Problem Statement

You need to build a model that is able to classify customer complaints based on the products/services. By doing so, you can segregate these tickets into their relevant categories and, therefore, help in the quick resolution of the issue.

You will be doing topic modelling on the **.json** data provided by the company. Since this data is not labelled, you need to apply NMF to analyse patterns and classify tickets into the following five clusters based on their products/services:

- Credit card / Prepaid card
- Bank account services
- Theft/Dispute reporting
- Mortgages/loans
- Others

With the help of topic modelling, you will be able to map each ticket onto its respective department/category. You can then use this data to train any supervised model such as logistic regression, decision tree or random forest. Using this trained model, you can classify any new customer complaint support ticket into its relevant department.

Group Facilitator : Name: Ashwin Kumar H Email ID: ashwinkpes@gmail.com Phone No:919886757058

Team Member Detail: Name: Sricharan H Email ID: sricharanh89@gmail.com Phone no: 919148562249

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mou
```

Pipelines that needs to be performed:

You need to perform the following eight major tasks to complete the assignment:

1. Data loading
2. Text preprocessing

3. Exploratory data analysis (EDA)
4. Feature extraction
5. Topic modelling
6. Model building using supervised learning
7. Model training and evaluation
8. Model inference

Install Swifter

```
pip install swifter
```

```
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: fsspec>=0.6.0 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: partd>=0.3.10 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: numpy>=1.18 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.9
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/python3.9/d
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.9
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: backcall in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: pickleshare in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: pygments in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: decorator in /usr/local/lib/python3.9/dist-packages (
```

```
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.9/dist
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/dist-packag
```

Install pip install textblob

```
pip install textblob
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: textblob in /usr/local/lib/python3.9/dist-packages (0.17
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.9/dist-packages (fro
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from nlt
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from n
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from nl
```

Importing the necessary libraries

```
import json
import numpy as np
import pandas as pd
import re, nltk, spacy, string
import en_core_web_sm
nlp = en_core_web_sm.load()
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from wordcloud import WordCloud, STOPWORDS
from collections import Counter
import swifter
from sklearn.feature_extraction.text import TfidfTransformer
```

```
from plotly.offline import plot
import plotly.graph_objects as go
import plotly.express as px

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from pprint import pprint


from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score
from pprint import pprint
```

Loading the data

The data is in JSON format and we need to convert it to a dataframe.

```
# Opening JSON file
f = open('/content/drive/MyDrive/ToDelete/complaints-2021-05-14_08_16.json',)

# returns JSON object as
# a dictionary
data = json.load(f)
df=pd.json_normalize(data)
```

Data preparation

```
# Inspect the dataframe to understand the given data.
df.head(3)
```

	_index	_type	_id	_score	_source.tags	_source.zip_code	_source.compla
0	complaint-public-v2	complaint	3211475	0.0	None	90301	3

```
#print the column names
list(df.columns)
```

```
['_index',
 '_type',
 '_id',
 '_score',
 '_source.tags',
 '_source.zip_code',
 '_source.complaint_id',
 '_source.issue',
 '_source.date_received',
 '_source.state',
 '_source.consumer_disputed',
 '_source.product',
 '_source.company_response',
 '_source.company',
 '_source.submitted_via',
 '_source.date_sent_to_company',
 '_source.company_public_response',
 '_source.sub_product',
 '_source.timely',
 '_source.complaint_what_happened',
 '_source.sub_issue',
 '_source.consumer_consent_provided']
```

```
#Assign new column names
df.columns = ['index', 'type', 'id', 'score', 'tags', 'zip_code','complaint_id', 'issue', 'da
       'state', 'consumer_disputed', 'product','company_response', 'company', 'submitted_via'
       'date_sent_to_company', 'company_public_response','sub_product', 'timely',
       'complaint_what_happened', 'sub_issue','consumer_consent_provided']
```

```
#Assign nan in place of blanks in the complaints column
df[df.loc[:, 'complaint_what_happened'] == ''] = np.nan
```

```
#Remove all rows where complaints column is nan
df = df[~df['complaint_what_happened'].isnull()]
```

```
# Make complaint_what_happened as tring
df['complaint_what_happened'] = df['complaint_what_happened'].astype(str)
```

Prepare the text for topic modeling

Once you have removed all the blank complaints, you need to:

- Make the text lowercase
- Remove text in square brackets
- Remove punctuation
- Remove words containing numbers

Once you have done these cleaning operations you need to perform the following:

- Lemmatize the texts
- Extract the POS tags of the lemmatized text and remove all the words which have tags other than NN[tag == "NN"].

```
# Write your function here to clean the text and remove all the unnecessary elements.
def clean_text(sent):
    sent = sent.lower() # Text to lowercase
    pattern = '[^\w\s]' # Removing punctuation
    sent = re.sub(pattern, '', sent)
    pattern = '\w*\d\w*' # Removing words with numbers in between
    sent = re.sub(pattern, '', sent)
    return sent

df_clean = pd.DataFrame(df['complaint_what_happened'].apply(clean_text))
df_clean.head(5)
```

	complaint_what_happened
1	good morning my name is xxxx xxxx and i apprec...
2	i upgraded my xxxx xxxx card in and was told ...
10	chase card was reported on however fraudulent...
11	on while trying to book a xxxx xxxx ticket ...
14	my grand son give me check for i deposit it i...

