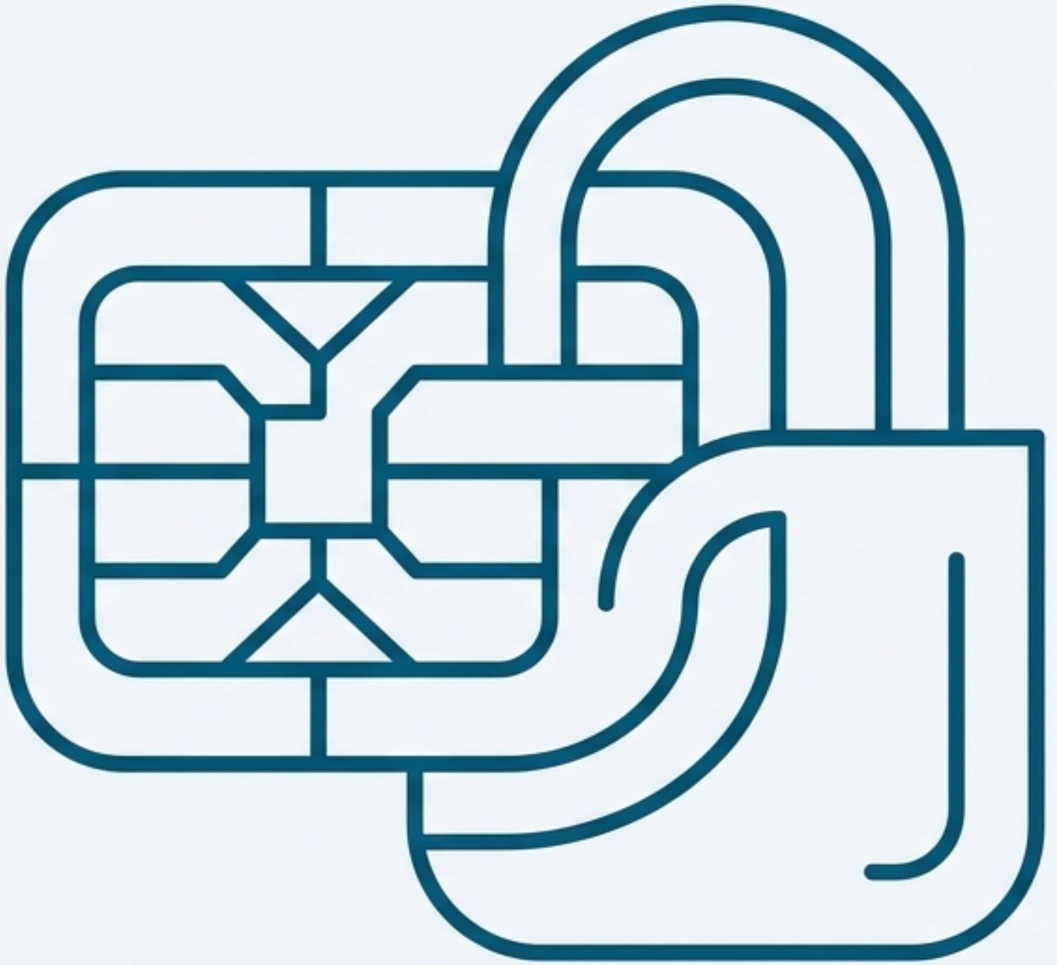
Building an Intelligent Fraud Detection System

