FML HW2 - Credit Score Classification

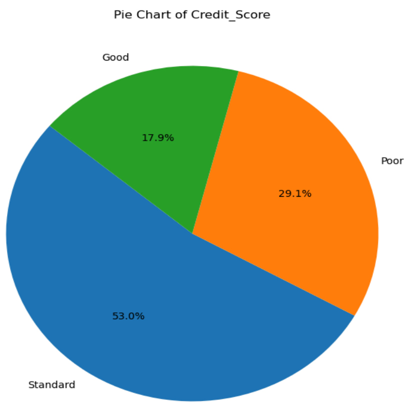
Team Detective

Tang Chia Tien, Tu Nhu Pham, Nicolas Reboullet, Ariel Torjmane

Section 1 - Feature Engineering

Data Overview:

- **Features for training:** float64(18), int64(2), object(6) => 26 features in total.
- **Target Variable:** object of 3 types('Poor', 'Good', 'Standard').



Feature Selection & Engineering (Step by Step)

1. **Unbalance of Data:**

The pie chart of the “Credit_Score” distribution shows the training dataset is very unbalanced, so we will have to fine tune the model if there's a "weights" parameter applicable to the model.

2. **Prevent Overfitting and Privacy Conflict:**
   Drop features 'Customer_ID', 'Name', and 'SSN'.

- **Overfitting Prevention:** These identifiers are unique to each individual and can lead to overfitting, as the model might learn patterns that are specific to the training data but don't generalize to new data.
- **Privacy:** 'Name' and 'SSN' are personally identifiable information (PII). Including PII in a model raises privacy concerns and can lead to potential legal implications, especially under regulations like GDPR.
- **Relevance:** Such features usually do not hold meaningful information that contributes to the predictive power of the model.

3. **Dropping the 'Month' column:**

It’s justified as it likely represents a time identifier without intrinsic predictive power for credit scoring and we are not doing time series forecasting, focusing the model on more impactful, behavior-driven financial features instead.

4. **Expand the object in 'Type of Loan':**

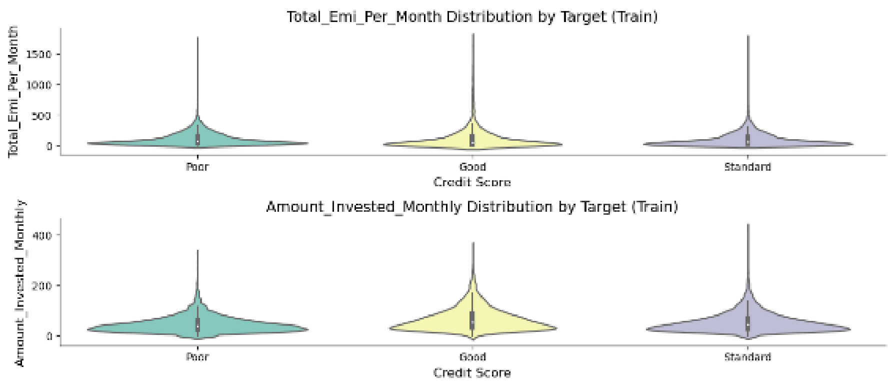
In transforming the 'Type of Loan' column into separate features, we enable the model to capture the unique influence of each loan category on the target variable. This approach avoids the potential pitfalls of ordinal encoding, where the model might improperly assume a hierarchy among loan types.

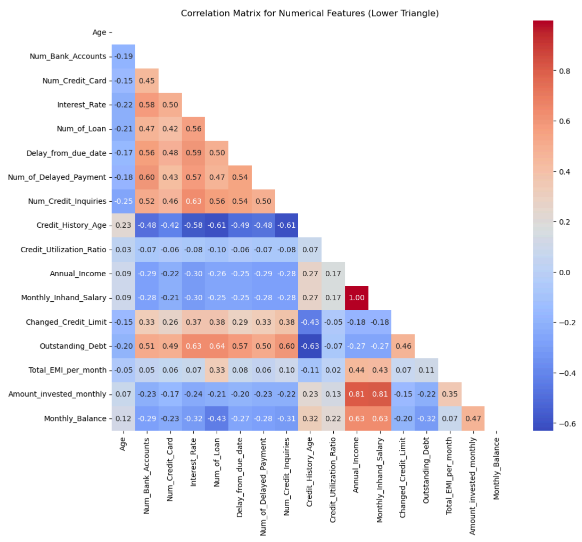
5. **Expand the object in 'Payment_Behaviour':**

