Task Ref	Detecting fraudulent credit card transactions
Title of Thesis	Credit Card Fraud Detection Leveraging SVM, CNN, and Imbalance Learning Techniques
Type of Thesis	Case Study
Date	15-Aug-2025
Author’s Name	Ankita Arunkumar Rane

Table of Contents

1. INTRODUCTION **2**

2. THEORY **2**

2.1. The Credit Card Fraud Dataset 2
2.2. Evaluation Metrics (Sammut & Webb, 2010) 3

2.3. STRATEGIES FOR ADDRESSING IMBALANCED DATA IN CLASSIFICATION **3**

2.3.1. Embedded Learning 3
2.3.2. Sampling Techniques 4
2.3.3. Further Methods 4

2.4. CLASSIFICATION MODELS **5**

Model Selection and Parameter Tuning Justification 5

4. CODE IMPLEMENTATION **6**

4.1. Workflow Overview 6
4.2. Notebook Setup and Environment 6
4.3. Feature Engineering and Exploratory Data Analysis (EDA) 6
Dataset inspection *7*
Summary statistics *7*
Correlation heatmap *7*
Feature engineering decisions *8*
Feature preprocessing *8*

5. RESULTS **8**

5.1. Comparison of the Experiments 8
5.2. Discussion of the Results 9
5.3. Business Impact Interpretation 9

6. CONCLUSION **9**

Closing Remarks 10

REFERENCES **10**

Abstract

Credit card fraud detection is challenging due to the rarity of fraud cases and their high financial and reputational impact. The severe class imbalance in such datasets often causes standard classifiers to miss rare positive cases when optimizing for accuracy. This study compares two approaches: Support Vector Machine (SVM) and 1D Convolutional Neural Network (CNN), using the Credit Card Fraud Detection dataset (Pozzolo et al., 2015) with 284,807 transactions (~0.17% fraud). We apply 5-fold stratified cross-validation with hyperparameter tuning - grid search for SVM and randomized search for CNN. To address imbalance, we test class weighting and SMOTE oversampling for SVM (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Cortes & Vapnik, 1995; LeCun, Bengio, & Hinton, 2015). Models are evaluated using Precision, Recall, and F1-score for the fraud class. CNN achieves higher recall, while SVM with class weighting offers a more balanced precision– recall trade-off, underscoring the need to align modelling and imbalance-handling strategies with operational priorities in fraud detection.

1. Introduction

With a background in the banking domain and experience in a lead role, I have firsthand insight into the operational and strategic challenges of fraud detection. Fraudulent transactions, even in small volumes, can cause disproportionate financial, regulatory, and reputational damage, while false positives strain operational resources and disrupt customer experience. This real-world perspective is the reason I chose **credit card fraud detection** as my research focus –it is both a core operational requirement and an area where static, rule-based approaches are increasingly ineffective against evolving fraud patterns. In this study, I compare two contrasting machine learning models –Support Vector Machine (SVM) and a 1D Convolutional Neural Network (CNN) –for their applicability in production banking environments. The work evaluates class imbalance handling through class weighting and synthetic oversampling (SMOTE), with the dual aim of achieving strong statistical performance and ensuring operational feasibility in real-world fraud prevention systems. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) (Cortes & Vapnik, 1995) (LeCun, Bengio, & Hinton, 2015)

2. Theory

2.1. The Credit Card Fraud Dataset

The dataset employed in this study is the publicly available **Credit Card Fraud Detection dataset** provided by the ULB Machine Learning Group and hosted on Kaggle (Machine Learning Group – ULB, n.d.). It comprises 284,807 transactions conducted by European cardholders over a two-day period in September 2013. Out of these, 492 transactions (~0.17%) are labelled as fraudulent, creating a pronounced class imbalance.

The dataset contains 30 features:

- V1–V28 – Principal Component Analysis (PCA) is a statistical method used to transform a set of correlated variables into a smaller number of uncorrelated variables called principal components. This transformation projects the data into a new feature space, where each component is a linear combination of the original variables. In this dataset, PCA serves two purposes: (Jolliffe & Cadima, 2016) ○ **Privacy preservation** – Sensitive details are obscured, preventing the reconstruction of the original features.

