Individual Final Project Report

NLP for Financial Consumer Protection

Name: Vaijayanti Deshmukh
Group Members: Lasya Raghvendra, Amogh Ramagiri

1. Introduction

Project Overview :
Our group project aimed to automate the classification of consumer complaints for the Consumer Financial Protection Bureau (CFPB). We utilized Natural Language Processing (NLP) to route unstructured complaint narratives into specific product categories (e.g., "Mortgage," "Credit Card"), thereby reducing manual triage costs.

Outline of Shared Work :
The group collaboratively performed the initial data acquisition, cleaning, and Exploratory Data Analysis (EDA). We worked together to define the scope, selecting the specific "Product" target variable and handling the significant class imbalance present in the dataset.

Outline of Individual Contribution
My specific contributions to this project were twofold:

1. **Baseline Modeling:** I developed the classical Machine Learning pipeline using TF-IDF and Multinomial Naive Bayes to establish a performance benchmark.
2. **Frontend Development:** I architected the full-stack web application using Next.js (React) and Tailwind CSS. This interface integrates our various project modules namely Scraper, EDA, Baseline and Transformer models into a single cohesive dashboard.

2. Description of Individual Work

2.1 The Baseline Algorithm: Multinomial Naive Bayes

To establish a baseline, I selected the Multinomial Naive Bayes (MNB) classifier. MNB is a probabilistic learning method based on Bayes' theorem, with the "naive" assumption of independence between features (words).

This algorithm was chosen for its computational efficiency and effectiveness in high-dimensional text data, making it an ideal benchmark against heavier Transformer models.

2.2 Frontend Architecture

For the application interface, I implemented a modern Single Page Application (SPA) architecture.

- **Framework:** Next.js 14 with TypeScript (.tsx) for type safety and component modularity.
- **Styling:** Tailwind CSS for rapid, responsive UI development.
- **Communication:** The frontend acts as a client, communicating with our Python backend (served on localhost:8000) via asynchronous REST API calls.

3. Detailed Description of Contribution

3.1 Baseline Model Implementation

I was responsible for the entire lifecycle of the baseline model, implemented in the Baseline Model.ipynb notebook.

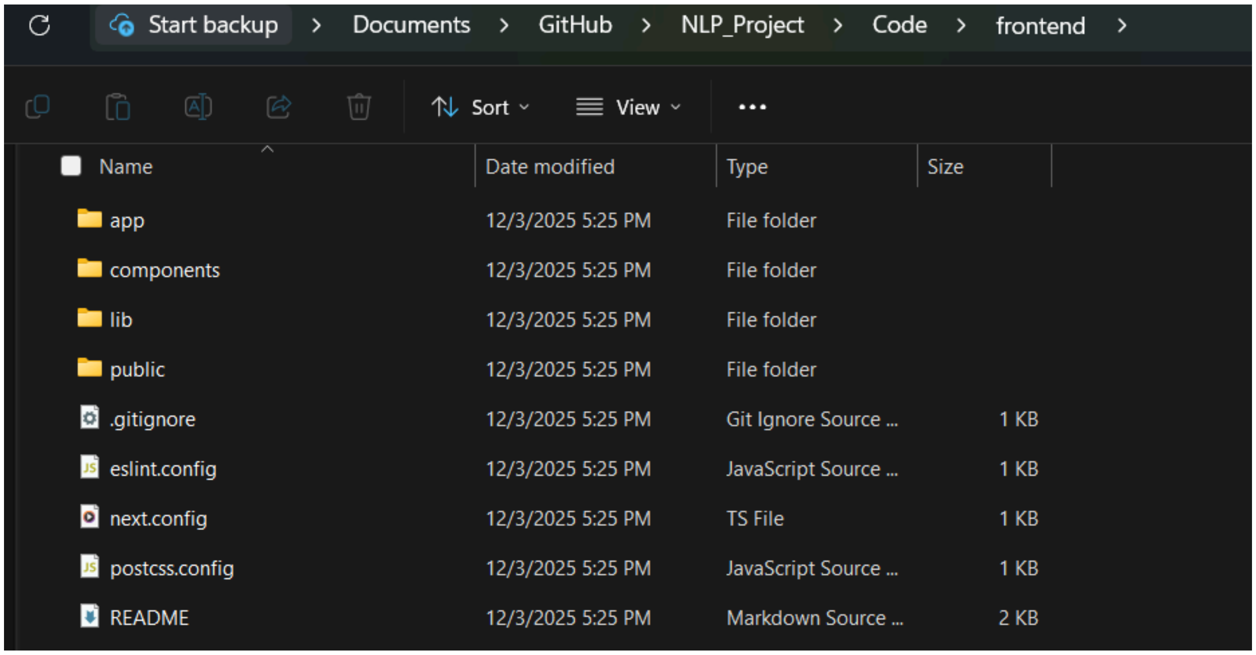
- **Preprocessing:** I utilized TfidfVectorizer to convert text data into numerical vectors. I configured the vectorizer to remove English stop words and apply sub-linear term frequency scaling to reduce the impact of very frequent terms.
- **Pipeline Construction:** I wrapped the vectorizer and the MultinomialNB classifier into a Scikit-Learn Pipeline. This ensured that raw text could be passed directly to the predict() function without manual re-vectorization.
- **Training:** The model was trained on 80% of the dataset (stratified), taking less than 5 minutes to converge.