```
#Write your function to Lemmatize the texts
def lemmmatize_text(text):
    sent = []
    doc = nlp(text)
    for token in doc:
        sent.append(token.lemma_)
    return " ".join(sent)

#Create a dataframe('df_clean') that will have only the complaints and the lemmatized complai
df_clean['complaint_lemmatized'] = df_clean['complaint_what_happened'].apply(lemmmatize_text)

df_clean.head(3)
```

	complaint_what_happened	complaint_lemmati
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I appre
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent

```
from textblob import TextBlob
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

```
pip install wordcloud
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: wordcloud in /usr/local/lib/python3.9/dist-packages (1.8
Requirement already satisfied: pillow in /usr/local/lib/python3.9/dist-packages (from w
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (fr
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/d
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-package
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import pickle
```

```
#Write your function to extract the POS tags
def get_POS_tags(text):
    sent = []
    blob = TextBlob(text)
    sent = [word for (word,tag) in blob.tags if tag=='NN']
    return " ".join(sent)

df_clean['complaint_POS_removed'] = df_clean['complaint_lemmatized'].apply(get_POS_tags)
```

```
#df_clean["complaint_POS_removed"] =  #this column should contain lemmatized text with all th


#The clean dataframe should now contain the raw complaint, lemmatized complaint and the compl
df_clean.head(3)
```

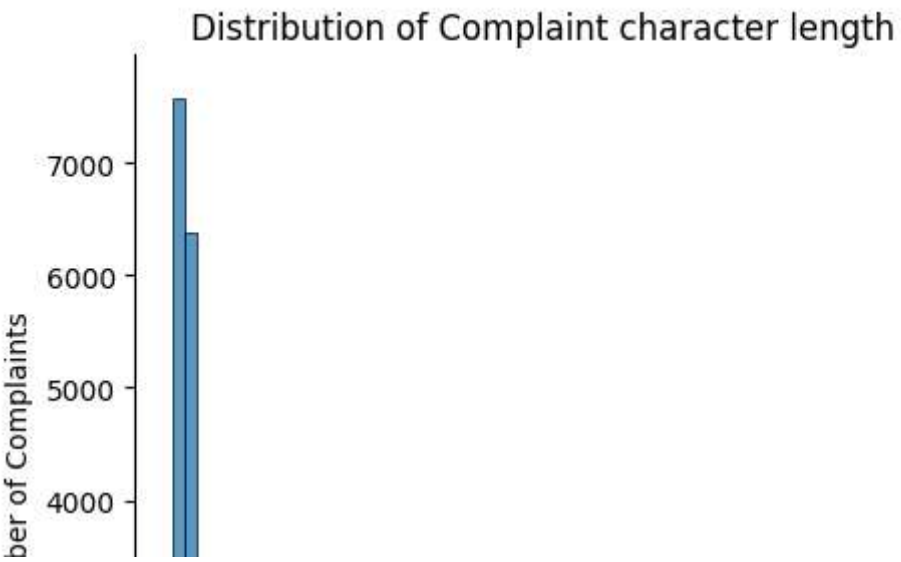
	complaint_what_happened	complaint_lemmatized	complai
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent u

Exploratory data analysis to get familiar with the data.

Write the code in this task to perform the following:

- Visualise the data according to the 'Complaint' character length
- Using a word cloud find the top 40 words by frequency among all the articles after processing the text
- Find the top unigrams,bigrams and trigrams by frequency among all the complaints after processing the text. ‘

```
# Write your code here to visualise the data according to the 'Complaint' character length
char_len = [len(each_sent) for each_sent in df_clean['complaint_POS_removed']]

sns.displot(char_len, kind='hist', bins=60)
plt.xlabel("Complaint character length")
plt.ylabel("Total number of Complaints")
plt.title("Distribution of Complaint character length")
plt.show()
```



Find the top 40 words by frequency among all the articles after processing the text.

```
#Using a word cloud find the top 40 words by frequency among all the articles after processir
stopwords = set(STOPWORDS)
wordcloud = WordCloud(max_font_size=60, max_words=40,
                      background_color="white", random_state=100,
                      stopwords=stopwords).generate(str(df_clean['complaint_POS_removed']))
plt.figure(figsize=[12,12])
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```


```
#Removing -PRON- from the text corpus
df_clean['Complaint_clean'] = df_clean['complaint_POS_removed'].str.replace('-PRON-', '')
```

Find the top unigrams,bigrams and trigrams by frequency among all the complaints after processing the text.

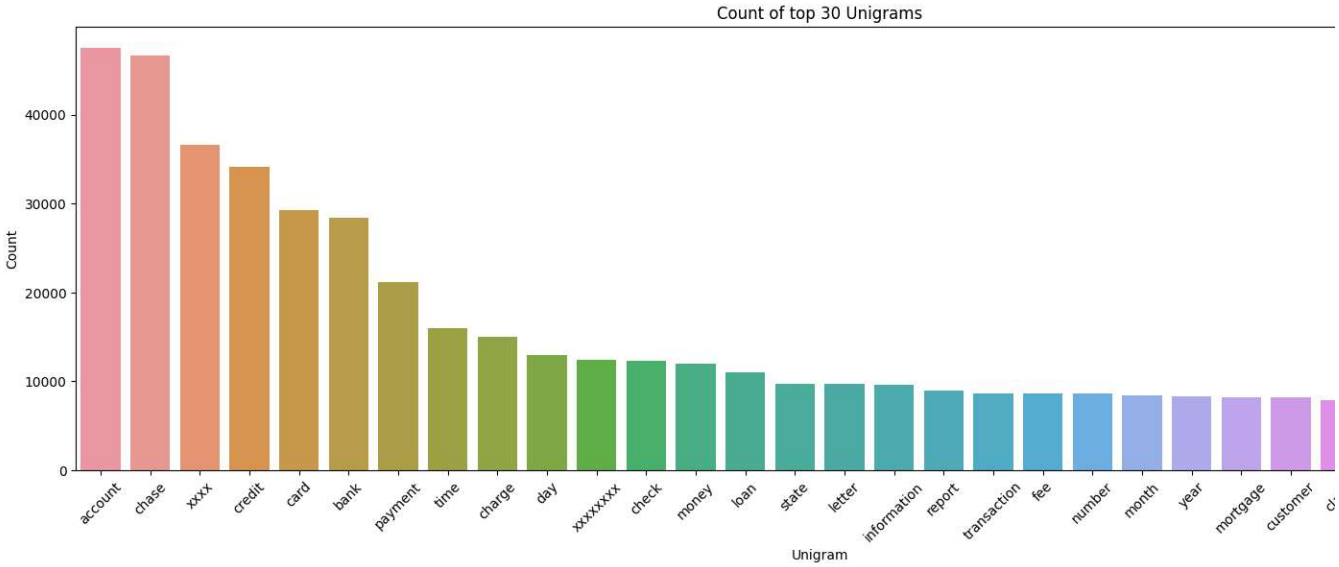
```
#Write your code here to find the top 30 unigram frequency among the complaints in the cleane
def get_top_ngrams(text, n=None, ngram=(1,1)):
  vec = CountVectorizer(stop_words='english', ngram_range=ngram).fit(text)
  bagofwords = vec.transform(text)
  sum_words = bagofwords.sum(axis=0)
  words_frequency = [(word, sum_words[0, index]) for word, index in vec.vocabulary_.items()]
  words_frequency = sorted(words_frequency, key = lambda x: x[1], reverse=True)
  return words_frequency[:n]
```

