1. Introduction

This project helps in exploring how well machine-learning models can categorize binary outcomes using a real-world dataset. Instead of treating everything as a single step, I divided the workflow into three notebooks: data preparation, Logistic Regression, and Random Forest, ensuring that each phase was clean, organized, and simple to understand.

2. Data Preparation

I began by loading the raw dataset and getting familiar with it, understanding which columns were numeric, which were categorical, and where the missing values were hiding.

From there, I cleaned the data by:

- Removing columns that added no real value (irrelevant IDs).
- Imputing missing values using medians to avoid skewing the data.
- Replacing infinite values.
- Encoding categorical variables so the models could handle them.
- Binning columns that had too many unique values, which helped simplify patterns.
- Running correlation checks and removing highly correlated features to avoid redundancy.
- Finally, I saved the cleaned dataset so both models would work on the exact same version of the data.

3. Logistic Regression Modeling (HG_Project3_PartA.ipynb)

I loaded the clean dataset and created a stratified split into training, validation, and test sets. Stratification was important because I wanted to preserve the proportion of the two classes.

Since Logistic Regression is sensitive to feature scale, I standardized the data using statistics from the training set only.

Next I tuned the model using RandomizedSearchCV, focusing on regularization strength and L1 penalty. Once I found the best parameters, I trained the final model and evaluated it using accuracy, precision, recall, F1-score, and AUC. I also visualized the confusion matrix to see where the model was getting confused.

I used Logistic Regression as a baseline model because it’s easy to interpret. It tells you how the features influence the outcome and works well when the relationship between the features and target is linear. It’s also fast to train, which made hyperparameter tuning very efficient.

4. Random Forest Modeling (HG_Project3_PartB.ipynb)

Using the same cleaned dataset and the same splits, I moved on to Random Forest. I tuned hyperparameters like number of trees, max depth, and minimum samples per split using RandomizedSearchCV.

After finding the best combination, I trained the final model and evaluated it using the same set of metrics as Logistic Regression to keep the comparison fair. I also generated a feature importance plot to understand which features the model relied on most.

Random Forest is great when the underlying patterns in the data are nonlinear or complex. Unlike Logistic Regression, it does not assume linearity. It learns multiple trees and averages their predictions, which makes it powerful and stable. It also handles interactions between features naturally, which often leads to higher accuracy.

5. Model Comparison & Results

Logistic Regression

	precision	recall	f1-score	support
0	0.88	0.90	0.89	8932
1	0.87	0.84	0.85	6777
accuracy			0.87	15709
macro avg	0.87	0.87	0.87	15709
weighted avg	0.87	0.87	0.87	15709

Random Forest

	precision	recall	f1-score	support
0	0.95	0.98	0.97	8932
1	0.97	0.94	0.96	6777
accuracy			0.96	15709
macro avg	0.96	0.96	0.96	15709
weighted avg	0.96	0.96	0.96	15709

6. Feature Importance

Random Forest helped highlight which features contributed most to the final predictions. Logistic Regression provided the direction of influence for each feature, which was equally helpful for interpretability.

Having both perspectives allowed me to understand not just which model performed better, but why the predictions were happening.

7. Recommendations

Based on all the evaluations, Random Forest is the model I would recommend, especially when the goal is strong predictive performance.
With 96% accuracy, it consistently makes correct predictions and handles subtle patterns extremely well.

Even though it didn't perform as well here, Logistic Regression still has value when interpretability or simplicity is needed. It's easier to explain to stakeholders and ideal for systems that need fast or lightweight models.

 # What Can Be Done Next

To push the project further, I could:

Try advanced ensemble models like XGBoost or Add SHAP analysis for model interpretability.