Progress Report 1 - Developing a Large Tabular Model for Predicting Credit Approvals and Analyzing Customer Behavior in Financial Institutions

Atharva Kulkani, Minh Dang, Nicolas A. Corgono, Ryu Sonoda, Varun Agarwal, Yu Zheng Lim

Oct 26, 2024

1. Problem Statement and Progress Overview

Financial institutions, like our industry partner TD Bank, are increasingly harnessing data science to enhance customer service, manage risks, and optimize operational efficiency. A major challenge lies in understanding and predicting customer behaviors and characteristics. This project aims to develop a tabular foundational model (TabFM) or a Large Tabular Model (LTM) for bank customers by leveraging publicly available data[1], to

a. predict if a customer will get approved for a credit product;
b. predict other variables in the dataset, and

assess both on similar metrics.

Currently, these can be done by traditional models like logistic regression and decision trees [1], which have been supplemented by advanced techniques such as gradient boosting [2] and deep learning [3, 4]. However, they often require extensive retraining for new tasks. On the other hand, TabFM leverages zero-shot and in-context learning, enabling it to adapt to new tasks with minimal retraining, thus improving model efficiency and effectiveness.

Our overall approach follows these stages:

1. Data preparation: Gather, explore, and preprocess the dataset. Split it into training and test sets.
2. Model Development: build baseline Models. Apply traditional ML models such as logistic regression and XGBoost
3. TabFM Implementation:
    a. Use zero-shot learning for initial predictions.
    b. Apply in-context learning with example prompts for task adaptation. (Choose another target variable to predict on)
4. Evaluation: Compare TabFM's performance with baseline models using sensible metrics. What features drive a customer's approval likelihood?
5. Test the model's generalization in addressing other use cases, using other target variables
6. Documentation and Knowledge Transfer

In this report, we address stages 1-2.

[1] https://www.kaggle.com/datasets/saurabhbadole/leading-indian-bank-and-cibil-real-world-dataset

2. Literature Review

The main research paper that TD advised us to focus on, published by Wen et al [1] from Microsoft, is the core reference for our work and serves as the basis for the work we intend to do in the second half of our project.

Abstract

Generative Tabular Learning (GTL) brings the powerful capabilities of large language models (LLMs) to tabular data, tapping into features like zero-shot generalization and in-context learning. By continuing to pre-train on 384 diverse datasets, GTL enables LLMs to understand numerical relationships and domain-specific knowledge. The approach is built on LLaMA-2 and is fine-tuned for performance in tabular tasks.

Introduction

Ensemble and tree-based models have been the go-to for predictive tasks, but they fall short when it comes to creating models that can generalize across different datasets with limited labeled data. This is where universal models are needed — models that can easily transfer to new datasets with just a few labeled examples. Pre-training deep learning models specifically for tabular data is a promising approach to fill this gap.

Two standout features of these models are: (1) After pre-training, they can quickly adapt to new tasks by using task-specific prompts, which allows for efficient knowledge transfer, and (2) they have in-context learning capabilities, meaning they can learn from a few labeled examples without needing to be fine-tuned.

Most previous work has focused too heavily on fine-tuning, missing out on key benefits like in-context learning and few-shot generalization. On top of that, traditional models struggle to handle tabular data formatted as language and often fail to incorporate critical domain knowledge.

GTL Framework

GTL introduces a pipeline that converts raw tabular data into an instruction-based format, designed to capture the relationships between numerical features and targets while also including meta-information. This approach is key to enabling zero-shot and few-shot learning without needing to update the model's parameters. For example, LLaMA-GTL achieves state-of-the-art performance in several few-shot learning scenarios.

While existing models, like TabPFN, focus mainly on classification, GTL goes beyond this by introducing Language-Interfaced Fine-Tuning (LIFT), which extends its use for more complex tabular tasks through language-based templates.

GTL for LLMs

The GTL process for converting tabular data into an instruction-based format consists of nine steps. It starts with gathering datasets from domains like shopping, health, and finance, followed by organizing the data into tables that include necessary details like columns and value explanations (optional). The problem at hand, whether classification or regression, will then be formatted for either zero-shot or in-context learning.

