## Model Training

### I. Feature Selection and Dimensionality Reduction:

1. Top 50 Columns Selection: From the original 797 columns, a focused selection of the columns most correlated with the target variable was made.
    a. Purpose and Benefits:
        i. Computational Efficiency: Crucial when using algorithms sensitive to the curse of dimensionality, like KNN.
        ii. Data Retention: The risk of noise interference is reduced.
        iii. Enhanced Imputation: Reduced feature sets can improve imputation quality, especially when feature similarity is accounted for.
        iv. Oversampling Precision: Having fewer but more relevant features can decrease the potential for generating noisy samples and limit the replication of non-essential features.
2. Further Refinement Using SelectKBest: The top 50 columns were further reduced to 20 using the SelectKBest method.
    a. Mechanism and Purpose: By using a score function that captures both linear and non-linear relationships, the most impactful columns were isolated. This ensures the model capitalizes on data that has a strong and meaningful predictive association with the target variable.

### II. Data Sampling and Imputation:

1. Stratified Sampling: A representative subset of the dataset was extracted.
    a. Purpose: Achieving computational efficiency while maintaining the integrity of the original class distribution of the target variable.
2. KNN Imputation: To address missing data, the KNN imputation method was applied.
    a. Mechanism and Benefits: By using the mean value from the 5 nearest neighbors, data patterns and relationships are captured, producing more contextual and reliable imputations compared to simpler methods.

### III. Data Splitting:

1. The data was divided into training and testing sets, ensuring unbiased evaluation and consistent results.
2. Considerations: Uniform splits for both original and imputed data sets were crucial to facilitate aligned evaluations. Moreover, IDs, while not used in modeling, were retained for traceability, ensuring that future insights could be mapped back to the original data.

### IV. Evaluation Metrics:

1. ROC-AUC: Evaluates the classifier's balance between true positive rate and false positive rate, encouraging the comprehensive discriminative ability of a model.
2. Fβ (β = 10): This emphasizes recall over precision, thus minimizing the FN rate.
3. Business Cost: Customized to heavily penalize false negatives, it ties the model's performance directly to potential business impacts.

## V. Model Training and Tracking Pipeline:

1. Integrated Training Framework:
    a. Pipeline for Preprocessing and Model Training
        i. Feature standardization
        ii. Oversampling to address class imbalance
        iii. Hyperparameter optimization via cross-validated randomized search
    b. Training Mechanism
        i. Executes training via the constructed pipeline
        ii. Generates plots of the ROC curve and confusion matrix
        iii. Calculates global feature importance
        iv. Archives to MLflow
2. MLflow Tracking
    a. Metrics, parameters, plots and models are tracked and logged, ensuring systematic documentation and facilitating post-hoc analysis.

## VI. Logistic Regression Model Training:

1. Model Description: Logistic Regression is a foundational algorithm for binary classification that models the probability of a given sample belonging to a particular category by applying a logistic function to a linear combination of features.
2. Hyperparameters:
    a. Regularization strength: Values from strong to weak.
    b. Penalization norm: Lasso, ridge or none.
    c. Optimization algorithm: Newton-CG, LBFGS or SAGA

## VII. LightGBM Classifier Model Training:

1. Model Description: The Light Gradient Boosting Machine is a gradient boosting framework designed for speed and efficiency. It constructs decision trees based on a leaf-wise strategy, allowing for faster convergence and improved accuracy with large datasets.
2. Oversampling and Imputation Strategies
    a. SMOTE cannot handle missing data, so the models are trained using imputed data and oversampling, followed by original and then imputed data without oversampling.
3. Hyperparameters:
    a. Learning Rate: Chosen to balance model's training speed and accuracy.
    b. Number of Estimators: Specified a wide range for diverse boosting iterations.
    c. Boosting Algorithm: Standard methods - gradient boosting and dropouts.
    d. Tree Complexity: Shallow trees with a moderate count of leaves to manage bias-variance tradeoff.
    e. Preventing Over/Underfitting:
        i. Set minimum data points per leaf for granularity.
        ii. Defined a fraction for features per tree to ensure diverse feature usage.
        iii. Determined bagging fraction to efficiently utilize the dataset.
    f. Bagging Frequency: Varied from none to every iteration to influence model's adaptability to new data.
    g. Regularization: Incorporated for feature selection and to enhance stability.