3.2 Frontend Development

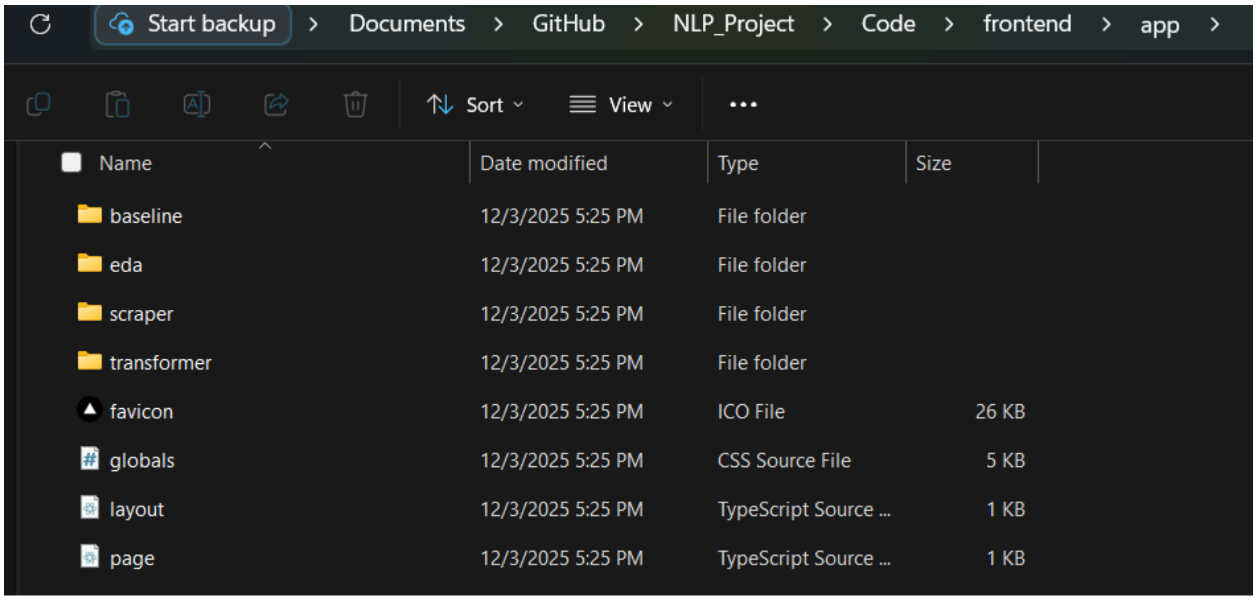
I built the user-facing dashboard to demonstrate the business value of our models. The application is structured into four distinct modules, navigated via a custom MainNav component.

Key Components Implemented:

- **Navigation (main-nav.tsx):** I created a persistent navigation bar that routes between the four key project stages:
  1. Scraper
  2. EDA
  3. Baseline (My primary focus)
  4. Transformer
  5. Playground