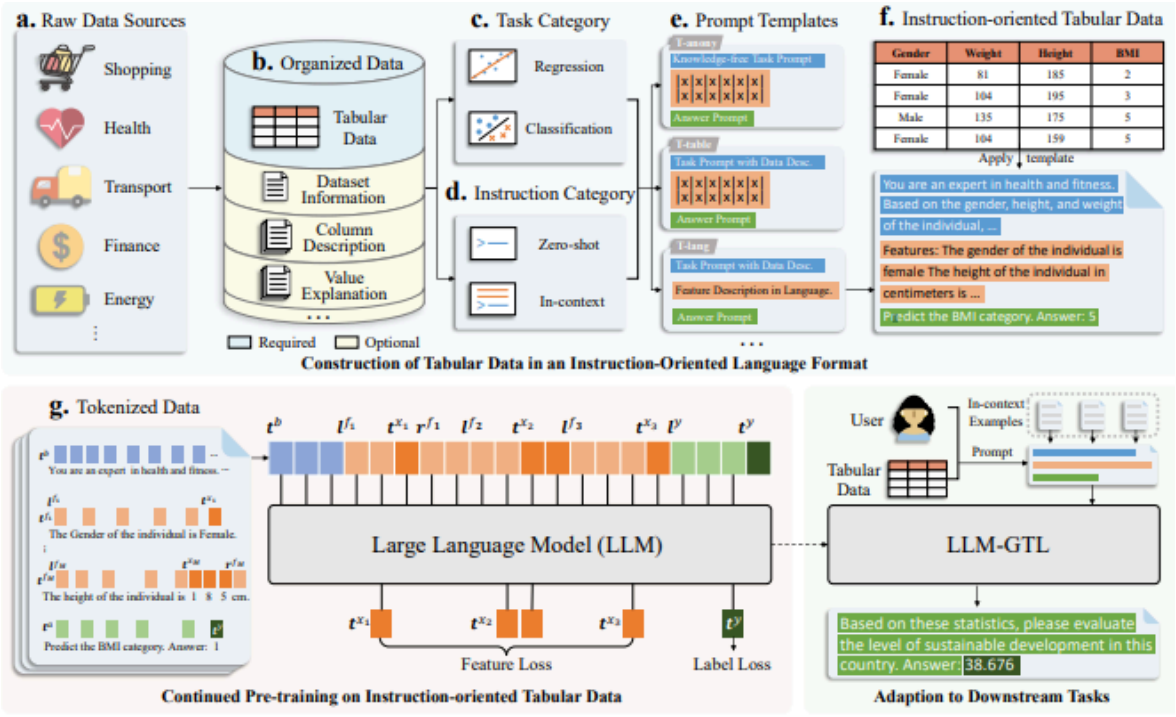

Fig. 1: Overall approach: GTL Pipeline for tabular data construction

The pipeline is briefly described as follows:

1. A variety of tabular datasets from different domains is collected, covering a broad range of predictive tasks.
2. For each task, the data and its meta-information are transformed from tabular format into a language-based, instruction-oriented format. This format includes:
    a. Task instructions that explain the background and goal of the task.
    b. Feature descriptions that provide details on the feature values.
    c. Answer descriptions that specify the prediction targets.
3. Pre-train an LLM on that data, to produce a GTL-enhanced LLM, or a TabFM.

Templates for Data Formatting

To further simplify the process, GTL uses three distinct templates for formatting data:

T-lang template: Converts tabular data into a natural language style. It mimics human language, which helps with LLM performance but can be repetitive when used for in-context examples.
T-table template: Displays data in a table using Markdown. This template is more efficient for handling in-context examples, although it may be more challenging for LLMs when it comes to long-term associations.
T-anony template: Similar to T-table, but it removes any meta-information, making it useful when background data isn't available.

These templates make GTL adaptable to a range of tasks and datasets, giving flexibility when it comes to using meta-information and including in-context examples.

Probabilistic Framework

GTL uses a probabilistic framework that captures relationships between features and target variables, taking into account the meta-information to give context. It works by representing the joint probability distribution of meta-information and text representations for features and targets, allowing GTL to model complex dependencies and nuances that traditional models might miss.

$$p(\boldsymbol{x}, y) = p(\boldsymbol{t}^{\boldsymbol{x}}, t^{y} | \boldsymbol{t}^{m}) = p(t^{y} | \boldsymbol{t}^{\boldsymbol{x}}, \boldsymbol{t}^{m}) \prod_{i=1}^{M} p(\boldsymbol{t}^{x_i} | \boldsymbol{t}^{<x_i}),$$

The first term represents joint probability distribution conditioned on meta information (t^m) and using text rep for features (t^x) and (t^y) for target.