## Handling Class Imbalance

### I. Data Sampling

1. Stratified sampling ensured the class distribution from the original dataset was retained. As a result, representation of the minority class remained consistent across data subsets.

### II. Feature Standardization

1. Prior to oversampling, feature standardization was integrated into the pipeline, ensuring an equal contribution of each feature to distance-based methods, such as SMOTE, facilitating unbiased synthetic sample generation.

### III. Oversampling with SMOTE (Synthetic Minority Over-sampling Technique)

1. An argument was provided to implement the SMOTE technique within the model training pipeline.
2. When SMOTE was activated, the pipeline incorporated this technique to artificially augment the minority class in the dataset.
3. SMOTE identifies k-nearest neighbors in the minority class and generates synthetic instances between these neighbors, enhancing class balance.

### IV. Model Selection and Characteristics

1. Logistic Regression
    a. Its hyperparameters, including regularization strength and penalty type, can indirectly aid in mitigating class imbalance effects.
2. LightGBM Classifier
    a. Hyperparameters like tree complexity, bagging, and regularization can be tuned to manage the bias-variance trade-off, allowing the model to better recognize the minority class.

### V. Model-specific Class Weights:

1. Class weights were computed for scenarios where oversampling was not used.
    a. For Logistic Regression:
        i. The compute_class_weight function was used to calculate balanced class weights.
        ii. A range of positive class weights was defined to allow the model to account for the class imbalance by adjusting the cost of misclassifications during hyperparameter tuning.
    b. For LGBMClassifier:
        i. The ratio of the negative class to the positive class was computed and used to adjust the scale_pos_weight parameter of the model, providing a way to handle class imbalance by setting a higher penalty on misclassifying the minority (positive) class.
        ii. A range of positive class weights was provided for hyperparameter tuning.

### VI. Diverse Experimentation Strategies:

1. Models underwent training using various combinations of strategies, such as:
    a. With or without SMOTE
    b. With or without imputed data
    c. Either using class weights or default weights
2. This diverse approach provided a robust evaluation framework, enabling identification of the most effective strategy to optimize model performance in the presence of class imbalance.

## Business Cost Function, Optimization Algorithm and Evaluation Metric

### I. Business Cost Function

1. Rationale: In credit assessment, misclassifications have asymmetric financial implications.
    a. Predicting a bad client as a good one (false negative) leads to loan defaults and subsequent financial losses.
    b. Misclassifying a good client as a bad one (false positive) results in foregone potential revenue.
    c. The potential revenue loss from a false positive is typically lesser than a loan default's cost. Hence, it's assumed that the cost of a false negative is ten times that of a false positive.
2. Implementation: The business_cost function in the business_cost_metric module computes costs as: (fn_cost × number of false negatives) + (fp_cost × number of false positives).

### II. Threshold Optimization

1. Rationale: The default threshold of 0.5 may not always be ideal, particularly when misclassification costs differ significantly.
2. Implementation: The threshold_optimization function in the business_cost_metric module identifies the optimal threshold using minimize_scalar from scipy.optimize. This optimal point turns predicted probabilities into binary classes, guiding the business cost computation.

### III. Optimization Algorithm

1. Objective: Enhance model performance via hyperparameter tuning.
2. Approach:
    a. Method: Adopt RandomizedSearchCV for efficient parameter selection.
    b. Efficiency: In contrast to the exhaustive GridSearchCV, RandomizedSearchCV samples from specified parameter distributions. This method is computationally efficient and often yields comparable results.
3. Outcome: Pinpoints and chooses the optimal parameter combination, optimizing the model as per the chosen metric.

