BIG DATA ANALYTICS

I - Context / Dataset :

In this project, the goal is to predict whether a client will default on their loan, using a rich and complex dataset. This dataset is made up of multiple tables from various internal and external sources, with different levels of aggregation.

We can break it down into three main types of data:

1. Static data (depth 0): These are directly linked to each loan case, identified by `case_id`. They contain basic information, such as the date when the loan decision was made.

2. Historical data (depth 1 and 2): These tables provide more detailed records, including payment histories, credit bureau data, and even tax registry information. They are indexed by specific columns, `num_group1` and `num_group2`, to track different interactions over time.

3. Transformed predictors: Some features have been processed to make them easier to analyze. For instance, we have transformations on amounts, dates, and days past due, indicated by notations like "P" for "Days Past Due" or "A" for amounts.

The challenge here is to **aggregate these historical records** for each `case_id` in a meaningful way. We need to condense raw data into useful features while keeping in mind that some external data, like credit bureau records, might not be available during the final evaluation phase.

In this competition, submissions are evaluated using a custom *gini stability metric*, which emphasizes both performance and consistency over time. This stability metric is crucial in scenarios where predictive accuracy may vary as external factors or client behaviors evolve, especially in real-world financial applications like loan approval.

How It Works:

1. **Average Performance (Gini Score)**
   A *gini* score is calculated for predictions corresponding to each `WEEK_NUM`

using the formula:
gini = 2 × AUC − 1,
where AUC (Area Under the Curve) measures how well the model distinguishes between classes. The mean of these weekly gini scores forms the foundation of the stability metric.

2. **Downward Trends (Falling Rate)**
   A linear regression, **a · x + b**, is fit through the weekly gini scores to detect any decline in predictive power over time. A penalty is applied to the model if the slope a is negative:
   falling_rate = min(0, a).
   This discourages models that perform well initially but drop off in accuracy later, which could cause lenders to misjudge long-term risk.
3. **Consistency (Variability Penalty)**
   The variability of the predictions is assessed by calculating the standard deviation of the residuals from the above linear regression. Higher variability implies wild fluctuations, which are undesirable in stable, real-world applications.

Final Stability Metric:

The final score is calculated as:
stability_metric = mean(gini) + 88.0 × min(0, a) − 0.5 × std(residuals).

Why Stability Matters:

Imagine a model that starts strong with an AUC of 0.75 in Week 1 but drops to 0.60 by Week 10. This drop would significantly penalize the stability score. In real-world lending, where client behavior constantly changes, it's vital for models not just to predict well today but to remain reliable over time. A sudden drop in performance could lead lenders to unknowingly approve riskier loans, exposing them to greater financial risk.

By prioritizing stability alongside accuracy, this metric helps ensure that models continue to deliver consistent, trustworthy predictions over time, ultimately supporting smarter, more sustainable lending decisions.

II- Model Introduction

With the competition's objective clear, various machine learning models were identified, each with distinct strengths:

Logistic Regression:

- **Purpose**: A baseline model commonly used in credit scoring.
- **Advantages**: Easy to implement and interpret, making it suitable for understanding the relationship between features and loan default predictions.
- **Limitations**: Struggles with complex, nonlinear relationships in the data, which is important for accurately predicting defaults based on various borrower profiles.

Decision Trees:

- **Purpose**: Captures nonlinear relationships, making them useful for classification tasks like predicting loan defaults.
- **Advantages**: Simple to understand, and provides transparency by revealing how decisions are made based on feature values.
- **Limitations**: Prone to overfitting, especially with deep trees, which can lead to poor generalization on new, unseen data

Random Forest:

- **Purpose**: An ensemble of decision trees that combines their predictions to improve stability and accuracy.
- **Advantages**: Less prone to overfitting compared to single decision trees, and provides feature importance, helping identify which factors influence default likelihood.
- **Limitations**: Computationally heavier, especially with large datasets, and can reduce model interpretability due to its complexity.

Gradient Boosting (LightGBM):

- **Purpose**: A powerful and efficient model that builds trees sequentially to correct previous errors, making it very effective for classification tasks like loan default prediction.
- **Advantages**: High performance, particularly on tabular data. Efficient handling of missing values and categorical features. Offers faster training times and better scalability compared to traditional gradient boosting methods.
- **Limitations**: Requires careful hyperparameter tuning to avoid overfitting and ensure optimal performance.