```
top_30words = get_top_ngrams(df_clean['Complaint_clean'].values.astype('U'), n=30, ngram=(1,1
```

```
df_unigram = pd.DataFrame(top_30words, columns=['unigram', 'count'])
df_unigram.head(3)
```

	unigram	count
0	account	47523
1	chase	46710
2	xxxx	36583

```
plt.figure(figsize=[20,6])
sns.barplot(x=df_unigram['unigram'], y=df_unigram['count'])
plt.xticks(rotation=45)
plt.xlabel("Unigram")
plt.ylabel("Count")
plt.title("Count of top 30 Unigrams")
plt.show()
```



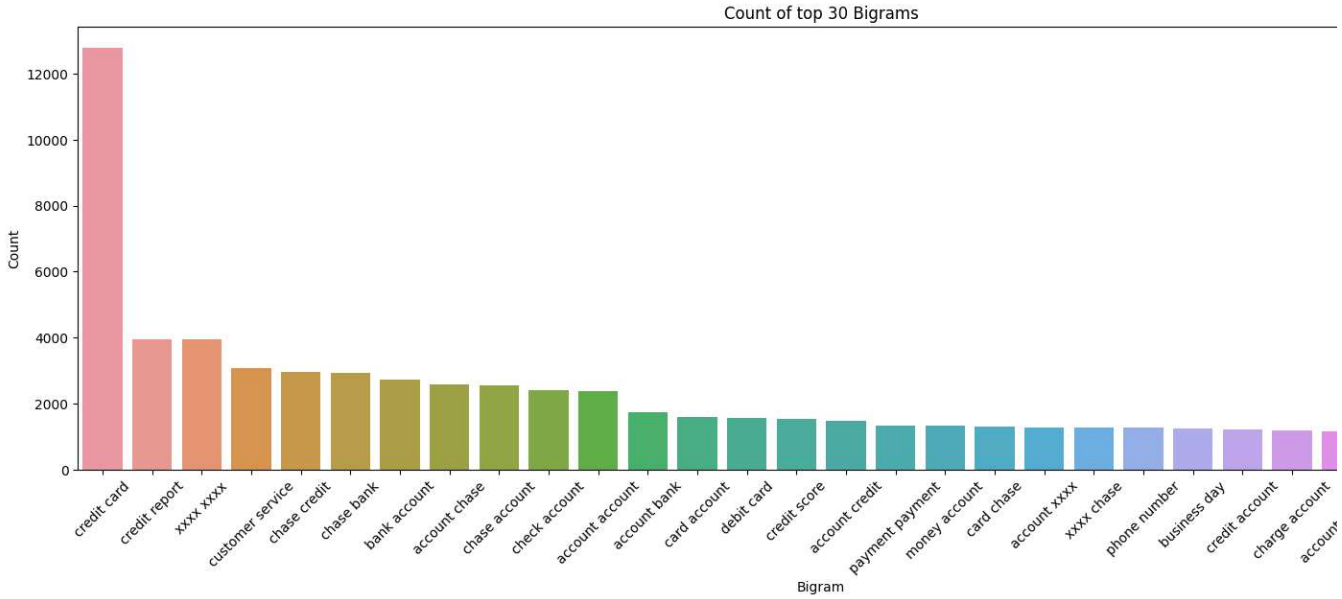

```
#Print the top 10 words in the unigram frequency
df_unigram.head(10)
```

	unigram	count
0	account	47523
1	chase	46710
2	xxxx	36583
3	credit	34154
4	card	29289
5	bank	28410
6	payment	21170
7	time	16027
8	charge	15000
9	day	12977

```
#Write your code here to find the top 30 bigram frequency among the complaints in the cleanec
top_30words = get_top_ngrams(df_clean['Complaint_clean'].values.astype('U'), n=30, ngram=(2,2