### IV. Evaluation Metric - Business Cost Metric

1. Purpose: Transition from solely statistical accuracy to assessing real financial impacts.
2. Workflow:
    a. Threshold Determination: Utilize threshold_optimization to find the best classification threshold.
    b. Class Assignment: Transform predicted probabilities into binary classes based on the derived threshold.
    c. Scoring:
        i. Cost Calculation: The business_cost function computes the associated business cost.
        ii. Normalization: Adjust the business cost by dividing it by the total prediction count for uniformity.
        iii. Alignment with Traditional Metrics: For a consistent comparison with metrics like AUC-ROC, where higher scores are better, the normalized business cost is inverted (1 - normalized cost). This adjustment ensures models are gauged based on their financial implications.

## Results Summary Table

The following table represents the best performing models from each combination of model type and training strategy, ranked based on their "Business Cost" score on the reserved test dataset. The models are listed in descending order of their performance.

| Model | Imputed | SMOTE | Class Weights | CV Metric | CV Score | Business Cost | | ROC-AUC | | Fβ (β = 10) | | Accuracy | | FN% Test | FP% Test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Train | Test | Train | Test | Train | Test | Train | Test | | |
| LGBMClassifier | False | False | True | Business Cost | 0.4476 | 0.5061 | 0.435 | 0.8081 | 0.6914 | 0.748 | 0.6648 | 0.6715 | 0.651 | 2.4 | 32.5 |
| LogisticRegression | True | False | True | ROC-AUC | 0.7198 | 0.4276 | 0.4165 | 0.7307 | 0.7039 | 0.548 | 0.523 | 0.7505 | 0.7405 | 3.6 | 22.35 |
| LogisticRegression | True | True | False | ROC-AUC | 0.7177 | 0.4181 | 0.4125 | 0.7285 | 0.7006 | 0.6242 | 0.6109 | 0.6791 | 0.669 | 2.85 | 30.25 |
| LGBMClassifier | True | False | True | Business Cost | 0.4367 | 0.4784 | 0.4105 | 0.7789 | 0.6991 | 0.5993 | 0.5107 | 0.763 | 0.7435 | 3.7 | 21.95 |
| LGBMClassifier | True | False | True | Fβ (β = 10) | 0.6359 | 0.4539 | 0.41 | 0.764 | 0.6968 | 0.689 | 0.6397 | 0.6642 | 0.644 | 2.6 | 33 |
| LGBMClassifier | False | False | True | ROC-AUC | 0.7318 | 0.4841 | 0.4095 | 0.78 | 0.6934 | 0.6791 | 0.5873 | 0.7046 | 0.684 | 3.05 | 28.55 |
| LGBMClassifier | False | False | True | Fβ (β = 10) | 0.6666 | 0.4814 | 0.4055 | 0.7889 | 0.6932 | 0.7608 | 0.6681 | 0.6344 | 0.617 | 2.35 | 35.95 |
| LogisticRegression | True | False | True | Business Cost | 0.433 | 0.4244 | 0.403 | 0.7307 | 0.7022 | 0.5421 | 0.5042 | 0.7518 | 0.7405 | 3.75 | 22.2 |
| LogisticRegression | True | True | False | Business Cost | 0.4531 | 0.4466 | 0.401 | 0.7408 | 0.7288 | 0.6587 | 0.647 | 0.6896 | 0.68 | 3.1 | 28.9 |
| LGBMClassifier | True | False | True | ROC-AUC | 0.7273 | 0.4725 | 0.399 | 0.7706 | 0.7022 | 0.5848 | 0.4921 | 0.7684 | 0.7455 | 3.85 | 21.6 |
| LogisticRegression | True | True | False | Fβ (β = 10) | 0.6634 | 0.4442 | 0.39 | 0.7321 | 0.7239 | 0.6679 | 0.6463 | 0.6794 | 0.669 | 3.1 | 30 |
| LogisticRegression | True | False | True | Fβ (β = 10) | 0.7157 | 0.435 | 0.39 | 0.7412 | 0.7284 | 0.7216 | 0.7045 | 0.624 | 0.615 | 2.5 | 36 |
| LGBMClassifier | True | True | False | ROC-AUC | 0.6914 | 0.4512 | 0.3585 | 0.7517 | 0.6619 | 0.6141 | 0.4896 | 0.7224 | 0.705 | 3.85 | 25.65 |
| LGBMClassifier | True | True | False | Business Cost | 0.4025 | 0.4222 | 0.348 | 0.7312 | 0.6605 | 0.5673 | 0.4651 | 0.7294 | 0.7125 | 4.05 | 24.7 |
| LGBMClassifier | True | True | False | Fβ (β = 10) | 0.6243 | 0.3881 | 0.338 | 0.6905 | 0.6374 | 0.6509 | 0.5706 | 0.6255 | 0.6215 | 3.15 | 34.7 |

