ORIE 5741 Final Project Report

Authors: Simon Tian, jt886 Cecilia Yang, xy96 Kayla Yang, cy354

1. Problem Statement

Accurate credit assessment is a critical aspect of individual financial management. While there were previously existing methods to achieve this outcome, these methods often involve hard inquiries that can adversely affect an individual's credit score. The project, *"Caffeinated Treasury Bot"* (CTB), seeks to address the challenge by providing a tool that predicts the probability of credit card approval for individuals, along with a guide on improving credit scores.

2. Importance of the Project

The project aims to empower individuals with accessible tools to manage their credit effectively with minimal prior knowledge. This project aligns with the goal of increasing transparency in individual credit score management, and provides the team with the opportunity to exercise the knowledge acquired through ORIE 4741/5741 in practice. By minimizing the negative impact of hard inquiries through the predictive capabilities of the project, banks can improve customer satisfaction and customers can benefit from financial literacy and risk-free credit checks.

3. The Dataset

After rigorous research and comparison, we decided to use the following dataset for the project: https://www.kaggle.com/datasets/parisrohan/credit-score-classification
These datasets provide comprehensive information on credit applications, outcomes, and individual credit risk factors, making them suitable for training the predictive models.

Features:
1. Age
2. Annual Income
3. Interest Rate
4. Number of Loan
5. Delay from Due Date
6. Changed Credit Limit
7. Outstanding Debt
8. Credit History Age
9. Total EMI per Month
10. Amount Invested Monthly
11. Monthly Balance
12. Number of Delayed Payment
13. Number of Bank Accounts
14. Number of Credit Cards
15. Number of Credit Inquiries
16. Credit Utilization Ratio

Label: Credit Score

4. Data Analysis Approach and Result

4.1 Data Cleaning

To reduce the eccentric noise of our dataset, we have performed the following procedures:

- **Data Type Conversion**:
  Ensuring that each numerical column has type float, while the categorical columns are of type object.
- **Outlier Removal**:

Percentile analysis was used, and the distribution of the data was plotted to determine a reasonable range of data. Then, the outliers beyond reasonable range were removed.

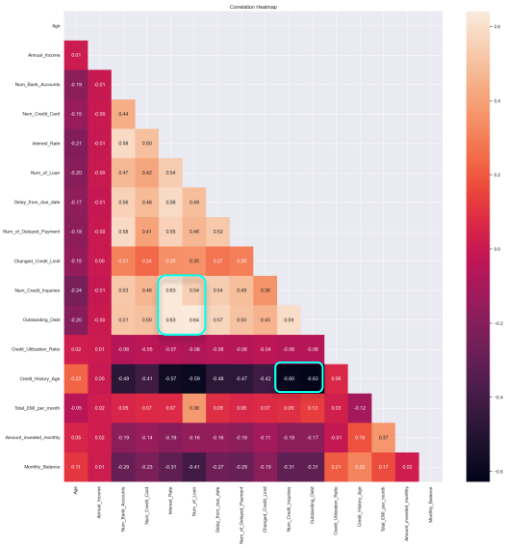
- **Missing Data Imputation**:
  For the missing data, we choose the Python *"rfill"* method which fills the data with its most recent value. This is because we have realized that our data is ordered in a way that each person has multiple rows, filling the missing data with the closest values ensures that the person is likely filled with his/her previous data.
- **Dropping Redundant Columns**:
  Due to the large size of the dataset, we decided to drop all *"NaN"* columns and repetitive rows to ensure data readiness.

4.2 Data Analysis

4.2.1 Correlation Map:

Observed from the correlation heatmap shown on the right, the following columns have the highest correlations among all parameter pairs:

Interest_Rate and Num_Credit_Inquiries	**0.63**
Interest_Rate and Outstanding_Debt	**0.63**
Num_of_Loan and Outstanding_Debt	**0.64**
Num_Credit_Inquiries and Credit_History_Age	**-0.60**
Outstanding_Debt and Credit_History_Age	**-0.63**

