**Predictive Model Plan**

|  |
| --- |
| **1. Model Logic (Generated with GenAI)** |

# Predictive Modeling Pipeline for Credit Delinquency

A predictive modeling pipeline for forecasting credit delinquency typically involves several sequential stages to transform raw data into actionable predictions.

# 1. Data Ingestion

* **Description**: This initial phase involves collecting and loading data from various sources. For this dataset, the data is ingested from a CSV file containing customer financial and behavioral information.
* **Action**: Load the Delinquency\_prediction\_dataset.csv into a data structure suitable for analysis (e.g., a Pandas DataFrame).

# 2. Data Preprocessing

* **Description**: Raw data often contains inconsistencies, missing values, and requires transformation before it can be used for modeling.
  + **Handling Missing Values**: Impute or remove missing data points. For example, for numerical features like Income or Loan\_Balance, imputation strategies like mean, median, or mode can be used. For categorical features, imputation with the most frequent category or a new 'Unknown' category can be applied.
  + **Encoding Categorical Variables**: Convert categorical features (e.g.,

Employment\_Status, Credit\_Card\_Type, Location, and monthly payment statuses) into a numerical format that machine learning models can understand. One-hot encoding or label encoding are common techniques.

* + **Feature Scaling**: Normalize or standardize numerical features (e.g., Age, Income, Credit\_Score, Loan\_Balance) to ensure that no single feature dominates the model due to its scale. Techniques like StandardScaler or MinMaxScaler can be used.
  + **Outlier Treatment**: Identify and manage extreme values that could negatively impact model performance.
* **Action**:
  + Fill missing values in Income, Credit\_Score, Loan\_Balance, and Month\_6 (e.g., with median/mode).
  + One-hot encode Employment\_Status, Credit\_Card\_Type, Location, Month\_1 through Month\_6.

# 3. Feature Engineering and Selection

* **Description**: This stage involves creating new features from existing ones and selecting the most relevant features to improve model performance and interpretability.
  + **Feature Engineering**: Create composite features (e.g., number of 'Missed' payments over the past 6 months, or ratio of Missed\_Payments to Account\_Tenure).
  + **Feature Selection**: Use statistical methods (e.g., correlation analysis, chi-squared tests), model-based methods (e.g., feature importance from tree-based models), or dimensionality reduction techniques (e.g., PCA) to select the most impactful features for predicting delinquency.
* **Action**: Create a new feature representing the total number of 'Late' or 'Missed' payments over

Month\_1 to Month\_6 columns. Select relevant features like Income, Credit\_Score,

Credit\_Utilization, Missed\_Payments, Loan\_Balance, Debt\_to\_Income\_Ratio, Account\_Tenure, and the engineered features.

# 4. Model Training

* **Description**: The preprocessed and selected data is split into training and testing sets. A machine learning model is then trained on the training data to learn the patterns associated with credit delinquency.
* **Action**: Split the dataset into 80% training data and 20% testing data. Train the chosen model(s) on the training set.

# 5. Model Evaluation

* **Description**: The trained model's performance is assessed on the unseen test data. Key metrics for classification problems like credit delinquency include Accuracy, Precision, Recall, F1-Score, and AUC-ROC.
* **Action**: Evaluate the model's performance using metrics such as Accuracy, Precision, Recall, and AUC-ROC to determine its effectiveness in predicting delinquent accounts.

# 6. Model Deployment (Optional, but part of a complete pipeline)

• **Description**: Once a model is trained and evaluated, it can be deployed into a production environment to make real-time predictions. This often involves integrating the model with existing systems.

# 7. Monitoring and Retraining (Ongoing)

• **Description**: Deployed models need continuous monitoring for performance degradation (e.g., concept drift) and periodic retraining with new data to maintain accuracy.

**Modeling Options for Predicting Delinquency**

Here are two modeling options, a simple and a complex one, for predicting credit delinquency:

# 1. Simple Model: Logistic Regression

* **Description**: Logistic Regression is a linear model used for binary classification. Despite its simplicity, it's highly interpretable, computationally efficient, and provides probability scores that can be directly used as risk scores. It assumes a linear relationship between features and the log-odds of the target variable.
* **Pros**: Highly interpretable coefficients, fast training times, good baseline model.
* **Cons**: May not capture complex non-linear relationships in the data.
* **Suitability**: Ideal for initial analysis, situations where interpretability is paramount, and when computational resources are limited.