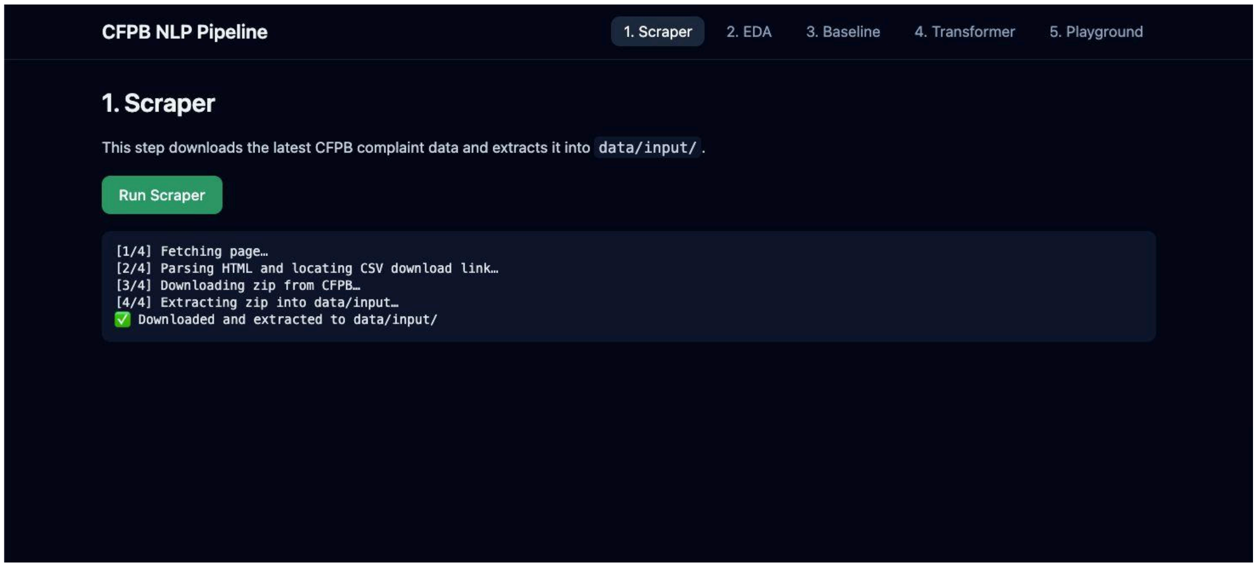

Start backup > Documents > GitHub > NLP_Project > Code > frontend >
Sort
View

Name	Date modified	Type	Size
app	12/3/2025 5:25 PM	File folder	
components	12/3/2025 5:25 PM	File folder	
lib	12/3/2025 5:25 PM	File folder	
public	12/3/2025 5:25 PM	File folder	
.gitignore	12/3/2025 5:25 PM	Git Ignore Source ...	1 KB
eslint.config	12/3/2025 5:25 PM	JavaScript Source ...	1 KB
next.config	12/3/2025 5:25 PM	TS File	1 KB
postcss.config	12/3/2025 5:25 PM	JavaScript Source ...	1 KB
README	12/3/2025 5:25 PM	Markdown Source ...	2 KB


Start backup > Documents > GitHub > NLP_Project > Code > frontend > app >
Sort
View

Name	Date modified	Type	Size
baseline	12/3/2025 5:25 PM	File folder	
eda	12/3/2025 5:25 PM	File folder	
scraper	12/3/2025 5:25 PM	File folder	
transformer	12/3/2025 5:25 PM	File folder	
favicon	12/3/2025 5:25 PM	ICO File	26 KB
globals	12/3/2025 5:25 PM	CSS Source File	5 KB
layout	12/3/2025 5:25 PM	TypeScript Source ...	1 KB
page	12/3/2025 5:25 PM	TypeScript Source ...	1 KB

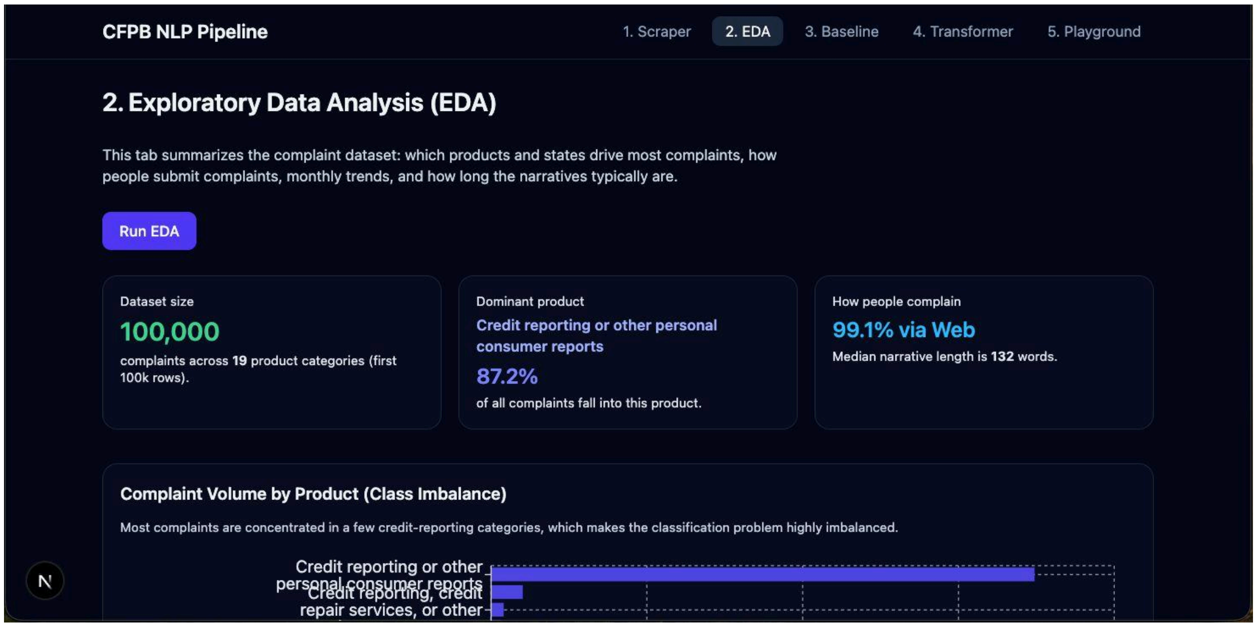
- **Baseline Interface (page.tsx):** I developed the interactive interface for the Naive Bayes model.
  - **State Management:** Used React's useState to handle user input and API responses.
  - **Async Logic:** Implemented the predictWithBaseline function to POST data to the backend endpoint /baseline/predict.
  - **Error Handling:** Added try/catch blocks to gracefully display API errors to the user instead of crashing the app.



