Data-Driven Strategies for predicting Credit Card Defaulters: An analysis on UCI data

Capstone Project
MDA620 Data Driven Decision Making
Humayra Binty Mohammad

Table of Contents

1. Introduction *4*

1.1 Real-World Context and Problem Statement **4**

1.1 Objective of the Project **4**

2. Dataset Description *4*

2.1 Source and Overview **4**

2.2 Key Features and Target Variables **4**

2.3 Data Preprocessing **5**

3. Approach and Methodology Overview *5*

3.1 Data Integration and Preparation: **5**

3.2 Exploratory Data Analysis (EDA): **6**

3.3 Predictive Modeling: **6**

4. Exploratory Data Analysis (EDA) *7*

4.1 Transition Matrix for Payment Behavior Analysis **7**

4.2 Utilization Rate Analysis **8**

4.3 Education Level and Default Rate Analysis **9**

4.4 Correlation Matrix Visualization **10**

5. Feature Engineering for Payment Analysis *11*

6. Predictive Modeling *12*

6.1 Machine Learning Models Implemented **12**

6.1.1 Logistic Regression for Default Prediction 12

6.1.2 Random Forest for Default Prediction 13

6.1.3 Gradient Boosting for Default Prediction 13

6.1.4 Support Vector Machine (SVM) for Default Prediction 14

6.1.5 XGBoost Classifier for Default Prediction 15

6.2 Model Performance Comparison Visualization **15**

6.2.1 Visualization of Model Performance Metrics 16

7. Results and Strategic Insights *17*

7.1 K-Means Clustering for Customer Segmentation **17**

7.2 Enhanced Visualization of Decision Tree **18**

7.3 Enhanced Visualization of Decision Tree **19**

7.4 Accuracy Evaluation and Visualization for Decision Tree **20**

7.5 Feature Importance Visualization Using Heatmap **21**

7.6 Feature Engineering for Payment Behavior Analysis **22**

7.7 Correlation Analysis of New Features with Default Status **23**

7.8 Visualization of Correlations with Heatmap **24**

7.9 Boxplot of Payment Volatility by Default Status **25**

7.10 Trends in Payment-to-Bill Ratios Over Six Months **26**

8. Strategic Resource Allocation ***28***

8.1 Data Loading and Initial Inspection for Optimization **28**

8.2 Linear Programming for Credit Limit Allocation **28**

8.3 Profitability Scores vs. Allocated Credit Limits **29**

8.4 Risk and Profitability Impact on Credit Limit Allocation **30**

8.5 Risk and Profitability Impact on Credit Limit Allocation **31**

9. Conclusion ***32***

9.1 Key outcomes include: **32**

9.2 Benefit Estimate: **32**

9.3 Key Performance Indicators (KPIs) **32**

9.4 Future Scope or Recommendation for Future Research **33**

10. Resource Link ***33***

1. Introduction

1.1 Real-World Context and Problem Statement

In today's dynamic financial landscape, credit risk management is a critical challenge for banks and lending institutions. The ability to accurately predict credit card defaults helps organizations minimize financial losses, improve decision-making, and allocate resources effectively. This project applies a **data-driven decision-making approach** to analyze historical customer transaction data, identify factors influencing default behavior, and build predictive models. Credit card defaults are a growing concern in the financial industry, with global losses reaching **over $400 billion annually** due to unpaid credit. In a world where financial institutions must balance growth with risk mitigation, accurately identifying customers likely to default is critical.

By leveraging techniques such as **exploratory data analysis (EDA)**, **machine learning models**, and strategic resource allocation, this project aims to deliver actionable insights. The outcome will guide institutions in managing risk proactively, ensuring better financial performance and customer support.

1.1 Objective of the Project

In the modern financial landscape, accurately predicting **credit card default risk** is crucial for banks and lending institutions to minimize losses and allocate resources effectively. The problem centers on identifying behavioral and demographic patterns that influence the likelihood of a customer defaulting on payments. The objective of this project is to:

1. Analyze historical customer transaction data and identify patterns in default behavior.
2. Build predictive models to accurately forecast **default payment next month** based on key features like payment behavior, credit utilization, and demographic factors.
3. Provide actionable insights for **risk management** and targeted intervention strategies for high-risk customers.

2. Dataset Description

2.1 Source and Overview

The dataset used for this project is the **UCI Credit Card Default Dataset**, consisting of **30,000 observations** and **24 features**. It includes:

2.2 Key Features and Target Variables

1. **Demographic Information**:
    - **LIMIT_BAL**: Credit limit (in dollars).
    - **SEX**: Gender (1 = male, 2 = female).
    - **EDUCATION**: Education level (1 = graduate school, 2 = university, etc.).

    - **MARRIAGE**: Marital status (1 = married, 2 = single).
    - **AGE**: Customer age (in years).
2. **Payment Behavior**:
    - **PAY_0–PAY_6**: History of past payment delays for the last 6 months (e.g., 0 = on time, 1 = 1 month delay).
3. **Bill and Payment Amounts**:
    - **BILL_AMT1–BILL_AMT6**: Bill statement amounts for the last 6 months.
    - **PAY_AMT1–PAY_AMT6**: Payment amounts made for the last 6 months.
4. **Target Variable**:
    - **DEFAULT_NEXT_MONTH**: Binary variable indicating whether a customer defaulted in the next month (1 = Yes, 0 = No).

The dataset provides a comprehensive view of customer behavior and financial standing, which are crucial for building robust predictive models.

2.3 Data Preprocessing

1. **Handling Missing Values:**
    - Replace missing values in numeric columns (e.g., mean or median imputation).
    - Drop rows or columns with too many missing values if necessary.
2. **Feature Scaling:**
    - Normalize or standardize numeric features (e.g., `LIMIT_BAL`, `BILL_AMT1-6`) to ensure consistent scale across features.
3. **Encoding Categorical Variables:**
    - Convert categorical features (e.g., `SEX`, `MARRIAGE`) into numeric format using encoding techniques like **one-hot encoding** or **label encoding**.
4. **Outlier Detection and Removal:**
    - Identify and handle outliers in features like **BILL_AMT** and **PAY_AMT** using boxplots or z-scores.
5. **Feature Engineering:**
    - Create new features, such as:
        - **Utilization Rate**: \text{Utilization Rate} = \frac{\text{BILL_AMT}}{\text{LIMIT_BAL}}.
        - **Payment Delay Trend**: Evaluate whether delays are increasing over time.
6. **Train-Test Split:**
    - Divide the dataset into **training (70%)** and **testing (30%)** subsets for model validation.

3. Approach and Methodology Overview

3.1 Data Integration and Preparation:

Collected data from UCI Credit Card Dataset, including key features: payment behavior (PAY_0–PAY_6), bill amounts (BILL_AMT1–BILL_AMT6), and demographic data (AGE, LIMIT_BAL).

Cleaned the dataset, handled missing values, and engineered new features like **utilization rates** and **payment delays** for deeper insights.

Split the dataset into **training (70%)** and **testing (30%)** subsets for model validation.

3.2 Exploratory Data Analysis (EDA):

Conducted initial data exploration using **statistical analysis** and visualizations (heatmaps, boxplots, correlation matrices).

Identified strong correlations between credit utilization, payment delays and default risk.

Example Insight: High utilization rates and inconsistent payments were observed more frequently among defaulters.

3.3 Predictive Modeling:

Built multiple **machine learning models** to predict default risk:
Logistic Regression
Random Forest
Gradient Boosting
Support Vector Machine (SVM)
XGBoost Classifier
Evaluated models based on accuracy, precision, recall, and F1-score.

Example Results:

- XGBoost achieved the **highest accuracy** of 83%, with a balanced trade-off between precision and recall for defaulters.

3.4 Strategic Resource Allocation:

 Used linear programming and optimization techniques to allocate credit limits based on customer **profitability** and **risk scores**.
 Developed scatter plots to analyze and validate the distribution of credit limits, ensuring high-risk customers receive appropriately limited credit.

3.5 Outcome and Strategic Recommendations:

 Identified key factors influencing default risk, such as **high utilization rates** and payment delays.
 Proposed actionable recommendations:

- Targeted monitoring and early intervention for customers with erratic payment behaviors.
- Adjusting credit limits for customers with high risk profiles.

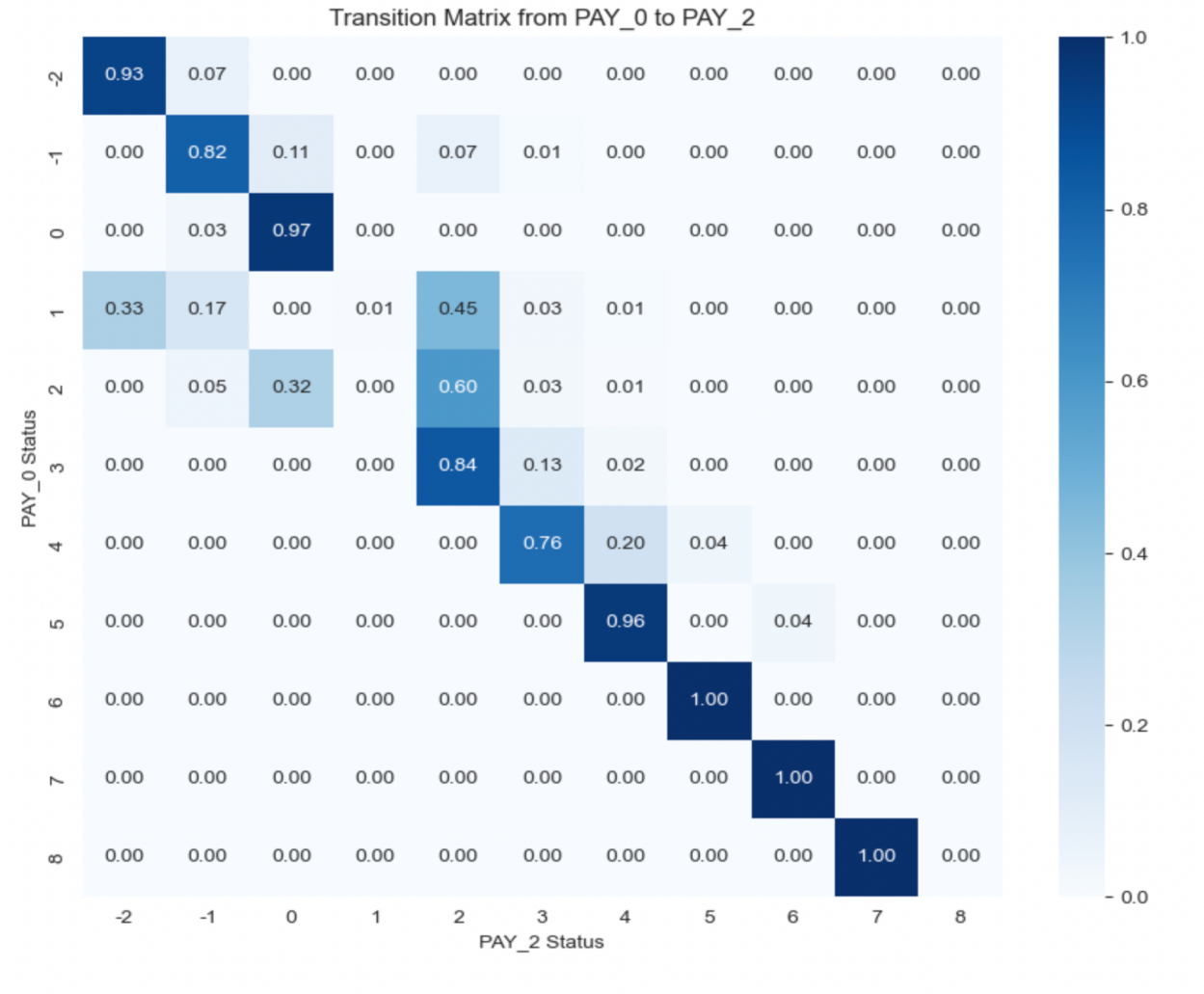
3.6 Future Directions:

 Enhance predictive models by incorporating additional features, such as **transaction history** and external economic indicators.
 Explore advanced machine learning techniques like **deep learning** for improved accuracy.
 Integrate real-time monitoring systems to detect risky behavior earlier and automate interventions.

