Machine Learning Project Blueprint

Credit Card Fraud Detection Example

Leon Motaung

November 10, 2025

1. Define the Problem

Clearly state the goal of your ML project:

- Predict whether a credit card transaction is fraudulent or not using historical transaction data.
- Identify success metrics:
  - Accuracy, Precision, Recall, F1-score (for imbalanced datasets)
  - ROC-AUC for classification tasks

2. Collect & Understand Data

- Gather datasets from CSV, databases, or APIs.
- Explore data: columns, types, missing values, duplicates, class distribution.
- Tools: `pandas`, `numpy`, `matplotlib`, `seaborn`, `plotly`

3. Data Preprocessing & Cleaning

- Handle missing values.
- Correct data types.
- Encode categorical variables.
- Normalize or scale features if needed.
- Detect and handle outliers.
- Feature engineering (e.g., scale amount column).

4. Exploratory Data Analysis (EDA)

- Visualize the data to find patterns:
  - Histograms, boxplots, scatterplots
  - Correlation heatmaps
  - Class distribution
- Understand imbalance in dataset (fraud vs normal)

5. Split Data

- Training set: 70–80%
- Test set: 20–30%
- Optional validation set for hyperparameter tuning

6. Choose ML Models

Baseline Models

Algorithm	Pros	Cons	Use Case
Logistic Regression	Simple, interpretable, fast	Can underfit complex patterns	Good first model for fraud detection
Decision Tree	Easy to visualize, handles non-linear	Overfits easily	Understand feature importance
K-Nearest Neighbors (KNN)	Simple, non-parametric	Slow on large datasets, sensitive to scaling	Small datasets or initial experiments

Ensemble Models (Recommended)

Random Forest	Handles imbalance well, robust to outliers, good accuracy	Larger model, slower inference	Standard go-to for fraud detection
Gradient Boosting (XGBoost, LightGBM, CatBoost)	High accuracy, handles missing values, can handle imbalance	Slower training, hyperparameter tuning needed	Best for tabular datasets
AdaBoost	Focuses on hard-to-predict cases	Sensitive to noisy data	Optional, less used in modern fraud detection

Neural Networks (Optional Advanced)

MLP (Multi-Layer Perceptron)	Can model complex relationships	Needs scaling, more compute	Tabular datasets with many features
Autoencoders	Unsupervised anomaly detection	Harder to interpret	Detect fraud as anomalies
RNN	Sequence modeling	Slower training	Transaction sequences or time-series fraud detection

7. Handle Imbalanced Data

- Resampling: SMOTE, ADASYN, undersampling
- Class weights in models (e.g., Random Forest, XGBoost)
- Anomaly detection: IsolationForest, OneClassSVM, Autoencoder

8. Train & Evaluate Models

- Use confusion matrix, precision, recall, F1-score, ROC-AUC.
- Evaluate baseline and ensemble models.

9. Optimize Model

- Hyperparameter tuning: GridSearchCV, RandomizedSearchCV, Optuna
- Feature selection

10. Deploy Solution

- Save model using `joblib` or `pickle`
- Deploy as REST API (Flask/FastAPI)
- Build dashboard for visualization (Streamlit, Dash, Plotly)

11. Monitor & Iterate

- Track model performance on new data
- Retrain periodically
- Monitor metrics like precision/recall drift

Recommended Algorithm Roadmap

1. Start simple: Logistic Regression, Decision Tree
2. Move to ensemble: Random Forest, XGBoost, LightGBM
3. Handle imbalance: SMOTE, class weights, anomaly detection
4. Optional advanced: Neural Networks / Autoencoders