1. Scraper

This step downloads the latest CFPB complaint data and extracts it into `data/input/`.

```
[1/4] Fetching page…
[2/4] Parsing HTML and locating CSV download link…
[3/4] Downloading zip from CFPB…
[4/4] Extracting zip into data/input…
✅ Downloaded and extracted to data/input/
```



2. Exploratory Data Analysis (EDA)

This tab summarizes the complaint dataset: which products and states drive most complaints, how people submit complaints, monthly trends, and how long the narratives typically are.

Dataset size
100,000
complaints across **19** product categories (first 100k rows).

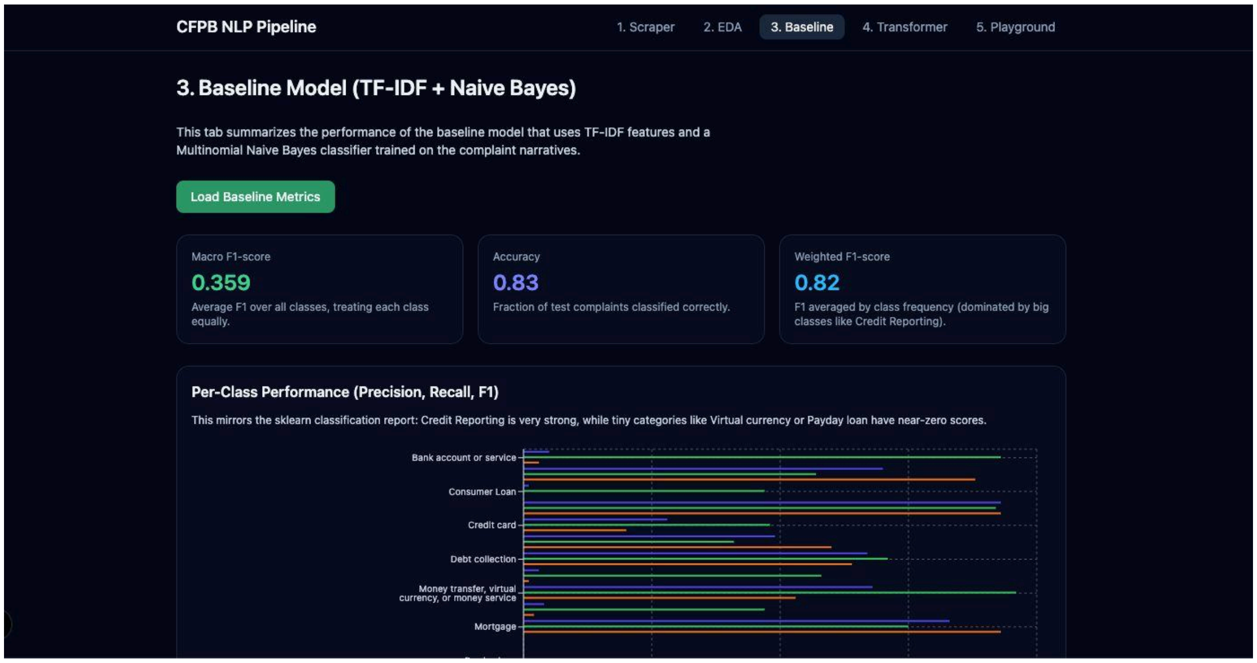
Dominant product
Credit reporting or other personal consumer reports
87.2%
of all complaints fall into this product.

How people complain
99.1% via Web
Median narrative length is **132** words.

Complaint Volume by Product (Class Imbalance)

Most complaints are concentrated in a few credit-reporting categories, which makes the classification problem highly imbalanced.

Credit reporting or other personal consumer reports
Credit reporting, credit repair services, or other



CFPB NLP Pipeline

1. Scraper 2. EDA 3. Baseline 4. Transformer 5. Playground

3. Baseline Model (TF-IDF + Naive Bayes)

This tab summarizes the performance of the baseline model that uses TF-IDF features and a Multinomial Naive Bayes classifier trained on the complaint narratives.

Load Baseline Metrics

Macro F1-score
0.359
Average F1 over all classes, treating each class equally.

