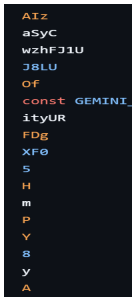
Project Title: AI-Driven Credit Risk Analytics and ECL Visualization Tool

0. Project Setup

Pull code from the github repo (https://github.com/ritviks-github/expected-credit-loss-calculation)
It consists of 3 folders namely client, backend, models. The client folder has the React frontend code for it, Backend code has the database models and different routes like login etc, and the Models folder has the code for Fast API for serving decision tree model to predict ECL from the dataset used. I have used mongoDB compass for database. To setup your own compass, in the db.js file in Backend folder, just add your own URI, there is a upload_excel.js file to add the dataset used in the mongoDB compass. And to login, you need to add user to mongoDB compass, there is a add_user.js file in which you can add dummy users. With role “cro” and “analyst”. In the ask_ai file in routes folder, i have added the gemini API key (it will work cuz of the way it is added to code).



That is all about the backend. For frontend just use **“npm run dev”** to start the react app, in this, just do **navigate to “/login”** route to start using.
To start the FastAPI, in Models run **“python main.py”**
The frontend port is 5173, backend port is 8080 and fastAPI port is 8000.

1. Project Overview

This project aims to build a tool for credit analysts and CROs (Chief Risk Officers) to review loan performance, calculate Expected Credit Loss (ECL), and make data-driven decisions. It integrates AI (Gemini), interactive visualizations, real-time chat, and document management, with secure login access for two roles.

2. Technology Stack

- **Frontend:** React.js (Vite), Recharts (for graphs)
- **Backend:** Fast API (for ML serving datapoints, ML model training using sklearn), ExpressJS
- **AI Integration:** Gemini (via prompt-based suggestions)
- **Database:** MongoDB (local via Compass)
- **Authentication:** JWT for auth, Bcrypt for password encryption

- **File Storage:** Local file system (uploads folder)
- **Chat System:** REST-based messaging system

3. Dataset Used

- **Source:** Kaggle - Bank Loan Modelling
- **Purpose:** To segment users (by gender, occupation, area, etc.) and compute ECL (Expected Credit Loss) for each.

4. User Roles

Analyst

- Can view segmented loan data.
- See ECL curves for each segment.
- Upload performance reports.
- Use Gemini AI to get suggestions based on trends.
- Communicate with CRO via a chat interface.
- Perform cross-segment comparisons.

CRO

- View all uploaded reports.
- Perform cross-segment comparisons.
- View full dataset and all analytics.
- Communicate with analysts via chat interface.

5. Key Features

ECL Graphs & Comparisons

- ECL data visualized using Recharts.
- Analysts and CROs can select filters to compare different segments.

AI Suggestions (Gemini)

- Analyst can input queries like "Should I reduce disbursement to rural self-employed women?"
- Gemini responds with insight based on data.
- It is a kind of a very minimal RAG pipeline (obviously without the embeddings of queries and data) in which the data of the dataset is augmented along with the user query.

Chat System

- REST-based.

- Stores each message with:
    - Content
    - Timestamp
    - Sender Role (Analyst or CRO)
- Enables real-time discussions on loan strategies.

Report Uploading

- Analyst can upload performance reports (PDFs).
- Metadata (file name, upload date, uploader) stored in MongoDB.
- Reports can be downloaded from frontend.

6. Security

- JWT used for secure login session handling.
- Bcrypt used to hash and store passwords securely in MongoDB.

7. Data Storage and Access

- MongoDB Compass used locally to store:
    - User credentials
    - Messages (chat logs)
    - Report metadata
- Uploaded reports stored in `/uploads` directory and fetched using unique filenames.

8. Development Highlights

- Built using modular Express routes for login, report handling, AI queries.
- Ensured role-based access control across UI.
- Debugged CORS issues and file serving routes.
- Created interactive and minimal UI with segment filter support.

9. Future Scope

- Switch to cloud MongoDB (e.g., Atlas).
- Enable real-time Socket.io-based chat.
- Add report summarization using LLMs

10. Use of AI while building

- Use v0.dev for getting the UI designing ideas

- Use of gemini for getting to know about interactivity provided by rechart JS library to display datapoints in a graph format interactively.

GitHub Repo: https://github.com/ritviks-github/expected-credit-loss-calculation