## Conclusions

1. LightGBM was the top performing model and featured most prominently among those best at minimizing business costs, though Logistic Regression also showed competitive results, particularly when tuned using ROC-AUC.
2. While imputation was a necessity for logistic regression, three of the top five LightGBM models, including the best one, did not utilize it, indicating that it should not be vital to reaching peak performance for that model type.
3. Oversampling showed a marginal benefit to logistic regression models, but was decidedly detrimental for LightGBM.
4. Employing class weights demonstrated a definite positive impact, especially in the penalization of false negatives.
5. A discernible trade-off exists between decreasing the FN rate and raising the FP rate. LightGBM exhibited a superior balance in optimizing business cost and maintaining accuracy.
6. While business cost was the primary metric, the top models also exhibited reasonable ROC-AUC scores. This suggests they achieved some compromise in reducing false negatives while still distinguishing well between classes.

## Global and Local Model Interpretability

### I. Approval Probabilities

1. For any applicant, invoking a model's predict_proba method provides the probabilities of both approval and denial. By multiplying these probabilities by 100 they can be interpreted as percentage chances for each respective class.

### II. Feature Importances

1. Traditional Methods:
    a. Logistic Regression Coefficients: Represent the change in log-odds for a unit change in a feature. They predominantly depict linear relationships between the feature and the outcome's odds.
    b. LightGBM Built-in Feature Importance: Derived from feature splits or total gain; may sometimes prioritize noisy features. It doesn't show directional influence.
    c. LightGBM Permutation Importance: Can be more reliable than simple counting of splits, but might be biased in cases where features are correlated. A strong feature might see its importance diluted if another correlated feature captures some of its predictive power during permutation.
2. SHAP (SHapley Additive exPlanations):
    a. Based in cooperative game theory, ensuring fair distribution of contributions.
    b. Quantifies how much each feature shifts a particular prediction away from the model's average prediction.
    c. Evaluates the average marginal contribution of a feature across all possible combinations of features, accounting for feature interplay.
    d. Global importance is calculated as the average SHAP value of a feature across all data instances.
3. Decision: SHAP was chosen over traditional methods to calculate feature importance.
4. Reasoning:
    a. Due to SHAP's proficiency in capturing complex relationships and feature interactions, it became the chosen interpretative tool.
    b. Model Agnosticism: Whether interpreting a linear model, or a complex gradient boosting model, SHAP offers a consistent framework for interpretation.
    c. Consistent API Responses / Dashboard Integration: Given the need for the archived model to serve consistent responses to the dashboard regardless of the model type, SHAP values ensure that interpretations remain consistent and reliable, eliminating the need to switch interpretation methodologies based on the model.
5. Transformation and Presentation:
    a. The SHAP values are transformed using max-abs scaling, so they range from -1 to 1. This makes it more intuitive to compare the relative importance of features while keeping the original sign and relative magnitude of each value.
    b. Horizontal bar charts, centered around zero, are used to depict both local and global feature importance.
    c. As an additional means of visually presenting the impact of individual features on an applicant's prediction, the API generates Kernel Density Estimation plots for a selected number of features. These plots not only display the distribution of values for both approved and denied applicants but also highlight where the applicant's value lies within those distributions and whether it contributed to their likelihood of approval or denial.

