✨TEAM TARA✨
Problem Statement-3
Complaint Classification

Problem statement:

Bank Customer Complaint Classification System

Objective:

Develop an intelligent system that aims to leverage natural language processing techniques to develop an efficient and accurate system for bank customer complaint analysis, classification. By automating the classification, financial institutions can enhance their dispute resolution capabilities and improve customer satisfaction.

Brief of the dataset:

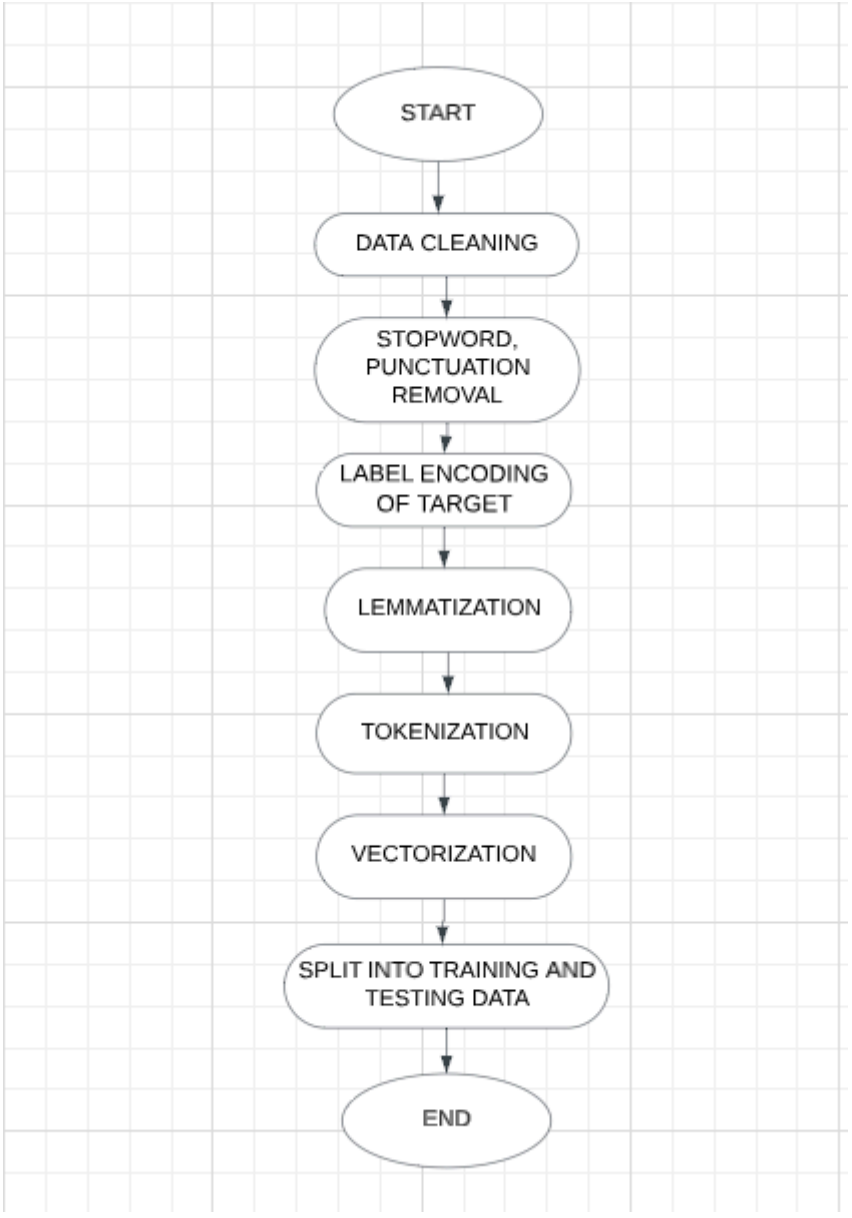
The dataset consists of 2 columns, Product and Narratives, in which Product contains the different types of complaints: credit card, mortgages and loans, debt collection, retail banking and credit reporting.
The data has 162420 rows in total.
We had to classify the complaints into the 5 categories accurately.