For the second equation: p(t^y | t^x, t^m) represents the probability of the target text representation given the feature text representations and meta-information and the summation part represents probability of each features text rep given all previous feature reps. Through this, GTL captures complex dependencies between target and features, models relation between features themselves and incorporates meta- information to provide context.

By using this probabilistic framework, GTL can generate predictions for new tabular data samples and potentially learn intricate patterns in the data that traditional tabular models might miss.

Zero-Shot and In-Context Learning

One of the most exciting features of GTL is its ability to handle zero-shot learning. This means it can make predictions for tasks it hasn't seen before, even when the data schema is unfamiliar, as long as meta-information is provided. In-context learning takes this a step further by introducing a few examples from the new task, allowing the model to make better predictions. With minimal fine-tuning, this can transition into few-shot learning, making GTL highly adaptable and efficient across different tasks.

3. Exploratory Data Analysis and Preprocessing

The literature review lays the groundwork for the work in the second half of our project: using TabFM to predict whether a customer would be approved for credit products, and to answer other business questions by predicting other target variables. The sections below address the first half of the project: to establish baseline results from baseline models, for TabFM to be compared against.

Dataset

This project utilizes three datasets: Internal Banking, External Banking, and an Unseen Dataset sourced from Kaggle.

1. **Internal Banking Dataset**: This dataset contains proprietary information collected and maintained by a bank, encompassing a wide range of customer-related data. It includes customer's loan portfolios, credit card usage, and other operational and financial metrics relevant to the bank's internal activities. It contains 51,336 rows and 26 features.
2. **External Banking Dataset**: The external dataset complements the internal data by providing additional credit information sourced from the Credit Information Bureau (India) Limited (CIBIL). This dataset includes detailed credit profiles, such as credit scores, loan repayment behavior, outstanding debts, and historical credit data, offering a broader view of customers' financial activities beyond the bank's internal systems. It contains 51,336 rows and 62 features.
3. **Unseen Dataset**: The unseen dataset is a small subset (100 rows, 42 features) derived from both the internal and external data, meant for model testing. However, it is limited in scope, as it does not contain the full feature set present in the other two datasets, hence prediction models trained using features from Internal and External may not work for Unseen. Due to its incomplete nature and small size, this dataset will not be used in the project. We instead merged the first 2 datasets, then derived the training and testing sets from that.

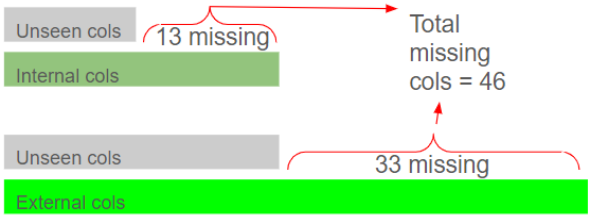

Fig. 2: Missing columns in Unseen dataset

Target Variable

For the purpose of Exploratory Data Analysis (EDA) and Modeling, we have combined the internal and external datasets using *PROSPECTID* column (primary key) and we will be using the *Approved_flag* feature as the target variable for analysis. *Approved_flag* takes 4 values: P1,

P2, P3, P4 with P1 signifying the least credit risk and P4 denoting the most risk and having the maximum chance of rejection.

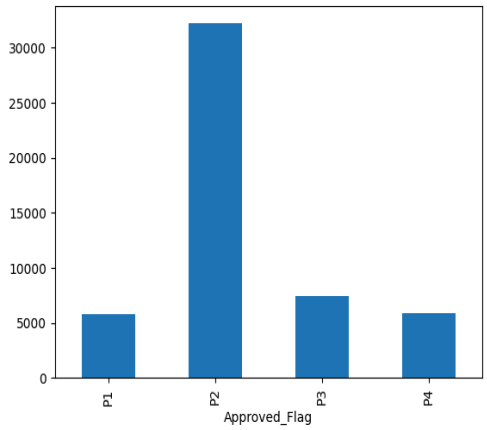

Fig. : Countplot of Approved flag across different tiers

The plot highlights a significant imbalance in the target variable, with a noticeably higher count for the P2 flag compared to the other categories.