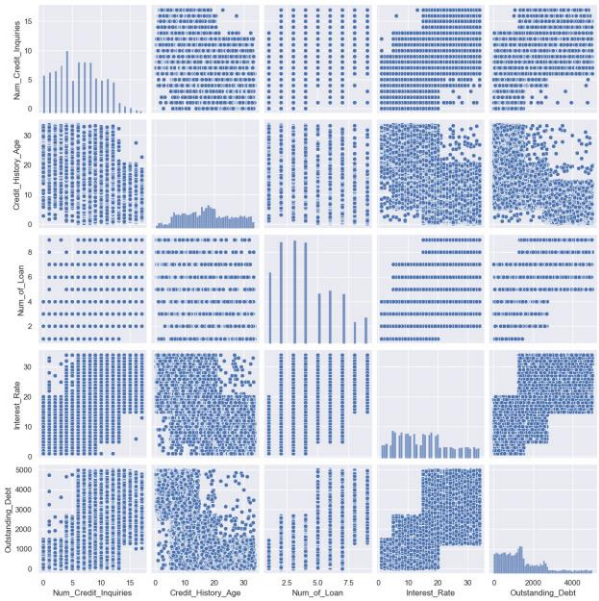


4.2.2 Pairplot

From the correlation plot on the right, we observed that:

- **Num_Credit_Inquiries** & **Num_of_Loan**: Positive correlation; an increase in one is associated with an increase in the other.
- **Credit_History_Age** & **Interest_Rate**: Potential negative correlation; a longer credit history might be associated with lower interest rates, but the correlation is low.
- **Outstanding_Debt**: Weak positive relationships with Num_of_Loan and Num_Credit_Inquiries; however, no clear linear pattern is visible.

Typically, highly correlated features provide redundant information. Therefore, from the above analysis. This analysis helps us to remove highly correlated features in feature selections.



4.2.3 Chi-square testing

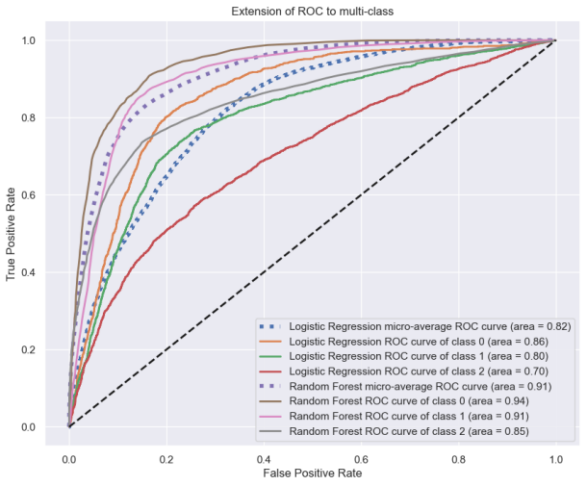
As we can see from the result of the Chi-square Testing, the features, ***Annual Income*** and ***Type of Loan*** are not statistically significant in predicting credit scores. Therefore, we may be considering dropping these features in our later feature selection process.

	Feature	Test Statistic	p-value	Significance
5	Interest_Rate	7072.6870	0.0000e+00	True
11	Num_Credit_Inquiries	5488.7343	0.0000e+00	True
8	Delay_from_due_date	4886.9547	0.0000e+00	True
4	Num_Credit_Card	4228.3615	0.0000e+00	True
3	Num_Bank_Accounts	4201.3578	0.0000e+00	True
13	Outstanding_Debt	4029.3863	0.0000e+00	True
15	Credit_History_Age	3912.9061	0.0000e+00	True
9	Num_of_Delayed_Payment	3710.6744	0.0000e+00	True
6	Num_of_Loan	2901.7266	0.0000e+00	True
16	Payment_of_Min_Amount	2103.3213	0.0000e+00	True
10	Changed_Credit_Limit	1115.0139	0.0000e+00	True
12	Credit_Mix	964.4133	0.0000e+00	True
20	Monthly_Balance	906.5111	0.0000e+00	True
1	Age	623.9846	5.2805e-268	True
18	Amount_invested_monthly	516.1178	2.5293e-222	True
19	Payment_Behaviour	275.2815	1.5045e-119	True
17	Total_EMI_per_month	28.5298	4.1453e-13	True
14	Credit_Utilization_Ratio	25.8714	5.8978e-12	True
0	Month	3.1105	4.4588e-02	True
2	Annual_Income	1.6452	1.9298e-01	False
7	Type_of_Loan	0.5858	5.5666e-01	False

4.3 Machine Learning with Baseline Models

During this stage, we did not perform any feature selections and added no treatment to our data. We trained three machine-learning models and observed the following results:

Logistic Regression Model	Random Forest Model	SVM Model
• Accuracy: 0.63 • Precision: 0.62 • Recall: 0.63 • F1 Score: 0.60 • AUC: 0.756943	• Accuracy: 0.78 • Precision: 0.78 • Recall: 0.78 • F1 Score: 0.78 • AUC: 0.8821221	• Accuracy: 0.65 • Precision: 0.65 • Recall: 0.65 • F1 Score: 0.65 • AUC: N/A



Based on the performance data and ROC curves of the three ML models we used, we observe that:

The Logistic Regression model has a moderate performance with an accuracy of 63%, precision of 62%, recall of 63%, F1 score of 60%, and an AUC of 0.76. The ROC curves show its ability to distinguish between classes, particularly class 0 with an AUC of 0.85 and class 1 with an AUC of 0.80, although performance drops for class 2 with an AUC of 0.70.

