Loan Default Risk Prediction – Technical Walkthrough

POC using Home Credit Dataset

1. Project Objective and Context

1.1. Objective

The goal of this project is to build a machine learning model capable of predicting the probability that a loan applicant will default on a credit product. Formally, this is modeled as a binary classification problem, where the target variable indicates:

- **1 (Default)** – The applicant is likely to default.
- **0 (No Default)** – The applicant is likely to repay successfully.

This is a core problem in the financial services industry, particularly in consumer credit risk analysis. Correctly identifying high-risk applicants allows lenders to minimize losses, adjust interest rates, and comply with regulatory requirements on risk management.

1.2. Project Framing

The project is framed as a **Proof of Concept (POC)**, designed to demonstrate the feasibility of:

- Using structured customer data to build predictive models.
- Integrating model outputs into a simplified web application.
- Deploying a modular and interpretable machine learning pipeline.

This framing supports iterative experimentation and serves as a starting point for future productionization, especially relevant for startups or innovation teams in fintech.

2. Data Acquisition and Loading

2.1. Dataset Source

The dataset used is from the public *Home Credit Default Risk* competition on Kaggle. It simulates a real-world dataset collected by a lending institution and includes client-level information ranging from personal demographics to loan history.

2.2. Structure and Content

The primary dataset, `application_train.csv`, includes over 300,000 records and 120+ features. Key types of variables include:

- **Personal details**: age, gender, family size, number of children.
- **Financials**: total income, credit amount, annuity, goods price.
- **Time-related data**: employment duration, registration date, birth date (in days).
- **Categorical features**: education level, housing type, contract type.

2.3. Data Loading Strategy

The data loading pipeline accounted for practical challenges such as:

- **Encoding**: Support for non-standard character encodings in CSV files.
- **Modular loading**: Files organized using a dictionary for cleaner access.
- **Memory management**: Dtypes optimized to reduce memory footprint.

This ensures the project remains scalable and maintainable for large datasets or multi-file pipelines.

3. Exploratory Data Analysis (EDA)

3.1. Missing Values Analysis

A column-wise null value analysis was performed. Features with a high proportion of missing values (e.g., over 40%) were reviewed for relevance. Some features had nulls due to optionality (e.g., phone numbers), while others reflected data collection gaps.

3.2. Feature Distributions

Histograms and KDE plots were used to assess skewness and detect anomalies. For example:

- `DAYS_EMPLOYED` had a high peak at 365243 days, indicating a placeholder or data entry error.
- `AMT_INCOME_TOTAL` and `AMT_CREDIT` showed positive skew, common in financial datasets due to high-income or high-credit outliers.

Log transformations were considered for normalization, though not always applied due to interpretability tradeoffs.

3.3. Target Imbalance

Only 8.07% of records were labeled as defaulters. This means the dataset is highly imbalanced — a crucial insight, as models trained naively on imbalanced data tend to favor the majority class (non-defaulters), leading to high accuracy but poor recall for defaults.

3.4. Correlation and Redundancy

A heatmap of Pearson correlations helped identify highly correlated features (e.g., `AMT_CREDIT` and `AMT_GOODS_PRICE`). Redundant features were flagged for removal or consolidation to reduce multicollinearity and improve model robustness.

4. Statistical Inference and Hypothesis Testing

Before modeling, inferential statistics were used to understand group differences and validate domain hypotheses:

Test	Purpose	Outcome
T-test: Income vs. Default	Compare mean income for default/no-default groups	Statistically signi
Chi-square: Gender vs. Default	Test independence between gender and default	Not significant (p
T-test: Credit Amount vs. Default	Compare mean credit amount	Significant differe

These results suggest that financial attributes have stronger predictive power than demographic ones such as gender. This evidence helps refine feature selection and supports model explainability.

