Name- Yashwant Bhosale
MIS- 612303039

Problem Statement #1: Bank Customer Complaint Classification Using NLP

Report about the solution:

1. **Understanding the problem statement:**In this project, we are tasked with developing an NLP classification model to categorize customer complaints submitted to the Consumer Financial Protection Bureau (CFPB) regarding various financial products. The dataset comprises a large number of complaints grouped into five distinct product categories: **credit reporting, debt collection, mortgages and loans, credit cards, and retail banking.** Given that the narratives are often raw and noisy, effective data preprocessing is crucial to enhance model performance.
2. **Approach:** So the approach will involve cleaning the data using various **preprocessing techniques**, followed by **vectorization** to convert text into a suitable format for analysis. Finally, we will experiment with **traditional machine learning algorithms, such as Logistic Regression and Random Forest,** to identify the model that achieves the highest accuracy in classifying the complaints efficiently.

Preprocessing

1. **Handling the null values:**

There aren't a lot of null values but it is important to take every possible action to improve the accuracy of the model so rows with null values are dropped.

2. **Handling duplicate rows**

In the dataset, many complaints have identical text but are classified into different categories. This duplication can lead to false positives and an artificially inflated accuracy when training the model. To address this issue, duplicate complaints are dropped, retaining only the first occurrence in the DataFrame. While this approach may not be ideal, it helps ensure that the model is trained on unique text entries, thereby reducing the risk of false positives during classification.

3. Handling Irrelevant Information

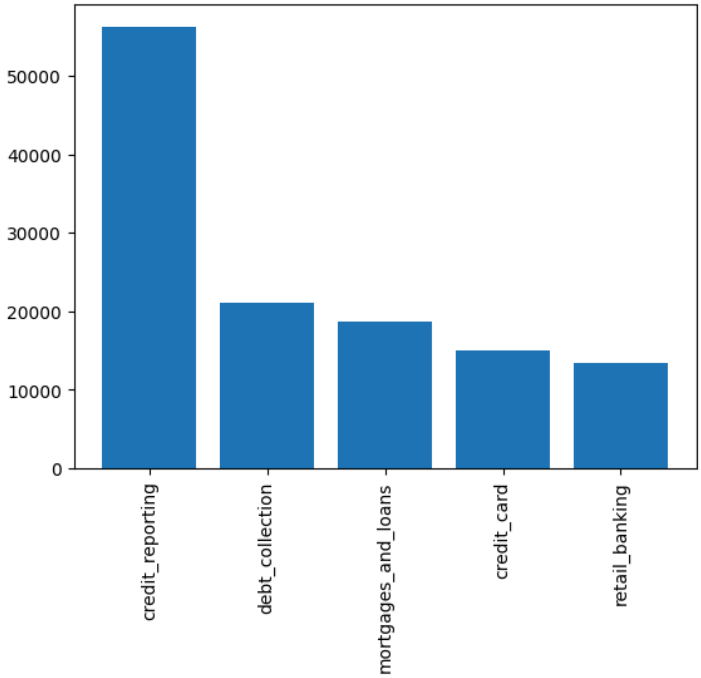
In the dataset, certain elements can be considered irrelevant information or noise, including punctuation marks, HTML tags, hyperlinks, stopwords, and placeholder expressions like "xxxx" that are used to hide sensitive information. These components do not contribute meaningfully to the classification task and can negatively impact the model's performance. Therefore, we will remove these irrelevant elements during the preprocessing phase to enhance the quality of the input data and improve the model's accuracy.

Resampling

During the data analysis phase, it was identified that the dataset has some serious imbalance; the complaints of the credit reporting class are irregularly high compared to others.
There are couple of approaches to deal with this kind of situations:

1. **Oversampling:** Here we increase the number of samples of minority classes by duplicating them.

2. **Undersampling:** Here we decrease the number of samples of the majority class. This is not recommended as this may lead to loss of valuable information.



3. **Synthetic Data Generation:** This approach involves creating synthetic samples for the minority classes rather than simply duplicating existing samples. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) generate new instances by interpolating between existing samples, helping to maintain diversity in the dataset while providing additional training data for the minority classes.