Process:

- The text data required tons of **pre-processing**.
- Our approach was to check for **duplicate and missing values** and remove them.
- We **label-encoded** the “product” column and checked the number of datapoints that were classified into each of the categories and realised that the distribution was *skewed*. This led us to **upsample** the data since downsampling meant dropping more than half of the dataset.
- We then converted all **strings to lowercase** and **removed the stopwords**.
- Next, we removed the punctuation marks and any other special characters in the dataset. We removed private credentials in the strings

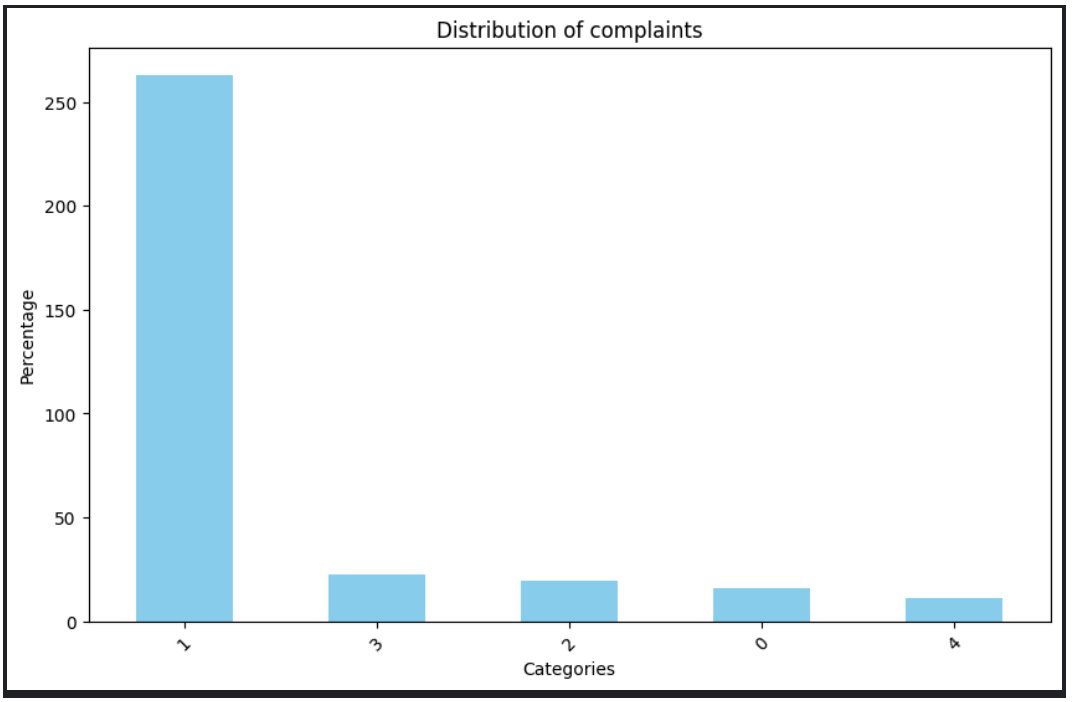
(which were denoted by ‘xxxxxx’) and any other back-to-back repeating characters and words.

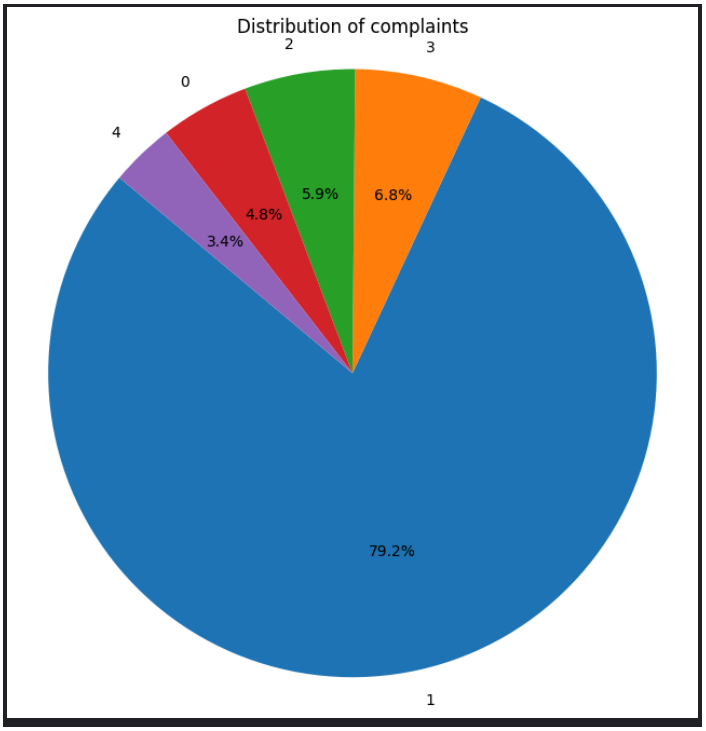
- We then **lemmatized the text** and then **tokenized** it.
- Afterwards, we vectorized the text using **Tf-Idf vectorization**.
- Post this, our data was ready to be split into Training and Testing data and to be fit into various models.