5. Feature Selection Strategy

Given the large number of raw features, a curated set of 10 numerical features was selected based on domain knowledge, exploratory findings, and computational considerations. The goal was to balance:

- **Simplicity** – Less risk of overfitting, easier debugging.
- **Interpretability** – Features easily explained to business stakeholders.
- **Predictive Power** – Features shown to correlate with the target.

Categorical features were temporarily excluded to focus on numeric modeling and pipeline clarity in the POC.

6. Modeling Approach

6.1. Models Selected

Two classifiers were chosen:

1. **Logistic Regression** – Linear model, interpretable coefficients.
2. **Random Forest** – Ensemble-based, captures non-linear patterns.

These models offer a balance between performance and transparency, critical in financial applications where explainability is often required by regulation.

6.2. Pipeline Components

A complete ML pipeline was built using `scikit-learn`'s `Pipeline` object, comprising:

- **Imputation**: Missing values filled using median values.
- **Standardization**: All features scaled to zero mean and unit variance.
- **Classifier**: Either logistic regression or random forest.

This modular structure improves maintainability and enables easy model swapping during experimentation.

6.3. Evaluation Metrics

Given the target imbalance, ROC AUC was selected as the primary evaluation metric. This measures the model's ability to separate the two classes and is insensitive to class imbalance.

Additional metrics used:

- **Precision**: Important in cases where false positives are costly.
- **Recall**: Captures how many true defaulters are identified.
- **Accuracy**: Reported but interpreted with caution.

7. Hyperparameter Tuning

7.1. Tuning Strategy

`GridSearchCV` was used for hyperparameter optimization. The grid was deliberately small to allow faster experimentation:

- `max_depth`: [10, 20]
- `n_estimators`: [50, 100]
- `min_samples_split`: [2, 5]

Each combination was evaluated using 5-fold cross-validation, with ROC AUC as the scoring function.

7.2. Tradeoffs

Limiting the parameter grid allowed for rapid prototyping, a key consideration in early-stage projects. This method prioritized quick insights over exhaustive optimization.

8. Feature Importance Analysis

8.1. Insights from Random Forest

Feature importances were extracted from the best-performing Random Forest model. The top contributors included:

- **AMT_CREDIT**: Total credit requested.

- **DAYS_BIRTH**: Age of the applicant (younger clients showed higher default rates).
- **AMT_INCOME_TOTAL**: Total declared income.

Understanding these patterns aids in model interpretation, regulatory compliance, and trust-building with business users.

9. Model Serialization and API Design

9.1. Model Saving

The final trained model was saved using `joblib`, enabling later reuse without retraining. This prepares the system for integration into other applications or services.

[11pt]article [a4paper, margin=1in]geometry amsmath, amssymb graphicx booktabs hyperref titlesec fancyhdr enumitem caption setspace longtable float

Loan Default Risk Prediction – Technical Walkthrough POC using Home Credit Dataset

10. Project Objective and Context

10.1. Objective

The goal of this project is to build a machine learning model capable of predicting the probability that a loan applicant will default on a credit product. Formally, this is modeled as a binary classification problem, where the target variable indicates:

- **1 (Default)** – The applicant is likely to default.
- **0 (No Default)** – The applicant is likely to repay successfully.

This is a core problem in the financial services industry, particularly in consumer credit risk analysis. Correctly identifying high-risk applicants allows lenders to minimize losses, adjust interest rates, and comply with regulatory requirements on risk management.

10.2. Project Framing

The project is framed as a **Proof of Concept (POC)**, designed to demonstrate the feasibility of:

- Using structured customer data to build predictive models.
- Integrating model outputs into a simplified web application.
- Deploying a modular and interpretable machine learning pipeline.

This framing supports iterative experimentation and serves as a starting point for future productionization, especially relevant for startups or innovation teams in fintech.

11. Data Acquisition and Loading

11.1. Dataset Source

The dataset used is from the public *Home Credit Default Risk* competition on Kaggle. It simulates a real-world dataset collected by a lending institution and includes client-level information ranging from personal demographics to loan history.