4. **ADASYN (Adaptive Synthetic Sampling)**: Similar to SMOTE, ADASYN focuses on generating synthetic samples for minority classes. However, it places greater emphasis on generating samples in regions where the density of minority class samples is lower, effectively adapting to the data distribution. This approach helps improve the model's ability to classify difficult-to-learn instances, further enhancing performance.

My approach of oversampling

In my attempts to address class imbalance, I initially used the `auto` attribute of the `RandomOverSampler`. However, this approach led to a decrease in model accuracy, as it merely duplicated existing samples without introducing meaningful variation. To improve the situation, I implemented an exponential sampling strategy that scales the number of samples based on their previous order.

The motivation behind this method is to create a significant difference in the number of samples across categories. The current distribution showed one class with a disproportionately high number of samples while others had relatively few. By applying exponential scaling, I aimed to adjust the sample sizes in a way that reduces the dominance of the majority class while still maintaining enough representation for the minority classes.

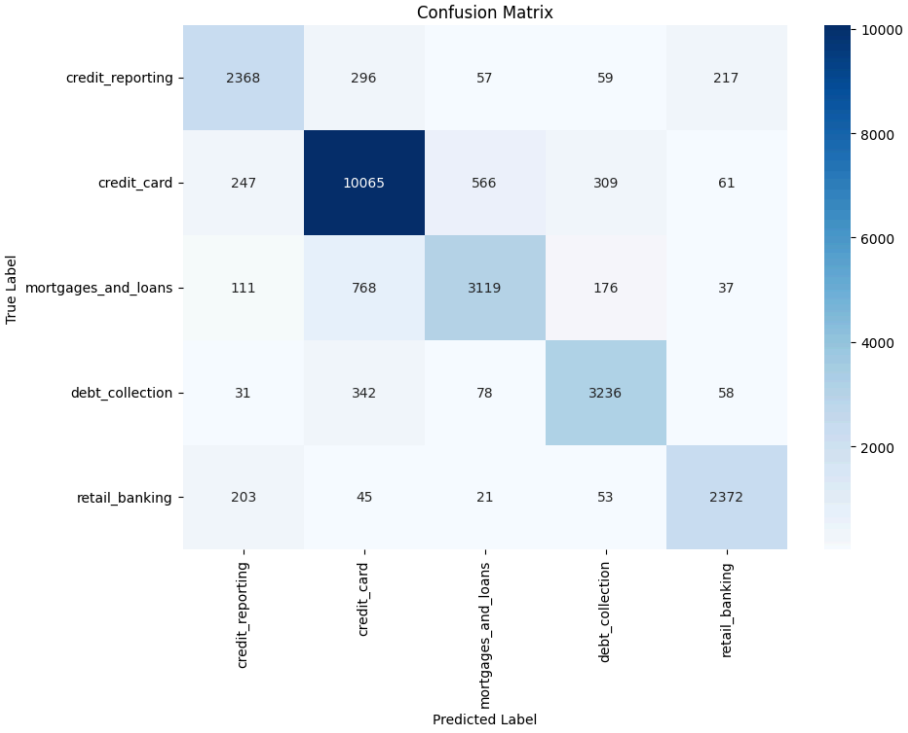
Although this approach may not be perfect, it has demonstrated a clear improvement in model accuracy compared to the traditional random oversampling methods. This strategy allows the model to learn from a more balanced dataset, thus enhancing its performance in classifying complaints accurately.