Per TD's requirements, we reframed the problem by converting it from a multiclass classification into a binary classification, where the output will indicate either 'Approved' or 'Not Approved.' Specifically, the P1, P2, and P3 flags will be grouped under 'Approved,' while P4 will be deemed as 'Not Approved'[2]. Since the project's objective is to accurately classify customers into these two broader categories, the individual flag distinctions will not be considered.

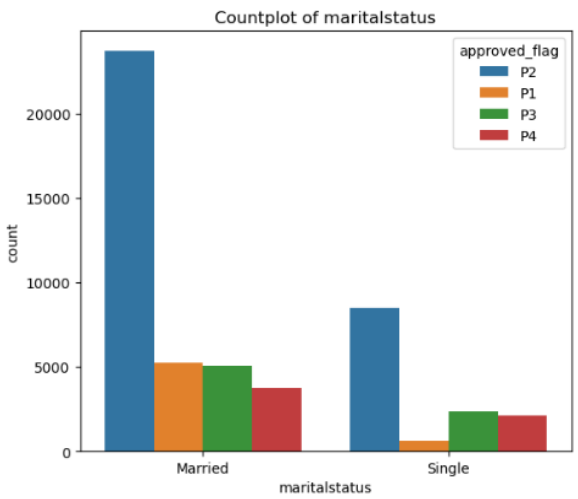

Fig : Distribution of Approved Flag w.r.t Marital Status

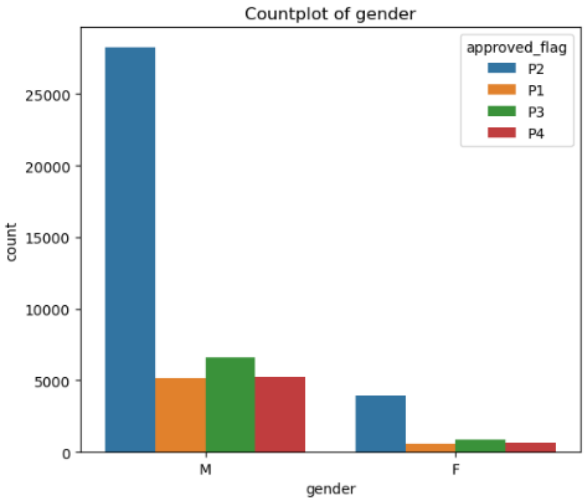

Fig : Distribution of Approved Flag w.r.t Gender

[2] Due to the lack of online information on the definitions of P1-4, we made this decision through consulting ChatGPT (which holds global knowledge from past till minimally 2021) and our industry mentor from TD

The analysis reveals a significant imbalance in the dataset in terms of both gender and marital status, with a notably higher proportion of males compared to females, and a greater number of married individuals compared to singles. However, despite these disparities, the distribution of the target variable within each gender category remains relatively similar.

The dataset reveals a notable prevalence of individuals with graduate degrees. In terms of age distribution, over 45% of participants fall within the 26 to 35 age range. Additionally, approximately 44% of the respondents have an annual income between $18,000 and $30,000.