For the notebook, the chosen model is **LightGBM** due to its high performance, efficient handling of tabular data, and fast training, making it an optimal choice for predicting loan defaults.

III - Introduction Notebook

Overview

This code was developed to predict whether a client will repay a loan, as part of the Kaggle *"Home Credit"* competition. It uses structured, spreadsheet-like data and follows a step-by-step process to prepare the data, train a machine learning model, and evaluate its performance.

Process Breakdown

1. Loading the Data

The code starts by loading multiple CSV files from the dataset containing client information, such as profiles, credit history, and other financial details. To handle this efficiently, it uses *Polars*—a fast, memory-friendly library designed for large datasets.

2. Cleaning and Preparing the Data

Next, the code converts columns into appropriate formats—for example, numerical values for income and categorical labels for education levels. It also creates new features (like *"maximum overdue payment amount"*) to improve the model's predictive power.

3. Combining the Data

All tables are merged using a unique client identifier (case_id). This step consolidates scattered information—personal details, financial behavior, and credit history into a single, unified dataset.

4. Training the Model

The model chosen for this task is *LightGBM*, a machine learning algorithm optimized for tabular data. LightGBM is ideal here because it works quickly with large datasets, handles categorical variables smoothly, and delivers strong results without needing extensive tuning.

5. Evaluating Performance

The model's accuracy is measured using the AUC score (Area Under the ROC Curve). This score reflects how well the model distinguishes between clients who repay loans and those who default. A higher AUC (closer to 1) means better performance. Additionally, a unique stability metric is calculated, specific to this competition. This metric penalizes models whose performance declines over time, ensuring the predictions remain reliable in the long run.

6. Generating Final Predictions

Finally, the model is applied to unseen test data. The results are saved in a submission.csv file, formatted for submission to Kaggle.

This code follows a standard data science pipeline:

1. Load/Clean Data → 2. Create Features → 3. Train a Model → 4. Evaluate → 5. Predict.

Why This Approach Matters

- Real-World Impact: Predicting loan defaults helps financial institutions manage risk and make informed lending decisions.
- Focus on Stability: Unlike most projects, this code doesn't just prioritize short-term accuracy. It also ensures the model stays reliable over time—a critical factor for real-world applications.

III- Data Analytics

The project followed a systematic data science approach, which included thorough data exploration, preprocessing, model selection, and rigorous performance evaluation to ensure the model's effectiveness in real-world applications.

Data Exploration: The Home Credit dataset provided comprehensive information about borrowers, including their credit history, demographic details (e.g., age, family status, and education level), financial data (such as income and debt history), and repayment behaviors. After running our notebook, we identified anomalies, missing values, and extreme outliers in the data. We also explored the relationships between different variables and identified key factors influencing loan default risk. Among these, the debt-to-income ratio, payment history, and employment status were found to be crucial in predicting a borrower's ability to repay a loan.

IV- Output interpretation

This results file contains two columns:

- The first column, case_id, is a unique identifier for each client—like a reference number. For example, clients labeled 57543 or 57549 are distinct individuals in the dataset.
- The second column, score, represents the model's predicted probability that a client will default on their loan. These scores range from 0 to 1, where 0 means

very low risk (the client is likely to repay) and 1 means very high risk (the client is likely to default).

Let me break this down with examples:

- A client with a score of 0.0045 (like case ID 57632) has an extremely low risk of default—almost a guaranteed repayment.
- A client with a score of 0.054 (like case ID 57549) carries a slightly higher risk. While still relatively low, this signals that the institution might want to review their profile more carefully.

Here's how you can use these scores:

- Rank clients by risk: Sort them from highest score (most risky) to lowest score (least risky). For instance, a client with a score of 0.05 would be **prioritized for monitoring** over someone with a score of 0.03, as the higher score indicates a higher risk of default, meaning they require closer scrutiny regarding loan repayment.
- Set a risk threshold: For example, the institution might decide to reject loans for anyone with a score above **0.05**, depending on how much risk they're willing to accept.

Key takeaways:

- All scores are probabilities, so they naturally fall between 0 and 1.
- Small differences (like 0.0535 vs. 0.0543) indicate clients with similar risk levels, the model doesn't see a major distinction between them.
- The safest client here is case ID 57632 with a near-zero score, while the riskiest in this sample is case ID 57549.