The Random Forest Model shows a good performance, with accuracy, precision, recall, and F1 score all at 78%, and an overall AUC of 0.88. The ROC curve indicates strong prediction ability across classes, with high AUCs for each class. This model represents a significant improvement over Logistic Regression in all areas.

The Support Vector Machine (SVM) showed a slightly better performance than Logistic Regression, with all metrics—accuracy, precision, recall, and F1 score—at 65%. The ROC curve shows that the Random Forest model has the highest curve positions across all classes, suggesting the best performance among the three models in distinguishing between different credit score classes.

4.4 Model Refinement with Feature Engineering:

4.4.1 Feature Engineering:

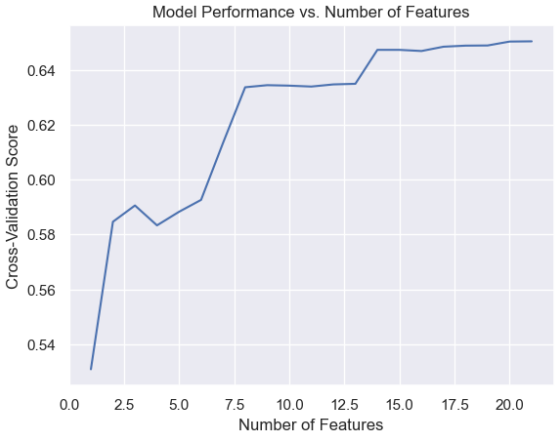
A set of feature selection techniques, including K-Best, Univariate Selection, and Lasso Regression is used to improve predictions.

The K-Best method allowed us to identify the top k features based on statistical tests, while univariate selection highlighted features based on their relationships with the target variable.

Via Lasso Regression, we shrank less important features to zero, ensuring only the most relevant features remained.

The plot on the right demonstrates that $k = 8$ provided the optimal balance, significantly improving model performance.

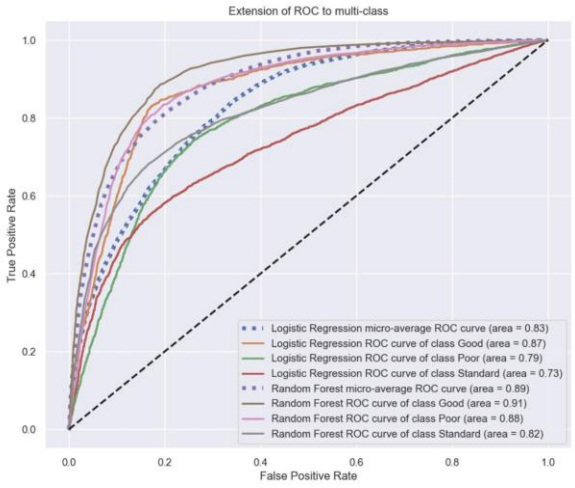
The final selected features include Changed Credit Limit, Outstanding Debt, Amount Invested Monthly, and others, all of which serve as key predictors.



4.4.2 Refined Model Performance

To improve the prediction accuracy of our models, we used the selected features from the feature selection process, and observed the following results:

Logistic Regression Model	Random Forest Model	SVM Model
• Accuracy: 0.63 • Precision: 0.63 • Recall: 0.63 • F1 Score: 0.63 • AUC: 0.77125823	• Accuracy: 0.73 • Precision: 0.73 • Recall: 0.73 • F1 Score: 0.73 • AUC: 0.8509849	• Accuracy: 0.64 • Precision: 0.65 • Recall: 0.64 • F1 Score: 0.64 • AUC: N/A



After engineering the features, **Logistic Regression** achieved an average ROC AUC of 0.83, while **Random Forest** had an AUC of 0.89. However, **the overall accuracy** of 63% revealed inconsistencies across different classes. Also, we observed that with the feature selection, the model accuracy, precision, recall, F1, and AUC all decreased compared to not adding the feature selection, which may suggest the inherent importance of some features in enhancing model accuracy, despite their insignificant appearance. If these features contain crucial information for the model, their removal can lead to a loss of predictive power.