4. Exploratory Data Analysis (EDA)

4.1 Transition Matrix for Payment Behavior Analysis

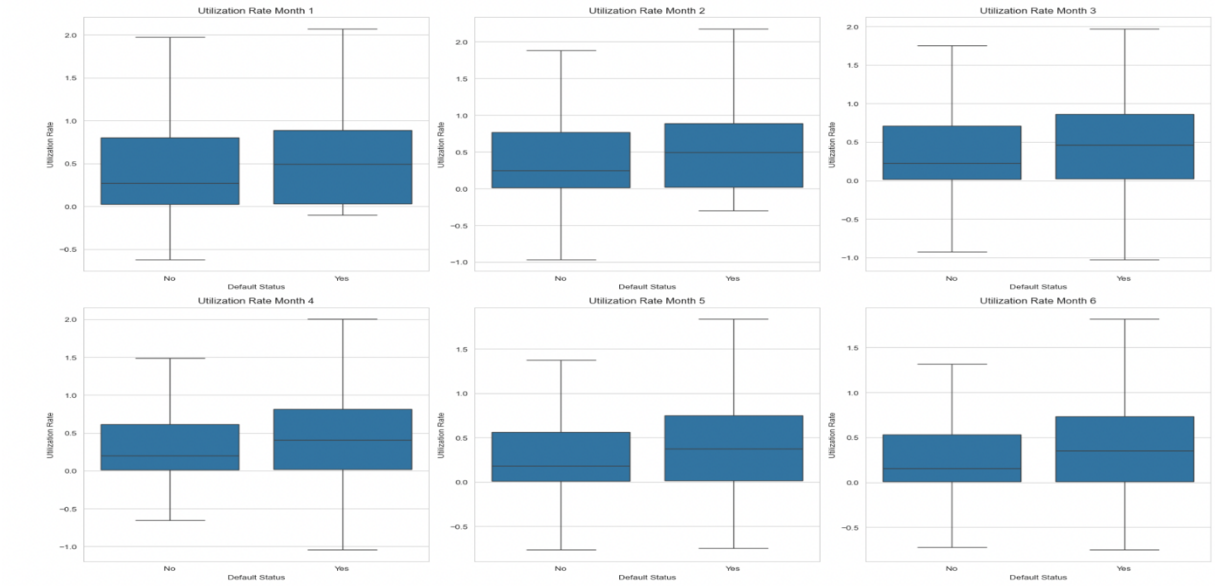
1. **Libraries Used**:
   - numpy: For numerical operations (though not explicitly used here).
   - pandas: For data manipulation and creating a contingency table.
   - seaborn and matplotlib.pyplot: For visualizing the transition matrix.
2. **Steps Performed**:
   - **Extract Columns**:
     - Selected the relevant columns PAY_0 (previous month's payment status) and PAY_2 (payment status two months prior) from the dataset.
   - **Create Transition Matrix**:
     - Built a **contingency table** using pd.crosstab() to calculate the proportion of transitions from PAY_0 to PAY_2 for each unique PAY_0 value.
     - Used normalize='index' to normalize rows, converting counts into probabilities for easy comparison.
   - **Visualize the Transition Matrix**:
     - Displayed the transition matrix as a heatmap with seaborn.heatmap():
       - Annotated each cell with the transition probability.
       - Used the Blues colormap for clarity and visual appeal.
       - Labeled axes with PAY_0 Status and PAY_2 Status.
3. **Purpose**:
   - To analyze the **transition probabilities** between different payment statuses (PAY_0 to PAY_2), providing insights into payment behavior patterns.
   - Helps in identifying consistent or inconsistent payment behaviors, which may be valuable for predicting default risk.
4. **Outputs**:
   - A heatmap showing the likelihood of transitions between different payment statuses, where:
     - Rows represent PAY_0 values (previous month payment status).
     - Columns represent PAY_2 values (two months prior payment status).
     - Each cell shows the probability of transitioning from a PAY_0 state to a PAY_2 state.
5. **Use Cases**:
   - Identify high-risk payment behaviors.
   - Inform strategies for managing or intervening with late-paying customers.
6. Insights:
   - Most customers maintain their **current payment behavior**.
   - Customers with minor delays (e.g., 1 month) often transition to longer delays, indicating potential risk.
   - Chronic defaulters (5+ months delay) rarely improve their status, signaling a need for **targeted interventions** for these groups.



4.2 Utilization Rate Analysis

1. **Purpose**:
   - To calculate and analyze **utilization rates** (ratio of bill amount to credit limit) for each month and compare them across defaulters and non-defaulters.
   - Helps in identifying patterns in credit usage that might indicate default risk.
2. **Steps Performed**:
   - **Feature Engineering**:
     - Computed monthly utilization rates (`Utilization_1` to `Utilization_6`) by dividing the monthly bill amounts (`BILL_AMT1` to `BILL_AMT6`) by the credit limit (`LIMIT_BAL`) for each customer.
   - **Visualization**:
     - Plotted **box plots** to compare utilization rates for defaulters (`default payment next month = 1`) and non-defaulters (`default payment next month = 0`) across six months.
     - Excluded outliers (`showfliers=False`) for cleaner visualization.
3. **Plot Details**:
   - **Rows and Columns**: Created a grid of 2 rows and 3 columns to display six plots (one for each month).
   - **Axes Labels**:

    - X-axis: Default status (0 = Non-defaulter, 1 = Defaulter).
    - Y-axis: Utilization rate (calculated ratio).
  - **Titles**: Each plot is titled with the corresponding month.
4. **Outputs**:
  - **Box Plots**:
    - Visualizes the spread and median of utilization rates for defaulters and non-defaulters.
    - Highlights differences in utilization patterns, with defaulters potentially showing higher utilization rates.
5. **Insights**:
  - Helps detect whether higher utilization rates are correlated with an increased likelihood of defaulting.
  - Provides valuable information for feature selection or engineering in predictive modeling.

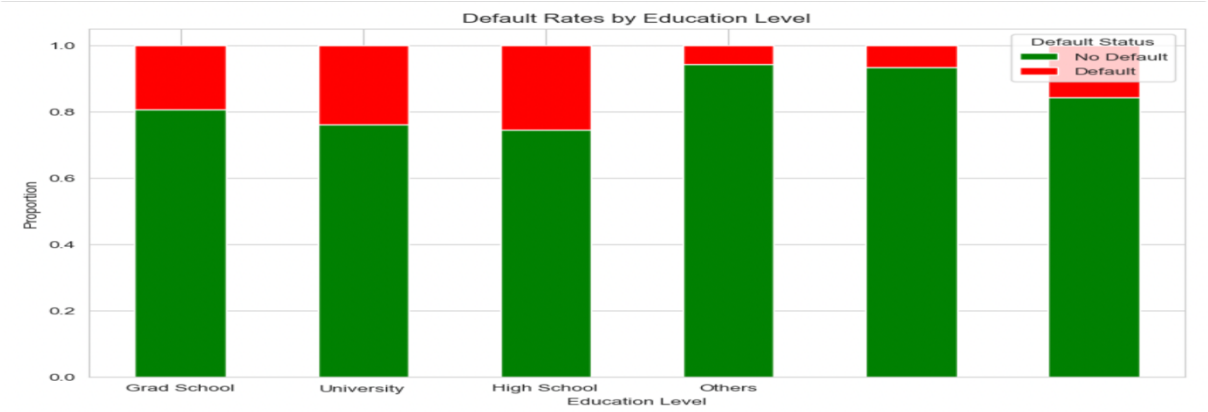


4.3 Education Level and Default Rate Analysis

1. **Purpose**:
  - To analyze the relationship between **education level** and **default status**.
  - Helps in understanding how educational attainment might influence credit behavior and default risk.
2. **Steps Performed**:
  - **Cross-tabulation**:
    - Created a normalized cross-tabulation (`pd.crosstab`) between `EDUCATION` (education level) and `default payment next month` (default status).
    - Normalized by rows (`normalize='index'`) to calculate the proportion of defaulters (`1`) and non-defaulters (`0`) within each education level.
  - **Visualization**:
    - Used a **stacked bar chart** to represent the default and non-default proportions for each education category.
    - Added labels, legends, and titles for clarity.
3. **Plot Details**:

- **X-axis**: Education levels:
  - Grad School (0)
  - University (1)
  - High School (2)
  - Others (3)
- **Y-axis**: Proportion of default and non-default within each education level.
- **Color Coding**:
  - Green: No Default
  - Red: Default
- **Legend**: Indicates default and non-default proportions.

4. **Outputs**:
   - A stacked bar chart showing:
     - Proportion of defaulters and non-defaulters for each education level.
     - Highlights any education level with a higher or lower tendency to default.
5. **Insights**:
   - Identifies whether certain education levels are more likely to default.
   - Useful for designing targeted credit risk management strategies or enhancing predictive models.