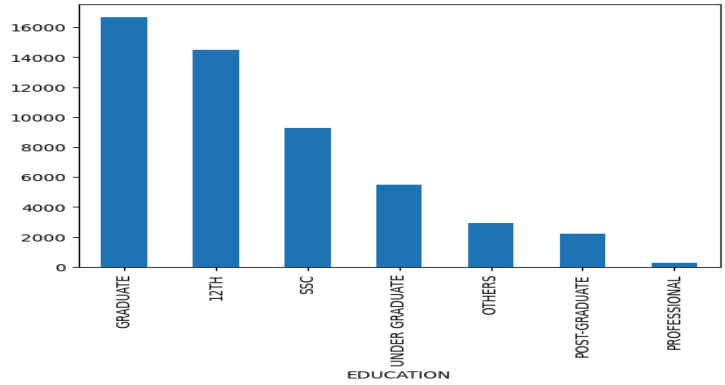

Fig : Countplot for Education Variable

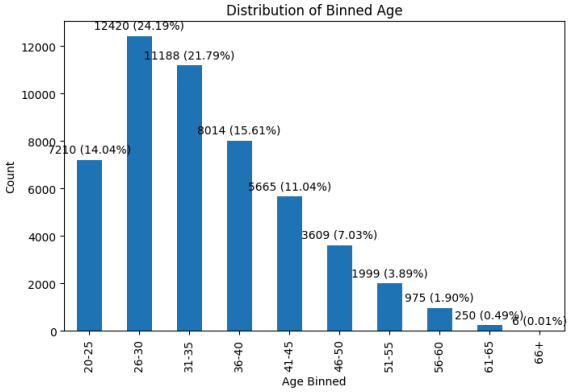

Fig : Countplot for Age Variable

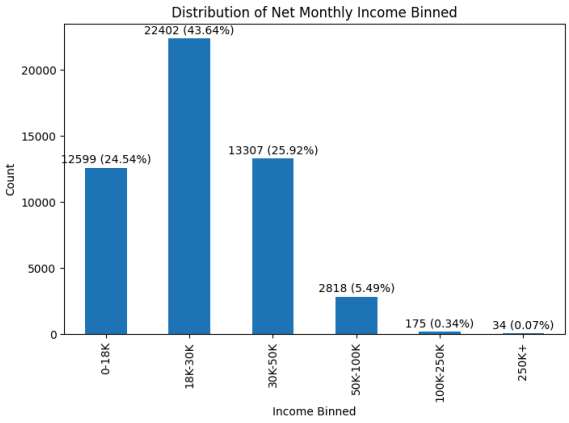

Fig : Countplot for Income Variable

<u>-99999 and missing values</u>

During our analysis of the dataset, we found multiple columns containing a significant number of “-99999” values. Specifically, eight features exhibited over 20% of these values, with *cc_utilization* and *pl_utilization* showing the highest percentages at 92.8% and 86.6%, respectively.

Column	Error Percentage
cc_utilization	92.792582
pl_utilization	86.557192
time_since_recent_deliquency	70.026882
max_delinquency_level	70.026882
time_since_first_deliquency	70.026882
max_unsec_exposure_inpct	45.149603
max_deliq_6mts	25.109085
max_deliq_12mts	21.100203

Fig. Features with high number of -99999 values

It is worth noting that “-99999” has an underlying meaning (eg. missing value indicator, and not necessarily 0), since there are already ‘0’ values present in many of these variables. We also noted that it is the only negative value found across all variables.

We distinguished two types of “-99999” columns: time-based (eg. *time_since_first_deliquency, time_since_recent_deliquency, time_since_recent_payment*) and quantity-based features (eg. *tot_enq, CC_enq, CC_enq_L6m*). We explored several possible combinations for preprocessing and concluded that the following process achieves the highest performance. For the time-based variables, such as time_since_first_deliquency, we created a new dummy column that takes value 0 when the variable has “-99999” value and 1 otherwise. For the quantity-based variables, such as "PL_enq" (number of personal loan queries), the zero value is already present on the column indicating that “-99999” necessarily has another meaning. For such values and features with many missing values, we used k-means to impute them.

<u>High percentage values exceeding 100%</u>

Additionally, we identified a few percentage-type columns containing observations with values exceeding 100%. For example, *pct_currentBal_all_TL* (which reflects the percentage of current balances across all accounts) and *max_unsec_exposure_inPct* (representing maximum unsecured exposure as a percentage). We kept them, as it is possible some of these people took huge loans.

4. Modeling

Due to the imbalance of the target variable, we used the F-1 score as the primary benchmark for all the models. Hyperparameter tuning for each model was conducted using 5-fold randomized cross-validation. The tuned hyperparameters for each model are summarized in Table 1.

Model	Tree based	Ensemble tree	Logistic Regression
Hyperparameter	Max depth Min sample split	# of estimators Max depth	Regularization Weight Regularization type

	Min sample leaf Criteria for split	Min sample split Min sample leaf	Solver Max iteration

Table 1: Model Hyperparameters

Initial Baseline

As an initial baseline model, we constructed a random forest by applying only one-hot encoding to all categorical variables (“minimal preprocessing ”). This yielded an F-1 score of 0.696. After applying the preprocessing steps described earlier (“best preprocessing”), the F-1 score slightly improved to 0.706 (Table 3).

Feature importance analysis (Fig) revealed that the variables related to enquiries were the most influential, while the time since the last delinquency or the age of the oldest account were also relevant, aligning with our prior expectations.