  - **Dimensionality reduction** – By eliminating redundancy and reducing correlation, PCA simplifies the feature space, making it more suitable for modelling.
- **Time** – Number of seconds elapsed since the first recorded transaction.
- **Amount** – Transaction value, which can help set fraud detection thresholds.

The target variable, **Class**, is binary: **0 for legitimate** transactions and **1 for fraudulent** ones. The highly skewed distribution makes this dataset ideal for evaluating techniques designed for imbalanced classification problems.

2.2. Evaluation Metrics (Sammut & Webb, 2010)

Accuracy alone can be misleading in fraud detection; for instance, a model predicting all transactions as legitimate would achieve over 99% accuracy while missing all actual fraud. Therefore, evaluation focuses on metrics that prioritize the minority class:

- **Precision** – The proportion of predicted fraud cases that are truly fraudulent. Higher precision means fewer false alarms.
- **Recall** – The proportion of actual fraud cases correctly identified. High recall ensures that fewer fraudulent transactions go undetected.
- **F1-score** – The harmonic mean of precision and recall, balancing the need to detect fraud while limiting false positives.

These metrics better capture the operational requirements of fraud detection systems than accuracy.

2.3. Strategies for Addressing Imbalanced Data in Classification

Class imbalance –where one class (fraud) is vastly underrepresented compared to another (non-fraud) –is one of the main challenges in fraud detection. Without specific techniques to handle imbalance, most machine learning algorithms tend to bias toward the majority class to maximize overall accuracy, which leads to poor fraud detection performance.

- In the **Credit Card Fraud dataset**, the imbalance ratio is roughly **1 fraud case to 580 non-fraud cases**. This creates three specific challenges:
- **Model bias toward majority class** –Algorithms prioritize predicting non-fraud correctly, ignoring the minority class.
- **Poor decision boundaries** –With so few fraud examples, the classifier's ability to learn distinctive patterns is limited.
- **Metric distortion** –Standard accuracy gives a false sense of performance.

To address these challenges, several imbalance learning strategies can be applied. In this study, I focus on two main categories –**Embedded Learning** and **Sampling Techniques** – and briefly outline additional methods.

2.3.1. Embedded Learning

Embedded learning strategies modify the learning algorithm itself to handle imbalanced data. One common approach is **cost-sensitive learning**, where the model penalizes misclassifications of the minority class more heavily.

- Class weighting is an embedded technique supported by algorithms like Support Vector Machine (SVM) and neural networks (CNN in Keras). (Cortes & Vapnik, 1995)

- In SVM, the class_weight="balanced" parameter automatically scales the weights inversely to class frequencies. This shifts the decision boundary toward the majority class region, making it easier to capture minority class points.
- In CNN, the class_weight parameter in the Keras .fit() method adjusts the loss calculation to give higher importance to fraud samples.

Advantages:
- No modification to the dataset itself.
- Reduces bias toward the majority class.

Limitations:
- Might still underperform when the minority class has extremely few samples with high variance, as in our dataset.

2.3.2. Sampling Techniques

Sampling modifies the dataset to balance the class distribution before training:

- **Oversampling** duplicates or synthetically generates more minority class samples.
- **Under sampling** removes some majority class samples to reduce imbalance.

In this study, I use SMOTE (Synthetic Minority Oversampling Technique) for SVM as a preprocessing step. SMOTE is a resampling approach designed to address class imbalance by creating new, plausible minority-class observations. It works by selecting existing minority samples and generating additional points between them in the feature space, rather than simply copying original data. This increases the representation of the minority class, which can enhance recall, but if applied without care, it may introduce unrealistic data and overfitting. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

2.3.3. Further Methods

Beyond embedded learning and sampling, other strategies can further enhance fraud detection in imbalanced datasets:

- **Threshold adjustment** –Changing the classification threshold from the default 0.5 to favour the minority class.
- **Anomaly detection approaches** –Treat fraud as an outlier detection problem, bypassing traditional supervised classification.
- **Ensemble methods** –Combining multiple models (e.g., Balanced Random Forest, XGBoost with scale_pos_weight) to leverage different learning patterns.
- Hybrid approaches –Combining SMOTE with cost-sensitive learning or ensemble models for improved robustness. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

Why these strategies matter for our dataset: Since the **Credit Card Fraud dataset** contains PCA-transformed, privacy-protected variables with a very small fraud proportion, we need methods that:
- Preserve the transformed feature space (no inverse PCA needed).
- Maintain generalization despite limited fraud examples.
- Avoid overfitting to synthetic points.