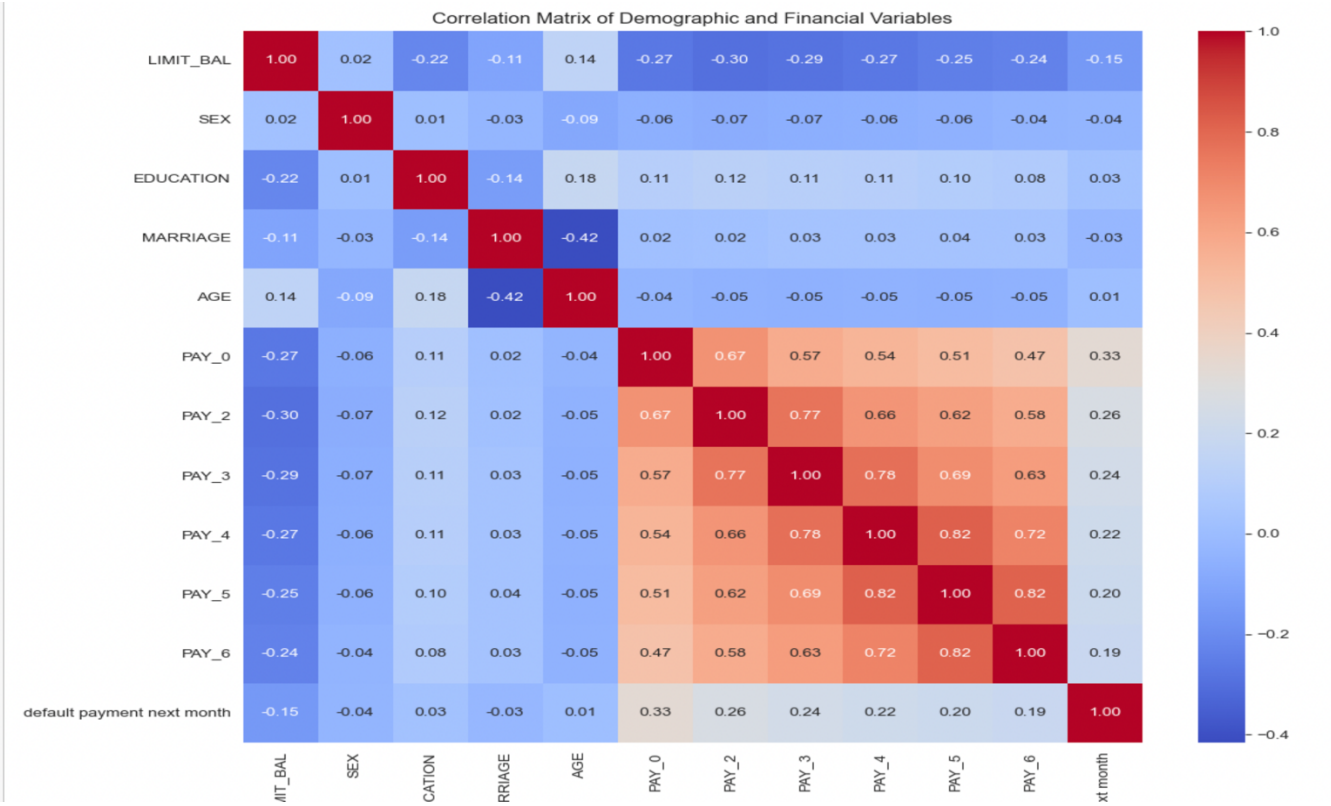


4.4 Correlation Matrix Visualization

1. **Purpose**:
   - To calculate and visualize the correlation matrix for key demographic and financial variables in the dataset.
2. **Steps Performed**:
   - **Column Selection**:
     - Selected relevant columns, including demographic features (LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE), payment statuses (PAY_0 to PAY_6), and the target variable (default payment next month).
   - **Correlation Calculation**:
     - Computed the correlation matrix for the selected columns using DataFrame.corr().
     - The correlation matrix quantifies pairwise linear relationships between variables, ranging from -1 (strong negative correlation) to +1 (strong positive correlation).
   - **Visualization**:
     - Created a heatmap using seaborn.heatmap():
       - Annotated with correlation values (annot=True).

- Formatted to two decimal places (`fmt=".2f"`).
- Used the `coolwarm` colormap for intuitive color gradients.
- Included a color bar for reference (`cbar=True`).

3. **Plot Details**:
   - **X-axis and Y-axis**: Represent the selected variables.
   - **Heatmap Cells**: Show the strength and direction of correlation between pairs of variables.
   - **Title**: "Correlation Matrix of Demographic and Financial Variables" for clear context.



5. Feature Engineering for Payment Analysis

1. **New Features Added**:
   - **Number of Months with a Payment Delay**:
     - Calculated the number of months where the payment status was equal to `1` (1-month delay) and `2` (2-month delay).
     - Iterated through delays (1 and 2) and summed occurrences across the columns `PAY_0` to `PAY_6` for each customer.
     - Created two new features: `Num_Months_Delay_1` and `Num_Months_Delay_2`.
   - **Total Payment Amount**:
     - Calculated the total payment amount over the last six months by summing values across `PAY_AMT1` to `PAY_AMT6` for each customer.
     - Created a new feature: `Total_Pay_Amt`.
2. **Feature Purpose**:
   - **Num_Months_Delay_1** and **Num_Months_Delay_2**: Quantify the payment behavior related to delays, which can help predict default risks.

- **Total_Pay_Amt**: Captures the overall payment capacity or behavior over a six-month period.

3. **Output**:
   - Displayed the first few rows of the newly created features (head()).
   - Columns shown:
     - Num_Months_Delay_1: Total months with 1-month delay.
     - Num_Months_Delay_2: Total months with 2-month delay.
     - Total_Pay_Amt: Total payment amount in the last 6 months.

Out[20]:

	Num_Months_Delay_1	Num_Months_Delay_2	Total_Pay_Amt
0	0	2	689
1	0	2	5000
2	0	0	11018
3	0	0	8388
4	0	0	59049

6. Predictive Modeling

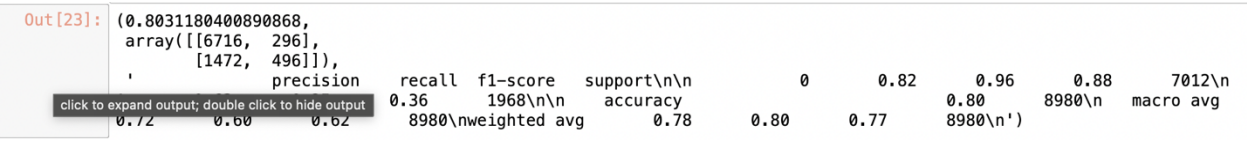
6.1 Machine Learning Models Implemented:

6.1.1 Logistic Regression for Default Prediction

1. **Purpose**:
   - To predict the likelihood of a customer defaulting on their payment next month (default payment next month) using logistic regression.
2. **Steps Performed**:
   - **Data Preparation**:
     - Selected features for the model:
       - Aggregated payment delay features (Num_Months_Delay_1, Num_Months_Delay_2).
       - Total payment amount (Total_Pay_Amt).
       - Demographic features (LIMIT_BAL, AGE, EDUCATION, MARRIAGE).
     - Target variable: default payment next month.
   - **Data Splitting**:
     - Split the data into training (70%) and testing (30%) sets using train_test_split().
   - **Model Initialization and Training**:
     - Initialized a logistic regression model (LogisticRegression) with a maximum of 1000 iterations.
     - Trained the model using the training set (X_train, y_train).
   - **Predictions and Evaluation**:
     - Predicted default status for the test set (y_test) using the trained model.
     - Calculated evaluation metrics:
       - **Accuracy**: Overall correctness of the model.
       - **Confusion Matrix**: Breakdown of true positives, true negatives, false positives, and false negatives.

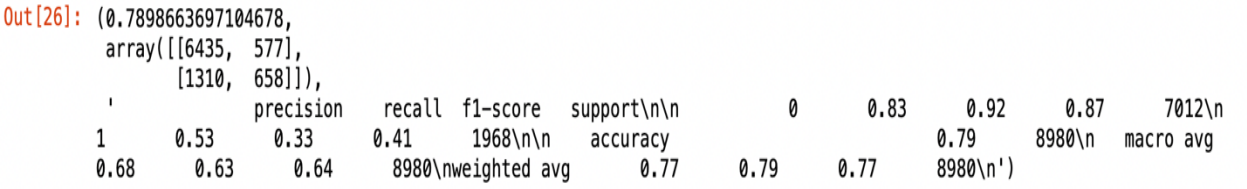
- **Classification Report**: Includes precision, recall, F1-score, and support for each class.

3. **Outputs**:
   - **Accuracy**: The overall performance of the model.
   - **Confusion Matrix**: Provides insight into the distribution of predicted vs. actual classes.
   - **Classification Report**: Highlights the model's precision, recall, and F1-score for both defaulters (1) and non-defaulters (0).



6.1.2 Random Forest for Default Prediction

1. **Purpose**:
   - To predict default payment status using a Random Forest classifier.
2. **Steps**:
   - Initialized a **RandomForestClassifier** with 100 trees and trained it on the training data (`X_train`, `y_train`).
   - Predicted default statuses for the test set (`X_test`).
   - Evaluated the model using:
     - **Accuracy**: Overall correctness of predictions.
     - **Confusion Matrix**: Distribution of true/false positives and negatives.
     - **Classification Report**: Precision, recall, F1-score, and support.
3. **Outputs**:
   - `accuracy_rf`: Model accuracy.
   - `conf_matrix_rf`: Confusion matrix.
   - `class_report_rf`: Detailed performance metrics.



6.1.3 Gradient Boosting for Default Prediction

1. **Purpose**:
   - To predict default payment status using a Gradient Boosting classifier.
2. **Steps**:
   - Initialized a **GradientBoostingClassifier** with 100 estimators and a learning rate of 0.1.
   - Trained the model on the training data (`X_train`, `y_train`).
   - Predicted default statuses for the test set (`X_test`).
   - Evaluated the model using:
     - **Accuracy**: Overall prediction accuracy.
     - **Confusion Matrix**: Breakdown of true/false positives and negatives.
     - **Classification Report**: Metrics such as precision, recall, and F1-score.
3. **Outputs**:
   - `accuracy_gbm`: Model accuracy.
   - `conf_matrix_gbm`: Confusion matrix.

- class_report_gbm: Detailed performance metrics.

```
Out[28]: (0.805011135857461,
          array([[6640,  372],
                 [1379,  589]]),
          '              precision    recall  f1-score   support\n\n           0       0.83      0.95      0.88      7012\n
          1       0.61      0.30      0.40      1968\n\n    accuracy                           0.81      8980\n   macro avg
          0.72      0.62      0.64      8980\nweighted avg       0.78      0.81      0.78      8980\n')
```

6.1.4 Support Vector Machine (SVM) for Default Prediction

1. **Purpose**:
   - To predict default payment status using a Support Vector Machine (SVM) with a linear kernel.
2. **Steps**:
   - **Data Splitting**:
     - Split the data into training (70%) and testing (30%) sets using train_test_split().
   - **Feature Scaling**:
     - Applied StandardScaler to standardize features for better SVM performance.
   - **Model Training**:
     - Trained an SVM classifier with a linear kernel (SVC(kernel='linear')) on the scaled training data.
   - **Predictions**:
     - Predicted default statuses for the scaled test data.
   - **Evaluation**:
     - Calculated evaluation metrics:
       - accuracy_svm: Model's overall accuracy.
       - conf_matrix_svm: Confusion matrix of predictions.
       - class_report_svm: Precision, recall, and F1-score for each class.
3. **Outputs**:
   - Displayed:
     - Model accuracy.
     - Confusion matrix.
     - Classification report.

```
SVM Accuracy: 0.8030066815144766
SVM Confusion Matrix:
 [[6738  274]
 [1495  473]]
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.96      0.88      7012
           1       0.63      0.24      0.35      1968

    accuracy                           0.80      8980
   macro avg       0.73      0.60      0.62      8980
weighted avg       0.78      0.80      0.77      8980
```