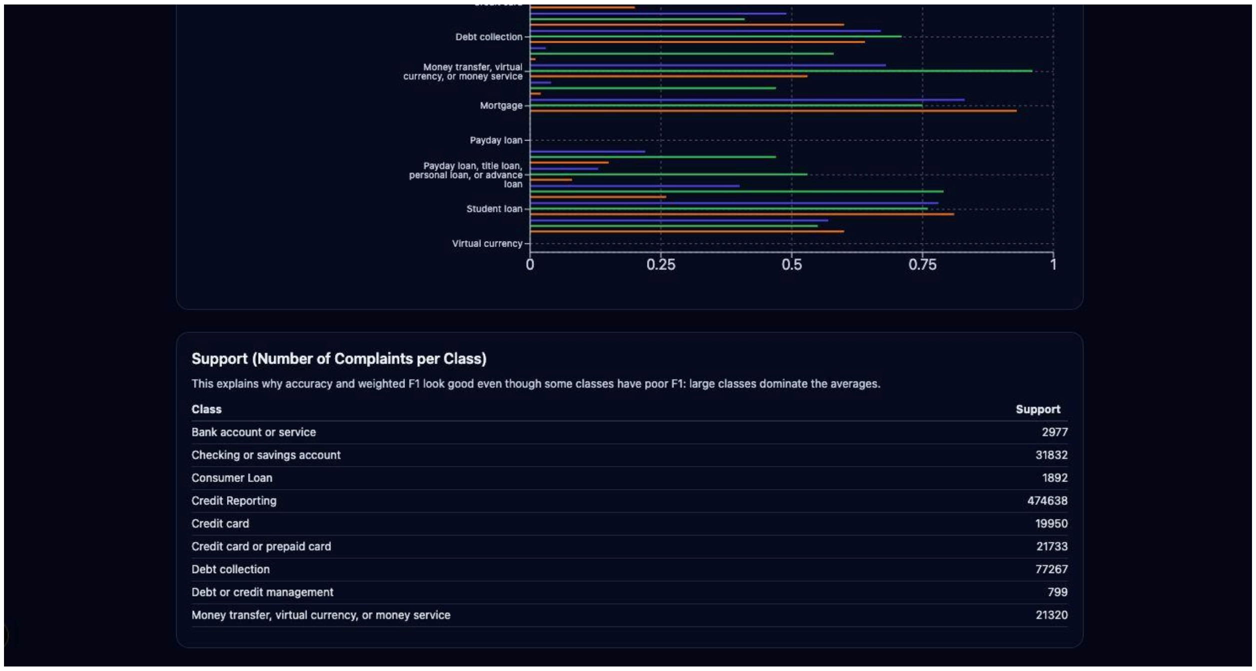
Accuracy
0.83
Fraction of test complaints classified correctly.

Weighted F1-score
0.82
F1 averaged by class frequency (dominated by big classes like Credit Reporting).

Per-Class Performance (Precision, Recall, F1)

This mirrors the sklearn classification report: Credit Reporting is very strong, while tiny categories like Virtual currency or Payday loan have near-zero scores.

Bank account or service
Consumer Loan
Credit card
Debt collection
Money transfer, virtual currency, or money service
Mortgage