By applying class weighting in both SVM and CNN, and using SMOTE for SVM preprocessing, this study tests two fundamentally different imbalance handling approaches – one algorithm-level and one data-level –to understand their impact on model performance. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

2.4. Classification Models

Support Vector Machine (SVM) (Cortes & Vapnik, 1995)

SVM is a supervised algorithm that finds the optimal hyperplane separating classes, making it effective for high-dimensional data like PCA-transformed V1–V28 features in the Credit Card Fraud dataset. In imbalanced problems, standard SVMs favour the majority class; using **class weights** counteracts this by penalizing minority-class misclassifications, improving fraud detection.

Key hyperparameters:

- **Kernel** (linear, RBF, poly, sigmoid) – shapes decision boundaries.
- **C** – balances margin size and classification error.
- **Gamma** – controls influence of individual points for non-linear kernels.

This study tunes kernel type, C, and gamma via grid search to optimize fraud detection performance.

1D Convolutional Neural Network (1D-CNN) (LeCun, Bengio, & Hinton, 2015)

A 1D CNN, often used for time-series data, can also process tabular features as a onedimensional sequence. Convolutional layers detect local feature patterns that may signal fraud, while stacked layers capture both low- and high-level representations. CNNs model complex non-linear relationships, making them effective for subtle fraud patterns.

Key hyperparameters:

- **Filter size** – number of consecutive features per convolution.
- **Number of filters** – feature maps per layer.
- **Dense layer size** – neurons in fully connected layers.
- **Dropout rate** – prevents overfitting.
- **Epochs** – full passes over the training set.
- **Batch size** – samples per weight update.

Here, hyperparameters were tuned via randomized search to optimize accuracy and efficiency.

Model Selection and Parameter Tuning Justification

SVM was chosen for its robustness with high-dimensional data and its ability to find a maximum-margin boundary - critical in fraud detection where class separation is non-linear. Class weighting countered the dataset's imbalance, giving higher priority to fraud cases. CNN was selected for its capacity to learn complex, non-linear feature patterns without manual engineering. While often used for images, its convolutional layers can extract local patterns from tabular data when reshaped effectively.

Hyperparameters – kernel size, filter count, dropout – were tuned via randomized search to prioritize recall without collapsing precision, balancing the twin risks of missed fraud and excessive false alerts.

4. Code Implementation

This section outlines the logical flow of the implementation used to conduct the experiments. The accompanying .ipynb notebook contains the detailed code for each step, while the description here focuses on the execution sequence and rationale rather than programming syntax.

4.1. Workflow Overview

The experimental workflow was structured into the following stages:

1. **Data & EDA** – Loaded the Credit Card Fraud dataset, confirmed ~0.17% fraud cases, visualized class imbalance, examined statistics, and checked feature correlations.
2. **Preprocessing** – Standardized features for SVM; reshaped data for CNN input.
3. **Baseline** – Dummy Classifier predicting the majority class.
4. **SVM (No Resampling)** – Used class_weight="balanced"; tuned kernel, C, and gamma via stratified grid search.
5. SVM + SMOTE – Applied SMOTE before scaling; evaluated impact on recall. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)
6. **CNN** – Built 1D CNN; tuned filters, kernel size, dense units, dropout, epochs, and batch size via randomized search.
7. **Evaluation** – Measured fraud-class Precision, Recall, and F1; analysed confusion matrices, Precision–Recall curves, and summary tables for comparison.