# 2. Complex Model: Gradient Boosting Machine (e.g., LightGBM or XGBoost)

* **Description**: Gradient Boosting Machines are powerful ensemble learning methods that build a strong predictive model by combining many weak learners (typically decision trees) in a sequential manner. Each new tree corrects the errors of the previous ones. LightGBM and XGBoost are optimized implementations known for their speed and accuracy.
* **Pros**: High predictive accuracy, can capture complex non-linear relationships and interactions between features, robust to outliers.
* **Cons**: Less interpretable than logistic regression, requires more computational resources and hyperparameter tuning, can be prone to overfitting if not carefully tuned.
* **Suitability**: Excellent for achieving high predictive performance in credit risk, where complex patterns might exist.

# Recommendation

For credit delinquency prediction, I recommend using a **Gradient Boosting Machine (specifically LightGBM or XGBoost)**.

• **Justification**: While Logistic Regression provides a good baseline and high interpretability, credit risk is often influenced by complex, non-linear interactions between various financial and behavioral features. Gradient Boosting Machines are adept at capturing these intricate patterns, leading to significantly higher predictive accuracy. In a domain like credit risk, where even a small improvement in accuracy can translate to substantial financial benefits (by accurately identifying high-risk individuals and reducing losses), the superior predictive power of a complex model outweighs the slight reduction in direct interpretability (which can still be addressed using techniques like SHAP values for explanation). The efficiency of implementations like LightGBM also makes them practical for real-world deployment.

# How the Delinquency Risk Model Transforms Customer Input Variables into a Final Risk Prediction

The transformation of customer input variables into a final risk prediction by a delinquency risk model follows a well-defined process:

1. **Data Ingestion**: Raw customer data is collected. This includes various attributes such as Age,

Income, Credit\_Score, Credit\_Utilization, Missed\_Payments, Loan\_Balance,

Debt\_to\_Income\_Ratio, Employment\_Status, Account\_Tenure, Credit\_Card\_Type, Location, and historical payment statuses (Month\_1 to Month\_6).

1. **Data Preprocessing**:
   * **Cleaning**: Any missing values in numerical fields (e.g., Income, Loan\_Balance, Credit\_Score) are handled. This could involve imputation using statistical measures (mean, median) or more advanced techniques. Categorical missing values (e.g. Month\_6) are also addressed.
   * **Feature Engineering**: New features might be created from existing ones to provide more predictive power. For example, a "Total Recent Missed Payments" feature could be derived by counting 'Late' or 'Missed' statuses across Month\_1 to Month\_6.
   * **Encoding Categorical Variables**: Non-numerical features like Employment\_Status, Credit\_Card\_Type, Location, and the payment statuses (Late, Missed, On-time) are converted into a numerical format. One-hot encoding is a common method for this, creating new binary columns for each unique category.
   * **Scaling Numerical Variables**: Numerical features with different scales (e.g., Income vs. Credit\_Utilization) are scaled to a standard range. This prevents features with larger values from disproportionately influencing the model.
2. **Feature Selection**: The preprocessed features are then passed through a feature selection process, where a subset of the most relevant features for predicting delinquency is chosen. This step helps in reducing dimensionality, improving model performance, and enhancing interpretability.
3. **Model Application (Inference)**: The preprocessed and selected input variables are fed into the trained machine learning model.
   * For a **Logistic Regression** model, each feature is multiplied by its learned coefficient, and these products are summed up, along with an intercept term. This sum is then passed through a sigmoid (logistic) function, which squashes the output to a value between 0 and 1, representing the probability of delinquency.
   * For a **Gradient Boosting Machine** (like LightGBM or XGBoost), the input features traverse through a series of decision trees. Each tree makes a prediction, and these predictions are combined (typically summed or averaged) to produce a final raw score. This raw score is then transformed by an activation function (often a sigmoid for binary classification) to output a probability of delinquency.
4. **Prediction Output (Risk Score)**: The final output of the model is a probability score, typically ranging from 0 to 1, indicating the likelihood of the customer becoming delinquent. A higher score signifies a higher risk of delinquency. This probability can then be converted into a binary prediction (delinquent/not delinquent) using a predefined threshold (e.g., if probability > 0.5, predict delinquent). This probability value itself is the "risk prediction."

In essence, the model acts as a sophisticated function that takes a vector of processed customer attributes and transforms them into a single probability score, quantifying the credit delinquency risk.

Sample Pseudocode for Building a Credit Risk Prediction Model

This pseudocode outlines the steps for building a credit risk prediction model using relevant features identified from the dataset.