#Print the top 10 words in the bigram frequency
df_bigram = pd.DataFrame(top_30words, columns=['bigram', 'count'])
df_bigram.head(10)
```

	bigram	count
0	credit card	12781
1	credit report	3955
2	xxxx xxxx	3951
3	customer service	3081
4	chase credit	2966
5	chase bank	2940
6	bank account	2728
7	account chase	2595
8	chase account	2564
9	check account	2413

```
plt.figure(figsize=[20,6])
sns.barplot(x=df_bigram['bigram'], y=df_bigram['count'])
plt.xticks(rotation=45)
plt.xlabel("Bigram")
plt.ylabel("Count")
plt.title("Count of top 30 Bigrams")
plt.show()
```

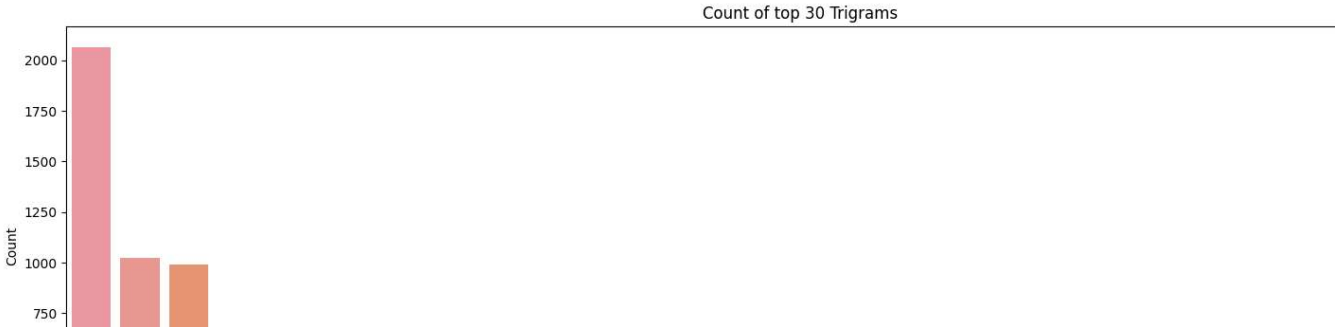


```
#Write your code here to find the top 30 trigram frequency among the complaints in the cleane
top_30words = get_top_ngrams(df_clean['Complaint_clean'].values.astype('U'), n=30, ngram=(3,3
```

```
#Print the top 10 words in the trigram frequency
df_trigram = pd.DataFrame(top_30words, columns=['trigram', 'count'])
df_trigram.head(10)
```

	trigram	count
0	chase credit card	2063
1	credit card account	1023
2	credit card company	992
3	credit card chase	632
4	credit card credit	515
5	inquiry credit report	481
6	charge credit card	421
7	xxxx xxxx xxxx	411
8	account credit card	398
9	card credit card	388

```
plt.figure(figsize=[20,6])
sns.barplot(x=df_trigram['trigram'], y=df_trigram['count'])
plt.xticks(rotation=45)
plt.xlabel("Trigram")
plt.ylabel("Count")
plt.title("Count of top 30 Trigrams")
plt.show()
```



The personal details of customer has been masked in the dataset with xxxx. Let's remove the masked text as this will be of no use for our analysis

```
df_clean['Complaint_clean'] = df_clean['Complaint_clean'].str.replace('xxxx','')


#All masked texts has been removed
df_clean.head(3)
```

	complaint_what_happened	complaint_lemmatized	complaint_POS_removed	C
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop bank cardmember service ask ...	m
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent upgrade date agent information orde...	ca date
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent a...	card report application identity consent servi...	card ident

Feature Extraction

Convert the raw texts to a matrix of TF-IDF features

max_df is used for removing terms that appear too frequently, also known as "corpus-specific stop words" max_df = 0.95 means "ignore terms that appear in more than 95% of the complaints"

min_df is used for removing terms that appear too infrequently min_df = 2 means "ignore terms that appear in less than 2 complaints"

```
#Write your code here to initialise the TfidfVectorizer

tfidf = TfidfVectorizer(min_df=2, max_df=0.95, stop_words='english')
```

Create a document term matrix using fit_transform

The contents of a document term matrix are tuples of (complaint_id,token_id) tf-idf score: The tuples that are not there have a tf-idf score of 0

```
#Write your code here to create the Document Term Matrix by transforming the complaints colum
dtm = tfidf.fit_transform(df_clean['Complaint_clean'])
dtm
```

```
<21072x7267 sparse matrix of type '<class 'numpy.float64'>'
        with 643743 stored elements in Compressed Sparse Row format>
```

Topic Modelling using NMF

Non-Negative Matrix Factorization (NMF) is an unsupervised technique so there are no labeling of topics that the model will be trained on. The way it works is that, NMF decomposes (or factorizes) high-dimensional vectors into a lower-dimensional representation. These lower-dimensional vectors are non-negative which also means their coefficients are non-negative.

In this task you have to perform the following:

- Find the best number of clusters
- Apply the best number to create word clusters
- Inspect & validate the correction of each cluster wrt the complaints
- Correct the labels if needed
- Map the clusters to topics/cluster names

```
from sklearn.decomposition import NMF
```

Manual Topic Modeling

You need to do take the trial & error approach to find the best num of topics for your NMF model.

The only parameter that is required is the number of components i.e. the number of topics we want. This is the most crucial step in the whole topic modeling process and will greatly affect how good your final topics are.