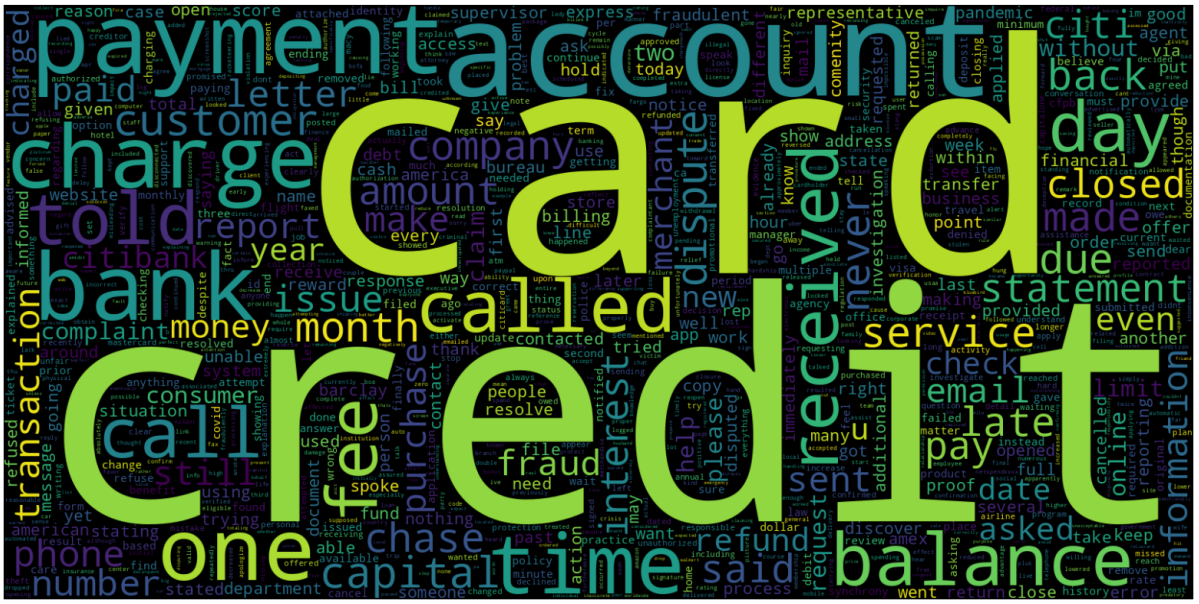
Visualisations:

