1. Introduction

Fraud detection in credit card transactions datasets is a well-known classification problem where the aim is to learn the basics of modelling and using appropriate tools to a model that identifies fraud transactions.

My project will demonstrate main principles and classical machine learning techniques and workflow practices for this problem. The dataset used contains 1 million transactions from Kaggle, with a fraud rate of ~10%.

In this document, I will demonstrate:

- Machine learning workflow
- Model comparison and selection
- Learning/validation curve analysis
- Hyperparameter turning
- Imbalance handling
- EDA and data interpretation

The data set used in this project is from Kaggle, and can be accessed from:

https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud

The features of the dataset are the following:

- Continuous features:
    - Distance from home
    - Distance from last transaction
    - Ratio to media purchase price
- Binary features
    - Repeated retailer
    - Used chip
    - Online transaction
    - Used pin
- Target:
    - Class (Fraud = 1 --- Non-fraud = 0)

Note that for the purposes of practicing and achieving the best learning outcomes, I did not consider the missing unit for each distance/ratio attribute as an issue.

2. Exploratory Data Analysis + Preprocessing

Class Imbalance:

- Fraud ~10%
- Non-Fraud ~90%

This level of imbalance is noticeable, but not extreme. Therefore, I tested oversampling to see how the model behavior changes. However, because the dataset is already highly informative (which is not usually the case in a real business context), resampling did not change model performance significantly. So for the final model training was done without oversampling.

Distribution:

- Continuous features:
    - All three features show right-skewed distributions.
    - Fraud transactions generally fall withing similar ranges to non-fraud, indicating no strong numeric separation.
- Binary features:
    - They showed uneven distributions, meaning each feature has a distinct pattern.
    - Fraud transactions have a noticeably higher proportion of online payments, making this feature an important signal. Other features did not show meaningful separation.

For cleaning and feature engineering, I did not perform much data cleaning and preparation since all features were already 1) clean 2) useful 3) and directly interpretable, so no feature engineering was performed. However, in a real business context It is obvious that a high amount of time will be spent at cleaning and preparing the data for analysis, but this will be for another project and practice idea.

For preprocessing I did some scaling using Sklearn's standard scaler (for model comparison and stability not for the final model presented). This scaling was done to non-binary features, while binary features were kept in a 1/0 format to preserve meaningfulness.

3. Model Selection and Evaluation.

I have tested a set of the most important machine learning models from Sklearn, which are 1) SVM 2) Random Forest 3) Logistic Regression. I have compared those models using 5-fold cross validation with metrics of f1 and recall.

Final model choice was: Random Forest Classifier. Even though all models performed strongly due to the dataset's high signal strength, Random Forest delivered the best performance for those metrics. This might be for several reasons such as the following:

- Features are highly informative already (if it was a real business scenario I would need to do feature engineering to achieve this).
- Tree-based models excel at:
    - Handling binary splits
    - Learning non-linear and "logical" conditions
    - Capturing contrast between features
- Trees do not rely on scaling
- Ensemble methods reduce variance and overfitting

Faud patterns implied logical conditions, which decision trees handle very well, especially ensembles of trees.

4. Hyperparameter Tuning

I used validation curves to see the effect of single hyperparameters (max_depth and n_esimators) on model performance.

Insights:

- Increasing max_depth improves performance up to around 7, then stabilizes
- Increasing n_estimators improves performance up to about 20, then stabilizes

I also performed grid search on the same parameters using GridSearchCV from Sklearn.

Conclusion:

The dataset is highly informative, so improvements beyond simple default parameters were marginal. This is understandable since clean and informative data reduced tuning complexity, which is not the case usually in dynamic real scenarios in businesses.

5. Final Performance

Key metric: Recall

I chose recall due to several factors:

- Fraud classification being a safety-critical problem
- False negatives (missing frauds) being extremely costly

Results on the test set:

- Recall ~ 0.99
- F1 score ~0.99
- False negatives 3 out of 17k

This performance is unusually high, because the dataset is closer to a theoretical example than a noisy real-world dataset. However, the purpose of this project is to analyze, understand, and explain the appropriate tools and techniques so this isn't a big problem since I demonstrated 1) proper workflow 2) modelling 3) understanding of evaluation metrics 4) model selection and reasoning behind decisions.

6. Learning Curve Analysis

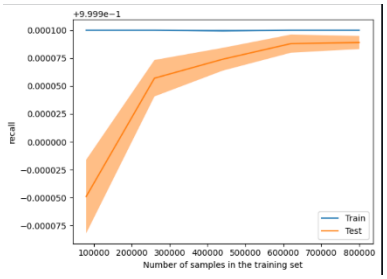

Figure 1. Learning Curve

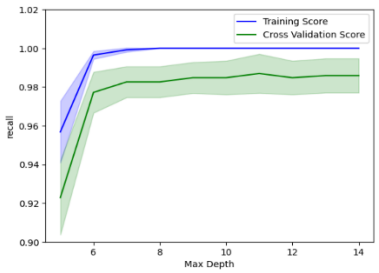

Figure 2. Validation curve of max_depth feature

The model quickly shows consistency with such highly informative features. With curves indicating that adding more data would result in very small performance gains, which confirms that the dataset is stable, predictive, and nearly ideal for supervised ML workflows.