```
#Load your nmf_model with the n_components i.e 5
num_topics = 5

#keep the random_state =40
nmf_model = NMF(n_components=num_topics, random_state=40)


nmf_model.fit(dtm)
len(tfidf.get_feature_names_out())
```

```
7267
```

```
H = nmf_model.components_


H
```

```
array([[4.22504203e-04, 0.00000000e+00, 2.26657998e-03, ...,
        0.00000000e+00, 8.23555252e-04, 8.89016785e-05],
       [0.00000000e+00, 3.84827477e-03, 0.00000000e+00, ...,
        2.45599794e-04, 0.00000000e+00, 1.70518816e-04],
       [1.88518908e-05, 1.00309551e-04, 0.00000000e+00, ...,
        0.00000000e+00, 9.08176225e-04, 0.00000000e+00],
       [1.23310321e-04, 2.16003817e-03, 0.00000000e+00, ...,
        2.74258126e-04, 3.07074409e-03, 5.44219302e-04],
       [1.10066620e-04, 3.18904608e-04, 1.34014823e-04, ...,
        8.37719628e-04, 3.22577908e-03, 0.00000000e+00]])
```

```
#Print the Top15 words for each of the topics
words = np.array(tfidf.get_feature_names_out())
topic_words = pd.DataFrame(np.zeros((num_topics, 15)), index=[f'Topic {i + 1}' for i in range
                           columns=[f'Word {i + 1}' for i in range(15)]).astype(str)
for i in range(num_topics):
    ix = H[i].argsort()[::-1][:15]
    topic_words.iloc[i] = words[ix]

topic_words
```

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9
Topic 1	account	bank	check	money	fund	chase	deposit	branch	day

```
#Create the best topic for each complaint in terms of integer value 0,1,2,3 & 4

topic_results = nmf_model.transform(dtm)
```

```
#Assign the best topic to each of the cmplaints in Topic Column

df_clean['Topic'] = topic_results.argmax(axis=1)
```

```
df_clean.head()
```

	complaint_what_happened	complaint_lemmatized	complaint_POS_removed	Complai
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop bank cardmember service ask ...	morning n bank car serv
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent upgrade date agent information orde...	card agen d informati
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent a...	card report application identity consent servi...	c applicatic conse
11	on while trying to book a xxxx xxxx ticket ...	on while try to book a xxxx xxxx ticket ...	try book xxxx ticket offer ticket card informa...	try book t t info
14	my grand son give me check	my grand son give I check	son chase account fund	son chas fund ban

```
#Print the first 5 Complaint for each of the Topics
df_clean=df_clean.groupby('Topic').head(5)
df_clean.sort_values('Topic')
```

	complaint_what_happened	complaint_lemmatized	complaint_POS_removed	Complai
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop bank cardmember service ask ...	morning n bank car serv
14	my grand son give me check for i deposit it i...	my grand son give I check for I deposit it i...	son chase account fund bank account pay money ...	son chas fund ban pay
17	with out notice jp morgan chase restricted my ...	with out notice jp morgan chase restrict my ac...	jp chase account debit card tuesday thursday b...	jp chas debit car thu
24	mishandling of this account by chase auto and ...	mishandle of this account by chase auto and xxxx	mishandle account auto xxxx	mishandl
27	i opened an account with chase bank on xxxx an...	I open an account with chase bank on xxxx and ...	account bank code bonus term everything accoun...	account b bo everything
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent upgrade date agent information orde...	card agen d informati
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent a...	card report application identity consent servi...	c applicatic conse
11	on while trying to book a xxxx xxxx ticket ...	on while try to book a xxxx xxxx ticket ...	try book xxxx ticket offer ticket card informa...	try book t t info
15	can you please remove inquiry	can you please remove inquiry	inquiry	
23	i have a chase credit card which is incorrectl...	I have a chase credit card which be incorrectl...	chase credit card datum credit report company ...	chase c datum cre cc
82	i recently called to ask chase bank why they r...	I recently call to ask chase bank why they rep...	bank credit bureau day payment info xxxx bill ...	bank cred day pay
58	i made a purchase of on xxxxxxxx i made payme...	I make a purchase of on xxxxxxxx I make paym...	purchase xxxxxxxx payment xxxxxxxx payment clo...	purchase payme date
167	a double payment from my chase debt cart to ch...	a double payment from my chase debt cart to ch...	payment debt cart credit card post request day...	payment credit requ
20	during the summer months i experience a declin...	during the summer month I experience a decline...	summer month income employment month payment e...	summ income em month pay
141	action taken by the company between the dates ...	action take by the company between the date of	action company date individual group contact c...	action date group c

After evaluating the mapping, if the topics assigned are correct then assign these names to the relevant topic:

- Bank Account services
- Credit card or prepaid card
- Theft/Dispute Reporting
- Mortgage/Loan
- Others

```
#Create the dictionary of Topic names and Topics

Topic_names = { 0:"Bank account services", 1:"Credit card / Prepaid card", 2:"Others",
               3:"Theft/Dispute reporting", 4:"Mortgages/loans" }
#Replace Topics with Topic Names
df_clean['Topic'] = df_clean['Topic'].map(Topic_names)
```

```
<ipython-input-159-6eb0c9e942ef>:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
```

```
df_clean.head(5)
```

	complaint_what_happened	complaint_lemmatized	complaint_POS_removed	Complaint_
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop bank cardmember service ask ...	morning nam bank cardm service
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent upgrade date agent information orde...	card agent u date information
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent a...	card report application identity consent servi...	carc application i consent
11	on while trying to book a xxxx xxxx ticket ...	on while try to book a xxxx xxxx ticket ...	try book xxxx ticket offer ticket card informa...	try book tick tick inform

Supervised model to predict any new complaints to the relevant Topics.

You have now build the model to create the topics for each complaints.Now in the below section you will use them to classify any new complaints.

Since you will be using supervised learning technique we have to convert the topic names to numbers(numpy arrays only understand numbers)