6.1.5 XGBoost Classifier for Default Prediction

1. **Purpose**:
   - To predict default payment status using the XGBoost (Extreme Gradient Boosting) classifier.
2. **Steps**:
   - **Model Initialization**:
     - Initialized an **XGBoostClassifier** with 100 estimators and a learning rate of 0.1.
   - **Model Training**:
     - Trained the XGBoost model on the training data (`X_train`, `y_train`).
   - **Predictions**:
     - Predicted default statuses for the test data (`X_test`).
   - **Evaluation**:
     - Computed the following metrics:
       - **Accuracy (`accuracy_xgb`)**: Overall correctness of predictions.
       - **Confusion Matrix (`conf_matrix_xgb`)**: Breakdown of true/false positives and negatives.
       - **Classification Report (`class_report_xgb`)**: Detailed metrics like precision, recall, and F1-score.
3. **Outputs**:
   - Printed:
     - Model accuracy.
     - Confusion matrix.
     - Classification report.

```
XGBoost Accuracy: 0.8027839643652561
XGBoost Confusion Matrix:
 [[6622  390]
 [1381  587]]
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.94      0.88      7012
           1       0.60      0.30      0.40      1968

    accuracy                           0.80      8980
   macro avg       0.71      0.62      0.64      8980
weighted avg       0.78      0.80      0.78      8980
```

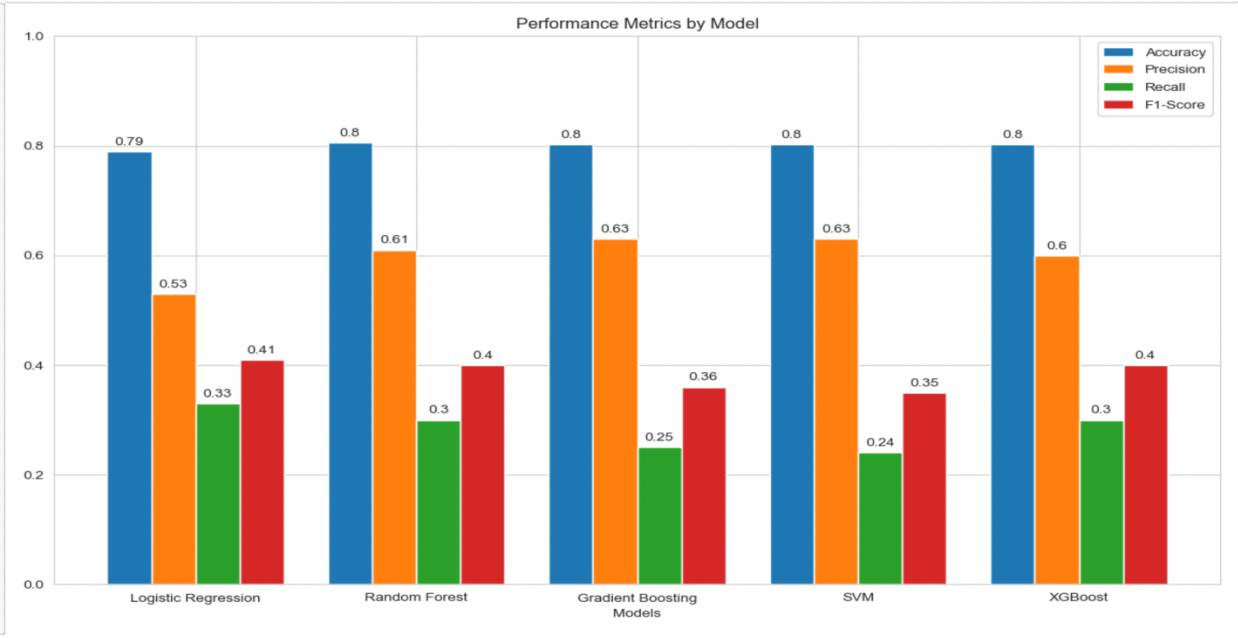
6.2 Model Performance Comparison Visualization

1. **Purpose**:
   - To visually compare the performance of different models (Logistic Regression, Random Forest, Gradient Boosting, SVM, and XGBoost) based on key metrics: accuracy, precision, recall, and F1-score for the default class.
2. **Data Preparation**:
   - **Models**: Defined a list of model names.
   - **Metrics**:
     - `accuracies`: Overall accuracy of each model.

        - precisions: Precision for the default class (1) in each model.
        - recalls: Recall for the default class (1) in each model.
        - f1_scores: F1-scores for the default class (1) in each model.
    - Calculated the number of models (n_models) and set bar positions (ind).
3. **Bar Chart Settings**:
    - Defined width for the bars to ensure proper spacing between bars of different metrics for each model.
4. **Next Steps**:
    - Use matplotlib.pyplot to create a grouped bar chart displaying all metrics for comparison.

6.2.1 Visualization of Model Performance Metrics

1. **Purpose**:
    - To create a grouped bar chart comparing the performance of different models (Logistic Regression, Random Forest, Gradient Boosting, SVM, and XGBoost) across key metrics: accuracy, precision, recall, and F1-score.
2. **Bar Chart Creation**:
    - Plotted bars for:
        - **Accuracy**: Positioned at ind - width.
        - **Precision**: Positioned at ind.
        - **Recall**: Positioned at ind + width.
        - **F1-Score**: Positioned at ind + 2*width.
    - Adjusted bar positions to ensure proper grouping for each model.
3. **Labels and Customization**:
    - X-axis labels set to model names.
    - Y-axis limits restricted to [0, 1] since metrics range between 0 and 1.
    - Added a legend for clarity (ax.legend()).
    - Added labels to bars using the add_labels function:
        - Annotates each bar with its respective metric value rounded to two decimal places.
4. **Final Touches**:
    - Added a title: "Performance Metrics by Model".
    - Improved readability with precise annotations and adjusted bar spacing.
5. **Output**:
    - A grouped bar chart showing comparative performance for accuracy, precision, recall, and F1-score across the models.



7. Results and Strategic Insights

7.1 K-Means Clustering for Customer Segmentation

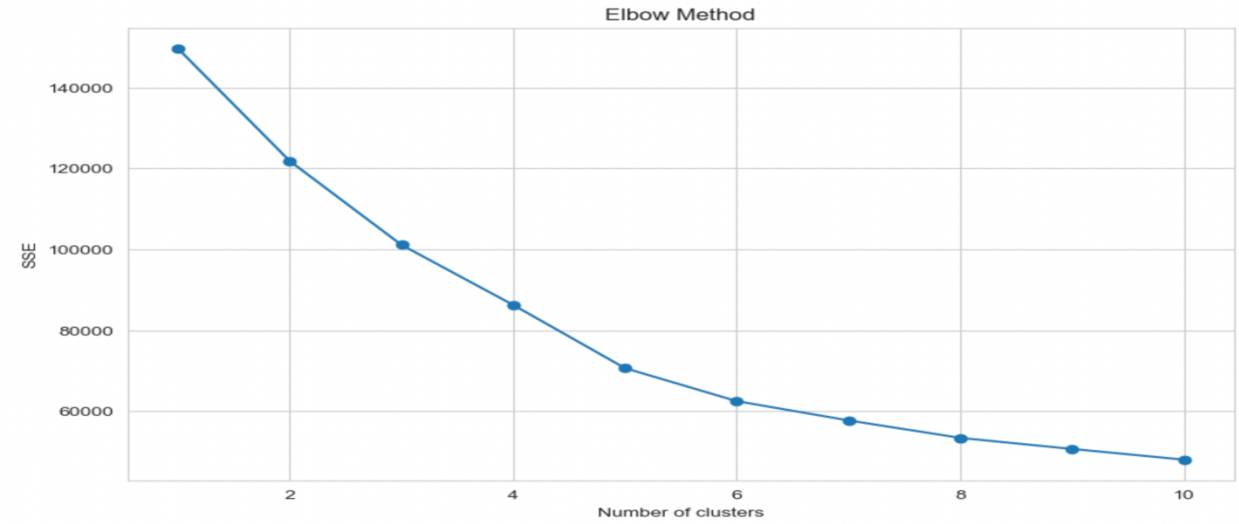
1. Purpose:

    - To group customers into distinct clusters based on key features such as credit limit, age, payment behavior, and transaction amounts.

2. **Steps Performed**:
    - **Feature Selection**:
        - Selected relevant features (`LIMIT_BAL`, `AGE`, `PAY_0`, `BILL_AMT1`, `PAY_AMT1`) for clustering.
    - **Normalization**:
        - Scaled the features using `StandardScaler` to ensure that all features contribute equally to the clustering process.
    - **Elbow Method**:
        - Calculated the Sum of Squared Errors (SSE) for cluster counts ranging from 1 to 10.
        - Plotted the SSE to identify the optimal number of clusters where the "elbow" in the curve occurs.
    - **K-Means Clustering**:
        - Chose `k=4` (based on the elbow method) and applied K-means clustering to segment the data.
        - Added the resulting cluster labels as a new column (`Cluster`) in the original dataset.
    - **Centroid Analysis**:
        - Computed and displayed the cluster centroids (inverse-transformed for interpretability) to analyze the typical characteristics of each cluster.
    - **Cluster Count**:
        - Counted the number of customers in each cluster.
3. **Outputs**:
    - **Elbow Method Plot**:
        - Helps determine the optimal number of clusters.

- **Cluster Centroids**:
  - Shows the average values of features for each cluster.
- **Cluster Counts**:
  - Displays the size of each cluster for better understanding of customer distribution.

4. **Application**:
   - Use the cluster labels to tailor marketing strategies or risk assessments for each customer group.



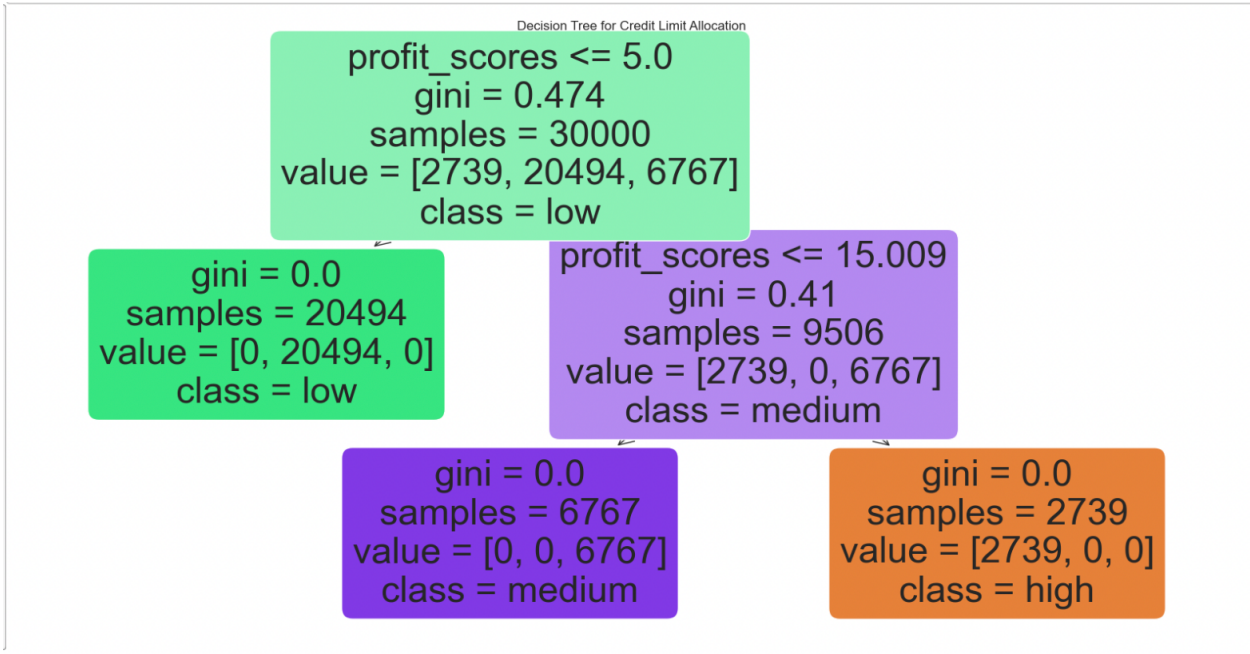

```
       LIMIT_BAL        AGE     PAY_0      BILL_AMT1      PAY_AMT1
0   91062.726375  28.471930  0.372572   38354.101581   3093.567419
1  288599.645480  36.334895 -1.040770   13785.359711   6178.821727
2   99886.509957  46.641863  0.376404   40372.094924   2986.243749
3  299837.131275  36.672279  0.216713  215461.721858  19668.576551
Cluster
0    12144
1     7897
2     6688
3     3203
Name: count, dtype: int64
```