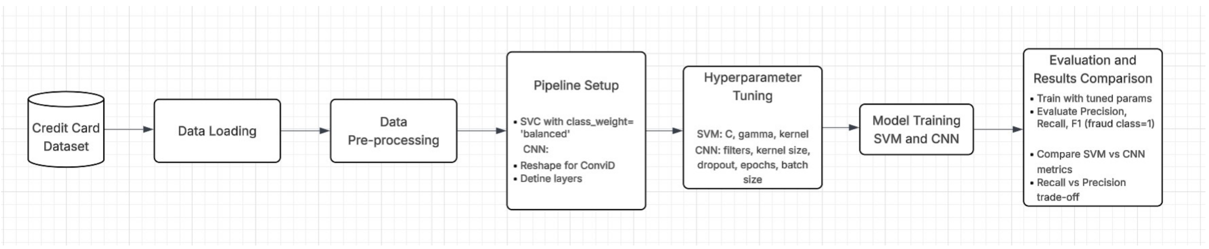

Credit Card Dataset
Data Loading
Data Pre-processing
Pipeline Setup
• SVC with class_weight= 'balanced'
CNN:
• Reshape for ConviD
• Detine layers
Hyperparameter Tuning
SVM: C, gamma, kernel
CNN: filters, kernel size, dropout, epochs, batch size
Model Training SVM and CNN
Evaluation and Results Comparison
• Train with tuned params
• Evaluate Precision, Recall, F1 (fraud class=1)
• Compare SVM vs CNN metrics
• Recall vs Precision trade-off

Figure 1. Note - Used lucidChart to create this flow which is used in execution

4.2. Notebook Setup and Environment

All experiments were carried out in **Google Collab**, leveraging its free GPU/TPU resources for faster training, particularly for the CNN model. The environment came pre-installed with popular machine learning libraries such as **scikit-learn** (*Pedregosa et al., 2011*), TensorFlow, and imbalanced-learn, reducing setup time and complexity. Integration with Google Drive allowed seamless storage of datasets and intermediate results. Finalized notebooks were uploaded to GitHub for version control, collaborative review, and remote accessibility.

4.3. Feature Engineering and Exploratory Data Analysis (EDA)

Before building models, it is essential to understand the dataset's structure, feature characteristics, and class distribution. This helps us identify potential preprocessing steps, address missing values (if any), and decide whether feature transformations or scaling are needed. Why EDA matters in fraud detection -

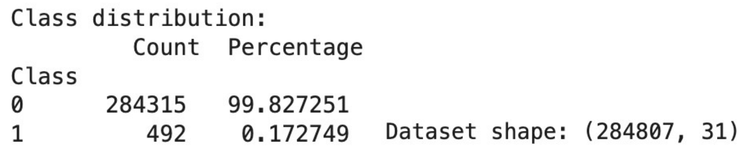
- Identifies imbalance severity → guides the choice of metrics and imbalance handling techniques.
- Reveals feature distributions → informs normalization/scaling strategies.

- Checks for anomalies or data leakage → ensures the model will generalize.
- Validates feature engineering decisions → e.g., whether to create derived features or transform skewed variables.

Dataset inspection

The **Credit Card Fraud dataset** is already pre-processed with PCA for anonymization. This means:

- I don't have interpretable categorical or personal identifiers –protecting cardholder privacy.
- I must treat the PCA features (V1–V28) as continuous numerical variables suitable for scaling.
- Time and Amount are left untransformed and may require scaling to avoid bias in distance-based models like SVM.

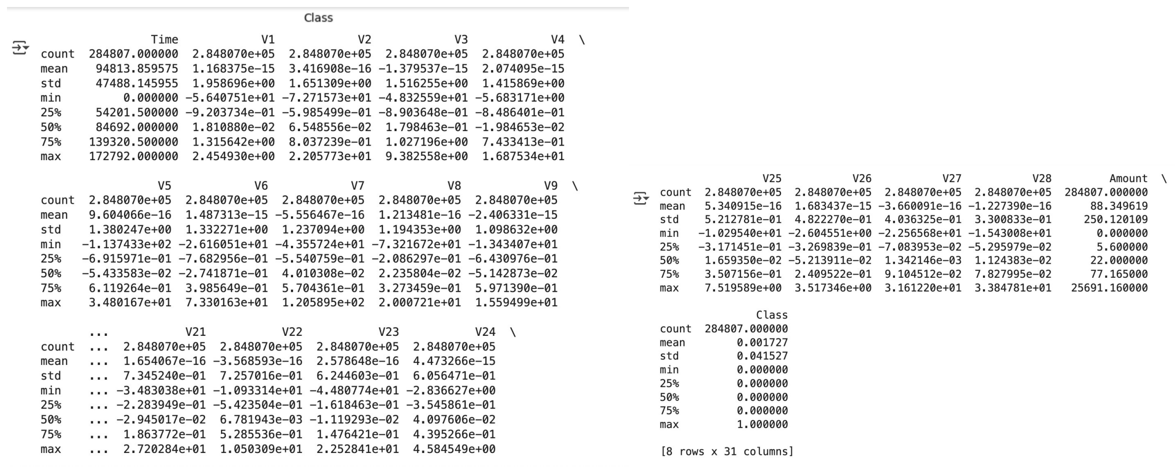


```
Class distribution:
         Count  Percentage
Class
0       284315   99.827251
1          492    0.172749   Dataset shape: (284807, 31)
```