11.2. Structure and Content

The primary dataset, `application_train.csv`, includes over 300,000 records and 120+ features. Key types of variables include:

- **Personal details**: age, gender, family size, number of children.
- **Financials**: total income, credit amount, annuity, goods price.
- **Time-related data**: employment duration, registration date, birth date (in days).
- **Categorical features**: education level, housing type, contract type.

11.3. Data Loading Strategy

The data loading pipeline accounted for practical challenges such as:

- **Encoding**: Support for non-standard character encodings in CSV files.
- **Modular loading**: Files organized using a dictionary for cleaner access.
- **Memory management**: Dtypes optimized to reduce memory footprint.

This ensures the project remains scalable and maintainable for large datasets or multi-file pipelines.

12. Exploratory Data Analysis (EDA)

12.1. Missing Values Analysis

A column-wise null value analysis was performed. Features with a high proportion of missing values (e.g., over 40%) were reviewed for relevance. Some features had nulls due to optionality (e.g., phone numbers), while others reflected data collection gaps.

12.2. Feature Distributions

Histograms and KDE plots were used to assess skewness and detect anomalies. For example:

- `DAYS_EMPLOYED` had a high peak at 365243 days, indicating a placeholder or data entry error.
- `AMT_INCOME_TOTAL` and `AMT_CREDIT` showed positive skew, common in financial datasets due to high-income or high-credit outliers.

Log transformations were considered for normalization, though not always applied due to interpretability tradeoffs.

12.3. Target Imbalance

Only 8.07% of records were labeled as defaulters. This means the dataset is highly imbalanced — a crucial insight, as models trained naively on imbalanced data tend to favor the majority class (non-defaulters), leading to high accuracy but poor recall for defaults.

12.4. Correlation and Redundancy

A heatmap of Pearson correlations helped identify highly correlated features (e.g., `AMT_CREDIT` and `AMT_GOODS_PRICE`). Redundant features were flagged for removal or consolidation to reduce multicollinearity and improve model robustness.

13. Statistical Inference and Hypothesis Testing

Before modeling, inferential statistics were used to understand group differences and validate domain hypotheses:

Test	Purpose	Outcome
T-test: Income vs. Default	Compare mean income for default/no-default groups	Statistically signi
Chi-square: Gender vs. Default	Test independence between gender and default	Not significant (p
T-test: Credit Amount vs. Default	Compare mean credit amount	Significant differe

These results suggest that financial attributes have stronger predictive power than demographic ones such as gender. This evidence helps refine feature selection and supports model explainability.

14. Feature Selection Strategy

Given the large number of raw features, a curated set of 10 numerical features was selected based on domain knowledge, exploratory findings, and computational considerations. The goal was to balance:

- **Simplicity** – Less risk of overfitting, easier debugging.
- **Interpretability** – Features easily explained to business stakeholders.
- **Predictive Power** – Features shown to correlate with the target.

Categorical features were temporarily excluded to focus on numeric modeling and pipeline clarity in the POC.

15. Modeling Approach

15.1. Models Selected

Two classifiers were chosen:

1. **Logistic Regression** – Linear model, interpretable coefficients.

2. **Random Forest** – Ensemble-based, captures non-linear patterns.

These models offer a balance between performance and transparency, critical in financial applications where explainability is often required by regulation.

15.2. Pipeline Components

A complete ML pipeline was built using `scikit-learn`'s `Pipeline` object, comprising:

- **Imputation**: Missing values filled using median values.
- **Standardization**: All features scaled to zero mean and unit variance.
- **Classifier**: Either logistic regression or random forest.

This modular structure improves maintainability and enables easy model swapping during experimentation.

15.3. Evaluation Metrics

Given the target imbalance, ROC AUC was selected as the primary evaluation metric. This measures the model's ability to separate the two classes and is insensitive to class imbalance.

