Executive summary

This project develops machine learning models to predict whether a borrower will default. Using a 255,347-row loan dataset, the analysis covers data cleaning, exploratory analysis, encoding, oversampling to address class imbalance, and two model baselines: Logistic Regression and Random Forest. The Random Forest achieved much higher reported performance (98% accuracy, ROC AUC = 0.98) than Logistic Regression (68% accuracy, ROC AUC = 0.745). However, several methodological issues in the workflow likely produced overly optimistic metrics for the Random Forest; recommended fixes and next steps are provided.

1. Project objective

Build a predictive model to classify loans into Default = 1 (defaulted) or Default = 0 (not defaulted), suitable for use in pre-approval screening or risk-scoring pipelines.

2. Data overview

- Rows**:** 255,347 records.
- Columns (after dropping LoanID): 17 features including numeric (Age, Income, LoanAmount, CreditScore, MonthsEmployed, NumCreditLines, InterestRate, LoanTerm, DTIRatio) and categorical (Education, EmploymentType, MaritalStatus, HasMortgage, HasDependents, LoanPurpose, HasCoSigner) plus the Default target.
- Missing values: None reported — all columns fully populated.
- Target imbalance (raw): 225,694 non-defaults vs 29,653 defaults (11.6% defaults overall).

Observations from the provided EDA: categorical variables display near-uniform counts across categories (e.g., Education categories have similar counts), which may indicate either a balanced synthetic dataset or a preprocessed sample.

3. Preprocessing & feature engineering

- Dropped LoanID.
- Label mapping applied to several categorical fields (Education, EmploymentType, HasMortgage, HasDependents, HasCoSigner).
- One-hot encoding for remaining categoricals (drop_first=True) and cast to integer types.
- Class imbalance handling: Random oversampling (RandomOverSampler) used to equalize classes from (225,694 vs 29,653) to (225,694 vs 225,694).
- Standard scaling applied to numeric features before modeling.

Important methodological notes / issues: the workflow performed oversampling on the full dataset *before* splitting into train/test, and the scaler appears to have been fit separately on the test set as well. Both practices can cause data leakage and overly optimistic test scores.

4. Modeling pipeline

Two supervised classifiers were trained:

1. Logistic Regression baseline linear model.
    - Inputs: scaled numeric + encoded categorical features.
    - Training / test split after oversampling: train_size = 302,429 rows, test_size = 148,959 rows.

2. Random Forest Classifier ensemble tree model with 100 estimators and default depth.

Both models were evaluated on the (resampled) test set.

5. Key results

Logistic Regression

- MAE: 0.32
- Accuracy: 0.68
- Classification report (macro avg F1 = 0.68) (balanced precision/recall across classes).
- ROC AUC: 0.7449.

Random Forest

- Accuracy: 0.98
- Classification report: precision / recall / f1 = 0.98 for both classes.
- Reported ROC AUC: reported as ~0.98 in one print statement and later logged as 1.00 during manual ROC plotting inconsistent values are present in the notebook.

Confusion matrices were plotted for both models in the notebook (showing near-balanced true/false rates for LR and very high true-positive and true-negative counts for Random Forest).

6. Interpretation & likely causes of observed performance

- The Logistic Regression baseline performs reasonably (accuracy = 68%, ROC AUC = 0.745) on the balanced (by oversampling) dataset this suggests some linear separability but limited discriminatory power.
- The Random Forest's extremely high performance (98%) is suspiciously high for this problem and dataset. Two likely contributors:

    1. Data leakage from oversampling before train/test split: oversampling the entire dataset prior to splitting can cause identical or overly similar synthetic samples to appear in both train and test sets, inflating test performance. The notebook shows oversampling performed before train_test_split.
    2. Improper scaler usage / fitting on test set: the notebook applies scaler.fit_transform to both training and test splits independently rather than fitting on train and transforming test another source of leakage.

Because of these procedural issues, the Random Forest metrics are likely over-optimistic and should not be taken at face value for production decisions until the pipeline is fixed and re-evaluated.

7. Limitations and risks

1. Oversampling before split → data leakage. This invalidates standard test-set evaluation.
2. No cross-validation / hyperparameter tuning reported. Model generalization not properly validated.
3. No feature importance or explainability results shown.

8. Conclusion

The project demonstrates a complete end-to-end modeling process on a large loan dataset and obtains promising baseline results (Logistic Regression). The Random Forest shows very strong performance in the notebook, but due to sampling and preprocessing ordering issues, this result should be revalidated with a corrected pipeline. After those corrections, tune and interpret the best model, then present business-focused metrics for deployment decisions.