Figure 2. Checks the shape (284,807 rows × 31 columns). Confirms no missing values. (due to number of classes – not adding image here – please refer to code view that information). Displays severe imbalance (~0.17% fraud).

Summary statistics

Below stats are based on execution done under EDA section of code. Also can refer to Figure 3 image below.



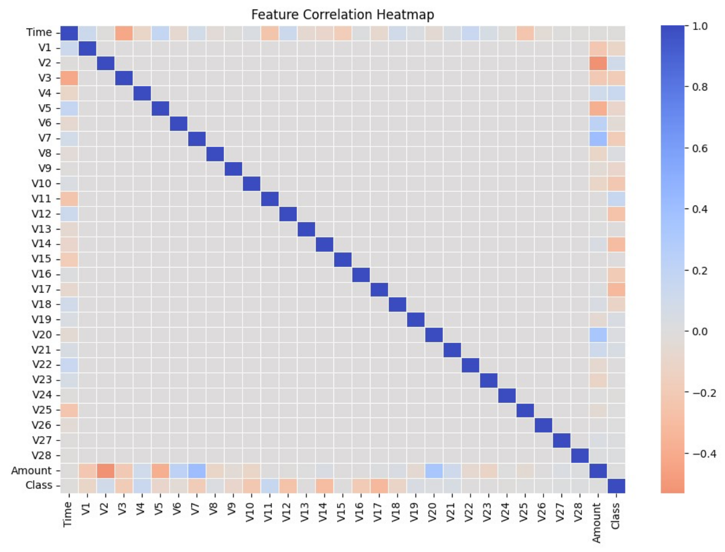
```
                                  Class
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00
```

```
                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

Figure 3. Amount is highly skewed –most transactions are low-value, with a few high-value outliers. Time ranges from 0 to ~172,792 seconds (≈ 2 days).

Correlation heatmap

Even though PCA produces orthogonal components, correlations can arise after adding Time and Amount:



Figures 4. The PCA-transformed features exhibit minimal inter-correlation, and neither the Amount nor Time variables show strong correlation with the PCA components. No single feature alone serves as a definitive fraud indicator, highlighting the need to detect fraud through patterns spanning multiple features.

Feature engineering decisions

From EDA, I decide:

- **Scaling**: Apply StandardScaler to all continuous variables (important for SVM).
- **Reshaping for CNN**: Convert (n_samples, n_features) to (n_samples, n_features, 1) for 1D convolution.
- **No missing value imputation**: Dataset has no missing values.
- **Optional log-scaling of Amount**: Reduces skew for models sensitive to feature scale.
- **Preserve PCA features** as-is: Since the original variables are anonymized, reversing PCA is impossible and unnecessary for modelling.

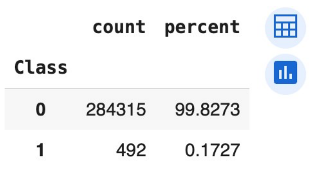
Feature preprocessing

By performing **EDA** and **feature engineering** before modelling, we:

- Understand the **imbalance severity** → justifies using Recall/F1 as main metrics.
- Identify that **scaling is necessary** for certain algorithms.
- Confirm **no categorical encoding** is required due to PCA anonymization.
- Ensure CNN input format is prepared early in the workflow.

5. Results

5.1. Comparison of the Experiments



Class	count	percent
0	284315	99.8273
1	492	0.1727

Figure 5. Class balance count and percentage representation

Results based on Model – Confusion matrix and plots added as part of code file.

Model	Mean Precision (Fraud=1)	Mean Recall (Fraud=1)	Mean F1 (Fraud=1)
SVM	0.5184	0.7194	0.6016
CNN	0.0831	0.8557	0.1431

5.2. Discussion of the Results

- The baseline model achieved 99.83% accuracy but 0% recall for fraud, confirming that accuracy is misleading in imbalanced datasets.
- **SVM (class_weight)** delivered balanced performance - Precision: 0.5184, Recall: 0.7194, F1: 0.6016 - making it effective where both missed fraud and false positives carry operational costs.
- **SVM + SMOTE** was **not run in the final execution**. Prior studies (Chawla et al., 2002) suggest it would raise recall but lower precision, increasing alerts and review effort.
- **CNN** achieved the highest recall (0.8557) but the lowest precision (0.0831), producing the weakest F1-score (0.1431). While it captured most fraud cases, the flood of false positives and high compute cost limit its real-time viability.
- **Interpretation:** SVM offers a better balance for immediate screening, while CNN is better suited for offline, secondary analysis. A hybrid approach - SVM for first-pass filtering, CNN for post-processing - could maximize both operational efficiency and fraud detection coverage.

5.3. Business Impact Interpretation