A Case Study in Handling Severely Imbalanced Data



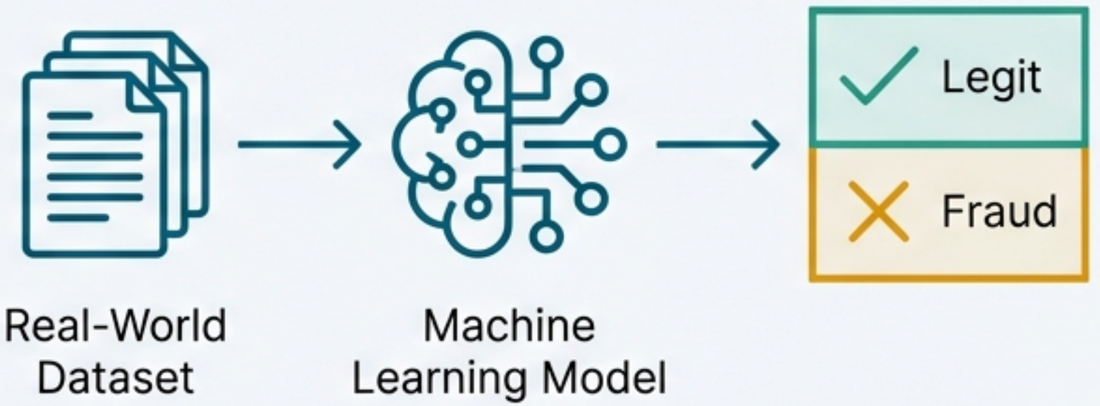
The Mission: Proactively Identify Fraudulent Transactions

Key Objective

Develop a machine learning model to accurately classify credit card transactions as either legitimate or fraudulent.

The Approach

- Utilize a real-world, anonymized dataset.
- Implement a Logistic Regression model, a powerful algorithm for binary classification problems.
- The entire project will be executed in Python using standard data science libraries (Pandas, Scikit-learn).



The Investigation Begins with a Real-World Transaction Dataset

Source

The data is a public dataset from Kaggle, containing transactions made by European cardholders.

Total Transactions

~284,000

Number of Features

30 (plus Class)

Feature Breakdown

Due to confidentiality, the original features have been transformed via Principal Component Analysis (PCA) into 28 numerical features (V1...V28). Two features remain untransformed: "Time" and "Amount". The target variable is "Class", where `0` is legitimate and `1` is fraudulent.

V1	V2	...	Amount	Class
0.084	-0.568	...	149.62	0
-1.359	-0.072	...	123.50	0
1.191	0.266	...	2.69	0
-0.966	-0.185	...	17.99	0

First, a Data Health Check Confirms a Clean Foundation

Initial Step

Check for missing or null values across all columns. A complete dataset is essential for reliable model training.

Method

Utilize the `.isnull().sum()` function in Pandas to count missing values per feature.

The Result

“We don’t have any missing values, which is a good thing.”

The dataset is complete and ready for analysis without the need for imputation.

A Hidden Flaw Threatens the Entire Project

The Discovery

An analysis of the 'Class' distribution reveals a severe imbalance between legitimate and fraudulent transactions.

The Numbers

- Legitimate Transactions (Class 0): 284,315
- Fraudulent Transactions (Class 1): 492