Other approaches:

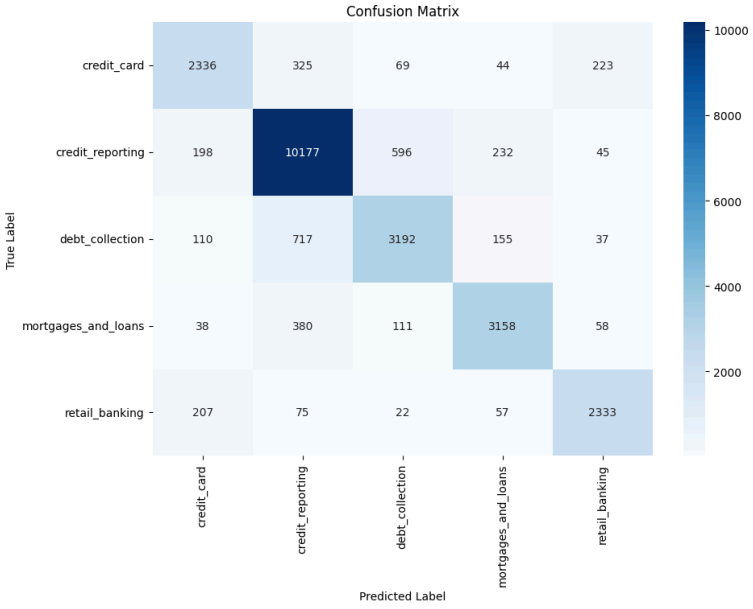
I also tried approaches like ADASYN, but it did not lead to a huge improvement in accuracy.

Overall, the accuracy is always in the range of 85-86 %.

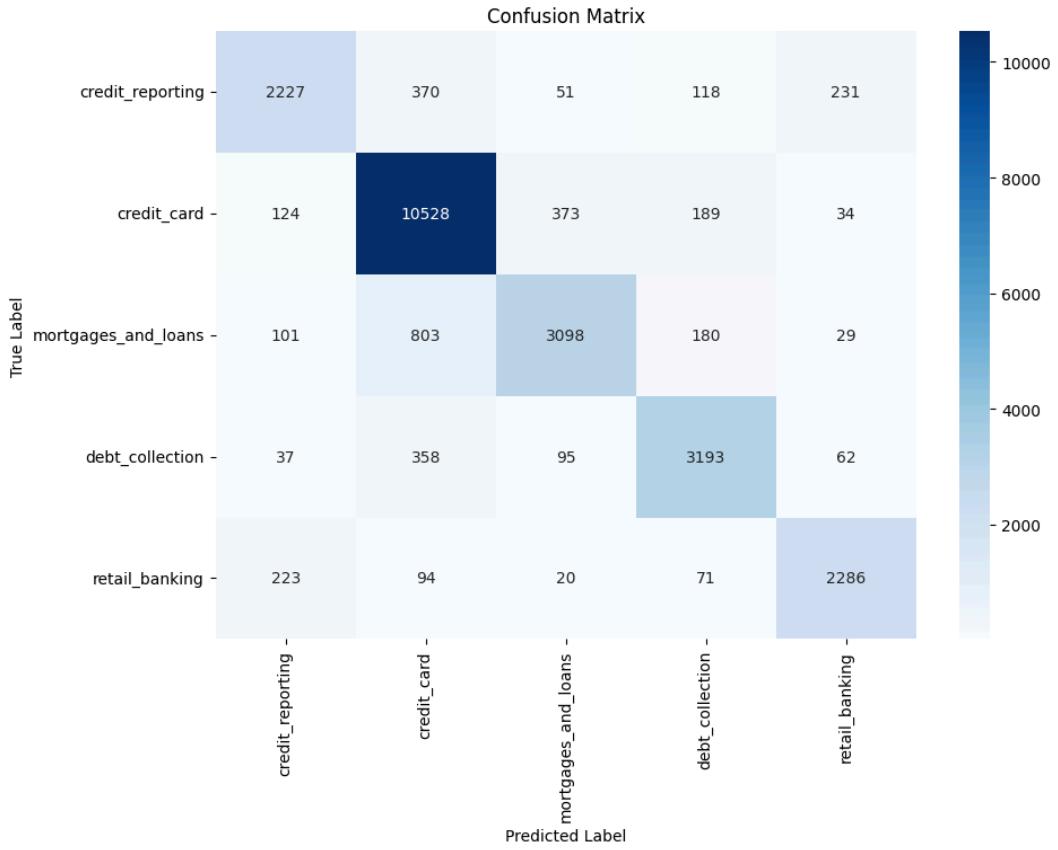
1. **Logistic regression:** Accuracy: 84.99