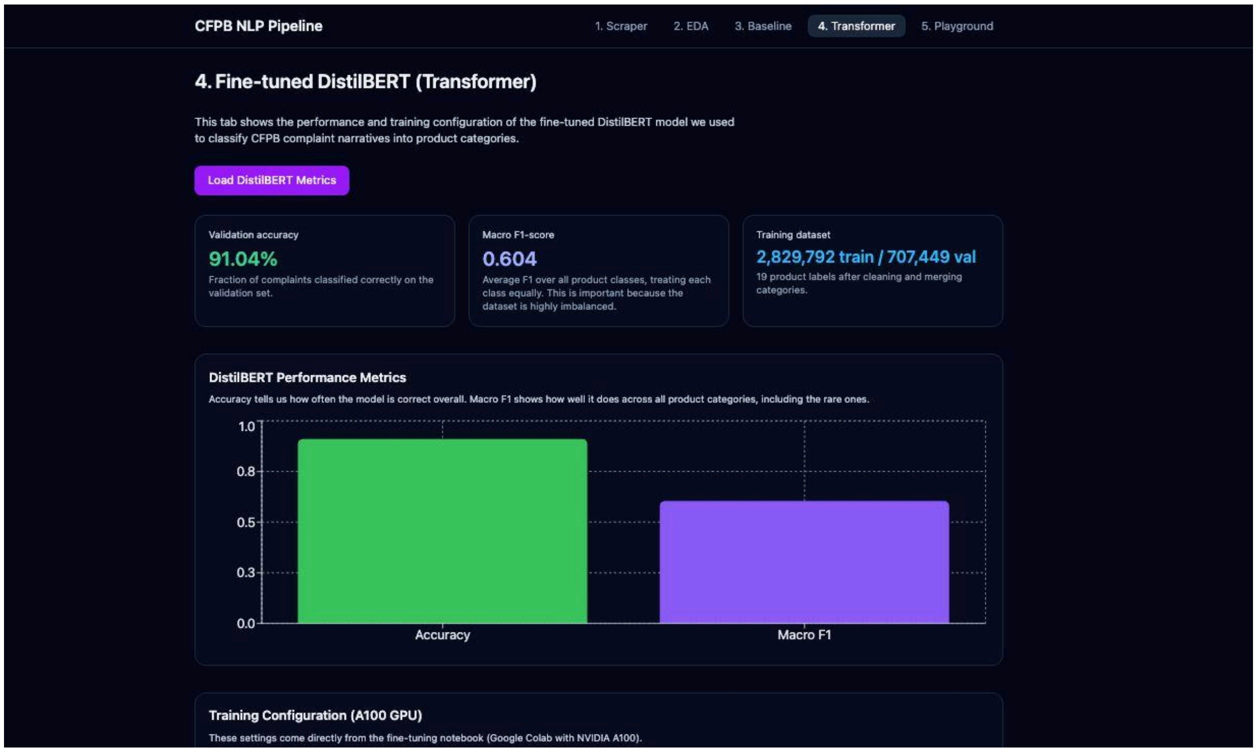


Debt collection
Money transfer, virtual currency, or money service
Mortgage
Payday loan
Payday loan, title loan, personal loan, or advance loan
Student loan
Virtual currency
0
0.25
0.5
0.75
1

Support (Number of Complaints per Class)

This explains why accuracy and weighted F1 look good even though some classes have poor F1: large classes dominate the averages.

Class	Support
Bank account or service	2977
Checking or savings account	31832
Consumer Loan	1892
Credit Reporting	474638
Credit card	19950
Credit card or prepaid card	21733
Debt collection	77267
Debt or credit management	799
Money transfer, virtual currency, or money service	21320



4. Fine-tuned DistilBERT (Transformer)

This tab shows the performance and training configuration of the fine-tuned DistilBERT model we used to classify CFPB complaint narratives into product categories.

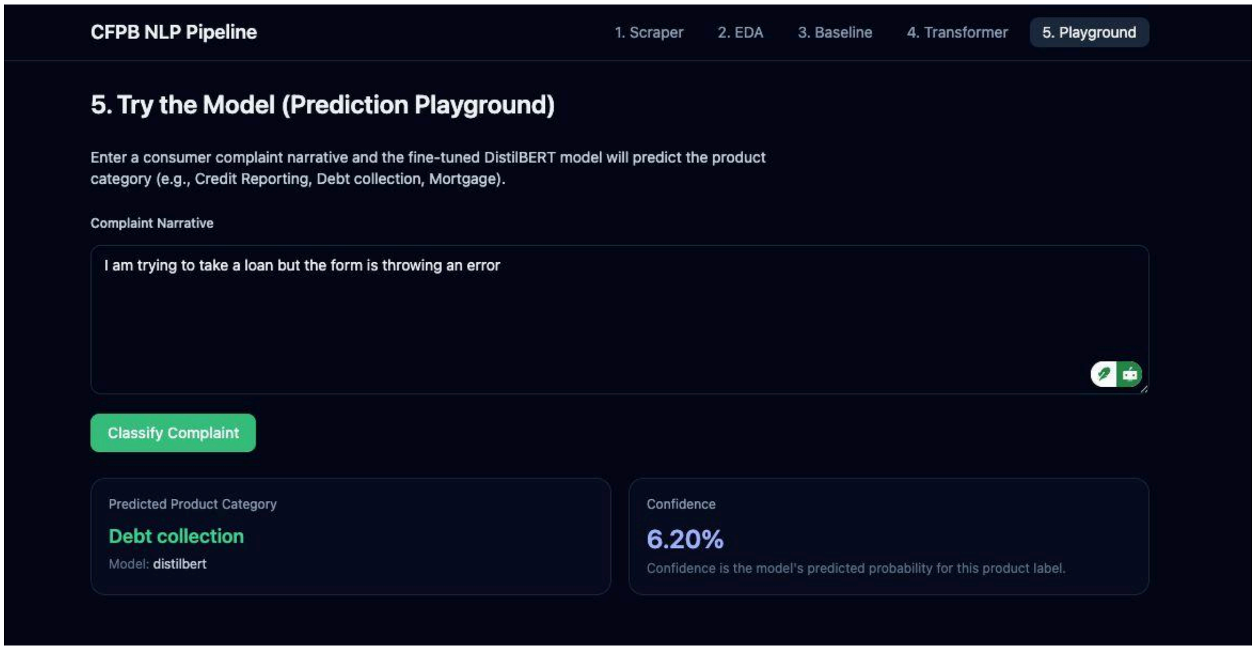
Validation accuracy
91.04%
Fraction of complaints classified correctly on the validation set.