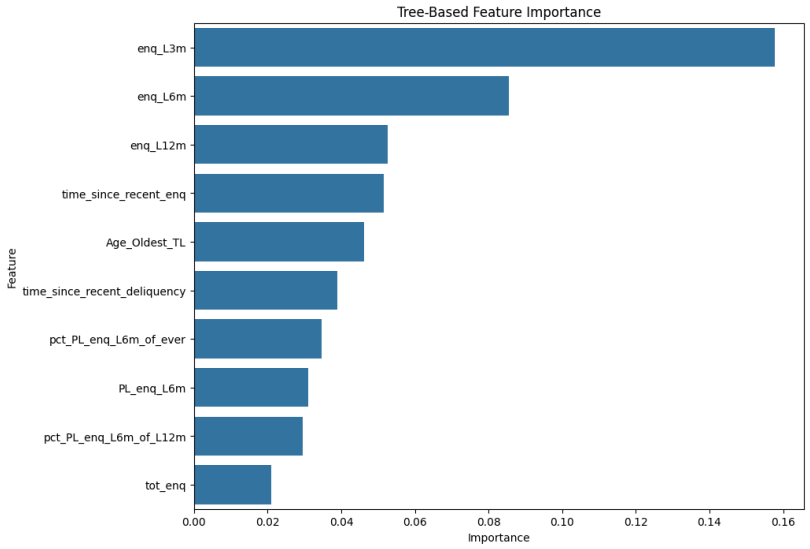

Fig. : Feature importance for random forest model with minimal preprocessing

Feature engineering

To improve the performance beyond minimal preprocessing, we implemented feature engineering. This involved creating 20 additional features by performing operations such as subtraction or division between existing features. Some new features are proportions of one variable over different periods (e.g. total opened account in the last 6 months/total opened account in the last 12 months). Others were the proportion between different types of variables (e.g. the number of secured accounts/the number of unsecured accounts). Using these new features, we constructed baseline models.

Baseline models

We then experimented with 4 different baseline models: decision tree, logistic regression, random forest, and XGBoost. These models were chosen for some of the following reasons:

Decision Tree	*Logistic Regression*	*Random Forest*	*XGBoost*
Simple and interpretable	Simple and interpretable	Can handle imbalanced data	Can handle missing data
Handles both numerical and categorical data	Handles both numerical and categorical data (with 1 hot encoding)	Handles both numerical and categorical data	Handles both numerical and categorical data (with 1 hot encoding)
Captures non-linear relationships	Probabilistic output aids decision making	Robust to overfitting	Robust to overfitting with built in regularization
Computationally efficient	Computationally efficient	Computationally efficient	Computationally efficient

Table 3: Benefits of baseline models

We split the data into train and test datasets. Feature engineering led to an improvement in the random forest model's test set's F1-score to 0.715, making a 0.02 increase over minimal preprocessing (Table 3).

Additionally, some of the newly engineered features such as *enq_L3m_divide_enq_L6m* (last 3 months' enquiries/last 6 months' enquiries) and *Age_Oldest_TL_Substract_Age_Newst_TL* (age of oldest opened account minus age of newest opened account), appeared in the top 10 most important features, indicating that feature engineering contributed to performance gains.