In summary, these scores act as a risk radar. The higher the number, the more caution is needed. Financial institutions can use this to make smarter, data-driven lending decisions, approving low-risk clients and flagging high-risk ones.

V - Issues Encountered with the Code

During the execution of the code on Google Colab, the process encountered a blockage at the third code block, primarily due to Google Colab's inability to access the Kaggle-stored data. To download this data, the Kaggle API is required, but it relies on specific permissions and properly configured API keys. Since these access credentials were not initially set up in the environment, the code failed to run.
To bypass this issue, our team opted to store the necessary files in a team member's Google Drive. This allowed that member to execute the code successfully. However,

when other team members attempted to replicate the process, they faced the same challenge: if a member did not have the required files in their own Google Drive or local machine, the code execution became impossible.
Simple Analogy: It's like trying to access a file on a cloud storage service without having the necessary permissions. Until proper access is configured (e.g., by obtaining an API key or transferring the file locally), the task will fail.

An additional challenge arose with the Polars library, used for data manipulation. While Polars is generally efficient, it can occasionally cause errors in specific scenarios, for instance, when variable names in the code clash with those reserved by the library. These conflicts are not systematic, but when they occur, they may lead to unexpected behavior or runtime errors.
Simple Analogy: Imagine using a file management app that contains a folder named "Documents," and you create another folder also called "Documents." The app might struggle to determine which folder to open, causing confusion. Similarly, naming conflicts between code variables and library functions can create unexpected issues.

VI - Interpretation of the AUC Scores

The **AUC** (Area Under the ROC Curve) is a metric that measures how well your model distinguishes between two classes—in this case, customers who will default on loans (target=1) and those who will not (target=0). Here's what the results mean:

1. **Train Set AUC (0.764)**:
    - This score indicates that the model performs **well on the training data**, correctly ranking ~76.4% of the defaulters above non-defaulters.
    - A higher train AUC suggests the model has learned patterns from the data, but it may also reflect slight **overfitting** (memorizing training data noise).
2. **Validation Set AUC (0.751)**:
    - On unseen validation data, the model achieves ~75.1% accuracy in ranking defaulters.
    - This score is critical because it shows **generalization ability**—how well the model performs on new, unseen cases during training.
3. **Test Set AUC (0.748)**:
    - The test set represents a final, unbiased evaluation. The ~74.8% score confirms the model's **stable performance** in real-world scenarios.
    - The small drop from validation to test (~0.3%) suggests minimal overfitting, which is ideal for reliability.

Impact on Home Credit's Decision-Making
With a stable model achieving consistent AUC scores (~0.75) and a robust stability metric, Home Credit can:

1. **Confidently Approve Loans**: Predict defaults more reliably, expanding access to credit for underserved groups while minimizing risk.
2. **Adapt Dynamically**: Adjust lending policies based on borrower risk patterns over time (e.g., seasonal fluctuations or economic shifts).
3. **Mitigate Future Risks**: Maintain performance during market volatility (e.g., recessions) or global crises (e.g., pandemics), ensuring long-term reliability.

This combination of accuracy and stability creates a **competitive advantage**, enabling smarter risk management and fostering financial inclusion—a critical differentiator in the consumer finance industry.

Model Evaluation:
Among all models tested, **LightGBM** proved to be the most effective. It outperformed other models in terms of predictive accuracy, speed, and stability. Importantly, it handled missing data and categorical variables seamlessly, which were key factors for this dataset. Additionally, its robust performance across different data subsets made it the ideal model for predicting loan defaults in this scenario.
Overall, **LightGBM** emerged as the optimal model for predicting loan defaults due to its accuracy, efficiency, and adaptability.

VII - Conclusion

This project has been a valuable learning experience, helping us understand how to analyze and interpret a machine learning pipeline, even without building it from scratch. By reviewing and dissecting the code, we strengthened our ability to read and debug workflows, such as using Polars for data loading and LightGBM for training.

We also tackled practical challenges, including managing Google Colab dependencies and setting up the Kaggle API, which enhanced our problem-solving skills. Additionally, we reflected on model selection, recognizing that LightGBM's speed and stability make it particularly well-suited for tabular data.

Another key takeaway was the importance of balancing metrics. The competition's emphasis on stability taught us to focus on long-term reliability rather than short-term improvements.

Ultimately, this project sharpened essential data science skills, such as critical thinking, adaptability, and the ability to connect technical insights to real-world business impact. We are grateful for this opportunity to learn and grow!