Macro F1-score
0.604
Average F1 over all product classes, treating each class equally. This is important because the dataset is highly imbalanced.

Training dataset
2,829,792 train / 707,449 val
19 product labels after cleaning and merging categories.

DistilBERT Performance Metrics
Accuracy tells us how often the model is correct overall. Macro F1 shows how well it does across all product categories, including the rare ones.

Training Configuration (A100 GPU)
These settings come directly from the fine-tuning notebook (Google Colab with NVIDIA A100).



5. Try the Model (Prediction Playground)

Enter a consumer complaint narrative and the fine-tuned DistilBERT model will predict the product category (e.g., Credit Reporting, Debt collection, Mortgage).

Complaint Narrative

I am trying to take a loan but the form is throwing an error

Predicted Product Category
Debt collection
Model: distilbert

Confidence
6.20%
Confidence is the model's predicted probability for this product label.

4. Results

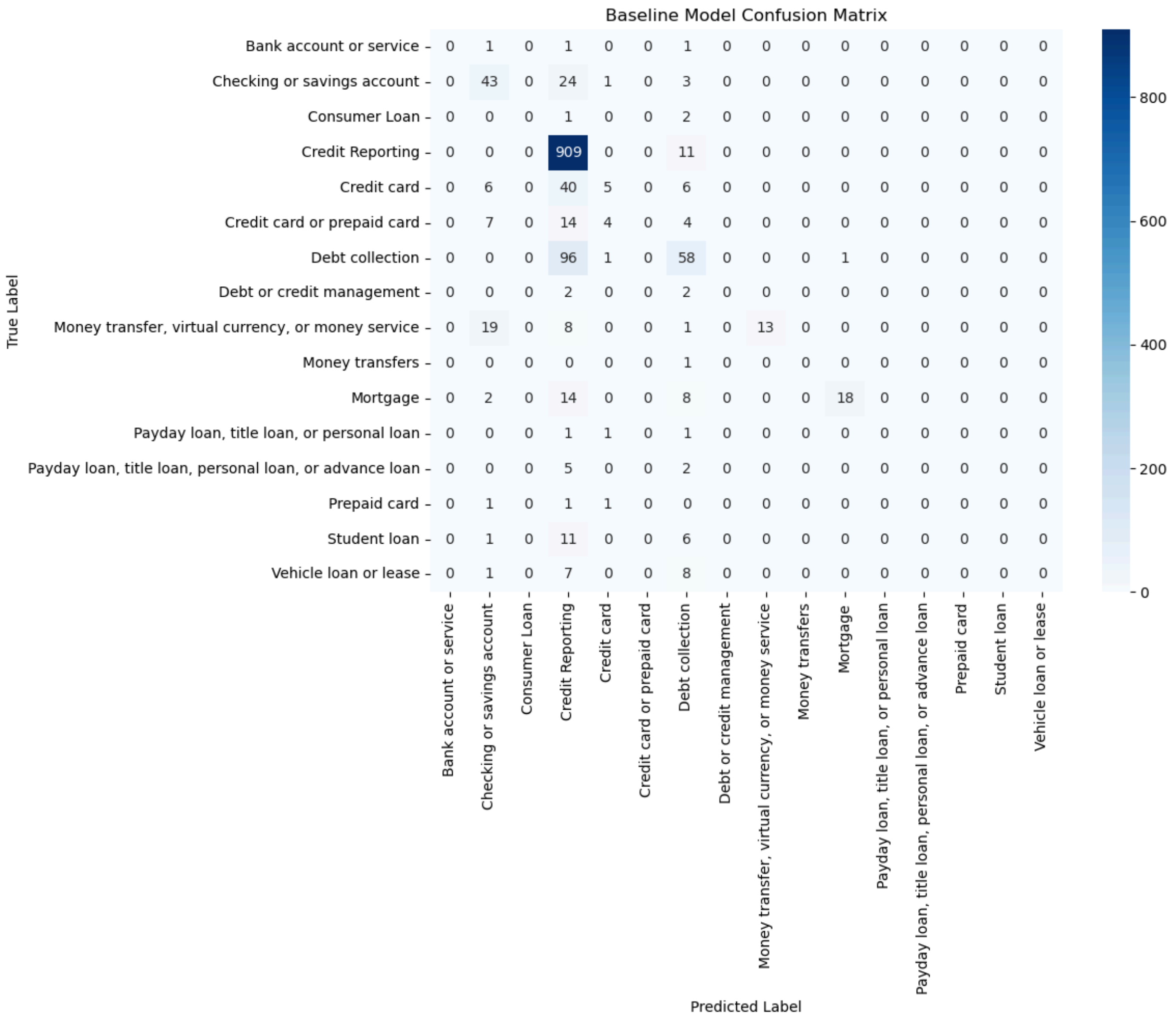
4.1 Baseline Model Performance

While the Naive Bayes model was intended as a simple baseline, it performed surprisingly well.