# 1. Data Ingestion

LOAD dataset FROM 'Delinquency\_prediction\_dataset.csv' INTO DataFrame named 'df'

# 2. Data Preprocessing

# Define target variable and features

TARGET\_VARIABLE = 'Delinquent\_Account'

NUMERICAL\_FEATURES = ['Age', 'Income', 'Credit\_Score', 'Credit\_Utilization', 'Missed\_Payments',

'Loan\_Balance', 'Debt\_to\_Income\_Ratio', 'Account\_Tenure']

CATEGORICAL\_FEATURES = ['Employment\_Status', 'Credit\_Card\_Type', 'Location']

MONTHLY\_PAYMENT\_FEATURES = ['Month\_1', 'Month\_2', 'Month\_3', 'Month\_4', 'Month\_5', 'Month\_6']

# Combine all features

ALL\_FEATURES = NUMERICAL\_FEATURES + CATEGORICAL\_FEATURES + MONTHLY\_PAYMENT\_FEATURES

# Handle Missing Values

FOR each feature IN NUMERICAL\_FEATURES:

IF feature HAS missing values IN df:

FILL missing values IN df[feature] WITH MEDIAN of df[feature]

FOR each feature IN CATEGORICAL\_FEATURES:

IF feature HAS missing values IN df:

FILL missing values IN df[feature] WITH MODE of df[feature]

FOR each feature IN MONTHLY\_PAYMENT\_FEATURES:

IF feature HAS missing values IN df:

FILL missing values IN df[feature] WITH MODE of df[feature]

# Feature Engineering: Total missed/late payments

CREATE new feature 'Total\_Late\_Missed\_Payments' IN df

INITIALIZE 'Total\_Late\_Missed\_Payments' to 0

FOR each month\_feature IN MONTHLY\_PAYMENT\_FEATURES:

IF df[month\_feature] IS 'Late' OR 'Missed':

INCREMENT 'Total\_Late\_Missed\_Payments'

# Add engineered feature to feature list

NUMERICAL\_FEATURES.APPEND('Total\_Late\_Missed\_Payments')

# Re-define ALL\_FEATURES after engineering

ALL\_FEATURES = NUMERICAL\_FEATURES + CATEGORICAL\_FEATURES + MONTHLY\_PAYMENT\_FEATURES

# Encoding Categorical Variables (One-Hot Encoding)

PERFORM ONE-HOT ENCODING on df FOR CATEGORICAL\_FEATURES AND MONTHLY\_PAYMENT\_FEATURES

STORE encoded DataFrame in 'df\_encoded'

# Select Features (after encoding, columns will be expanded)

FEATURES\_FOR\_MODELING = list of all columns in 'df\_encoded' EXCEPT TARGET\_VARIABLE and 'Customer\_ID'

# Align column names for consistency (especially for new data during prediction)

# This step is crucial in a real pipeline to ensure consistent feature sets.

# 3. Data Splitting

SPLIT df\_encoded INTO X (features) AND y (target: TARGET\_VARIABLE)

SPLIT X AND y INTO X\_train, X\_test, y\_train, y\_test (e.g., 80% train, 20% test)

# 4. Model Training (Using Recommended Gradient Boosting Machine - LightGBM as example)

# Import necessary libraries (conceptual)

# FROM sklearn.ensemble IMPORT RandomForestClassifier (for simple model option)

# FROM lightgbm IMPORT LGBMClassifier (for complex model option)

# FROM sklearn.linear\_model IMPORT LogisticRegression (for simple model option)

# FROM sklearn.preprocessing IMPORT StandardScaler (for scaling)

# FROM sklearn.metrics IMPORT accuracy\_score, precision\_score, recall\_score, roc\_auc\_score, f1\_score

# Instantiate the model

# simple\_model = LogisticRegression(solver='liblinear') # Example of simple model complex\_model = LGBMClassifier(random\_state=42) # Example of complex model # Train the model

TRAIN complex\_model WITH X\_train, y\_train

# 5. Model Evaluation

PREDICT probabilities ON X\_test USING complex\_model.predict\_proba()

GET delinquency\_probabilities (the probability of class 1)

PREDICT classes ON X\_test USING complex\_model.predict()

GET predicted\_delinquency

# Calculate Evaluation Metrics

ACCURACY = CALCULATE ACCURACY\_SCORE(y\_test, predicted\_delinquency)

PRECISION = CALCULATE PRECISION\_SCORE(y\_test, predicted\_delinquency)

RECALL = CALCULATE RECALL\_SCORE(y\_test, predicted\_delinquency)

F1\_SCORE = CALCULATE F1\_SCORE(y\_test, predicted\_delinquency)

AUC\_ROC = CALCULATE ROC\_AUC\_SCORE(y\_test, delinquency\_probabilities)

PRINT "Model Evaluation Metrics:"

PRINT "Accuracy:", ACCURACY

PRINT "Precision:", PRECISION

PRINT "Recall:", RECALL

PRINT "F1-Score:", F1\_SCORE

PRINT "AUC-ROC:", AUC\_ROC

# Further steps (not in pseudocode, but part of deployment/monitoring):

# SAVE trained\_model\_object

# DEPLOY model AS an API endpoint

# MONITOR model performance OVER TIME

# RETRAIN model PERIODICALLY with new data

The recommended model for credit delinquency prediction is a **Gradient Boosting Machine (e.g., LightGBM or XGBoost)**. This is a powerful ensemble learning technique that builds a strong predictive model by sequentially combining many weak decision trees, with each new tree correcting the errors of the previous ones. It is chosen for its high predictive accuracy and ability to capture complex, non-linear relationships within the data, which are often present in credit risk scenarios.