```
#Create the dictionary again of Topic names and Topics

Topic_names = { "Bank account services":0, "Credit card / Prepaid card":1, "Others":2,
               "Theft/Dispute reporting":3, "Mortgages/loans":4 }
#Replace Topics with Topic Names
df_clean['Topic'] = df_clean['Topic'].map(Topic_names)
```

```
df_clean.head()
```

	complaint_what_happened	complaint_lemmatized	complaint_POS_removed	Complai
1	good morning my name is xxxx xxxx and i apprec...	good morning my name be xxxx xxxx and I apprec...	morning name stop bank cardmember service ask ...	morning n bank car serv
2	i upgraded my xxxx xxxx card in and was told ...	I upgrade my xxxx xxxx card in and be tell b...	card agent upgrade date agent information orde...	card agen d informati
10	chase card was reported on however fraudulent...	chase card be report on however fraudulent a...	card report application identity consent servi...	c applicatic conse
11	on while trying to book a xxxx xxxx ticket ...	on while try to book a xxxx xxxx ticket ...	try book xxxx ticket offer ticket card informa...	try book t t info
14	my grand son give me check	my grand son give I check	son chase account fund	son chas fund ban

```
#Keep the columns"complaint_what_happened" & "Topic" only in the new dataframe --> training_d
training_data = df_clean[['complaint_what_happened', 'Topic']]
```

```
training_data.head()
```

	complaint_what_happened	Topic
1	good morning my name is xxxx xxxx and i apprec...	0
2	i upgraded my xxxx xxxx card in and was told ...	1

Apply the supervised models on the training data created. In this process, you have to do the following:

- Create the vector counts using Count Vectoriser
- Transform the word vecotr to tf-idf
- Create the train & test data using the train_test_split on the tf-idf & topics

```
#Write your code to get the Vector count
vect = CountVectorizer()
X_train_cnt = vect.fit_transform(training_data['complaint_what_happened'])
pickle.dump(vect.vocabulary_, open("count_vector.pk1", "wb"))
#Write your code here to transform the word vector to tf-idf
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_cnt)
pickle.dump(tfidf_transformer, open('tfidf.pk1', "wb"))
```

You have to try atleast 3 models on the train & test data from these options:

- Logistic regression
- Decision Tree
- Random Forest
- Naive Bayes (optional)

Using the required evaluation metrics judge the tried models and select the ones performing the best

```
# Write your code here to build any 3 models and evaluate them using the required metrics
```

```
# Importing LogisticRegression from sklearn
from sklearn.linear_model import LogisticRegression
# Importing Train, Test Split
from sklearn.model_selection import train_test_split
```

```
# Train, Test Split
X_train, X_test, y_train, y_test = train_test_split(X_train_tfidf, training_data['Topic'], te
```

```
logreg = LogisticRegression(random_state=42, solver='liblinear').fit(X_train, y_train)


# Getting the score of the base model
logreg.score(X_test, y_test)
```

```
0.14285714285714285
```

```
logreg_grid = {"C": [100, 10, 5, 4, 3, 2, 1, 1.0, 0.1, 0.01],
               "solver": ["liblinear"]}


logreg_hpt = GridSearchCV(LogisticRegression(random_state=42),
                          param_grid=logreg_grid,
                          cv=5,
                          verbose=True,
                          n_jobs=-1)

# Fit random hyperparameter search model
logreg_hpt.fit(X_train, y_train);
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700: UserWarni

The least populated class in y has only 2 members, which is less than n_splits=5.
```

```
# get best params
logreg_hpt.best_params_
```

```
{'C': 100, 'solver': 'liblinear'}
```

```
logreg_hpt.score(X_test, y_test)
```

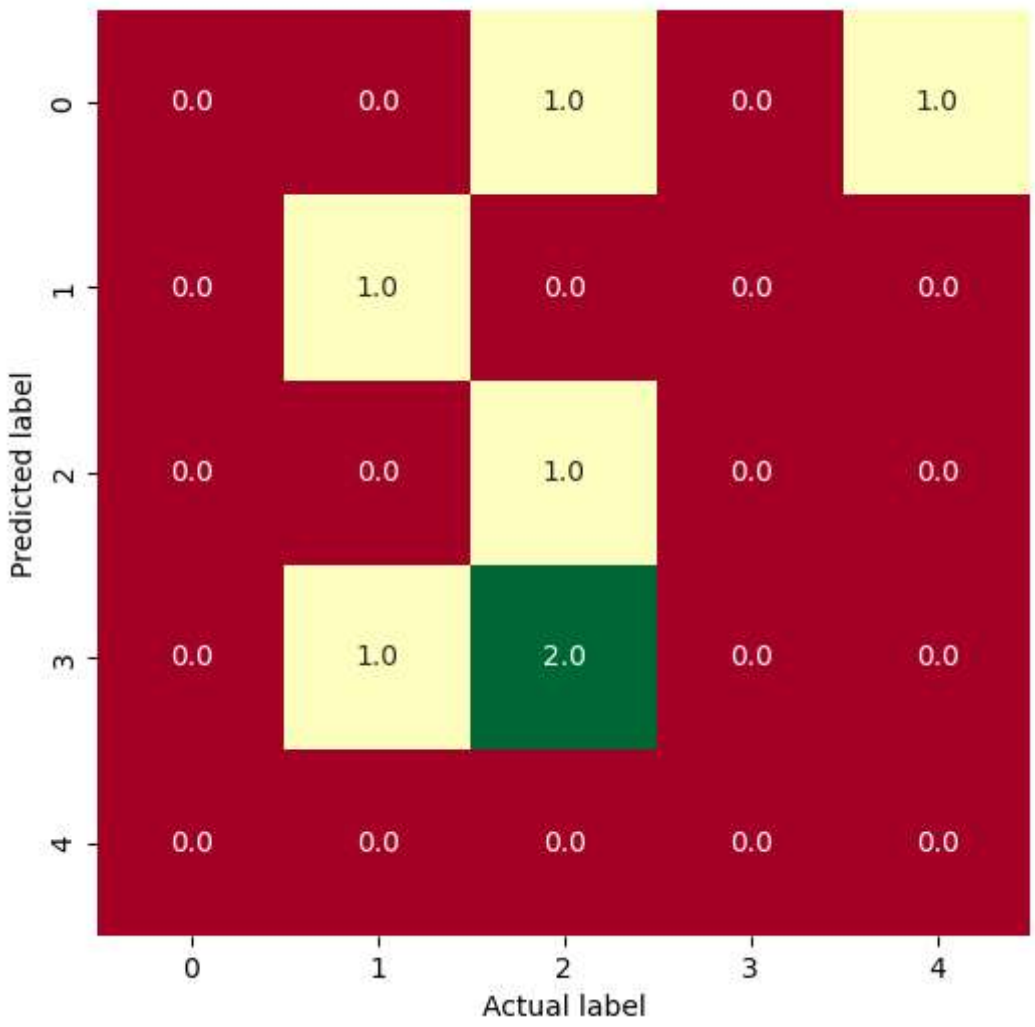
```
0.2857142857142857
```