1. Distribution of complaints across product categories

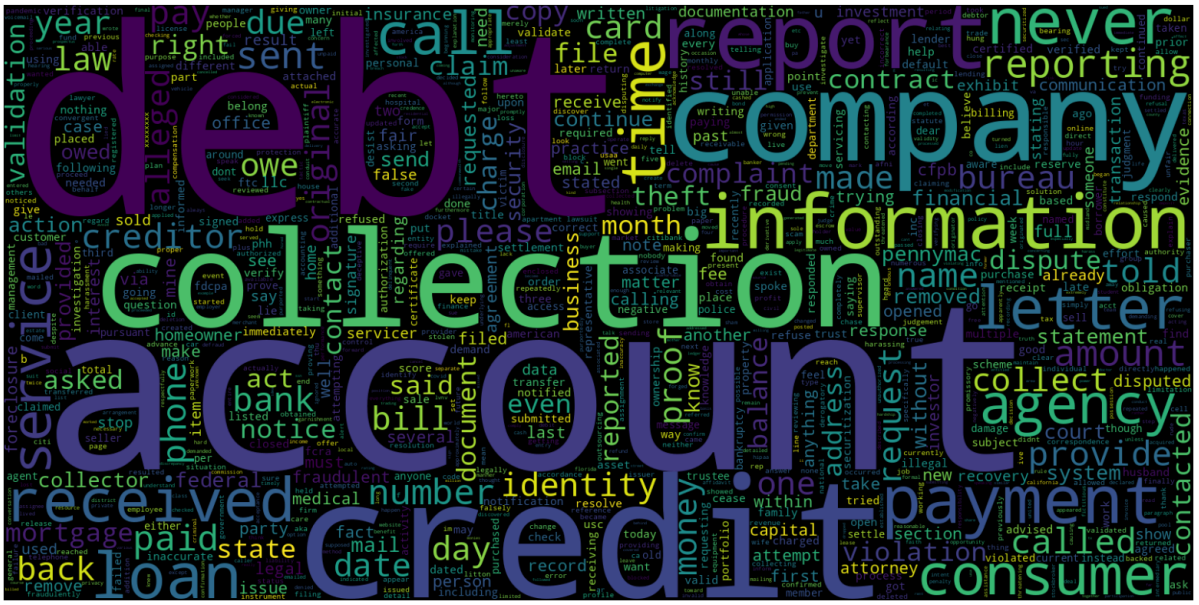




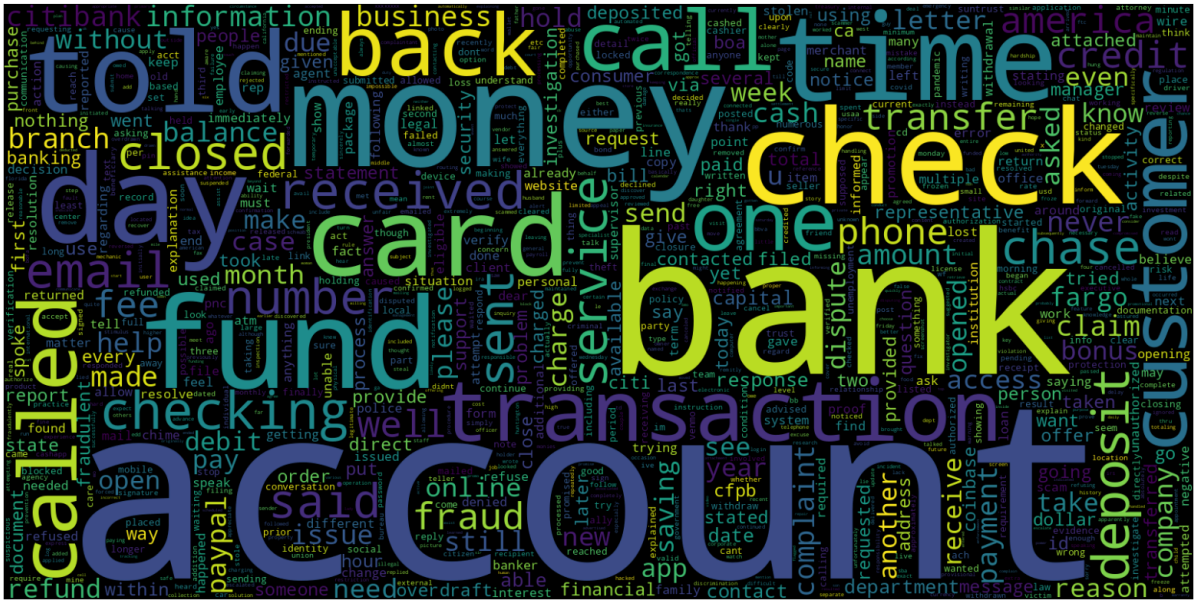
2. Word cloud visualisations for each of the categories
   Category 1: credit_card