Additional metrics used:

- **Precision**: Important in cases where false positives are costly.
- **Recall**: Captures how many true defaulters are identified.
- **Accuracy**: Reported but interpreted with caution.

16. Hyperparameter Tuning

16.1. Tuning Strategy

`GridSearchCV` was used for hyperparameter optimization. The grid was deliberately small to allow faster experimentation:

- `max_depth`: [10, 20]
- `n_estimators`: [50, 100]
- `min_samples_split`: [2, 5]

Each combination was evaluated using 5-fold cross-validation, with ROC AUC as the scoring function.

16.2. Tradeoffs

Limiting the parameter grid allowed for rapid prototyping, a key consideration in early-stage projects. This method prioritized quick insights over exhaustive optimization.

17. Feature Importance Analysis

17.1. Insights from Random Forest

Feature importances were extracted from the best-performing Random Forest model. The top contributors included:

- **AMT_CREDIT**: Total credit requested.
- **DAYS_BIRTH**: Age of the applicant (younger clients showed higher default rates).
- **AMT_INCOME_TOTAL**: Total declared income.

Understanding these patterns aids in model interpretation, regulatory compliance, and trust-building with business users.

18. Model Serialization and API Design

18.1. Model Saving

The final trained model was saved using `joblib`, enabling later reuse without retraining. This prepares the system for integration into other applications or services.

19. Flask Web Application with Professional UI

19.1. Objective of the App

To complement the machine learning model, a lightweight web application was developed using the Flask framework. This application serves as a user interface for model inference, allowing users to submit loan applicant information and receive default risk predictions in real time.

The purpose of the app is threefold:

1. Provide an interactive demo for stakeholders.
2. Simulate a front-end interface for possible integration into existing banking workflows.
3. Enable manual testing and validation of the machine learning pipeline.

19.2. Technology Stack

- **Backend:** Flask (Python)
- **Frontend:** HTML5, CSS3 (professionally styled with responsiveness)
- **Serving:** Localhost (development mode), extensible for Docker or cloud deployment

19.3. App Architecture

The application follows a minimal but modular architecture:

- `app.py` – Flask app entry point. Loads the trained model and handles routing.
- `templates/form.html` – The main UI form for user input.
- `static/style.css` – Custom stylesheet providing a clean, professional design.
- `model.pkl` – Serialized machine learning model loaded at runtime.

19.4. User Flow

1. The user navigates to the main page.
2. A styled HTML form is presented with input fields for selected features (e.g., income, credit amount, days employed).
3. Upon form submission, Flask receives the input, preprocesses it, and makes a prediction using the model.
4. The result (default / non-default) is displayed along with the probability score in a formatted result card.

19.5. Design and UI Considerations

The interface was designed to be:

- **Minimal and clean** – Intuitive layout with clear labels.
- **Responsive** – Usable on desktop and tablet devices.
- **Professional** – Fonts, spacing, and color palette chosen to resemble fintech applications.

Example UI elements include:

- Form fields with floating labels and real-time validation.
- Submit button styled with hover and focus effects.
- Result section highlighted with conditional formatting (e.g., red for default, green for no default).

19.6. Deployment Readiness

While currently running in development mode, the app is structured to support:

- **Production deployment via WSGI (e.g., Gunicorn)**
- **Dockerization** – Suitable for cloud platforms like GCP or AWS.
- **CI/CD integration** – Modular codebase allows for testing and deployment automation.

19.7. Benefits and Use Cases

This application enhances the machine learning project by:

- Demonstrating usability to non-technical stakeholders.
- Providing a real-time decision support interface.
- Serving as a baseline for further development (e.g., user authentication, data storage).

Conclusion

This project presents a robust and interpretable machine learning solution for predicting loan default risk using structured client data. Through careful data exploration, statistical validation, feature selection, and model evaluation, a modular and scalable POC was developed. The pipeline demonstrates clear potential for real-world adoption in financial services environments that value transparency, accountability, and operational efficiency.