Same as 4., we split the Payment Behavior according to its 2 parts: 'Amount_Spent' & 'Value_of_Payments' to catch more info from each part. There are 3 types of Amount_Spent(small, medium, large) and 2 types of Value_of_Payments(small, large).

6. **Based on the corr heat map and violin plots for the various features by 'Credit_Score' category, we created and deleted some features:**
   - ❖ **Credit Utilization Ratio:** We've adopted a binary approach to classify Credit_Utilization_Ratio, distinguishing between 'High' (over 30%) and 'Low' (up to 30%) utilization. This decision aligns with the credit industry's benchmark, where maintaining a utilization ratio below 30% is generally seen as indicative of responsible credit management.
   - ❖ **Credit Delay Status:** Creating a binary class based on Delay_from_due_date exceeding a threshold (mean delay in 'Poor' credit score category) effectively highlights cases with significantly late payments, a key indicator of credit risk, enhancing the model's ability to differentiate between risk levels.

- **Income Related Features:** We drop 2 of them because their corr is between 1.0 and 0.7 which means 2 of them are redundant and provide no more information and 'Monthly_Inhand_Salary' is a direct reflection of liquid income of 'Monthly_Balance'. Also, Normalizing income by the cost of living or creating ratios of income to other financial obligations might provide additional insight.





- **Investment Proportion and EMI-to-income ratio Features:** "Amount_invested_monthly' and 'Total_EMI_per_month' have wider distributions in some credit score categories. Creating features like Investmen Proportion or EMI-to-income ratio could be predictive. Additionally, 'Total_EMI_per_month' is likely correlated with 'Outstanding_Debt', which can serve as a proxy for the same underlying information. And also for 'Amount_invested_monthly', it's similar to 'Total_EMI_per_month', the spread of investment amounts does not visibly differ across credit score categories, which might limit its utility in distinguishing between them. So we will drop them after creating the new features.
- **Credit History Age:** If older credit histories correlate with better credit scores, a categorical feature indicating the age of the credit history (e.g., new, established, long-standing) might be informative. Use the median of "Credit History Age" based on "Credit_Score" type as criteria of categorizing.
- **'Payment_of_Min_Amount':** 'Payment_of_Min_Amount' shows a similar distribution pattern across all credit score classes, indicating it may not significantly differentiate between them. By removing it, we focus the model's learning on more distinctive features.

Section 2 - Model Tuning and Comparison

Model Name	Hyperparameter	Accuracy Rate (CV=5)	Kaggle Score
Logistic Regression	max_iter=1000, penalty="l2", solver="lbfgs", class_weight = "balanced"	0.663 (σ: 0.002)	0.663
KNN	n_neighbors=2, p=1	0.798 (σ: 0.002)	0.8

SVM	class_weight = "balanced", kernel = 'poly', C=10,random_state=42	0.663 (σ: 0.003)	0.690
Decision Tree	class_weight = "balanced", criterion="entropy", min_samples_split=3, random_state=42	0.772 (σ: 0.004)	0.774
Ensemble Learning			
Gradient Boosting	n_estimators=100, max_depth = 10, min_samples_split=3, random_state=42	0.799 (σ: 0.0007)	0.805
Random Forest	n_estimators=150, min_samples_split=3, class_weight = "balanced", random_state=42	0.814 (σ: 0.002)	0.819

Logistic Regression:

- **Parameter Tuning**: Max_iter was set to 1000 for full convergence on complex datasets, and class_weight to "balanced" to adjust for imbalanced classes. "L2" penalty for Ridge regularization prevents overfitting, and the "lbfgs" solver is effective for small datasets with "l2" penalty.
- **Preventing Overfitting:** Regularization ("l2" penalty) and balanced class weights prevent model bias and overfitting. Cross-validation ensures generalizability by testing on various data subsets.
- **Additional Commentary:** The model's simplicity serves as a baseline, highlighting potential needs for a more complex model due to the dataset's characteristics.