4.5 Model Refinement with Stacking Models

4.5.1 Stacking Model

Next, we improved the predictions using the stacking model and obtained the following results:

```
Combined Model (Stacking) Evaluation:
              precision    recall  f1-score   support

        Good       0.66      0.58      0.61      1356
        Poor       0.76      0.69      0.72      2873
    Standard       0.73      0.80      0.76      4708

    accuracy                           0.73      8937
   macro avg       0.72      0.69      0.70      8937
weighted avg       0.73      0.73      0.73      8937

Accuracy: 0.7294
```

Stacking combines the strengths of multiple models to form a unified, more accurate meta-model. In particular, the combined model achieved an accuracy of 72.94%, a significant improvement over the baseline models. The team believes that the use of the stacking model will provide the following benefits:

1. **Robustness:** By leveraging multiple models, the stacking model mitigates errors and improves predictive accuracy.
2. **Generalization:** The diversity of base models allows the ensemble to generalize well across varied data distributions.
3. **Applicability:** Stacking models can be applied to diverse problems and leverage multiple model architectures to improve performance.

The above performance result clearly shows how the stacking model outperforms individual models, with improved accuracy and recall across the *"Good," "Poor,"* and *"Standard"* classes.

4.5.2 K-fold Cross Validation of Stacking Models

To further verify the accuracy of the stacking models, we next performed a K-fold cross-validation of the stacking models and obtained the following results:

```
Stacking Model (k-Fold Cross-Validation) Evaluation:
Accuracy Scores for each fold: [0.72496363 0.7288496  0.72549239 0.73724261 0.73030439]
Mean Accuracy: 0.7294
Standard Deviation: 0.0044
```

In summary, our approach to feature engineering ensured that the model received the most relevant information. The stacking model refined our predictions by utilizing diverse base models, resulting in an overall improvement in performance. With an accuracy of 72.94%, the final ensemble model outperformed its baseline counterparts significantly.

5. User Testing

After the completion of model performance optimization and verification, the team decided to create a user-friendly interface to collect standardized inputs and generate meaningful predictions for users. The team then decided to obtain the following 9 features from the user input and use them as the model parameters:

Numerical Features
1. Number of Credit Card
2. Number of Credit Inquiries
3. Number of Bank Accounts
4. Outstanding Debts
5. Number of Delayed Payment
6. Change in Credit Limit
7. Monthly Investment

Categorical Features
1. Credit Mix (Good, Poor, Standard)
2. Payment of Minimum Amount

Since some categorical data types cannot be understood by the pre-trained models directly, we applied a column transforming method to parse these categorical data types. This information will then be stored into a "preprocessor" model for later application on the predictive pretrained models. We utilized the following three libraries from SKLearn to achieve this goal:

1. **StandardScaler**
   This function will standardize all the numerical features by scaling the mean to 0 and variance to 1, which increases the predictive accuracy of the models.
2. **OneHotEncoder**
   Like its name suggests, this function will convert the input categorical data into numerical values for the models to understand. We also ignore any anomalous categorical inputs to prevent unexpected behaviors of the models.
3. **ColumnTransformer**
   This function will transform the columns based on the two functions previously specified.

After fitting the *"preprocessor"* model, we imported the pre-trained models, including:
a. Logistic Regression Model.
b. Random Forest Model.
c. SVM Model.
d. CatBoost Model

After acquiring the inputs from the user, we parsed them into the predefined 9 features and made predictions based on these inputs. The following section demonstrates 3 different scenarios when testing the CTB model.

5.1 Good Credit Mix

User Input
Number of Credit Cards: **1**
Number of Credit Inquiries in the last six months: **0**
Number of Bank Accounts: **1**
Outstanding Debt: **0**
Number of Delayed Payments: **0**
Changes in Credit Limit: **0**
Amount Invested Monthly: **10000**
Credit Mix (1 for Good, 2 for Bad, 3 for Unknown): **1**
Payment of Min. Amount (Yes/No): **No**

Model Predictions
Logistic Regression Prediction: ['**Good**']
Random Forest Prediction: ['**Good**']
SVM Prediction: ['**Standard**']
CatBoost Prediction: **Standard**

As shown above, when a relatively good credit mix scenario is presented to the pre-trained models, the yielded output is 50% *"Good"* and 50% *"Standard"*.