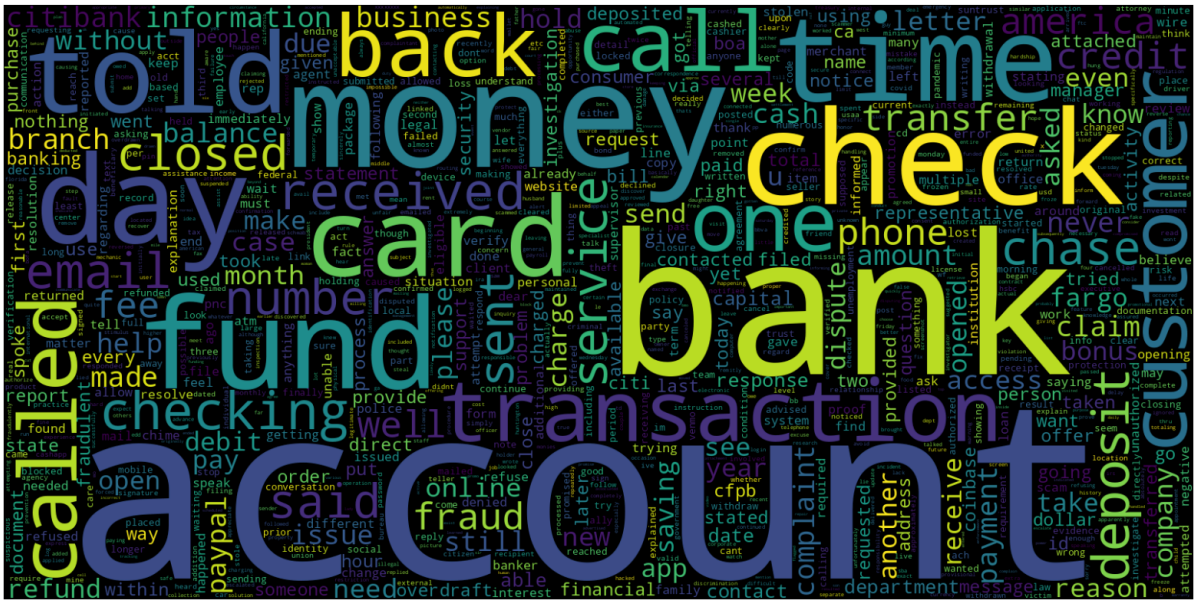
   Category 2: debt_collection



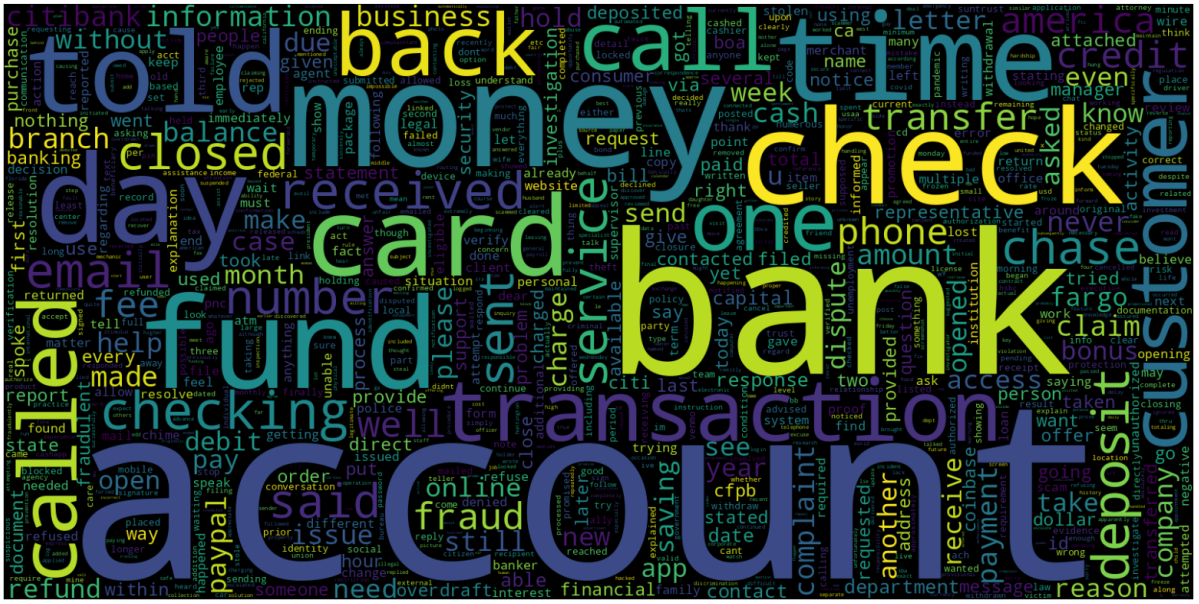
Category 3: retail_banking



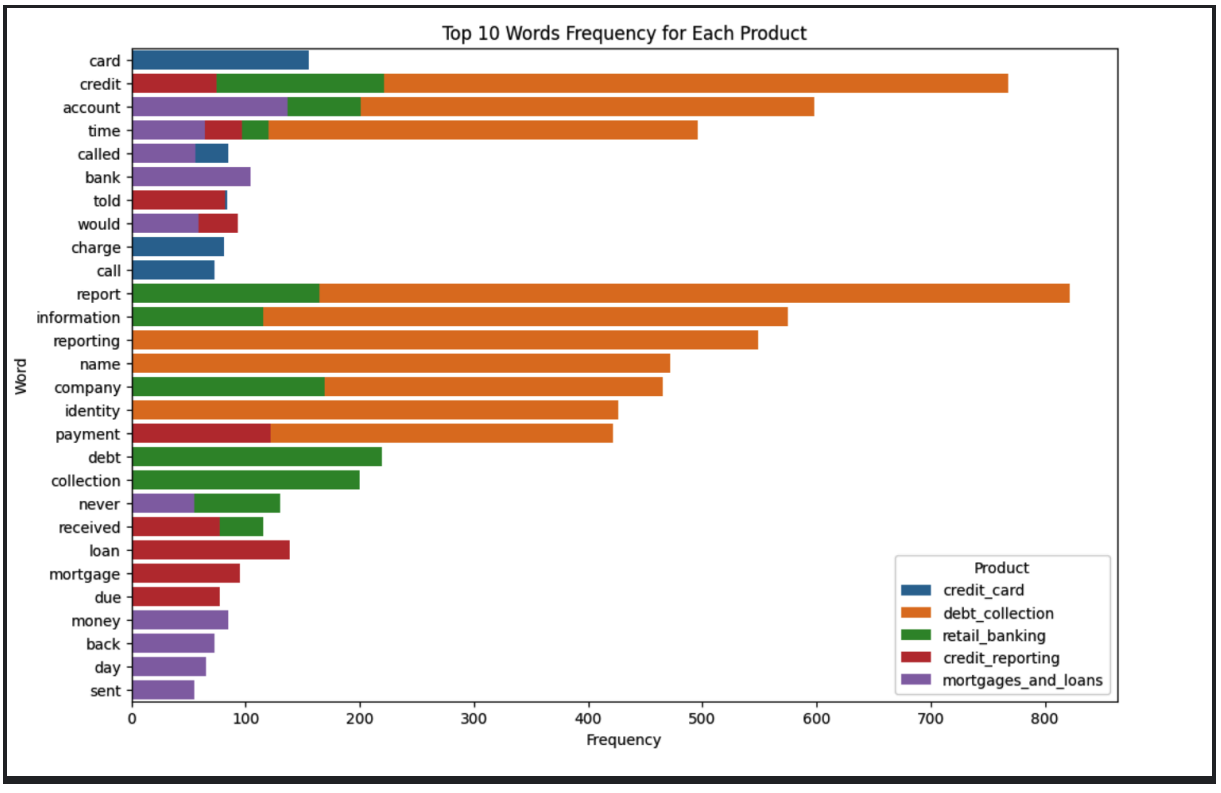
Category 4: credit_reporting



Category 5: mortgages_and_loans



3. Top 10 words in each category



Models used:

1. Multinomial Naive Bayes

```
In [745]: mnb = MultinomialNB()    #MULTINOMIALNB
          mnb.fit(X_train, y_train)

Out[745]: MultinomialNB()

In [746]: y_pred = mnb.predict(X_test)
          accuracy = accuracy_score(y_pred, y_test)
          f1 = f1_score(y_pred, y_test, average="weighted")

          print("Accuracy:", accuracy)
          print("F1 Score:", f1)

          Accuracy: 0.8064774336555631
          F1 Score: 0.8077670653379518
```

The accuracy wasn't as high as we expected it to be, hence we moved onto trying more models.