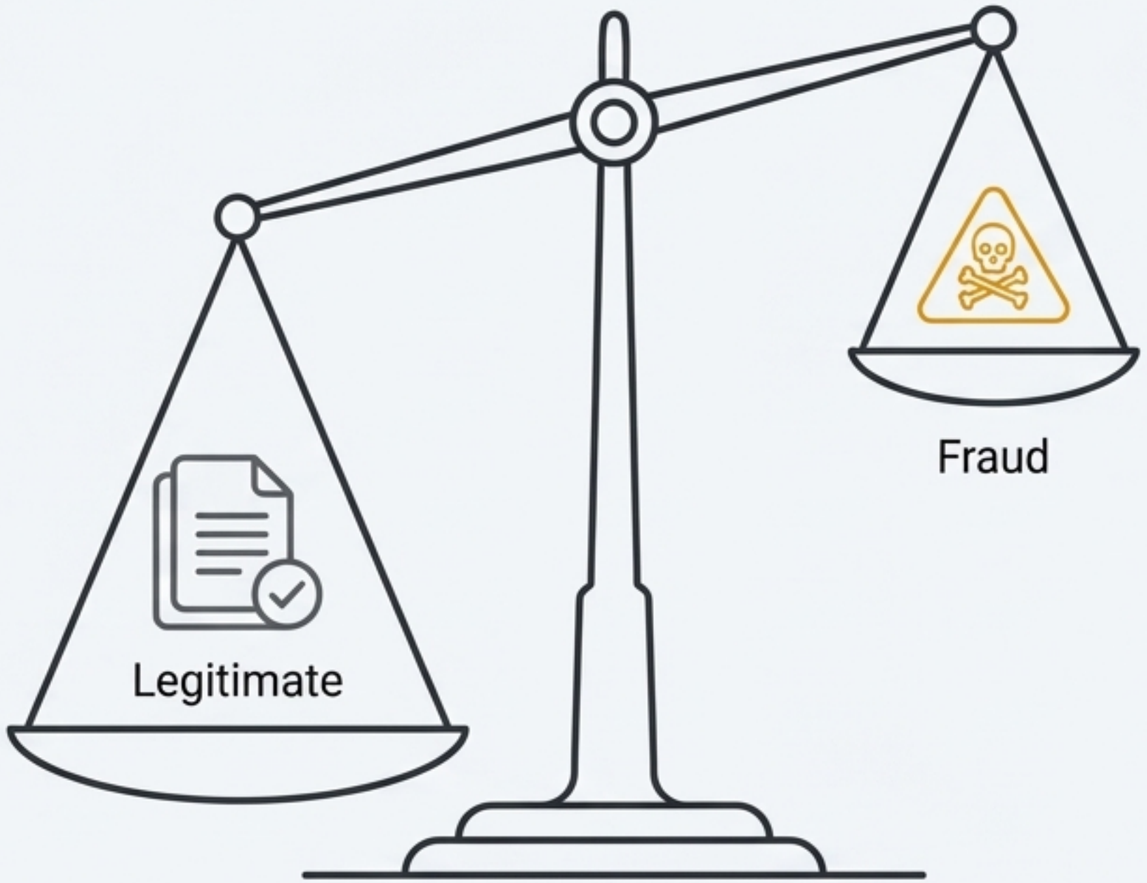
The Implication

Fraudulent transactions represent only a tiny fraction of the data (~0.17%).

Why This Imbalance Is a Critical Failure Point

The Bias Trap

If we train a model on this raw data, it will be overwhelmingly exposed to legitimate transactions.

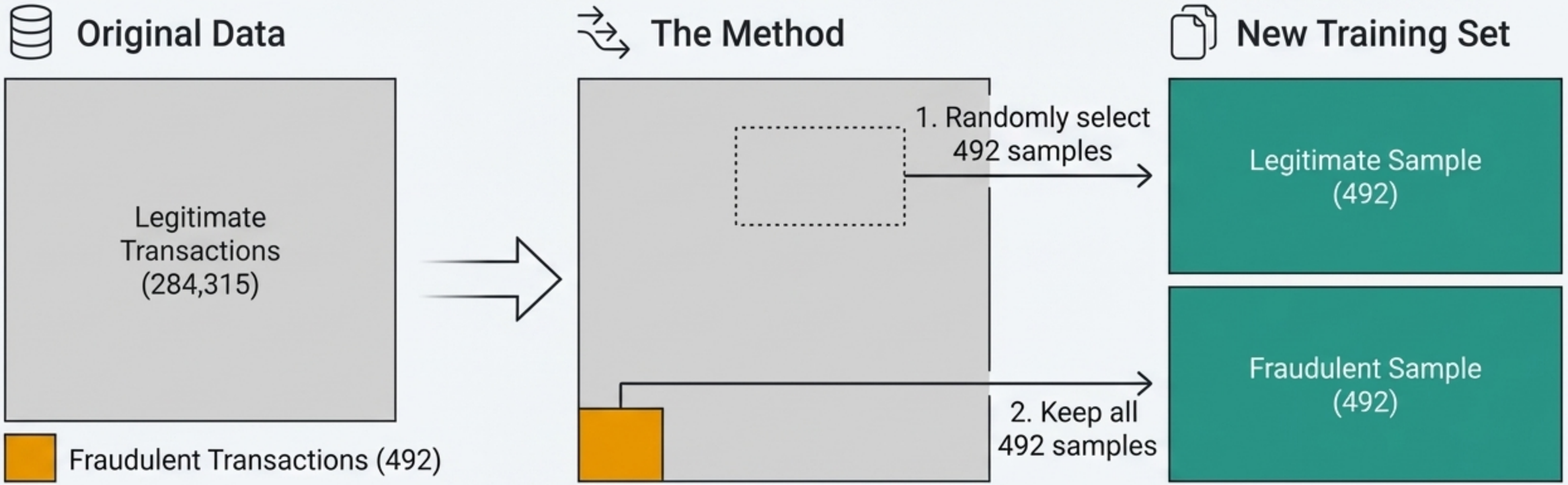


The Perverse Outcome

The model can achieve over 99% accuracy simply by predicting "legitimate" for every transaction. This creates a model that appears accurate but is completely useless, as it will never detect fraud.

