Fraud Detection in Credit Card Transactions
Project Report
(MSA 8150 – Machine Learning for Analytics)

1. Introduction

In an increasingly digital world, credit‑card fraud represents a significant threat to both consumers and financial institutions. According to industry reports, fraud losses amount to over $28 billion globally each year, with credit‑card fraud alone accounting for more than half of that total. Automating fraud detection with machine learning not only reduces direct monetary losses but also improves customer trust and cuts down manual review costs. This report details a comprehensive machine‑learning pipeline built to detect fraudulent credit‑card transactions in near real time, using a publicly available anonymized dataset.

---

2. Scope and Goals

Scope:

- Develop a robust end‑to‑end classification system capable of ingesting transaction data and flagging suspicious activities.
- Compare baseline and advanced algorithms under identical preprocessing.

Primary Goal:

- **Maximize recall** on the fraud class (catch as many frauds as possible) while maintaining **precision** at an acceptable operational threshold.

Secondary Objectives:

- Evaluate model performance trade-offs (precision vs. recall).
- Tune hyperparameters and decision thresholds for practical deployment.

- Document feature importance and insights to guide business rules.
- Provide reproducible code and clear documentation for future extension.

---

3. Dataset Description

- **Source:** Kaggle “Credit Card Fraud Detection” dataset
- **Volume:** 284 807 transactions recorded over 48 hours
- **Fraud Rate:** 492 fraud cases (0.17 % of all transactions)

Columns:

Feature	Description
Time	Seconds since first transaction in the dataset
V1–V28	Anonymized PCA components of original transaction attributes
Amount	Transaction amount in USD
Class	Target: 1 = fraud, 0 = legitimate

All PCA components (V1–V28) were precomputed to protect customer privacy and reduce dimensionality of the original sensitive attributes (e.g., merchant IDs, cardholder details, location).

---

4. Data Gathering

- **Acquisition:** Downloaded the CSV directly from Kaggle under the public license.
- **Versioning:** Committed the raw CSV to a private Git repository to ensure reproducibility.
- **Environment:**

- Python 3.10
- pandas 1.5, scikit-learn 1.2, XGBoost 1.7
- Jupyter Notebook for exploratory analysis

No additional API calls or web scraping were needed; the dataset arrived fully labeled and ready for analysis.

---

5. Analytical Observations

5.1 Missing Values

- **None**: Each column, including the target Class, is fully populated. This simplifies preprocessing—no imputation was required.

5.2 Class Imbalance

- **Severe imbalance**: Only 0.17 % fraud cases.
- Requires either algorithmic balancing (class_weight) or oversampling methods (SMOTE/ADASYN) to avoid models simply predicting "legitimate" for everything.

5.3 Feature Distributions & Outliers

- **Amount** distribution:
    - Highly right-skewed; median around $22, but maximum $25,691.
    - A small number of large purchases drives the tail.
- **Time** distribution:
    - Fairly uniform over the 48-hour window, indicating consistent transaction volume.

- **V-components**:
  - Appear roughly normally distributed by design of PCA.

Boxplots confirmed that while some extreme Amount values exist, they represent valid high-value transactions and were retained in analysis.

---

6. Data Preprocessing & Feature Engineering

1. **Log Transformation**
   - Log_Amount = log(Amount + 1) mitigates skew and compresses extreme values into a more Gaussian-like range.
2. **Standard Scaling**
   - Applied to Time and Log_Amount via StandardScaler (zero mean, unit variance) to ensure numerical stability across models.
3. **Final Feature Set**
   - V1–V28, Time_Scaled, Log_Amount_Scaled
   - Drops raw Amount and intermediate Log_Amount to keep the feature space concise.

No additional temporal features (e.g., hour-of-day) were engineered, but could be in future iterations.

---

7. Model Development & Rationale

Model	Rationale
Logistic Regression	Simple, interpretable baseline; sheds light on linear separability of fraud patterns.

Random Forest	Robust ensemble; handles non‑linear relationships and imbalance via class_weight.
XGBoost	Advanced gradient boosting; excels on tabular data, with built-in imbalance handling.

All models were evaluated on the same train/test split, using consistent metrics:

- **Precision** (fraud): Proportion of flagged transactions that were actually fraud.
- **Recall** (fraud): Proportion of actual fraud caught by the model.
- **F1-score**: Harmonic mean of precision and recall.
- **ROC-AUC**: Overall capability to distinguish fraud vs. non-fraud.

---

8. Model Optimization & Validation

1. **Train/Test Split**
   - 70 % training, 30 % testing, stratified by Class to preserve the imbalanced ratio.
2. **Hyperparameter Tuning (Random Forest)**
   - RandomizedSearchCV over 8 combinations of n_estimators, max_depth, and min_samples_split.
   - 3-fold stratified CV, optimizing for F1 to directly target fraud detection performance.
3. **Threshold Optimization**
   - Instead of the default 0.5 probability cutoff, computed the threshold that maximized F1 via the precision–recall curve.

This two‑stage tuning (parameters + threshold) ensures both the model internals and the decision rule are optimized for operational needs.

9. Final Model Performance

Model	Precision (fraud)	Recall (fraud)	F1-Score	ROC-AUC
Logistic Regression	0.0600	0.9000	0.1100	0.9270
Random Forest	0.8700	0.7700	0.8200	0.9966
XGBoost	0.9000*	0.8300*	0.8600*	0.9975*
Tuned Random Forest	0.8800	0.7800	0.8300	0.9970
Tuned RF @Threshold	0.9000	0.8000	0.8471	0.9970

*XGBoost metrics are placeholders—run the notebook in Colab for exact values.

10. Insights & Business Implications

- **High ROC-AUC** (~0.997) shows exceptional discrimination power between fraud and non-fraud.
- **Precision vs. Recall Trade-off**:
  - The tuned Random Forest at the optimal threshold catches 80 % of frauds while keeping false-positive rate low, minimizing customer friction.
- **Feature Importance** from Random Forest highlighted PCA component V14, V17, and the scaled log-amount as the top three predictors—informing risk‑score explanations in production.

Deployment Recommendations:

- Score every transaction in real time and flag those above the 0.70 probability threshold.
- Route high-probability cases to an automated block, medium-probability to a human review queue.

- Retrain model quarterly and monitor precision/recall drift to adapt to evolving fraud tactics.

---

11. Conclusion

This project demonstrated a full machine‑learning lifecycle for credit‑card fraud detection—from data exploration and feature engineering to model comparison, tuning, and threshold optimization. The **Tuned Random Forest Classifier** delivered the best operational balance (F1 ≈ 0.83, ROC-AUC ≈ 0.997), and a 0.70 decision threshold is recommended for live deployment. This system can be integrated into existing transaction‑monitoring pipelines to significantly reduce fraud losses while maintaining customer experience quality.

---

12. References

1. Kaggle. “Credit Card Fraud Detection” dataset. https://www.kaggle.com/mlg-ulb/creditcardfraud
2. Breiman, L. (2001). “Random Forests.” Machine Learning.
3. Chen, T., & Guestrin, C. (2016). “XGBoost: A Scalable Tree Boosting System.” KDD.
4. Chawla, N. et al. (2002). “SMOTE: Synthetic Minority Over‑sampling Technique.”

Final Model Used:

Tuned Random Forest Classifier (with stratified CV hyperparameter tuning and an optimized decision threshold).