```
# Load pickled model
pickle.dump(logreg_hpt, open("logreg_model.pk1", "wb"))


# Make predictions on test data
logreg_model = pickle.load(open("logreg_model.pk1", "rb"))


# Load pickled model
logreg_model = pickle.load(open("logreg_model.pk1", "rb"))
```

```
# Make predictions on test data
y_pred = logreg_model.predict(X_test)


# Print Confusion Matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[0 0 1 0 1]
 [0 1 0 0 0]
 [0 0 1 0 0]
 [0 1 2 0 0]
 [0 0 0 0 0]]
```

```
# Visualize Confusion Matrix with heatmap
fig, ax = plt.subplots(figsize=(6, 6))
ax = sns.heatmap(confusion_matrix(y_test, y_pred),
                 annot=True,
                 cbar=False,
                 cmap="RdYlGn", fmt = '0.1f')
plt.xlabel("Actual label")
plt.ylabel("Predicted label")
plt.show()
```



```
Topicnames_target = ["Bank account services", "Credit card / Prepaid card", "Others", "Theft/

# Print Classification Report
print(classification_report(y_test, y_pred, target_names = Topicnames_target))
```

```
                            precision    recall  f1-score   support

     Bank account services       0.00      0.00      0.00         2
Credit card / Prepaid card       0.50      1.00      0.67         1
                    Others       0.25      1.00      0.40         1
   Theft/Dispute reporting       0.00      0.00      0.00         3
            Mortgages/loans      0.00      0.00      0.00         0

                  accuracy                           0.29         7
                 macro avg       0.15      0.40      0.21         7
              weighted avg       0.11      0.29      0.15         7

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.
```

Decision Tree Classifier

```
# Importing DecisionTreeClassifier from sklearn
from sklearn.tree import DecisionTreeClassifier

# Train, Test Split
X_train, X_test, y_train, y_test = train_test_split(X_train_tfidf, training_data['Topic'], te
```

```
dt = DecisionTreeClassifier(random_state=42).fit(X_train, y_train)


# Getting the score of the base model
dt.score(X_test, y_test)
```

```
0.14285714285714285
```

```
dt_grid = {"max_depth": [3, 5, 10],
           "min_samples_split": np.arange(2, 20, 2),
           "min_samples_leaf": np.arange(1, 20, 2)}


# Setup grid hyperparameter search for LogisticRegression
dt_hpt = GridSearchCV(DecisionTreeClassifier(random_state=42),
                                param_grid=dt_grid,
                                cv=5,
                                verbose=True,
                                n_jobs=-1)

# Fit random hyperparameter search model
dt_hpt.fit(X_train, y_train);
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700: UserWarni

The least populated class in y has only 2 members, which is less than n_splits=5.