Conclusion: This biased learning means its predictions for the minority class (fraud) will be poor.

The Strategy: Create a Balanced Dataset Through Under-Sampling



This ensures the model gives equal weight to learning the patterns of both classes, preventing bias towards the majority.

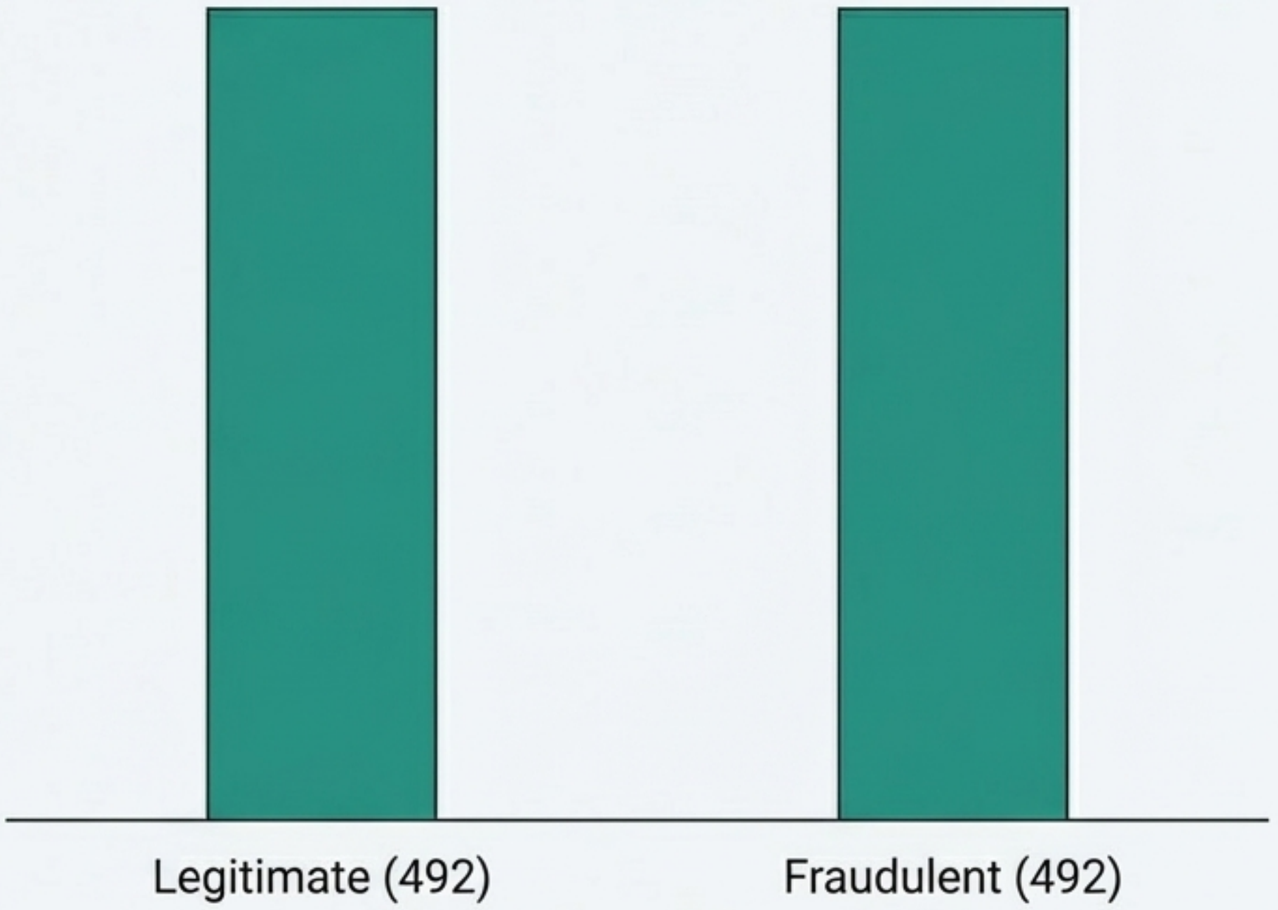
The Result: A Perfectly Uniform and Unbiased Dataset