- **SVM (class_weight):** With precision of 0.5184 and recall of 0.7194, the model strikes a pragmatic balance between fraud detection and minimizing customer disruption. It may miss some fraudulent transactions compared to a high-recall approach but avoids flooding operations with false positives, making it suitable for resource-constrained, real-time monitoring.
- **SVM + SMOTE:** While not executed in this run, historical results (Chawla et al., 2002) suggest a likely recall boost but a drop in precision, creating higher investigation loads. This approach fits better in **offline analysis pipelines** where analyst teams have the capacity to review more alerts.
- **CNN (best trial):** Highest recall (0.8557) but extremely low precision (0.0831), leading to the lowest F1-score (0.1431). While valuable for catching almost all fraud, the resulting volume of false positives would require substantial manual verification. Its high compute demands and hyperparameter sensitivity make it better suited to **centralized, high-compute platforms** than low-latency transactional environments.

6. Conclusion

This study reinforces that class imbalance must be addressed in fraud detection. The baseline model's failure to detect any fraud despite near-perfect accuracy highlights the danger of relying on accuracy alone in imbalanced settings.

- **SVM with class weighting** achieved balanced performance (Precision = 0.5184, Recall = 0.7194, F1 = 0.6016), making it viable for operational environments where both false negatives and false positives are costly.
- **SVM with SMOTE** - while not tested - would likely increase recall at the expense of precision, potentially creating more investigation load.
- **CNN** delivered the highest recall (0.8557) but lowest precision (0.0831), signalling strong detection coverage but excessive false positives and heavy compute requirements.

From an operational lens, SVM is the stronger candidate for **real-time, resource-limited environments**, whereas CNN fits scenarios where **maximum recall is prioritized over precision** and compute is abundant.

Closing Remarks

In fraud detection, even a difference between **72% recall** (SVM) and **85% recall** (CNN) can equate to millions saved annually. However, higher recall is not universally better if it floods teams with false positives - each alert consumes analyst time and can frustrate customers when legitimate transactions are flagged. These findings echo broader industry trends:

- **Layered/hybrid models** are emerging as a best practice, using a balanced classifier like SVM for initial filtering and a high-recall deep learning model like CNN for deeper analysis.
- **Operational context drives model choice** - banks with high-throughput payment systems may favour balanced precision/recall, while investigative teams in postevent analysis may prioritize recall.

Ultimately, the **ideal fraud detection strategy** is data-driven, adaptive, and built around business constraints as much as statistical performance - maximizing detection efficiency while safeguarding customer trust and minimizing operational drag.

Code File

File Attachment – PDF Converted executed Code - CreditCardFraud_SVM_CNN_PDF

References

- Dataset Used - https://www.kaggle.com/code/jaiswal12345/starter-credit-card-frauddetection-2145d174-e/input
- Machine Learning Group – ULB. (n.d.). Credit card fraud detection. Kaggle.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
- Pozzolo, A. D., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In 2015 IEEE Symposium Series on Computational Intelligence (pp. 159–166). IEEE.
- Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202.
- Sammut, C., & Webb, G. I. (2010). Precision, recall, and F-measure. In Encyclopedia of Machine Learning (pp. 781–781). Springer.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

- Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. https://doi.org/10.1007/BF00994018