Fitting 5 folds for each of 270 candidates, totalling 1350 fits
```

```
# Check best parameters
dt_hpt.best_params_
```

```
{'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 8}
```

```
# Evaluating the tuned model
dt_hpt.score(X_test, y_test)
```

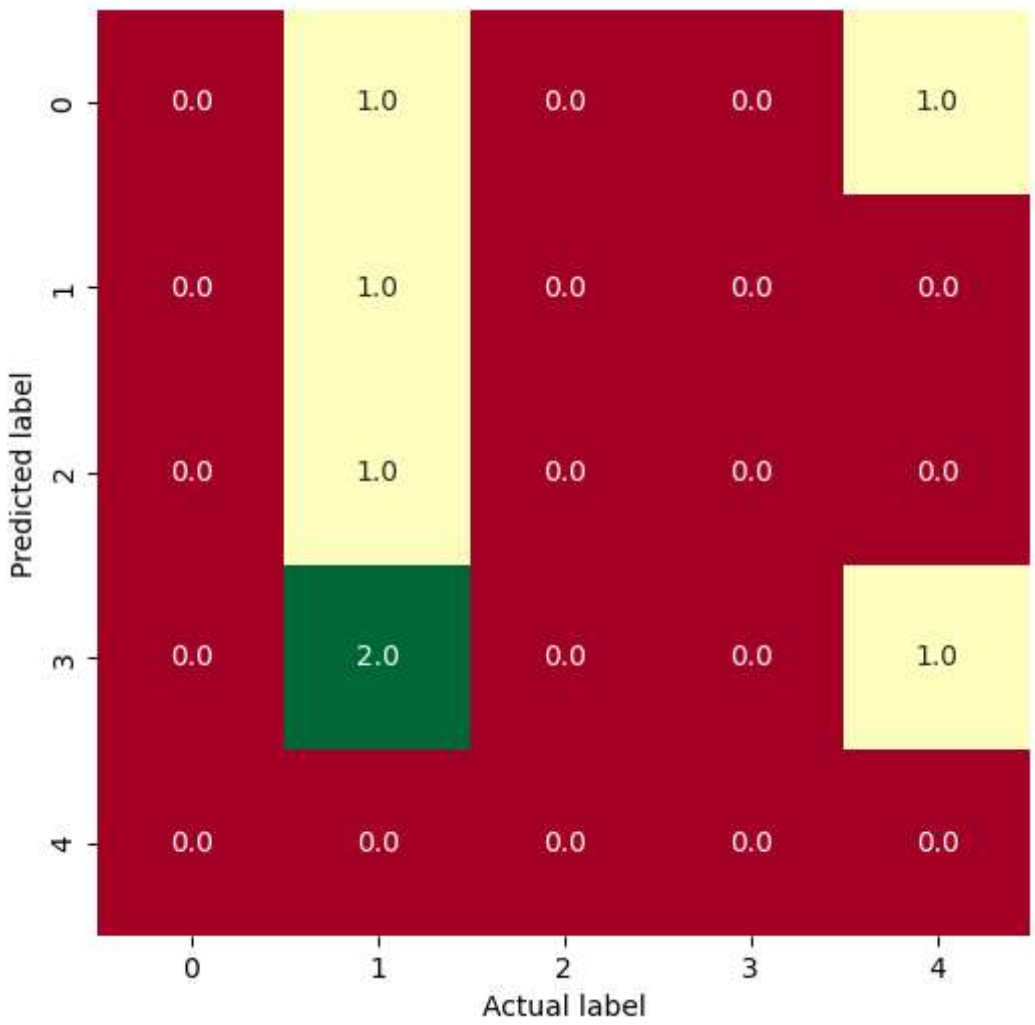
```
0.14285714285714285
```

```
# Save Logistic Regression Model
pickle.dump(dt_hpt, open("dt_model.pk1", "wb"))


# Load pickled model
dt_model = pickle.load(open("dt_model.pk1", "rb"))
```

```
# Make predictions on test data
y_pred = dt_model.predict(X_test)


# Print Confusion Matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[0 1 0 0 1]
 [0 1 0 0 0]
 [0 1 0 0 0]
 [0 2 0 0 1]
 [0 0 0 0 0]]
```

```
# Visualize Confusion Matrix with heatmap
fig, ax = plt.subplots(figsize=(6, 6))
ax = sns.heatmap(confusion_matrix(y_test, y_pred),
                 annot=True,
                 cbar=False,
                 cmap="RdYlGn", fmt = '0.1f')
plt.xlabel("Actual label")
plt.ylabel("Predicted label")
plt.show()
```



```
Topicnames_target = ["Bank account services", "Credit card / Prepaid card", "Others", "Theft/

# Print Classification Report
print(classification_report(y_test, y_pred, target_names = Topicnames_target))
```

```
                            precision    recall  f1-score   support

     Bank account services       0.00      0.00      0.00         2
Credit card / Prepaid card       0.20      1.00      0.33         1
                    Others       0.00      0.00      0.00         1
   Theft/Dispute reporting       0.00      0.00      0.00         3
           Mortgages/loans       0.00      0.00      0.00         0

                  accuracy                           0.14         7
                 macro avg       0.04      0.20      0.07         7
              weighted avg       0.03      0.14      0.05         7

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undefin

Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.
```

Random Forest Classifier

```
# Importing Random Forest Classifier from sklearn
from sklearn.ensemble import RandomForestClassifier
# Importing Train, Test Split
from sklearn.model_selection import train_test_split
```

```
# Train, Test Split
X_train, X_test, y_train, y_test = train_test_split(X_train_tfidf, training_data['Topic'], te


rf = RandomForestClassifier(random_state=42).fit(X_train, y_train)


# Getting the score of the base model
rf.score(X_test, y_test)
```

```
0.2857142857142857
```

HyperParameter Tuning

```
rf_grid = {"n_estimators": np.arange(10, 1000, 50),
           "max_depth": [3, 5, 10],
           "min_samples_split": np.arange(2, 20, 2),
           "min_samples_leaf": np.arange(1, 20, 2)}


# Setup random hyperparameter search for LogisticRegression
rf_hpt = RandomizedSearchCV(RandomForestClassifier(random_state=42),
                            param_distributions=rf_grid,
                            cv=5,
                            verbose=True,
                            n_jobs=-1)

# Fit random hyperparameter search model
rf_hpt.fit(X_train, y_train);
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700: UserWarni

The least populated class in y has only 2 members, which is less than n_splits=5.
```

```
# Check best parameters
rf_hpt.best_params_
```

```
{'n_estimators': 860,
 'min_samples_split': 10,
 'min_samples_leaf': 13,
 'max_depth': 5}
```

```
# Evaluating the tuned model
rf_hpt.score(X_test, y_test)
```

```
0.0
```

Overall Results

We created three supervised models (Logistic Regression, Decision Tree Classifier and Random Forest Classifier) to predict any new complaints to the relevant Topics.

Results

Decission Tree classifier was the best perfromer in our case

✓ 0s completed at 3:17 PM