New Dataset Composition

- Legitimate Transactions: 492
- Fraudulent Transactions: 492
- Total Training Samples: 984

Confirmation

“Now you can see we have a uniformly distributed dataset with 492 fraudulent transactions and 492 normal transactions.”



Validating the Sample: Do the Data Distributions Still Hold?

Critical Check

We must ensure our new, smaller dataset is representative. A key step is to compare the statistical properties (like the mean) of the features for both classes.

Observation

“We can see that the mean values are very different... This difference is very important for us, and this is how the machine learning model can find the difference between normal and fraudulent transactions.”

Conclusion

The significant difference in feature means between the two classes persists in the new dataset, confirming that the crucial predictive patterns have been preserved.

Comparison of Feature Means in New Dataset

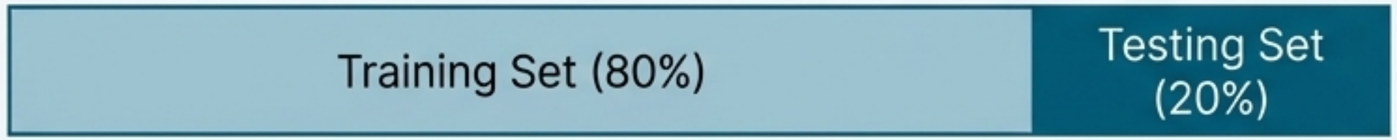
Feature	Mean (Class 0 - Legitimate)	Mean (Class 1 - Fraudulent)
V4	-0.057	**7.116**
V11	-0.093	**11.335**
V12	0.088	**-12.449**

Splitting the Data and Choosing Our Model

Step 1: Feature and Target Separation

`X` contains all the feature columns (V1-V28, Time, Amount).
`Y` contains the target 'Class' label (0 or 1).

Step 2: Splitting for Training and Testing



The training set is used to teach the model, while the testing set is held back to evaluate its performance on unseen data.

Step 3: Model Selection

- **Algorithm:** Logistic Regression.
- **Reason:** A highly effective and interpretable model for binary classification problems.

Evaluating Performance: High Accuracy on Training Data

The Process

→ 1. Train the Logistic Regression model using the `X_train` and `Y_train` data.

→ 2. Use the trained model to make predictions on the same `X_train` data.

→ 3. Compare the model's predictions to the true labels (`Y_train`) to calculate the accuracy.

Interpretation

"We have got an accuracy score of 0.947... this means that out of 100 predictions our model has made, 94 to 95 predictions are correct. It's a very good accuracy score."

Training Data Accuracy

94.7%

The True Test: Performance on Unseen Data

The Importance of Test Accuracy

“The accuracy score on test data is very important.” This metric tells us if the model has genuinely learned patterns or just memorized the training data (overfitting).

The Process

→ Use the *same* trained model to make predictions on the unseen `X_test` data.

→ Compare these new predictions to the true labels (`Y_test`).

Analysis

The test accuracy is very close to the training accuracy. This is an excellent outcome, indicating our model is robust and not overfitted.

Test Data Accuracy

93.1%

The Lesson: Data Strategy is the Foundation of Model Success

Challenge

We started with a dataset where fraud was a needle in a haystack fraud was a needle in a haystack (0.17%).

Strategy

By applying under-sampling, we created a balanced environment created a balanced environment where the model could learn effectively.

Result

We built a Logistic Regression model with high and consistent model with high and consistent accuracy (~94%) on both seen and unseen data.

The Key Insight: A powerful model is only as good as the data it's trained on. Solving the fundamental problem of data imbalance was more critical than any complex model tuning. This methodical approach prevented building a model that was 99% accurate but 100% useless.