KNN:

- **Parameter Tuning:** The n_neighbors hyperparameter was set to 2 for optimal cross-validation accuracy, after testing values from 1 to 10. The 'uniform' weights parameter was chosen over 'distance', and the Minkowski p parameter was set to 1 (Manhattan distance), more effective for this dataset than Euclidean distance.
- **Preventing Overfitting:** Despite overfitting risks, a lower n_neighbors value improved performance, confirmed by cross-validation for robustness across data subsets.
- **Additional Commentary:** The KNN model's superiority over Logistic Regression suggests the dataset has local structures better captured by KNN. However, KNN's higher computational cost at prediction time and its efficacy in capturing non-linear relationships should be considered for deployment.

SVM:

- **Parameter Tuning:** "Balanced" class_weight was used to prioritize minority classes. The 'poly' kernel outperformed 'rbf' and 'linear' in tests, and the regularization parameter C was optimally set to 10, after comparing with 0.1 and 1, balancing margin maximization and classification error.
- **Preventing Overfitting:** Balanced class weights and a higher C value, combined with the 'poly' kernel, mitigate overfitting while capturing complex data patterns.
- **Additional Commentary:** While computationally intensive, particularly for large datasets, it could benefit from further exploration of kernel parameters and input space dimensionality for improved performance.

Decision Tree:

- **Parameter Tuning:** Class_weight was set to "balanced" to counter class imbalances. The "entropy" criterion for information gain was chosen over "gini" and "log_loss" for better split quality. Min_samples_split was optimized to 3, balancing underfitting and overfitting.

- **Preventing Overfitting:** Min_samples_split was limited to 3, ensuring nodes split only with more than two samples, promoting generalization. The 'entropy' criterion aids in making informative splits by reducing uncertainty significantly.
- **Additional Commentary:** The Decision Tree's simplicity may limit its ability to handle complex data, but it offers clear insights into feature importance and decision logic..

Gradient Boosting:

- **Parameter Tuning:** ‘n_estimators’ was set to 100, optimizing the number of boosting stages for balance between model complexity and training efficiency. ‘max_depth’ was tuned to 10 to capture complex patterns without memorizing the data(*balancing the need to model complex feature interactions with the risk of learning noise*), in combination with ‘min_samples_split’=3 to avoid overfitting by preventing splits on small subsets.
- **Preventing Overfitting:** `max_depth`=10 and `min_samples_split`=3 together control tree complexity in the ensemble, minimizing overfitting.

Random Forest:

- **Parameter Tuning:** n_estimators were set to 150 for an optimal number of trees in the ensemble, balancing robustness and pattern recognition. min_samples_split was determined to be 3, avoiding overly fine-grained splits and overfitting. The class_weight was set to "balanced" to give more weight to the minority class, addressing class imbalance.
- **Preventing Overfitting:** Balanced class weight and a minimum sample split of 3 prevent the model from overfitting and focusing on noise.
- **Additional Commentary:** Choosing n_estimators=150 and min_samples_split=3 reflects the dataset's complexity, allowing for a sufficiently large but not overly complex ensemble. This ensures the model captures essential data patterns while avoiding overfitting.

Section 3: Conclusion

In our comparative study of six machine learning models, Random Forest emerged as the top performers due to its ensemble methods, effectively handling complex data patterns and demonstrating robustness against overfitting. It led with the highest accuracy, closely followed by Gradient Boosting and K-Nearest Neighbors. Decision Tree showed moderate results, while Logistic Regression and SVM lagged behind, likely due to their limited ability to capture complex relationships in the data.
Thus, We choose Random Forest as our best model to predict the “Credit Score” in this Kaggle Competition.