## Limitations and Potential Improvements

### I. Feature Engineering and Selection

Limitations:

A majority of the features were derived using general statistical functions such as min, max, mean, sum, and variance applied to grouped tables. Given the probable correlations between many of the features, there was a noticeable, albeit not significant, inclination towards overfitting, especially in the LightGBM models. No domain-specific feature engineering was performed.

Potential Improvements:

Consulting someone with domain expertise or undertaking a thorough investigation into credit risk analysis could facilitate more insightful feature engineering. While the current methodology involves Pearson correlations followed by SelectKBest for feature reduction, an upfront use of SelectKBest while conserving a larger proportion of the original feature set, or even employing advanced techniques like the Boruta algorithm or recursive feature elimination for individual models, might yield a more ideal feature set.

### II. Data Constraints

Limitations:

Training and validation were restricted by computational and time constraints.

Potential Improvements:

Leveraging a larger sample size for training and validation could reveal more subtle patterns, thereby improving model robustness and generalizability.

### III. Hyperparameter Tuning

Limitations:

The choice of RandomizedSearchCV, though efficient, might not have canvassed the hyperparameter space as comprehensively as other techniques.

Potential Improvements:

Implementing GridSearchCV or Bayesian optimization could enable a more thorough exploration of the hyperparameter space, and providing more comprehensive ranges for many parameters would allow for more finely tuned models.

### IV. Model Diversity

Limitations:

The limited number of models explored, consisting of logistic regression and the LightGBM classifier, might have missed out on specific architectures better suited for this kind of problem.

Potential Improvements:

Expanding the array of models explored could enhance performance. This could entail testing other gradient-boosted trees, support vector machines, deep neural networks, and even ensemble methods such as stacking or boosting.

### V. Handling Class Imbalance

Limitations:

The approach to address class imbalance only included SMOTE and class weighting.

Potential Improvements:

Diversifying techniques to manage class imbalance could range from data-centric interventions like alternative oversampling methods (e.g., ADASYN) and undersampling techniques like Tomek links to algorithmic adjustments like optimizing for alternative metrics such as AUC-PR.

# Data Drift Analysis

## I. Objective

The objective of this analysis was to identify potential data drift in a production environment. Data drift refers to unforeseen changes in the distribution of the model's input data over time. Such changes can result in degraded model performance if not managed appropriately.

## II. Data Utilized

1. Reference Data: Derived from the application_train dataset, which represents the data used during the model training phase.
2. Current Data: Derived from the application_test dataset, representing new client data post-deployment.
3. Data Preparation: Only features determined through prior feature engineering and selection were considered for drift detection.

## III. Data Drift Report Generation:

The "Evidently" library was used, which was designed for analyzing and visualizing data quality, drift, and performance metrics, aiding in the continuous monitoring of machine learning models in production.

1. Default values were used for the statistical test threshold and confidence level.
    a. Given there are only two datasets separated by an unknown time interval, it's not feasible to gauge the volatility or consistent trajectory of data changes over multiple points in time. In such scenarios it's prudent to rely on the defaults, otherwise it might introduce arbitrariness into the drift detection process.
2. Statistical Test Used: Wasserstein Distance
    a. The Evidently documentation recommends this test for numerical features with more than 5 unique values in datasets with more than 1000 observations.
    b. Often referred to as the Earth Mover's Distance, it quantifies the difference between two probability distributions. It represents the minimum cost of transporting data points to transform one distribution into the other, providing a holistic view of distributional differences, capturing both location and shape change.

## IV. Results

1. Out of 20 features analyzed, 2 exhibited signs of potential data drift, or 10% of columns.
    a. Features with detected drift (crossing the 0.1 threshold):
        i. BURO_STATUS_1_MEAN_MEAN: Drift score of 0.112
        ii. CC_CNT_DRAWINGS_ATM_CURRENT_MEAN: Drift score of 0.143
2. Detected features did not significantly exceed the threshold, indicating potential but not definitive drift.

## V. Conclusion

While most features remained stable, ten percent showed potential signs of drift. The dataset as a whole didn't exhibit any overarching drift. However, given that drift can emerge over time, continuous monitoring and periodic drift analysis are recommended to ensure model robustness and performance in production.