Predictive Model Plan

1. Model Logic (Generated with GenAI)

The core logic of the predictive model for customer delinquency involves a structured process to transform raw customer data into a risk prediction. Below is a step-by-step description of this process.

Problem Definition

- **Define Delinquency:** Clearly define what constitutes a 'delinquent' account. Based on the provided dataset, this would involve examining columns such as *Delinquent_Account* and the *Month_1* to *Month_6* columns. If *Delinquent Account* is a binary flag (0 or 1), it could directly serve as the target variable. Otherwise, a delinquent account could be defined as an account with a certain number of missed payments over a specified period.
- **Target Variable:** Identify the target variable. From the given data description, *Delinquent_Account* (if it is a binary variable) or deriving a new one from *Month_1* to *Month_6* (e.g. if a customer has a certain number of 'Late' or 'Missed' payments).

Data Preprocessing

- **Handle Missing Values:** Identify and handle any missing values. This might involve imputation (mean, median, mode, or more sophisticated methods) or removal of rows/columns depending on the extent of missingness.
- **Handle Outliers:** Detect and address outliers in numerical features (e.g., *Income*, *Loan_Balance*). This could involve capping, transformation, or removal.
- **Encode Categorical Variables:** Convert categorical features (e.g., *Employment_Status*, *Credit_Card_Type*, *Location*, and the *Month* columns if they are to be used as features) into numerical representations using techniques like one-hot encoding, label encoding, or target encoding.
- **Feature Scaling:** Scale numerical features (e.g., *Age*, *Income*, *Credit_Score*, *Loan_Balance*, *Debt_to_Income_Ratio*, *Credit_Utilization*, *Missed_Payments*, *Account_Tenure*) to a standard range (e.g., using StandardScaler or MinMaxScaler) to ensure that no single feature dominates the model due to its scale.

Feature Engineering

- **Payment History Aggregation:** Create aggregate features from the *Month_1* to *Month_6* columns (e.g., total number of *Missed* or *Late* payments, percentage of on-time payments).
- **Debt-to-Income Ratio Interaction:** Create interaction terms if *Debt_to_Income_Ratio* needs to be considered along with other features such as *Income* or *Loan_Balance*.
- **Age Bins:** Bin *Age* into categories (e.g., young, middle-aged, senior) to capture non-linear relationships.
- **Credit Score Categories:** Categorize *Credit_Score* into ranges (e.g., poor, fair, good, excellent).

Feature Selection

- **Correlation Analysis:** Identify highly correlated features and consider removing one of them to reduce multicollinearity.

- **Univariate Feature Selection:** Use statistical tests (e.g., Chi-squared for categorical features, ANOVA for numerical features) to select features that are most strongly related to the target variable.
- **Recursive Feature Elimination (RFE):** Use a model-based feature selection technique where features are recursively removed, and the model is trained until the optimal subset of features is found.
- **Feature Importance from Tree-based Models:** Use feature importance scores from models like Random Forest or Gradient Boosting to identify the most impactful features.

Model Selection

- **Classification Algorithms:** Consider various classification algorithms suitable for predicting a binary outcome (delinquent/not delinquent):
    - **Logistic Regression:** A good baseline model, interpretable.
    - **Decision Trees:** Simple, interpretable, can handle non-linear relationships.
    - **Random Forest:** Ensemble method, robust to overfitting, good performance.
    - **Gradient Boosting (e.g., XGBoost, LightGBM):** High-performing ensemble methods, often win Kaggle competitions.
    - **Support Vector Machines (SVM):** Effective in high-dimensional spaces.
    - **Neural Networks:** Can capture complex non-linear relationships, but require more data and computational resources.
- **Handling Imbalanced Data:** If the number of delinquent accounts is significantly lower than non-delinquent accounts (which is common in credit datasets), address class imbalance using techniques like:
    - **Resampling:** Oversampling the minority class (e.g., SMOTE) or undersampling the majority class.
    - **Cost-sensitive Learning:** Assigning different misclassification costs to each class.

Model Training

- **Data Splitting:** Split the preprocessed data into training, validation, and test sets.
    - **Training Set:** Used to train the model.
    - **Validation Set:** Used for hyperparameter tuning and model selection.
    - **Test Set:** Used for final, unbiased evaluation of the chosen model's performance on unseen data.
- **Cross-Validation:** Implement k-fold cross-validation during training to get a more robust estimate of model performance and reduce variance.
- **Hyperparameter Tuning:** Optimize model hyperparameters using techniques like GridSearchCV or RandomizedSearchCV to find the best combination of parameters.

Model Evaluation

- **Metrics for Classification:** Evaluate model performance using appropriate metrics, especially considering the potential for class imbalance:
    - **Accuracy:** Overall correctness (less reliable with imbalanced data).
    - **Precision:** Proportion of correctly predicted positive cases among all predicted positive cases.
    - **Recall (Sensitivity):** Proportion of correctly predicted positive cases among all actual positive cases.
    - **F1-Score:** Harmonic mean of precision and recall.
    - **ROC AUC (Receiver Operating Characteristic - Area Under Curve):** Measures the ability of the model to distinguish between classes.

    - **Confusion Matrix:** Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.
- **Business Impact:** Translate model performance metrics into business terms (e.g., cost of false positives vs. false negatives).

Modeling Options for Predicting Delinquency

- Predicting delinquency involves selecting between models like Logistic Regression and Gradient Boosting (e.g., XGBoost).
- **Logistic Regression** is a straightforward approach, predicting binary outcomes by estimating the relationship between features and the log-odds of the outcome. It offers clear interpretability, fast training, and serves as a solid baseline. However, its linearity assumption may overlook complex patterns, and it is sensitive to outliers, limiting its predictive power on intricate datasets.
- **Gradient Boosting**, particularly XGBoost, leverages an ensemble of decision trees to iteratively improve predictions, capturing non-linear relationships and feature interactions effectively. It excels in predictive accuracy and robustness, handling outliers and missing data with ease while providing insights through feature importance metrics. Despite its computational demands, complexity in hyperparameter tuning, and reduced interpretability, XGBoost's ability to handle complex, real-world data makes it a powerful choice for delinquency prediction.

