Fraud Detection System Project Report

1. Project Overview

This project implements a machine learning solution for detecting fraudulent online payment transactions. The system processes large-scale transaction data efficiently and employs multiple classification algorithms for fraud detection.

1.1 Objectives

- Detect fraudulent transactions in online payments
- Compare performance of different machine learning models
- Handle large datasets efficiently through memory optimization
- Provide comprehensive model evaluation metrics

2. Dataset Analysis

2.1 Dataset Features

- Transaction Type: Categorical (CASH_OUT, PAYMENT, CASH_IN, TRANSFER, DEBIT)
- Amount: Numerical (transaction value)
- oldbalanceOrg: Numerical (original account balance)
- newbalanceOrig: Numerical (new account balance after transaction)
- isFraud: Binary target variable (0: legitimate, 1: fraudulent)

2.2 Data Processing Pipeline

1. Memory-Optimized Loading
    - Chunk-based processing (chunk_size=100,000)
    - Optimized data types (float32, int8, category)
    - Garbage collection implementation
2. Preprocessing Steps
    - Categorical encoding of transaction types

```
type_mapping = {
    "CASH_OUT": 1,
    "PAYMENT": 2,
    "CASH_IN": 3,
    "TRANSFER": 4,
    "DEBIT": 5
}
```

- Missing value handling using median imputation
- Feature scaling using StandardScaler
- Infinite value handling

3. Model Implementation

3.1 Decision Tree Classifier

- Configuration:
  - Random state: 42
  - Default parameters for interpretability

3.2 K-Nearest Neighbors (KNN)

- Configuration:
  - n_neighbors: 5
  - Euclidean distance metric

3.3 Neural Network (MLP)

```
MLPClassifier(
    hidden_layer_sizes=(50,),
    max_iter=100,
    early_stopping=True,
    validation_fraction=0.1,
    n_iter_no_change=10,
    random_state=42
)
```

. Evaluation Metrics

4.1 Performance Metrics

- Accuracy scores
- Precision, Recall, F1-score
- ROC curves and AUC scores
- Cross-validation scores (5-fold)
- Training time measurements

4.2 Visualization Suite

1. Model Accuracy Comparison
    - Bar plot of accuracy scores
    - Error bars for statistical significance
2. Confusion Matrices
    - 3x1 subplot layout
    - Color-coded heatmaps
    - Annotated values
3. ROC Curves
    - Multiple model comparison
    - AUC scores in legend
    - Baseline reference line
4. Cross-validation Scores
    - Mean scores with error bars
    - Standard deviation visualization

5. Technical Implementation Details

```
# Chunk processing
chunk_size = 50000
X_chunks = []
y_chunks = []

# Memory cleanup
del chunk, X_chunk, y_chunk
gc.collect()
```

5.2 Data Pipeline

```
# Feature selection

features = ["type", "amount", "oldbalanceOrg", "newbalanceOrig"]

# Data splitting

train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

6. Results Analysis

6.1 Model Performance

- Training times
- Cross-validation scores
- Classification reports
- Confusion matrices
- ROC curves

6.2 Comparison Metrics

- Model accuracy comparison
- Performance trade-offs
- Memory usage
- Processing speed

7. Future Enhancements

7.1 Technical Improvements

- Hyperparameter optimization
- Additional model implementations
- Feature importance analysis
- GPU acceleration

7.2 Functionality Extensions

- Real-time prediction capabilities
- API implementation
- Automated model retraining
- Extended visualization options

8. Dependencies

- scikit-learn
- pandas
- numpy
- matplotlib
- seaborn
- Python 3.x