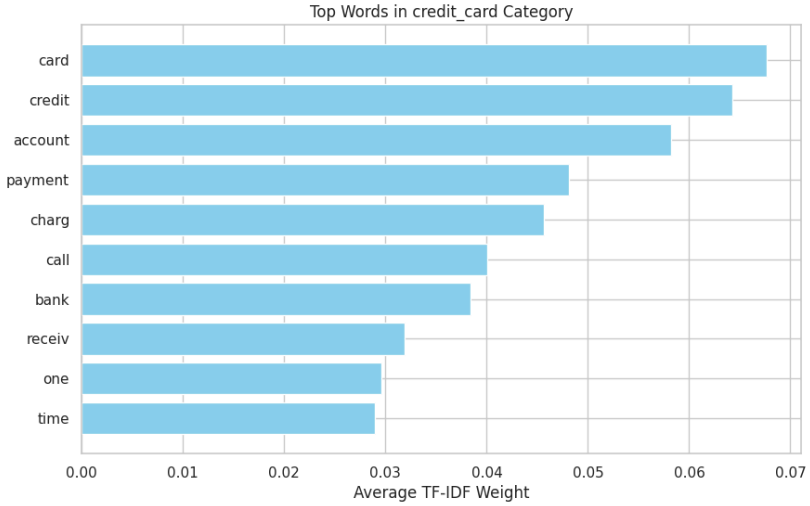
2. **Xgboost:** Accuracy: 85.1415 %

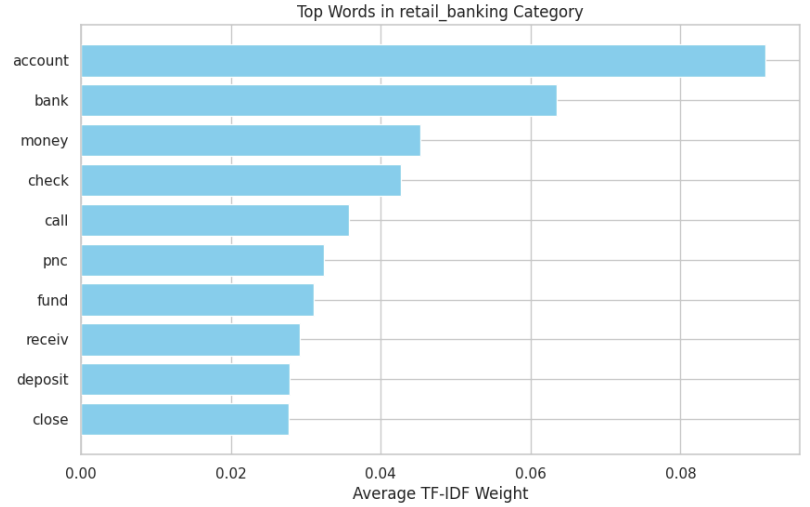


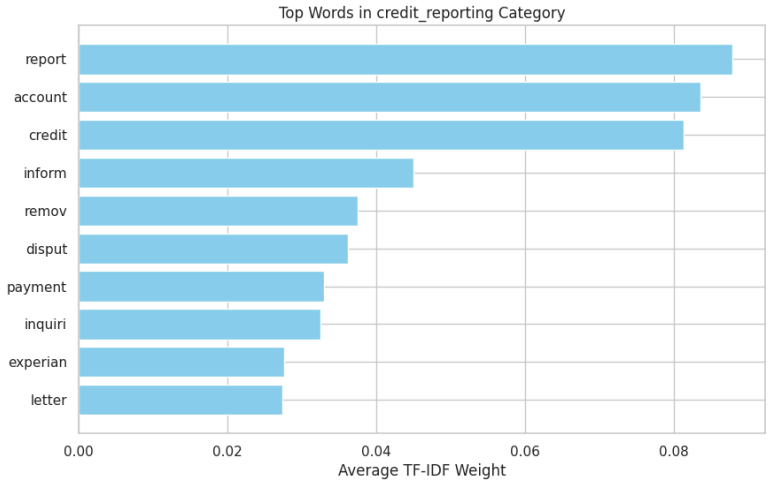
3. **RandomForestClassifier:** Accuracy: **85.68 %**