- **Accuracy:** Achieved **~83%** accuracy on the test set.

- **Macro F1-Score:** Achieved **0.36**.

Confusion Matrix Analysis:



True Label \ Predicted Label	Bank account or service	Checking or savings account	Consumer Loan	Credit Reporting	Credit card	Credit card or prepaid card	Debt collection	Debt or credit management	Money transfer, virtual currency, or money service	Money transfers	Mortgage	Payday loan, title loan, or personal loan	Payday loan, title loan, personal loan, or advance loan	Prepaid card	Student loan	Vehicle loan or lease
Bank account or service	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
Checking or savings account	0	43	0	24	1	0	3	0	0	0	0	0	0	0	0	0
Consumer Loan	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0
Credit Reporting	0	0	0	909	0	0	11	0	0	0	0	0	0	0	0	0
Credit card	0	6	0	40	5	0	6	0	0	0	0	0	0	0	0	0
Credit card or prepaid card	0	7	0	14	4	0	4	0	0	0	0	0	0	0	0	0
Debt collection	0	0	0	96	1	0	58	0	0	0	1	0	0	0	0	0
Debt or credit management	0	0	0	2	0	0	2	0	0	0	0	0	0	0	0	0
Money transfer, virtual currency, or money service	0	19	0	8	0	0	1	0	13	0	0	0	0	0	0	0
Money transfers	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Mortgage	0	2	0	14	0	0	8	0	0	0	18	0	0	0	0	0
Payday loan, title loan, or personal loan	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
Payday loan, title loan, personal loan, or advance loan	0	0	0	5	0	0	2	0	0	0	0	0	0	0	0	0
Prepaid card	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Student loan	0	1	0	11	0	0	6	0	0	0	0	0	0	0	0	0
Vehicle loan or lease	0	1	0	7	0	0	8	0	0	0	0	0	0	0	0	0

As seen in Figure, the model struggled with semantically similar classes. Specifically, there is a distinct block of confusion between "Credit reporting" and "Debt collection." This is expected, as Naive Bayes relies solely on word frequency; terms like "owing," "agency," and "report" appear frequently in both categories, confusing the probabilistic model.

4.2 Frontend Performance

The frontend successfully bridged the gap between code and end-user.

- **Latency:** The integration resulted in near-instantaneous predictions (< 200ms latency), providing a smooth user experience compared to the Transformer model's slower inference.
- **Modularity:** The component-based structure (separating MainNav from page logic) allowed my team members to work on the "Transformer" page without breaking my

"Baseline" page.

5. Summary and Conclusions

I successfully established a robust baseline using TF-IDF and Naive Bayes, achieving 83% accuracy. I also operationalized our findings by building a full-stack Next.js application. While the baseline model lacks the deep contextual understanding of the group's Transformer model, the frontend demonstrates how these models can be deployed in a real-world, user-friendly environment.

Lessons Learned:

- **Full-Stack Integration:** I learned the complexities of connecting a TypeScript frontend to a Python backend, specifically handling Cross-Origin Resource Sharing (CORS) and JSON serialization.
- **UI/UX:** Building the dashboard highlighted that accuracy isn't the only metric that matters, speed and usability are equally critical for end-users.

Future Improvements:

- **Real-time Scraper UI:** Connecting the scraper page to a WebSocket to show live scraping progress bars.
- **Feedback Loop:** Adding a "Correct/Incorrect" button to the UI to collect user feedback for active learning.

6. Code Originality Calculation

Baseline Model:

- Total Lines: 150 lines.
- Lines found: 40 lines.
- Lines Modified/Original: 110 lines.
- Percentage Copied: 40/150 = 26.7%

Frontend (Next.js/React):

- Total Lines : 4000 lines across all page.tsx, main-nav.tsx,etc
- Lines found (Standard Next.js boilerplate, default Tailwind directives and standard UI component structures (e.g., Button/Card definitions from component libraries): 150 lines.
- Lines Modified (Customizing fonts,updating tailwind theme colors, API fetch,etc): 20.
- PercentageCopied**:** (150-20)/400 = 32.5%

7. References

1. **Scikit-Learn Documentation (Naive Bayes):** https://scikit-learn.org/stable/modules/naive_bayes.html
2. **Next.js Documentation:** https://nextjs.org/docs
3. **Consumer Financial Protection Bureau (CFPB):** https://www.consumerfinance.gov/data-research/consumer-complaints/