2. Justification for Model Choice

Gradient Boosting, specifically XGBoost, is highly suitable for Geldium's delinquency prediction needs due to its superior predictive accuracy. In the financial domain, even marginal improvements in accuracy can yield significant benefits by reducing losses and optimizing lending strategies. XGBoost's exceptional performance on tabular data ensures precise identification of high-risk accounts, enabling better credit risk assessment and decision-making. Its ability to capture complex, non-linear relationships between variables further enhances its effectiveness, addressing the intricacies of financial behaviors without requiring extensive manual feature engineering.

XGBoost's robustness and scalability make it ideal for real-world financial datasets, which often contain outliers, noise, and missing values. Its capability to handle these imperfections reduces the need for aggressive pre-processing, saving time and resources. Furthermore, XGBoost is optimized for performance on large-scale datasets, ensuring seamless integration with Geldium's extensive customer base. Additionally, the model provides feature importance metrics, offering valuable insights into the drivers of delinquency, which can guide strategic interventions and policy refinements.

While less interpretable than logistic regression, XGBoost's trade-off in transparency is outweighed by its substantial gains in accuracy and flexibility. By leveraging XGBoost, Geldium can proactively mitigate risks, prioritize resource allocation effectively, and enhance customer interventions, ultimately maintaining a competitive edge in the financial landscape. This approach ensures data-driven, nuanced decision-making aligned with the institution's goals of profitability, risk management, and compliance.

3. Evaluation Strategy

My evaluation strategy for this financial risk prediction model encompasses a comprehensive set of metrics designed to assess accuracy, robust performance for imbalanced data, and crucial fairness considerations.

Accuracy Metrics

These metrics quantify how well the model predicts the true outcome.

- **Accuracy**: The proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances.

$$\frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: The proportion of true positive predictions among all positive predictions made by the model.

$$\frac{TP}{TP + FP}$$

- **Recall (Sensitivity or True Positive Rate)**: The proportion of true positive predictions among all actual positive instances.

$$\frac{TP}{TP + FN}$$

- **F1-Score**: The harmonic mean of Precision and Recall. It provides a single score that balances both metrics.

$$2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)$$

- **ROC AUC (Receiver Operating Characteristic - Area Under the Curve)**: Measures the ability of the model to distinguish between classes. It represents the probability that the model ranks a randomly chosen positive instance higher than a randomly chosen negative instance.

Fairness and Bias Checks:

- **Demographic Parity (Statistical Parity)** ensures that the proportion of positive predictions, such as credit approvals, is roughly equal across different groups, regardless of their actual outcomes. This metric helps identify whether the model's decisions are balanced across groups, with significant differences potentially indicating disparate impact.
- **Equal Opportunity** focuses on ensuring that the True Positive Rate (Recall) is equal across different groups. It evaluates whether the model identifies those who should receive a positive outcome—such as truly non-delinquent individuals—equally well across all demographic groups.
- **Equal Accuracy** requires that the overall accuracy, or correct classification rate, is consistent across different groups. This metric provides a broader comparison of the model's predictive performance, ensuring equitable treatment in terms of overall correctness.

- **Disparate Impact (80% Rule)** is a legal and ethical guideline that mandates the selection rate for a protected group be at least 80% of the selection rate for the group with the highest selection rate. Mathematically, this is expressed as:

$$\frac{\textbf{\textit{Selection Rate}}\ (\textbf{\textit{Protected Group}})}{\textbf{\textit{Selection Rate}}\ (\textbf{\textit{Most Favored Group}})} \geq 0.80$$

- **Bias Amplification** measures whether the model exaggerates societal biases present in the training data, leading to outcomes more skewed than those observed in the real world. This metric is critical in understanding if the model exacerbates pre-existing inequalities, thus ensuring fairness in its predictions.

Bias Mitigation Techniques in Predictive Modeling for Credit Risk

Bias mitigation techniques can be applied at various stages of the machine learning pipeline: pre-processing, in-processing, and post-processing. These approaches help reduce bias while maintaining the model's predictive utility.

- **Pre-Processing Techniques (Before Training):** Pre-processing methods address bias by modifying the training data. **Fairness Through Relabeling** adjusts labels in the training set to correct historical biases, such as over-classifying a group as delinquent. **Reweighting** assigns higher weights to samples from underrepresented or disadvantaged groups, ensuring the model pays attention to their patterns during training. **Disparate Impact Remover (DIR)** transforms feature values to reduce bias linked to sensitive attributes while retaining their predictive utility.
- **In-Processing Techniques (During Training):** In-processing methods modify the learning algorithm during training. **Adversarial Debiasing** uses two competing networks—a predictor and an adversary. The predictor learns to make accurate predictions while minimizing its dependency on sensitive attributes. **Regularization with Fairness Constraints** incorporates a fairness penalty in the model's objective function, encouraging it to optimize for both accuracy and fairness metrics, such as balancing False Positive Rates across groups.
- **Post-Processing Techniques (After Training):** Post-processing methods adjust predictions after training. **Equalized Odds (Threshold Adjustment)** modifies classification thresholds to equalize True Positive and False Positive Rates across groups. **Reject Option Classification (ROC)** defers decisions on ambiguous predictions to human experts or applies separate rules for disadvantaged groups. **Calibration** adjusts predicted probabilities to ensure consistency between predicted and observed outcomes across demographics, improving fairness and accuracy.