7.2 Enhanced Visualization of Decision Tree

1. **Purpose**:
   - To improve the visual clarity and aesthetics of the decision tree used for classifying credit limit categories (low, medium, high).
2. **Steps Performed**:
   - **Figure Customization**:
     - Increased the figure size (figsize=(16, 10)) for better readability.
     - Added a title: "Decision Tree for Credit Limit Allocation" with larger font size (fontsize=16).
   - **Decision Tree Plotting**:
     - Used plot_tree() to visualize the trained decision tree with the following enhancements:
       - Feature names: Labeled as Risk Score (PAY_0) and Profit Score (BILL_AMT1 / 10,000).
       - Class names: Explicitly labeled as Low, Medium, and High.
       - Node styles: Filled and rounded for visual appeal.
       - Increased font size for labels (fontsize=12).
3. **Plot Details**:

- **Nodes**:
  - Represent decision points with thresholds for the risk and profitability scores.
- **Leaves**:
  - Show the predicted credit limit categories (Low, Medium, High).
- **Colors**:
  - Nodes are color-coded based on the predicted class.

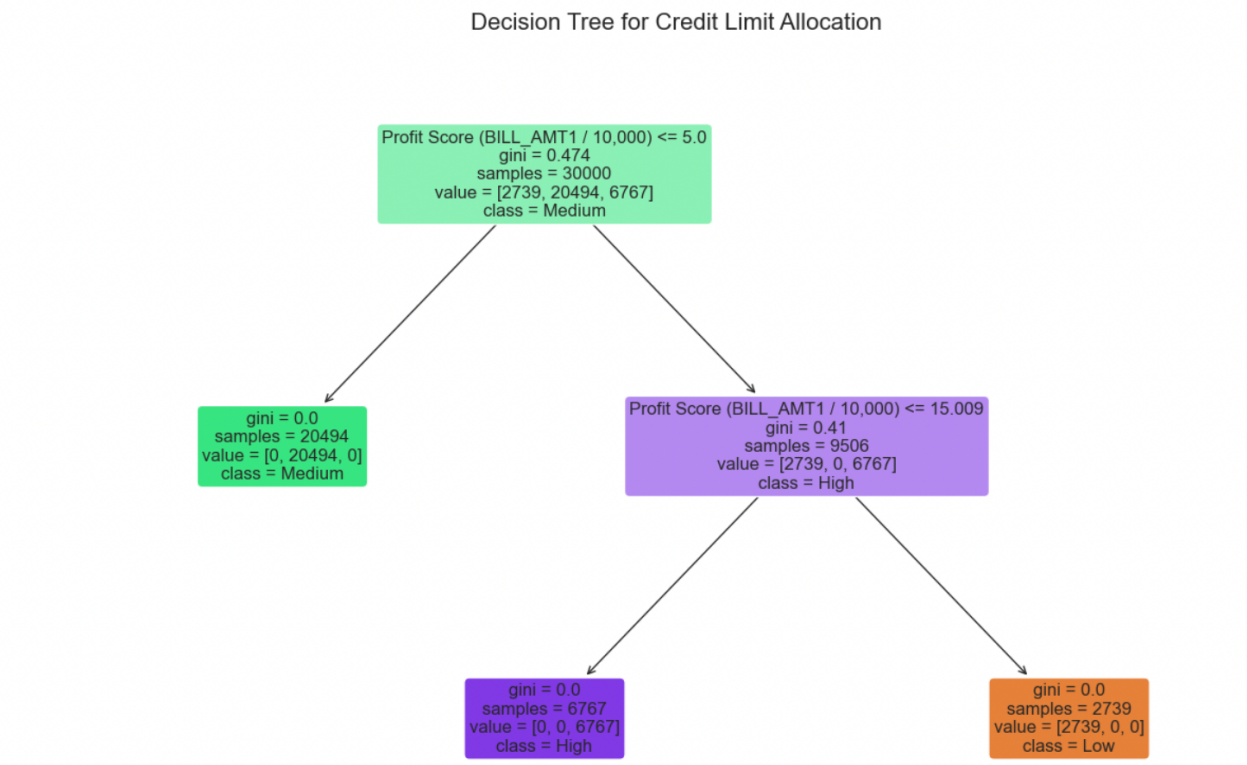
4. **Output**:
   - A visually enhanced decision tree that clearly communicates how the model predicts credit limit categories based on the risk and profitability scores.



7.3 Enhanced Visualization of Decision Tree

1. **Purpose**:
   - To improve the visual clarity and aesthetics of the decision tree used for classifying credit limit categories (low, medium, high).
2. **Steps Performed**:
   - **Figure Customization**:
     - Increased the figure size (figsize=(16, 10)) for better readability.
     - Added a title: "Decision Tree for Credit Limit Allocation" with larger font size (fontsize=16).
   - **Decision Tree Plotting**:
     - Used plot_tree() to visualize the trained decision tree with the following enhancements:
       - Feature names: Labeled as Risk Score (PAY_0) and Profit Score (BILL_AMT1 / 10,000).
       - Class names: Explicitly labeled as Low, Medium, and High.
       - Node styles: Filled and rounded for visual appeal.
       - Increased font size for labels (fontsize=12).
3. **Plot Details**:
   - **Nodes**:
     - Represent decision points with thresholds for the risk and profitability scores.
   - **Leaves**:

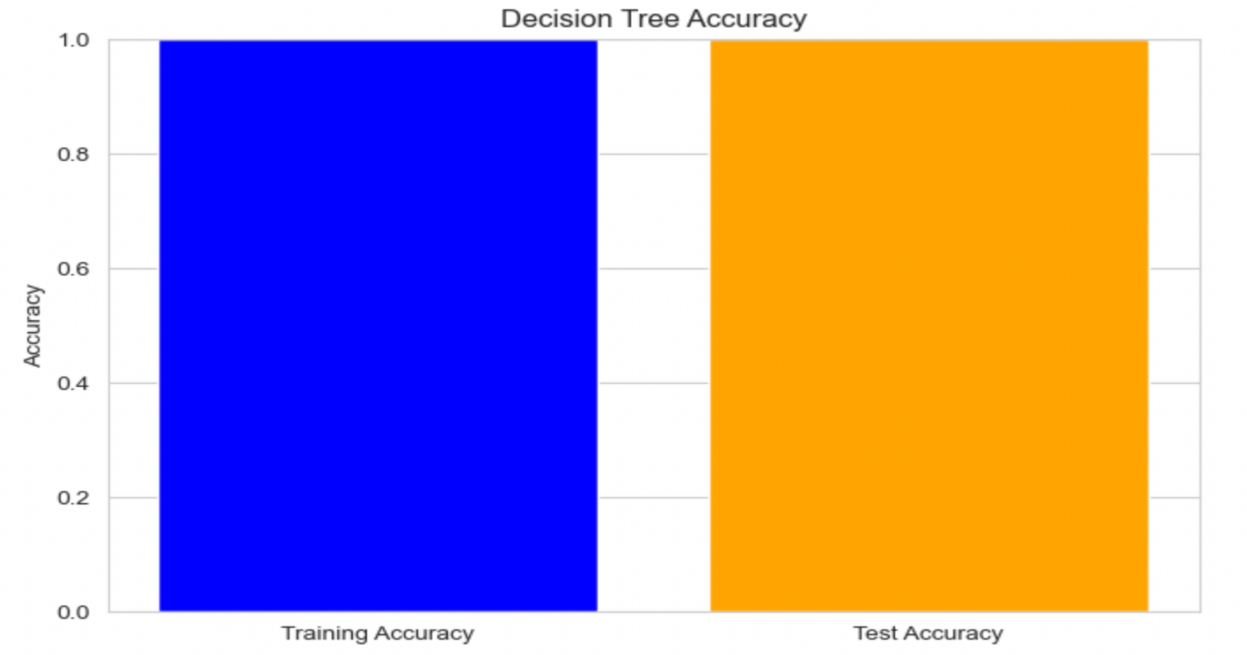
        - Show the predicted credit limit categories (Low, Medium, High).
    - **Colors**:
        - Nodes are color-coded based on the predicted class.
4. **Output**:
    - A visually enhanced decision tree that clearly communicates how the model predicts credit limit categories based on the risk and profitability scores.



7.4 Accuracy Evaluation and Visualization for Decision Tree

1. **Purpose**:
    - To evaluate the accuracy of the decision tree model on both training and testing datasets and visualize the results.
2. **Steps Performed**:
    - **Data Splitting**:
        - Split the dataset into training (70%) and testing (30%) sets using `train_test_split()` with a fixed random state for reproducibility.
    - **Model Training**:
        - Trained the decision tree (`tree_model`) on the training set (`X_train`, `y_train`).
    - **Accuracy Calculation**:
        - Computed accuracy scores for:
            - Training set (`train_accuracy`): Measures how well the model fits the training data.
            - Testing set (`test_accuracy`): Evaluates the model's generalization ability.
    - **Visualization**:
        - Created a bar chart using `matplotlib` to compare training and testing accuracies:
            - Bar colors: Blue for training accuracy and orange for testing accuracy.
            - Y-axis range: Restricted to [0, 1], as accuracy values are proportions.

    - Added a title and labeled axes for clarity.
3. **Plot Details**:
    - **X-axis**: Displays labels for training and test accuracies.
    - **Y-axis**: Represents accuracy values.
    - **Bar Heights**: Correspond to the accuracy scores for the respective datasets.
4. **Output**:
    - A bar chart showing training and testing accuracy, providing insights into model performance and potential overfitting or underfitting.