5.2 Poor Credit Mix

User Input
Number of Credit Cards: **5**
Number of Credit Inquiries in the last six months: **15**
Number of Bank Accounts: **5**
Outstanding Debt: **30000**
Number of Delayed Payments: **20**
Changes in Credit Limit: **0**
Amount Invested Monthly: **0**
Credit Mix (1 for Good, 2 for Bad, 3 for Unknown): **2**
Payment of Min Amount (Yes/No): **No**

Model Predictions
Logistic Regression Prediction: ['**Poor**']
Random Forest Prediction: ['**Standard**']
SVM Prediction: ['**Poor**']
CatBoost Prediction: **Standard**

As shown above, when a relatively poor credit mix scenario is presented to the pre-trained models, the yielded output is 50% *"Poor"* and 50% *"Standard*

5.3 Extreme High Input

User Input
Number of Credit Cards: **2134**
Number of Credit Inquiries in the last six months: **2134**
Number of Bank Accounts: **2134**
Outstanding Debt: **1234**
Number of Delayed Payments: **1234**
Changes in Credit Limit: **1234**
Amount Invested Monthly: **1234**
Credit Mix (1 for Good, 2 for Bad, 3 for Unknown): **1**
Payment of Min Amount (Yes/No): **No**

Model Predictions
Logistic Regression Prediction:[**Poor**]
Random Forest Prediction: ['**Standard**']
SVM Prediction: [**Poor**]
CatBoost Prediction: ['**Standard**']

Based on the observations, the models provided the same prediction as the *"Poor Credit Mix"* test scenario. This occurrence is possibly due to the significant impact of certain features, such as Number of Credit Inquiries, Outstanding Debt, and Number of Delayed Payments. In both *"Poor Credit Mix"* and *"Extreme High Input"* scenarios, these features all had large values, which resulted in the same output.

6. Potential Impact

From the client's perspective, this project offers a dual benefit: understanding how to manage and improve credit scores and assessing their likelihood of credit card approval in advance. With a final prediction accuracy of 72.94%, we are moderately confident that our project will offer a decent tool for people to assess their credit performance without negatively impacting their credit scores. We believe that this tool will change how people make decisions because with the information this credit score predictor provides them, they will be able to get a clear idea of their credit stance and make more informed financial decisions.

7. Future Improvements

The team identified three aspects to enhance the CTB project as a practical tool for users:

1. **Better Prediction Accuracy:** By incorporating more comprehensive and sophisticated models, we will be able to further improve the model to above 90% accuracy.
2. **LLM User Input Parsing:** The utilization of LLM will be able to provide a better user experience instead of a series of questionnaire. The LLM can also parse unexpected inputs and provide a prediction based on the context.
3. **Graphical User Dashboard:** Currently, the model outputs can only provide limited information to the user. A graphical output will further improve user interpretability.

8. Conclusion

In conclusion, this project has successfully developed a robust credit score prediction model that can empower individuals to better manage their financial standing. By exploring various machine learning approaches and refining the model through feature engineering, the team was able to achieve a prediction accuracy of **72.94%** using a stacking ensemble model. This represents a significant improvement over the baseline models, showcasing the power of leveraging diverse algorithms to enhance predictive performance.

The key benefits of this project are multifaceted. First and foremost, it provides users with valuable insights into their current credit profile and the likelihood of credit card approval. This empowers them to make more informed financial decisions without the risk of negatively impacting their credit scores through unnecessary hard inquiries. By understanding their creditworthiness, users can strategize ways to improve their standing, such as paying down outstanding debts, increasing credit limits, or addressing any discrepancies in their credit history.

Finally, this project serves as a testament to the team's ability to apply the concepts learned in ORIE 4741/5741 to a real-world challenge in the financial services industry. By delivering a user-friendly, data-driven solution, the team has demonstrated the potential for innovative fintech applications to transform the way individuals navigate the credit landscape.

9. GitHub Repository

https://github.com/SimonSaysGiveMeSmile/Caffeinated-Treasury-Bot-ORIE-4741-

10. Team Contribution and Course Code

Simon Tian	Kayle Yang	Cecilia Yang
ORIE 5741	ORIE 4741	ORIE 4741
Data Cleaning Data Analysis User Interface Model Debugging Presentation Report Write-up	Baseline ML Models Feature Engineering Model Refinements Presentation Report Write-up	Data Cleaning Data Analysis Model Evaluations Code Consolidations Presentation Report Write-up