import callbacks from keras.import isofping.metriks.classification_report, acc @wallution metrics from sklaam.metrics import roc_urve, precision_recall.curve #equalition metrics			

Figure 31: Important libraries for neural network

Refer figure 31 for importing all libraries

5.5 Artificial Neural Network(ANN)

- Deep Neural Network



Figure 32: ANN with five hidden dense layer

Refer code from Figure 32 for implementing ANN with fixed 5 hidden layers and fixed number of neurons.

Exp	eriment 4 Neural Network
[]	<pre>def build_DNN(input_shape, neurons, dropout, layers): # Basic neural network mode """</pre>
	Basic deep neural network model using keras.
	Parameters:
	input_shape : Shape of the input data
	neurons : Number of neurons for the input Dense layer
	dropout : dropout for the neural network
	layers : Number of the layers for the neural network
	Returns:
	model : deep neural network model
	<pre>model = Sequential()</pre>
	<pre>model.add(Dense(neurons, input_dim=input_shape, activation='relu'))</pre>
	<pre>model.add(Dropout(dropout))</pre>
	for i in range(0,layers):
	<pre>model.add(Dense(neurons, activation='relu'))</pre>
	<pre>model.add(Dropout(dropout))</pre>
	model.add(BatchNormalization())
	neurons = int(neurons/2)
	<pre>model.add(Dense(74, activation = 'softmax'))</pre>
	<pre>model.compile(loss='sparse_categorical_crossentropy', metrics = ['accuracy'],</pre>
	<pre>print(model.summary())</pre>
	return model

Figure 33: ANN with four hidden dense layer

Refer code from Figure 33 for implementing ANN with 4 hidden layers or dynamic number of hidden layers with descending number of neurons per layer, keras library has been used for implementation of ANN, 'Adam' as an optimizer and 'Softmax' activation function (Molino et al.; 2018).

5.6 Bidirectional Long Short Term Memory (BLSTM)



Figure 34: BLSTM using Five hidden layers

```
Experiment 5 LSTM
[] def get_model():
    model = Sequential()
    #model.add(Input(shape=(maxlen,)))
    model.add(Embedding(num_words + 1, embedding_size, input_length=X_train_token.shape[1], weights=[embedding_matrix], trainable=True))
    model.add(Bidirectional(LSTM(128, return_sequences=True)))
    #model.add(LSTM(128, recurrent_dropout=0.2, return_sequences=True))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(128, activation='softmax')))
    model.add(Dense(1ass_count, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    print(model.summary())
    return model
```

Figure 35: BLSTM using four hidden layers

Refer Figure 34 and Figure 35 for code of BLSTM⁴.

Note: All the required notebook settings have already been configured in 'Project_x18196292.ipynb' file. If execution is required then import the notebbook file in google colab and execute. Also, all intermediate supporting files have also been included in code artefacts.

References

K. S. Manjunatha, H. A. and Guru, D. S. (2019). Data Analytics and Learning, Vol. 43, Springer Singapore. URL: http://link.springer.com/10.1007/978-981-13-2514-4

⁴https://medium.com/@jonathan_hui/nlp-word-embedding-glove-5e7f523999f6

Molino, P., Zheng, H. and Wang, Y. C. (2018). COTA: Improving the speed and accuracy of customer support through ranking and deep networks, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 586–595.