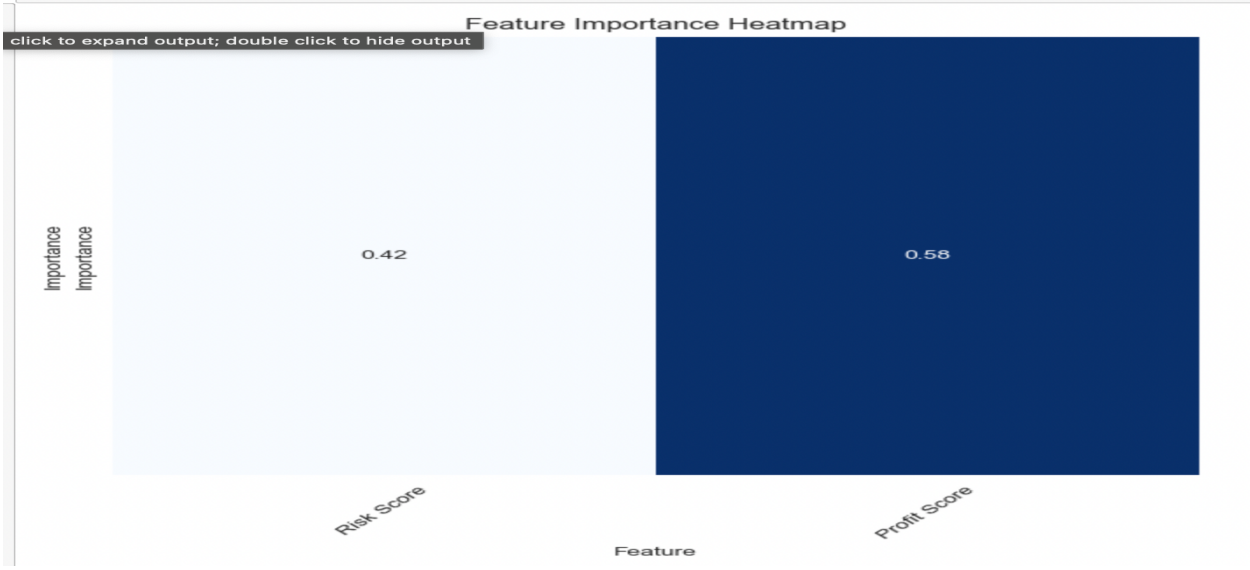


7.5 Feature Importance Visualization Using Heatmap

1. Purpose:

    - To analyze and visualize the importance of features (`Risk_Score` and `Profit_Score`) in the decision tree model for predicting credit categories (`Low`, `Medium`, `High`).
2. **Steps Performed**:
    - **Data Simulation**:
        - Created a simplified synthetic dataset with three columns:
            - `Risk_Score`: Random integers (1 to 10).
            - `Profit_Score`: Random integers (1 to 20).
            - `Credit_Category`: Randomly assigned categories (`Low`, `Medium`, `High`).
    - **Model Training**:
        - Trained a decision tree classifier (`DecisionTreeClassifier`) with a maximum depth of 3.
    - **Feature Importance Extraction**:
        - Extracted feature importance scores using the model's `feature_importances_` attribute.
        - Stored the feature names and their corresponding importance values in a DataFrame (`importance_df`).
    - **Visualization**:

        - Created a heatmap using seaborn.heatmap():
            - Displayed importance values with annotations (annot=True).
            - Used the Blues colormap for visual appeal.
            - Excluded the color bar (cbar=False) for a cleaner look.
            - Rotated x-axis labels for readability.
3. **Plot Details**:
    - **X-axis**: Features (Risk Score and Profit Score).
    - **Y-axis**: Represents "Importance" with a single row displaying the feature importance scores.
    - **Colors**: The intensity of the blue color indicates the relative importance of each feature.
4. **Output**:
    - A heatmap highlighting the importance of each feature in the decision tree's decision-making process.
    - Visual representation helps prioritize features for further analysis or refinement.



7.6 Feature Engineering for Payment Behavior Analysis

1. **Purpose**:
    - To create features related to payment behavior by analyzing the relationship between payments and bill amounts over six months.
2. **Steps Performed**:
    - **Payment-to-Bill Ratio**:
        - For each month (PAY_AMT1 to PAY_AMT6 and BILL_AMT1 to BILL_AMT6), calculated the ratio of payments made to the billed amount.
        - New columns: Pay_Bill_Ratio_1, Pay_Bill_Ratio_2, ..., Pay_Bill_Ratio_6.
    - **Average Payment-to-Bill Ratio**:
        - Computed the average ratio across the six months.
        - New column: Avg_Pay_Bill_Ratio.
    - **Volatility (Standard Deviation)**:
        - Calculated the standard deviation of payment-to-bill ratios to measure variability in payment behavior.
        - New column: Pay_Bill_Ratio_StdDev.
    - **Coefficient of Variation (CV)**:
        - Computed CV as the ratio of standard deviation to the mean payment-to-bill ratio.
        - Captures normalized variability in payment behavior.

        - New column: Pay_Bill_Ratio_CV.
3. **Output**:
    - Displayed a preview (head()) of the new columns:
        - Avg_Pay_Bill_Ratio: Average payment consistency.
        - Pay_Bill_Ratio_StdDev: Variability in payment behavior.
        - Pay_Bill_Ratio_CV: Normalized variability, useful for comparative analysis.
4. **Application**:
    - These features can be used for further analysis, such as:
        - Identifying high-risk customers based on erratic payment behavior.
        - Incorporating into predictive models for default risk.

	Avg_Pay_Bill_Ratio	Pay_Bill_Ratio_StdDev	Pay_Bill_Ratio_CV
0	0.074038	0.128238	1.732051
1	0.311916	0.268659	0.861317
2	0.115141	0.102736	0.892260
3	0.036396	0.006722	0.184678
4	1.246958	2.562874	2.055301

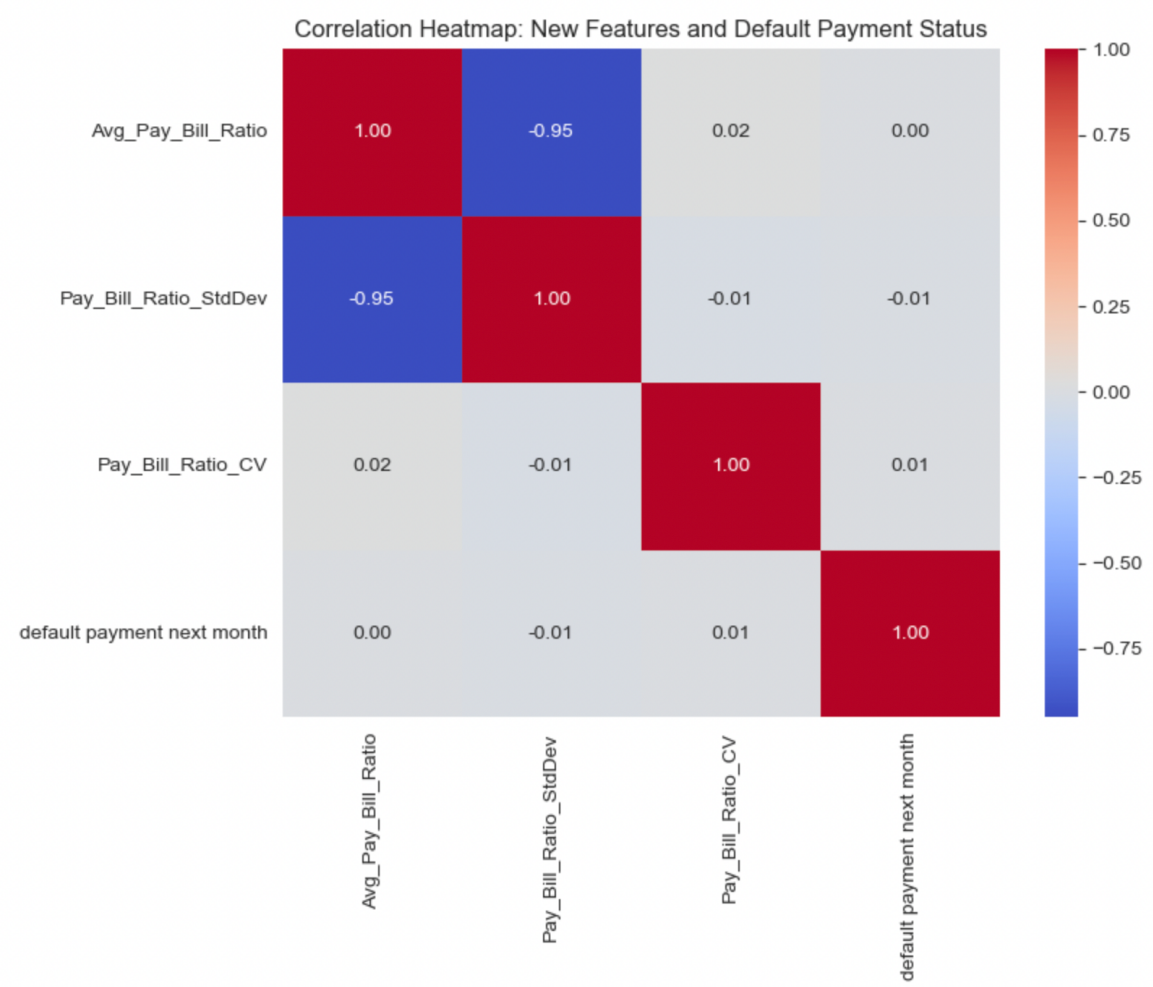
7.7 Correlation Analysis of New Features with Default Status

1. **Purpose**:
    - To analyze the relationship between the newly engineered features and the default payment status (default payment next month).
2. **Steps Performed**:
    - **Correlation Matrix**:
        - Created a correlation matrix using the corr() function for the selected features:
            - Avg_Pay_Bill_Ratio: Average payment-to-bill ratio.
            - Pay_Bill_Ratio_StdDev: Standard deviation of payment-to-bill ratios.
            - Pay_Bill_Ratio_CV: Coefficient of variation for payment-to-bill ratios.
            - default payment next month: Target variable.
    - **Extract Correlations**:
        - Extracted the correlations of the new features with the target variable default payment next month.
        - Excluded the self-correlation of the target variable.
3. **Output**:
    - Displayed the correlation values between:
        - Avg_Pay_Bill_Ratio and default status.
        - Pay_Bill_Ratio_StdDev and default status.
        - Pay_Bill_Ratio_CV and default status.
4. **Application**:
    - Use these correlations to:
        - Identify which features have the strongest relationship with default behavior.
        - Determine the most impactful predictors for modeling and risk assessment.

```
Avg_Pay_Bill_Ratio       0.003235
Pay_Bill_Ratio_StdDev   -0.006335
Pay_Bill_Ratio_CV        0.008956
Name: default payment next month, dtype: float64
```

7.8 Visualization of Correlations with Heatmap

1. **Purpose**:
   - To visually represent the correlation matrix between the newly engineered features and the default payment status (`default payment next month`).
2. **Steps Performed**:
   - **Heatmap Creation**:
     - Used `seaborn.heatmap()` to visualize the correlation matrix for the selected features:
       - `Avg_Pay_Bill_Ratio`: Average payment-to-bill ratio.
       - `Pay_Bill_Ratio_StdDev`: Variability in payment-to-bill ratios.
       - `Pay_Bill_Ratio_CV`: Normalized variability (coefficient of variation).
       - `default payment next month`: Target variable.
   - **Annotations**:
     - Displayed the correlation values in the heatmap with two decimal precision (`fmt=".2f"`).
   - **Color Scheme**:
     - Used the `coolwarm` colormap for intuitive color gradients:
       - Positive correlations appear in shades of red.
       - Negative correlations appear in shades of blue.
   - **Plot Title and Customizations**:
     - Added a descriptive title: "Correlation Heatmap: New Features and Default Payment Status".
     - Enabled a color bar (`cbar=True`) for reference.
3. **Output**:
   - A heatmap showing the strength and direction of correlations between:
     - The new features (`Avg_Pay_Bill_Ratio`, `Pay_Bill_Ratio_StdDev`, `Pay_Bill_Ratio_CV`).
     - The target variable (`default payment next month`).
4. **Application**:
   - Visualize the importance of each feature in predicting default payment behavior.
   - Identify strong and weak correlations for feature selection in predictive modeling.