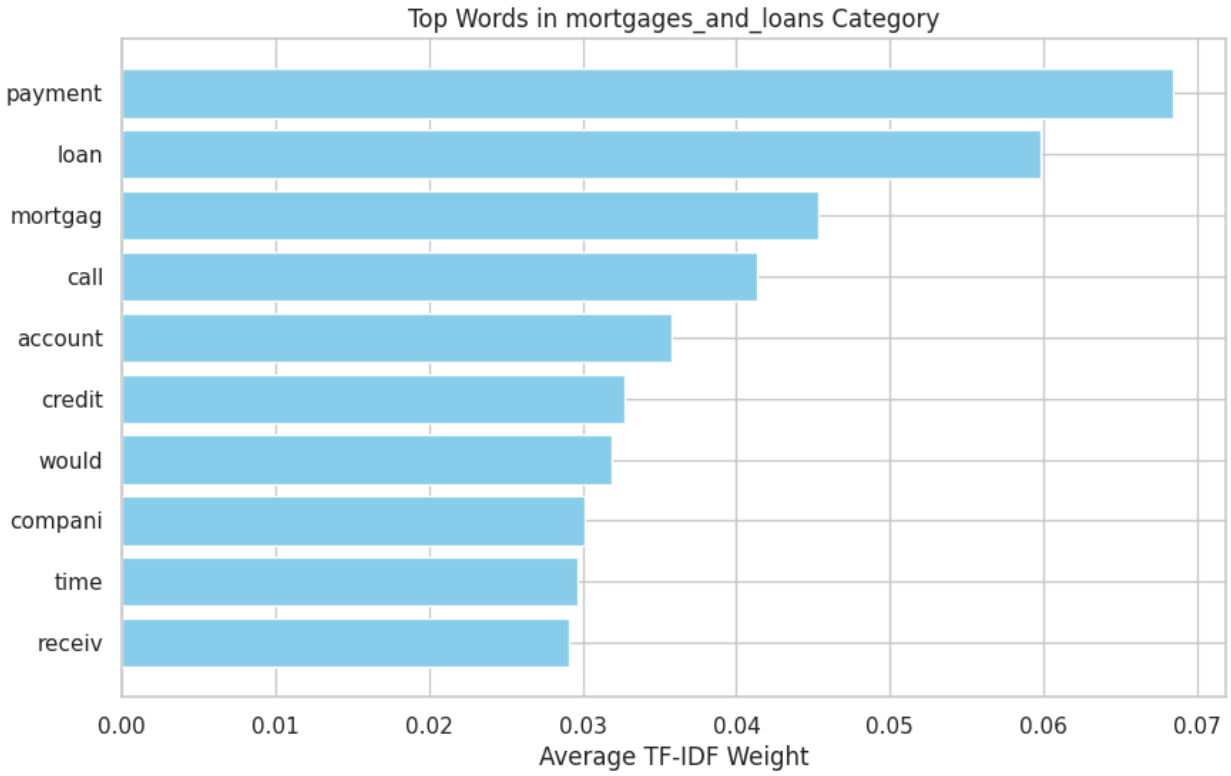


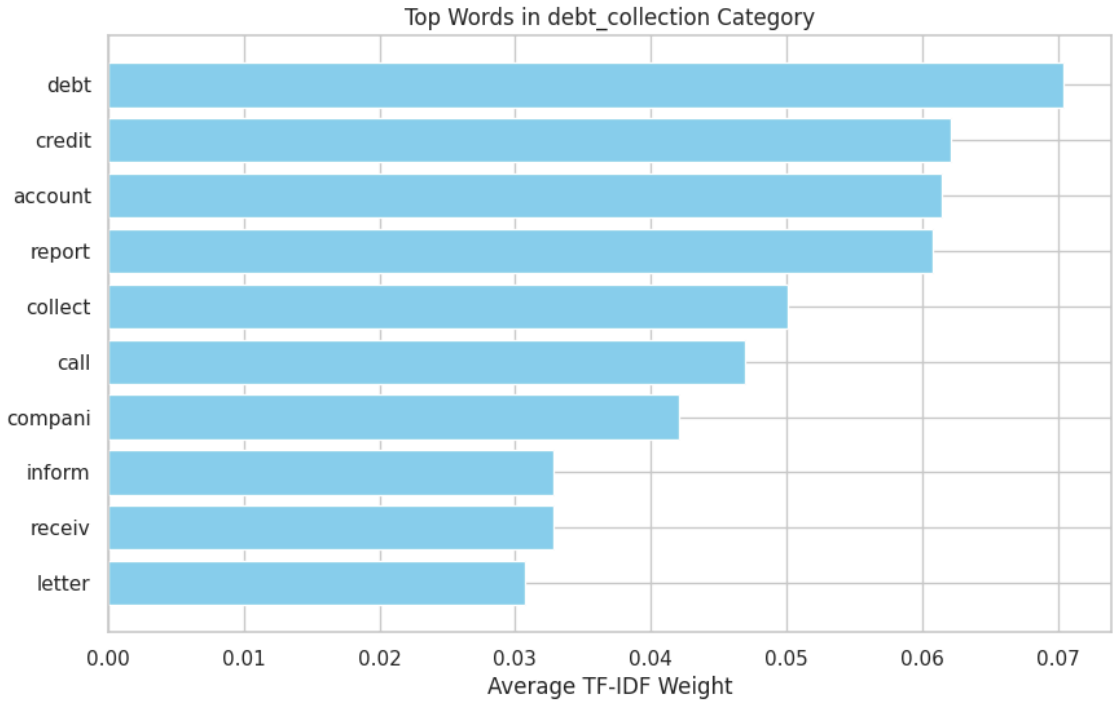
One common thing that was observed in all the confusion matrices, the model consistently fails to classify credit card and credit reporting, credit card and mortgages and loans, mostly because the common weights of the words in these classes are very similar.


Top Words in credit_card Category
card
credit
account
payment
charg
call
bank
receiv
one
time
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Average TF-IDF Weight


Top Words in retail_banking Category
account
bank
money
check
call
pnc
fund
receiv
deposit
close
0.00
0.02
0.04
0.06
0.08
Average TF-IDF Weight


Top Words in credit_reporting Category
report
account
credit
inform
remov
disput
payment
inquiri
experian
letter
0.00
0.02
0.04
0.06
0.08
Average TF-IDF Weight


Top Words in mortgages_and_loans Category
payment
loan
mortgag
call
account
credit
would
compani
time
receiv
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Average TF-IDF Weight


Top Words in debt_collection Category
debt
credit
account
report
collect
call
compani
inform
receiv
letter
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Average TF-IDF Weight

Routing Mechanism

I tried to make a simple web app with react frontend and flask based backend to facilitate this classification mechanism.

1. Model was pickled using the pickle package
2. Then it was loaded into the memory by the pickle loader by the flask backend.
3. Then api routes are created so that clients can make requests using their complaints to the server.
4. Server will then feed the input to the model by first vectorizing it by prefitted vectorizer and predict the category which is sent back to the user.

some key features:

1. basic role based authentication
2. jwt based verification
3. user can submit and view the category of the complaint
4. admin can view statistics and recent complaints
5. techstack used: Flask for backend, Reactjs for frontend, Mongodb for database
6. Frontend deployed on: Vercel, Backend deployed on: Render

Future scope of this web app may be to create multiple roles and an admin of a specific category can see and respond to specific complaints or forward it to higher authority.

User login: https://bank-complaints-classification.vercel.app/
Admin login: https://bank-complaints-classification.vercel.app/adminlogin

credentials for testing:

```
{
  "admin" : {
    "email" : "admin@gmail.com",
    "password" : "admin@123"
  },
  "user" : {
    "email" : "user@gmail.com",
    "password" : "user@123"
  }
}
```