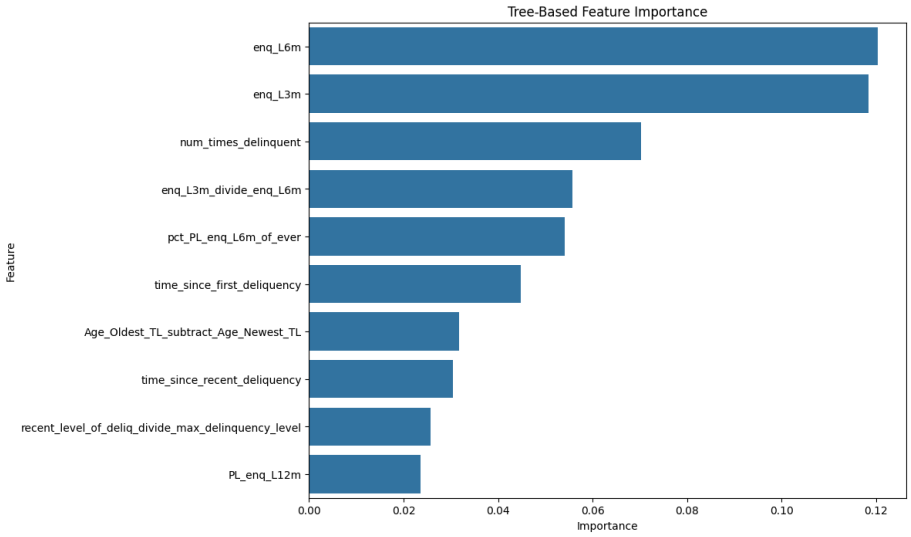

Fig. : Feature importance for random forest model with feature engineering

In addition, to address the data imbalance, we employed SMOTE (Synthetic Minority Over-sampling Technique) to upsample the minority class. Since SMOTE does not allow NaN, we dropped the features with NaN values. This yielded the F-1 score of 0.710 (Table 3) on the test set. However, SMOTE eventually performed poorer than baseline models that employed feature engineering, hence was removed as a preprocessing technique moving forward.

Across all baseline models' test sets' performances, we observe as expected, that due to the simplicity of its structure, the decision tree model underperformed compared to random forest by 0.012 in F-1 Score. Interestingly, logistic regression, which we initially expected to underperform due to the simplicity, performed nearly as well as random forest, achieving an F-1 score of 0.713. In conclusion, XGBoost outperformed all the models with a significantly higher F-1, 0.758.

The performance of each model is summarized in Table 3.

Feature Engineering	No	No	No	Yes	Yes	Yes	Yes
Model	Random Forest (minimal preprocess)	Random Forest (best preprocess)	Random Forest (SMOTE + drop features)	Logistic regression (best preprocess)	Decision tree (best preprocess)	Random Forest (best preprocess)	XGBoost (best preprocess)
F-1 score	0.696	0.706	0.710	0.713	0.703	0.715	**0.758**

Table 3: Models' performance

5. Goals and Next Steps

Our goals and next steps are summarized below:

1. Implementation of TabFM model: We will work on implementing the TabFM model using the GTL pipeline framework as mentioned in the literature review with the goal of checking the capability of TabFM to generate predictions for tabular data samples and potentially learn intricate patterns in the data that traditional tabular models might miss.
2. Traditional ML models and TabFM for comparisons: Once the TabFM implementation is completed and the predictions are noted using it, we will compare the results with the traditional ML model performances. The insight gained from this would be valuable in understanding the future applications of the TabFM model in the banking sector over classical ML models for tabular data.
3. Test for model adaptability with a second target variable: As part of the project, our goal is also to see TabFM model's easy adaptability to other use cases with minimal

engineering efforts. We will use a secondary target variable (after discussing with TD mentors) for checking the model's generalization to address other use cases.

4. Documentation & Knowledge Transfer: We will work on documenting all data processing steps, model configurations, feature engineering, workflow followed and model performance to develop a knowledge transfer guide for foundation and extension in the future. The updated codes will be pushed to GitHub as well.

Ethical considerations:

We're currently working with a Kaggle dataset that contains no Personally Identifiable Information (PII). Similarly, if the proposed model is adopted at TD, data privacy compliance is ensured as we were told that the same dataset schema is being used at TD and steps are taken to mask the PII data before project use.

Conclusion

The first phase of the project explored the dataset and developed traditional ML baseline models for predicting credit approvals and analyzing customer behavior. The next phase will showcase TabFM's application in credit approval predictions and its ability to adapt to other financial use cases with minimal reengineering through zero-shot and in-context learning.

6. Contribution

- Yu Zheng Lim (yl5451): EDA, literature review, report writing
- Ryu Sonoda (rs4493): EDA, baseline modeling, report writing
- Atharva Kulkarni (ak5070): EDA and Literature Review
- Nicolas Alejandro Cogorno (nac2216): EDA, writing of the report, among others
- Minh Dang (nd2802): Goals & Next Steps & Conclusion
- Varun Agarwal (va2515): Team Captain: set up meetings and manage progress. EDA and report writing.

References

[1] Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, Jiang Bian. 2023. From Supervised to Generative: A Novel Paradigm for Tabular Deep Learning with Large Language Models. In KDD.

[2] Tianqi Chen and Carlos Guestrin. 2016.XGBoost: A scalable tree boosting system. In KDD.

[3] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021.Revisiting deep learning models for tabular data. In NeurIPS.

[4] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023.TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In ICLR.