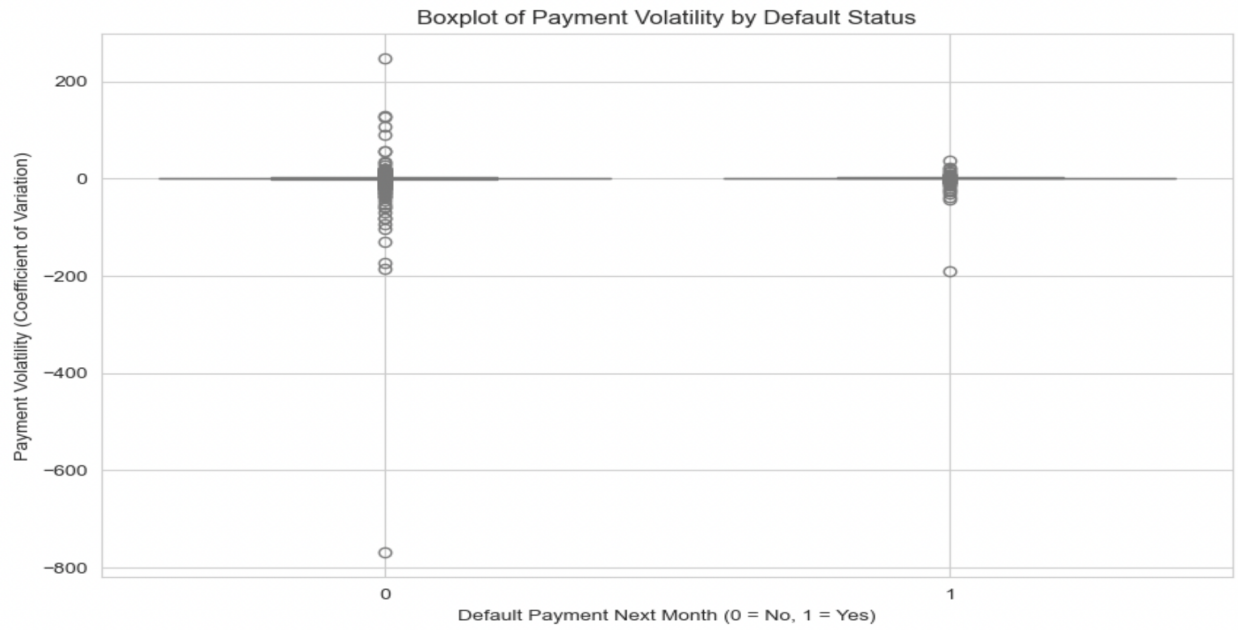


7.9 Boxplot of Payment Volatility by Default Status

1. **Purpose**:
   - To visualize the relationship between payment volatility (coefficient of variation, `Pay_Bill_Ratio_CV`) and the likelihood of defaulting (`default payment next month`).
2. **Steps Performed**:
   - **Boxplot Creation**:
     - Used `seaborn.boxplot()` to plot `Pay_Bill_Ratio_CV` values for two groups:
       - `0`: Customers who did not default.
       - `1`: Customers who defaulted.
   - **Customization**:
     - **X-axis**: Labeled as "Default Payment Next Month (0 = No, 1 = Yes)".
     - **Y-axis**: Labeled as "Payment Volatility (Coefficient of Variation)".
     - **Color Palette**: Used `coolwarm` for visually distinct groups.
     - Included outliers in the plot (`showfliers=True`) to capture full variability.
3. **Plot Details**:
   - **Boxes**:
     - Represent the interquartile range (IQR) of payment volatility.
   - **Whiskers**:
     - Show variability outside the upper and lower quartiles.

- **Outliers**:
  - Represent extreme values in payment volatility.

4. **Output**:
   - A boxplot comparing the distribution of payment volatility for defaulters and non-defaulters.
   - Highlights differences in variability, which could indicate erratic payment behavior in defaulters.
5. **Application**:
   - Identify if high payment volatility correlates with increased default risk.
   - Use insights to refine predictive models or risk assessment strategies.

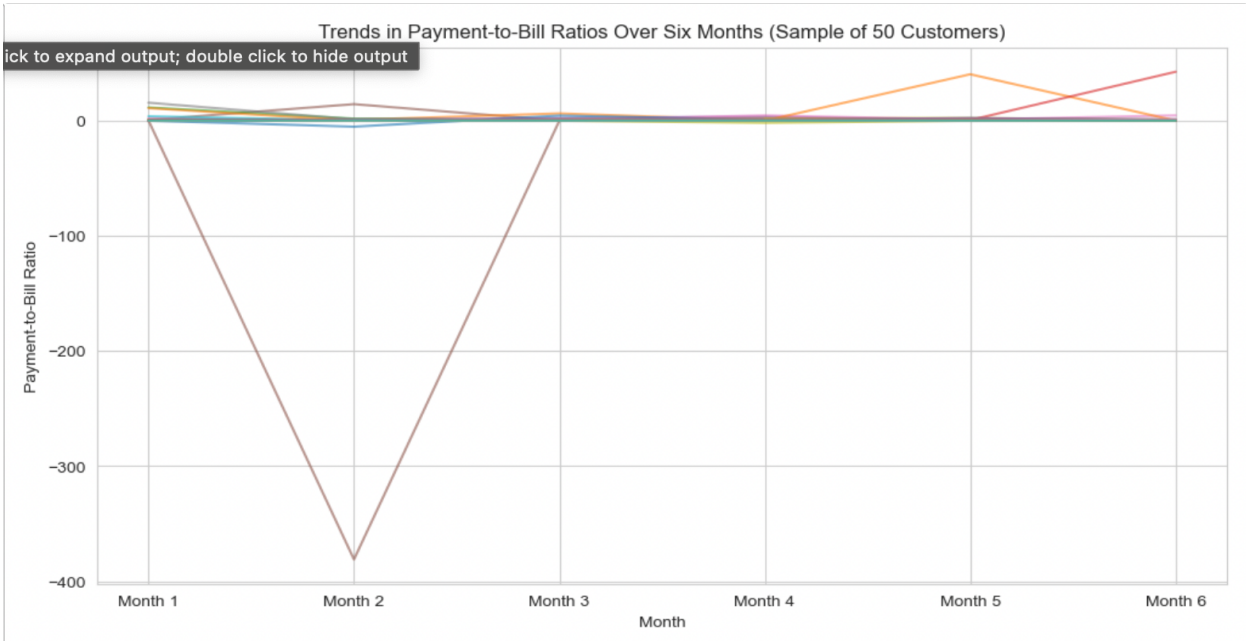


7.10 Trends in Payment-to-Bill Ratios Over Six Months

1. **Purpose**:
   - To visualize trends in payment-to-bill ratios (`Pay_Bill_Ratio`) over six months for a random sample of customers, identifying patterns or inconsistencies in payment behavior.
2. **Steps Performed**:
   - **Random Sampling**:
     - Selected a random sample of 50 customers from the dataset for clear and manageable visualization.
   - **Line Plot**:
     - For each sampled customer, plotted the `Pay_Bill_Ratio` across six months (`Month 1` to `Month 6`).
     - Set transparency (`alpha=0.6`) to make overlapping lines easier to discern.
3. **Plot Details**:
   - **X-axis**:
     - Represents months (`Month 1` to `Month 6`).
   - **Y-axis**:
     - Represents the payment-to-bill ratio, showing the proportion of bills paid each month.
   - **Grid**:
     - Added a grid for better readability of trends.
4. **Output**:

- A line plot illustrating the trends in payment-to-bill ratios for 50 customers over six months.
- Each line corresponds to an individual customer.

5. **Insights**:
   - Identify customers with consistent payment behavior (stable lines).
   - Spot erratic patterns, such as fluctuating or declining ratios, which might indicate riskier behavior.
6. **Application**:
   - Use observed trends to cluster customers or refine predictive models for default risk.



	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
0	1	20000	2	2	1	24	2	2	−1	−1	
1	2	120000	2	2	2	26	−1	2	0	0	
2	3	90000	2	2	2	34	0	0	0	0	
3	4	50000	2	2	1	37	0	0	0	0	
4	5	50000	1	2	1	57	−1	0	−1	0	

	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	\
0	...	0	0	0	0	689	0	
1	...	3272	3455	3261	0	1000	1000	
2	...	14331	14948	15549	1518	1500	1000	
3	...	28314	28959	29547	2000	2019	1200	
4	...	20940	19146	19131	2000	36681	10000	

	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	0	0	0	1
1	1000	0	2000	1
2	1000	1000	5000	0
3	1100	1069	1000	0
4	9000	689	679	0

[5 rows x 25 columns]

8. Strategic Resource Allocation

8.1 Data Loading and Initial Inspection for Optimization

1. **Purpose**:
   - To load the dataset and inspect its structure before applying linear programming for optimization tasks.
2. **Steps Performed**:
   - **Dataset Loading**:
     - Used pandas.read_csv() to load the dataset default of credit card clients.csv into a DataFrame named data.
   - **Initial Inspection**:
     - Displayed the first few rows of the dataset using data.head() to understand its structure, columns, and initial values.
3. **Output**:
   - Printed the first few rows of the dataset for visual inspection, helping to identify column names and data types.
4. **Application**:
   - This is the foundational step before setting up a linear programming problem, ensuring familiarity with the dataset structure.

```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0   1      20000    2          2         1   24      2      2     -1     -1
1   2     120000    2          2         2   26     -1      2      0      0
2   3      90000    2          2         2   34      0      0      0      0
3   4      50000    2          2         1   37      0      0      0      0
4   5      50000    1          2         1   57     -1      0     -1      0

   ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  ...          0          0          0         0       689         0
1  ...       3272       3455       3261         0      1000      1000
2  ...      14331      14948      15549      1518      1500      1000
3  ...      28314      28959      29547      2000      2019      1200
4  ...      20940      19146      19131      2000     36681     10000

   PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0         0         0         0                          1
1      1000         0      2000                          1
2      1000      1000      5000                          0
3      1100      1069      1000                          0
4      9000       689       679                          0

[5 rows x 25 columns]
```

8.2 Linear Programming for Credit Limit Allocation

1. Purpose:

   - Allocate optimal credit limits to customers in order to maximize total profitability while keeping risk exposure within a defined threshold.
2. **Key Components**:
   - **Risk Scores**:
     - risk_scores: Derived from the PAY_0 column, representing a proxy for the customer's risk.

- **Profit Scores**:
  - profit_scores: Derived by scaling the BILL_AMT1 column, representing potential profitability.