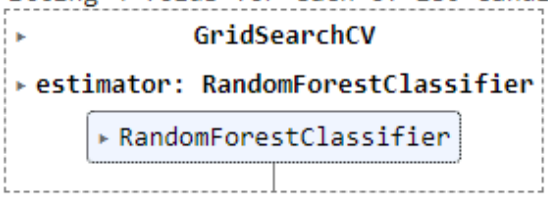
2. Random Forest Classifier

```
y_pred = classifier_rf.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuracy)
print("F1 Score:", f1)

Accuracy: 0.5030283710551482
F1 Score: 0.6693531283138918
```

The accuracy score was even lower, hence we decided to hyperparameter tune the model -

```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)

params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}

from sklearn.model_selection import GridSearchCV

# Instantiate the grid search model
grid_search = GridSearchCV(estimator=rf,
                           param_grid=params,
                           cv = 4,
                           n_jobs=-1, verbose=1, scoring="accuracy")

grid_search.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits
```




```
[ ] y_pred= grid_search.predict(X_test)
    accuracy = accuracy_score(y_pred, y_test)
    print(accuracy)
```

```
0.7287217086388269
```

The accuracy improved to 72.8%

3. XGBoost
   We further decided to use Boosting algorithms.

```
In [747]: import xgboost as xgb      #XGBOOST
          from xgboost import XGBClassifier
          xgb_cl = xgb.XGBClassifier()
          xgb_cl.fit(X_train, y_train)
          predxgb = xgb_cl.predict(X_test)
          accuracy_score(y_test, predxgb)

Out[747]: 0.8874864027256122
```

4. LightGBM
   We decided to use LightGBM since empirically it has a slightly better accuracy than XGBoost and is about 6 times faster than it.

```
Out[769]: LGBMClassifier()

In [770]: y_pred=clf.predict(X_test)

In [771]: accuracy_score(y_test, y_pred)

Out[771]: 0.8817806785296471
```

   We observed that both the boosting algorithms - XGBoost and LightGBM gave the highest accuracy.

   We then took a user input (complaint) and then predicted the class of the complaint by using XGBoost since it gave us the highest accuracy. We added an additional validation of not classifying the data into any class if a non-english word or gibberish was inputted.

```
Enter your complaint narrative: credit card stolen, no security
Predicted category: credit_card
```

```
Enter your complaint narrative: me roban la tarjeta de credito
Invalid complaint: The complaint is not in English.
```

```
Enter your complaint narrative: abc
Invalid complaint: Narrative is empty or too short.
```

Sentiment Analysis:

We have further done Sentiment Analysis in order to Identify how the user is feeling while submitting a complaint.
We executed it using SentimentIntensityAnalyzer from NLTK library, which gave us 92% accuracy.

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
   Unnamed: 0           product  \
0           0       credit_card
1           1       credit_card
2           2    retail_banking
3           3  credit_reporting
4           4  credit_reporting

                                           narrative sentiment
0  purchase order day shipping amount receive pro...  Positive
1  forwarded message date tue subject please inve...  Positive
2  forwarded message cc sent friday pdt subject f...  Negative
3  payment history missing credit report speciali...   Neutral
4  payment history missing credit report made mis...   Neutral
Total Positive Rows: 91
Total Negative Rows: 317
Total Rows: 442
Accuracy: 92.3076923076923 %
```

Further, we did training and found efficacy by taking input complaints from the user and then further doing Sentiment Classification, which gave us 98.64% accuracy.

```
Enter your narrative: good
Predicted Sentiment: Positive
Updated Total Positive Rows: 107
Updated Total Negative Rows: 330
Updated Total Rows: 443
Updated Accuracy: 98.64559819413093 %
```

Chatbot:

We further created a chatbot for the same, which can be used in the Industry later on for analysing employee behaviour.

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
Please enter your complaint: horrible service
We're sorry to hear that.We will improve :(
Would you like to submit another complaint? (yes/no): no
Thank you for using our service. Goodbye!
```

Future Scope:

We could build a recommendation system that would give appropriate solutions to a user’s complaint.
The Complaint classification, coupled with Sentiment Analysis, can provide valuable insights for the company when an employee raises a ticket or issue.
The dataset can be enhanced with additional features to facilitate more comprehensive visualisation, thereby enabling deeper insights into the product. This enriched dataset can be leveraged to inform and guide efforts aimed at enhancing the product and its associated processes.

References:

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html
https://neptune.ai/blog/xgboost-vs-lightgbm#:~:text=Both%20the%20algorithms%20perform%20similarly,choice%20for%20machine%20learning%20experiments.
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://rspsciencehub.com/article_23795_3d7d1d4dfc63d072724eafdb92a5bcbf.pdf
https://www.geeksforgeeks.org/python-lemmatization-with-nltk/