Based on common credit risk factors and the features identified in the dataset, the top 5 input features highlighted for predicting credit delinquency are:

1. **Credit\_Score**: A fundamental indicator of an individual's creditworthiness.
2. **Credit\_Utilization**: The amount of credit a person is using relative to their total available credit, indicating how heavily they rely on credit.
3. **Missed\_Payments**: A direct measure of past payment behavior, highly indicative of future delinquency.
4. **Income**: Reflects an individual's financial capacity to manage debt.
5. **Debt\_to\_Income\_Ratio**: Compares an individual's monthly debt payments to their gross monthly income, indicating their debt burden.

|  |
| --- |
| **2. Justification for Model Choice** |

My model choice of a Gradient Boosting Machine (such as LightGBM or XGBoost) directly aligns with Geldium's presumed goal of minimizing financial risk and optimizing lending decisions. By leveraging the superior predictive accuracy of this complex model, Geldium can more precisely identify customers at high risk of delinquency, leading to a reduction in loan defaults and associated financial losses. This advanced capability allows for more strategic risk management, enabling Geldium to refine its lending policies, potentially offer tailored products, and ultimately enhance profitability through data-driven insights into customer

|  |
| --- |
| **behavior.3. Evaluation Strategy** |

Evaluating the performance of a credit delinquency prediction model is crucial to ensure its reliability and effectiveness in real-world lending scenarios. The evaluation strategy will focus on a comprehensive set of metrics and considerations:

**Key Metrics**

1. **Accuracy**
   * **Definition**: Accuracy measures the proportion of correctly classified instances (both delinquent and non-delinquent) out of the total instances. o **Formula**: (True Positives + True Negatives) / Total Observations
   * **Interpretation**: While intuitive, accuracy can be misleading in imbalanced datasets (where one class significantly outnumbers the other, common in delinquency prediction, as delinquencies are rare). A high accuracy might simply reflect the model's ability to correctly predict the majority class. Therefore, it should be considered alongside other metrics, especially for credit risk where identifying the minority class (delinquent customers) is often the primary goal.
2. **F1 Score**
   * **Definition**: The F1 Score is the harmonic mean of Precision and Recall. It provides a single metric that balances both the precision (correct positive predictions) and recall (identifying all actual positives).
   * **Formula**: 2 \* (Precision \* Recall) / (Precision + Recall)
   * **Interpretation**: A high F1 Score indicates that the model has a good balance between correctly identifying delinquent accounts (Recall) and minimizing false positives (Precision). This is particularly important for credit risk: high recall prevents missed opportunities to flag risky customers, while high precision minimizes falsely flagging good customers, which could lead to missed business.
3. **AUC-ROC (Area Under the Receiver Operating Characteristic Curve)**
   * **Definition**: The AUC-ROC curve plots the True Positive Rate (Recall) against the False Positive Rate at various classification thresholds. The AUC (Area Under the Curve) provides a single scalar value that represents the model's ability to distinguish between the two classes across all possible thresholds.
   * **Interpretation**: An AUC score ranges from 0 to 1. An AUC of 0.5 suggests the model performs no better than random chance, while an AUC of 1.0 indicates a perfect classifier. In credit risk, a higher AUC means the model is better at ranking customers by their delinquency risk, effectively separating delinquent from non-delinquent accounts. This is valuable as it allo o ws Geldium to set an appropriate risk threshold based on its business appetite.

|  |  |
| --- | --- |
| 4. | **Fairness Checks**   * **Definition**: Fairness checks involve evaluating the model's performance across different demographic groups (e.g., by age, gender, location, or employment status) to ensure that predictions are not biased against any specific group. This is crucial for ethical AI and regulatory compliance. * **Methodology**: This involves analyzing the aforementioned metrics (Accuracy, F1 Score, Precision, Recall) for predefined sensitive groups within the data. Statistical parity, equalized odds, and demographic parity are common fairness metrics. * **Interpretation**: If the model exhibits significantly different performance metrics (e.g., lower recall for a certain age group, or higher false positive rates for a particular location) across different groups, it indicates bias. This would necessitate further investigation, potentially leading to adjustments in feature engineering, model selection, or the application of fairness-aware machine learning techniques to mitigate discrimination and ensure equitable lending practices. Geldium must ensure that its credit risk model adheres to principles of fairness and avoids perpetuating or amplifying existing biases. |