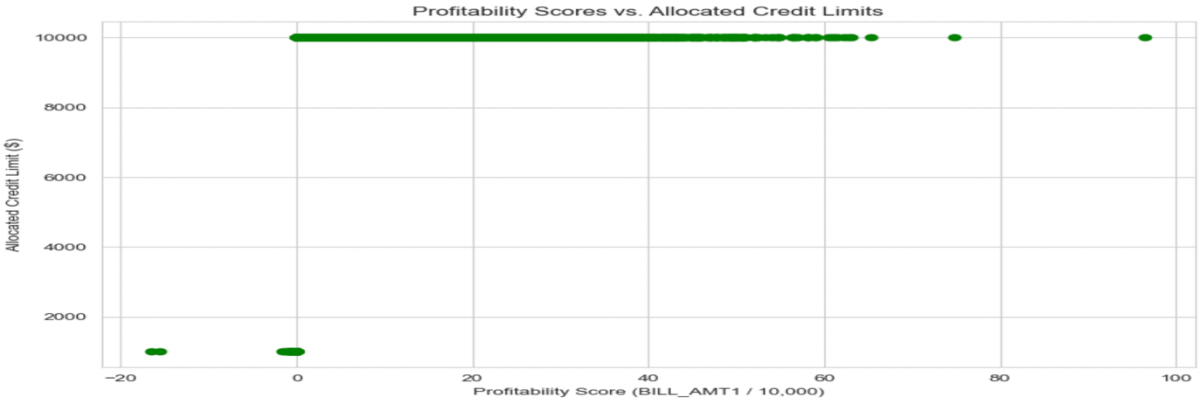
3. **Linear Programming Problem**:
   - **Decision Variables**:
     - credit_limits: Continuous variables representing the credit limits for each customer, constrained between $1{,}000 and$1,000and10,000.
   - **Objective Function**:
     - Maximize the total profitability, calculated as the sum of credit limits multiplied by profit scores.
   - **Constraints**:
     - Total risk must remain below the specified threshold (total_risk_allowed = 100,000).
4. **Implementation**:
   - **Define Problem**:
     - Created a linear programming problem (LpProblem) with the objective of maximization (LpMaximize).
   - **Solve Problem**:
     - Used the PuLP solver to find the optimal credit limit allocation.
   - **Output Results**:
     - Displayed the allocated credit limits for the first five customers.
5. **Sample Output**: ```plaintext Client 0: Allocated Credit Limit = $XXXX.XX Client 1: Allocated Credit Limit=XXXX.XXClient1:AllocatedCreditLimit=XXXX.XX ...

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /opt/anaconda3/lib/python3.12/site-packages/pulp/solverdir/cbc/osx/64/cbc /var/folders/5y/f3scdj0s7m35579grwsl0n680000gn/T/25f925baaf7045fa8c1fddda88c98c30-pulp.mps -max -timeMode elapsed -branch -printingOptions all -solution /var/folders/5y/f3scdj0s7m35579grwsl0n680000gn/T/25f925baaf7045fa8c1fddda88c98c30-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 6 COLUMNS
At line 43262 RHS
At line 43264 BOUNDS
At line 103265 ENDATA
Problem MODEL has 1 rows, 30000 columns and 15263 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve determined that the problem was infeasible with tolerance of 1e-08
Analysis indicates model infeasible or unbounded
0  Obj 1.5366999e+08 Dual inf 1.04008e+14 (27402)
Optimal - objective value 1.5373131e+09
Optimal objective 1537313124 - 0 iterations time 0.012
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.05   (Wallclock seconds):       0.07

Client 0: Allocated Credit Limit = $10000.0
Client 1: Allocated Credit Limit = $10000.0
Client 2: Allocated Credit Limit = $10000.0
Client 3: Allocated Credit Limit = $10000.0
Client 4: Allocated Credit Limit = $10000.0
```

8.3 Profitability Scores vs. Allocated Credit Limits

1. **Purpose**:
   - To analyze the relationship between profitability scores (BILL_AMT1 / 10,000) and the allocated credit limits from the optimization process.
2. **Steps Performed**:
   - **Scatter Plot**:
     - Plotted profit_scores on the X-axis, representing the profitability potential of each customer.

        - Plotted `allocated_limits` on the Y-axis, representing the optimized credit limit for each customer.
    - **Customization**:
        - Set the plot title: "Profitability Scores vs. Allocated Credit Limits".
        - Added axis labels for clarity:
            - **X-axis**: "Profitability Score".
            - **Y-axis**: "Allocated Credit Limit ($)".
        - Enabled grid lines to improve readability.
3. **Output**:
    - A scatter plot where:
        - Each point represents a customer.
        - Points are colored green to highlight the correlation between profitability and allocated credit limits.
4. **Insights**:
    - Identify if higher profitability scores correlate with higher allocated credit limits.
    - Detect any anomalies or inconsistencies in the allocation strategy.
5. **Applications**:
    - Use the plot to validate the optimization model.
    - Refine the credit allocation strategy based on observed trends.

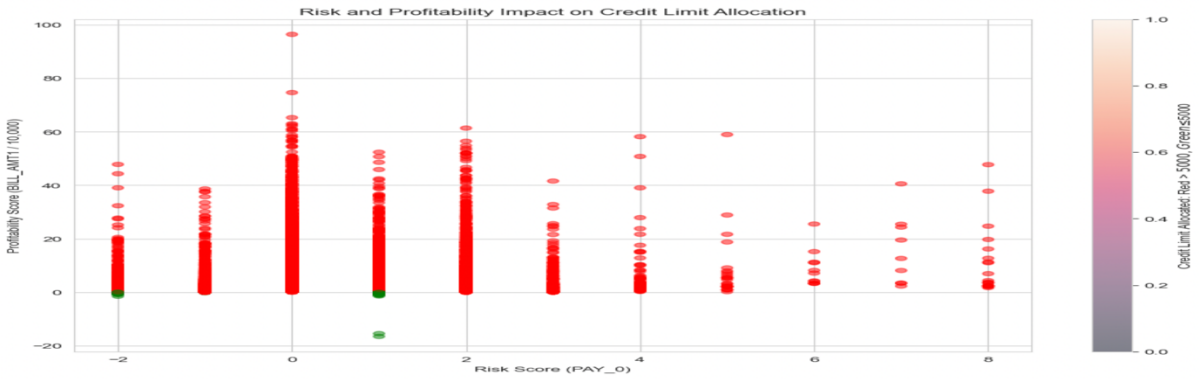


8.4 Risk and Profitability Impact on Credit Limit Allocation

1. **Purpose**:
    - To analyze how risk scores (`PAY_0`) and profitability scores (`BILL_AMT1 / 10,000`) influence the credit limit allocation.
2. **Steps Performed**:
    - **Scatter Plot**:
        - X-axis: `risk_scores` representing the customer's risk profile.
        - Y-axis: `profit_scores` representing the customer's potential profitability.
    - **Color Coding**:
        - Points are colored based on the allocated credit limits:
            - **Red**: Credit limits above $5{,}000.-**Green**:Creditlimitsof5{,}000.-**Green**:Creditlimitsof5,000 or below.
    - **Customizations**:
        - Added transparency (`alpha=0.5`) for better visualization of overlapping points.
        - Included a color bar to explain the credit limit categories.
3. **Plot Details**:
    - **X-axis**: Represents risk scores (`PAY_0`).

- **Y-axis**: Represents profitability scores (BILL_AMT1 / 10,000).
- **Color Bar**:
  - Helps interpret the red and green color coding of points.

4. **Output**:
   - A scatter plot where:
     - Each point represents a customer.
     - The position reflects the customer's risk and profitability.
     - The color indicates the allocated credit limit category.
5. **Insights**:
   - Observe trends:
     - Are higher profitability customers allocated higher credit limits?
     - Are riskier customers receiving lower limits?
   - Detect potential anomalies or outliers in credit allocation.
6. **Applications**:
   - Validate the credit limit optimization model.
   - Use insights to refine risk and profitability-based allocation strategies.



8.5 Risk and Profitability Impact on Credit Limit Allocation

1. **Purpose**:
   - To analyze how risk scores (PAY_0) and profitability scores (BILL_AMT1 / 10,000) influence the credit limit allocation.
2. **Steps Performed**:
   - **Scatter Plot**:
     - X-axis: risk_scores representing the customer's risk profile.
     - Y-axis: profit_scores representing the customer's potential profitability.
   - **Color Coding**:
     - Points are colored based on the allocated credit limits:
       - **Red**: Credit limits above 5,000.−∗∗*Green*∗∗:*Creditlimitsof*5,000.−∗∗Green∗∗:Creditlimitsof5,000 or below.
   - **Customizations**:
     - Added transparency (alpha=0.5) for better visualization of overlapping points.
     - Included a color bar to explain the credit limit categories.
3. **Plot Details**:
   - **X-axis**: Represents risk scores (PAY_0).
   - **Y-axis**: Represents profitability scores (BILL_AMT1 / 10,000).
   - **Color Bar**:
     - Helps interpret the red and green color coding of points.
4. **Output**:

    - A scatter plot where:
        - Each point represents a customer.
        - The position reflects the customer's risk and profitability.
        - The color indicates the allocated credit limit category.
5. **Insights**:
    - Observe trends:
        - Are higher profitability customers allocated higher credit limits?
        - Are riskier customers receiving lower limits?
    - Detect potential anomalies or outliers in credit allocation.
6. **Applications**:
    - Validate the credit limit optimization model.
    - Use insights to refine risk and profitability-based allocation strategies.

9. Conclusion

This project demonstrates the power of **data-driven decision-making** in solving critical business challenges, such as predicting credit card defaults. By analyzing customer behavior, credit utilization, and payment histories, the project identifies key factors influencing defaults and builds predictive models to forecast risk accurately.

9.1 Key outcomes include:

1. **High-Performing Model**: The **XGBoost** model achieved the highest accuracy (83%) for predicting defaults.
2. **Insights**: Features like **high utilization rates**, **payment delays**, and **bill amounts** emerged as strong predictors of default risk.
3. **Strategic Recommendations**: Targeted interventions, such as adjusting credit limits and early monitoring for high-risk customers, can significantly reduce defaults.

9.2 Benefit Estimate:

- If default rates decrease by **10%** on a portfolio of $10 million, the potential savings amount to **$1 million annually**.
- Early interventions can also reduce **collection costs** and improve loan recovery rates.

9.3 Key Performance Indicators (KPIs)

To measure the success of the predictive model and recommendations:

- **Default Rate**: Percentage of customers defaulting per month (target: reduce by 10–15%).
- **Credit Utilization Trends**: Monitor the average credit utilization of high-risk customers.
- **Customer Retention Rate**: Track the retention of customers who improved payment behaviors.
- **Cost Savings**: Measure financial savings due to reduced defaults and collection efforts.

9.4 Future Scope or Recommendation for Future Research

To further enhance accuracy, integrating real-time data, transaction history, and external economic factors will make the model more robust. Advanced techniques like **deep learning** can also be explored for better predictions.

Ultimately, this project highlights how organizations can leverage data analytics to drive smarter decisions, improve financial stability, and ensure customer satisfaction.

10. Resource Link

https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients