# Anomaly Detection in Trade Data Using Machine Learning Techniques

**Authors:** Aditya Jain, Malhar Vhatkar  
**Institution:** Thapar University

## Table of Contents

1. [Introduction](#introduction)
2. [Objectives](#objectives)
3. [Dataset Description](#dataset-description)
4. [Data Preprocessing](#data-preprocessing)
5. [Anomaly Detection Techniques](#anomaly-detection-techniques)
   * [Isolation Forest](#isolation-forest)
   * [One-Class SVM](#one-class-svm)
   * [Autoencoder](#autoencoder)
6. [Results and Evaluation](#results-and-evaluation)
7. [Visualizations](#visualizations)
   * [Histograms](#histograms)
   * [Correlation Heatmap](#correlation-heatmap)
   * [3D PCA Anomaly Plots](#3d-pca-anomaly-plots)
8. [Conclusion](#conclusion)
9. [References](#references)
10. [Acknowledgments](#acknowledgments)

## Introduction

In the global trade industry, detecting anomalies in transaction data is crucial for preventing fraud, ensuring compliance, and maintaining financial integrity. This project focuses on applying machine learning techniques to detect anomalies in trade data. We utilize unsupervised learning methods to identify unusual patterns that could indicate fraudulent activities.

This report documents the methodologies used, the results obtained, and the conclusions drawn from applying three anomaly detection techniques: Isolation Forest, One-Class Support Vector Machine (SVM), and Autoencoder neural networks.

## Objectives

* To generate a synthetic dataset that simulates trade transactions with embedded anomalies.
* To preprocess the data effectively for machine learning models.
* To apply Isolation Forest, One-Class SVM, and Autoencoder for anomaly detection.
* To evaluate and compare the performance of these models.
* To visualize the results using histograms, correlation heatmaps, and 3D PCA plots.
* To document the findings in a comprehensive report.

## Dataset Description

A synthetic dataset named enhanced\_trade\_data.csv was generated to simulate trade transactions. The dataset contains 1,000 records with the following features:

* **TransactionID**: Unique identifier for each transaction.
* **Date**: Date of the transaction.
* **Amount**: Monetary value of the transaction.
* **Commodity**: Type of commodity traded (Gold, Silver, Oil, Wheat).
* **Country**: Country involved in the transaction (USA, China, Germany, India).
* **TradeVolume**: Volume of trade.
* **TradeType**: Type of trade (Domestic, International).
* **TransactionTime**: Time of the transaction within a day.
* **Anomaly**: Label indicating whether the transaction is normal (0) or anomalous (1).

**Anomaly Injection:**

* A contamination rate of 10% was used to introduce anomalies into the dataset.
* Anomalies were generated by modifying the Amount, TradeVolume, TradeType, and TransactionTime features.

## Data Preprocessing

Data preprocessing steps included:

1. **Loading the Data**: The dataset was loaded using pandas.
2. **Encoding Categorical Variables**:
   * Label Encoding was applied to Commodity, Country, and TradeType using LabelEncoder.
   * Encoders were saved using joblib for future use.
3. **Date Conversion**:
   * The Date feature was converted to ordinal format to represent it as numerical data.
4. **Feature Scaling**:
   * MinMaxScaler was used to scale the features to a range between 0 and 1.
   * The scaler was saved using joblib.
5. **Train-Test Split**:
   * The data was split into training and testing sets with an 80-20 split.
   * Stratification was used to maintain the proportion of anomalies in both sets.

## Anomaly Detection Techniques

### Isolation Forest

* **Description**: An ensemble algorithm that isolates anomalies by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.
* **Parameters**:
  + n\_estimators: 100
  + max\_samples: 'auto'
  + contamination: 0.10
  + random\_state: 42
* **Model Persistence**: The trained model was saved using joblib.

### One-Class SVM

* **Description**: An unsupervised learning algorithm that learns a decision function for novelty detection, classifying new data as similar or different to the training set.
* **Parameters**:
  + kernel: 'rbf' (Radial Basis Function)
  + gamma: 'auto'
  + nu: 0.10
* **Model Persistence**: The trained model was saved using joblib.

### Autoencoder

* **Description**: A type of neural network that learns to reconstruct its input. Anomalies are detected based on the reconstruction error.
* **Architecture**:
  + **Encoder**:
    - Input layer with size equal to the number of features.
    - Hidden layers with sizes 64, 32, 16, and a bottleneck layer with size 8.
  + **Decoder**:
    - Mirrors the encoder layers in reverse order.
* **Training Parameters**:
  + num\_epochs: 500
  + learning\_rate: 0.001
  + Loss function: Mean Squared Error (MSE)
* **Model Persistence**: The trained model was saved using PyTorch's state\_dict.

## Results and Evaluation

**Performance Metrics Used**:

* **Precision**: Proportion of true positives among all positive predictions.
* **Recall**: Proportion of true positives among all actual positives.
* **F1-Score**: Harmonic mean of precision and recall.
* **Accuracy**: Proportion of correct predictions among all predictions.

**Training Set Performance**:

| Model | Train Precision | Train Recall | Train F1-Score | Train Accuracy |
| --- | --- | --- | --- | --- |
| Isolation Forest | 1.00 | 1.00 | 1.00 | 1.00 |
| One-Class SVM | 1.00 | 1.00 | 1.00 | 1.00 |
| Autoencoder | 1.00 | 1.00 | 1.00 | 1.00 |

**Testing Set Performance**:

| Model | Test Precision | Test Recall | Test F1-Score | Test Accuracy |
| --- | --- | --- | --- | --- |
| Isolation Forest | 1.00 | 1.00 | 1.00 | 1.00 |
| One-Class SVM | 1.00 | 1.00 | 1.00 | 1.00 |
| Autoencoder | 1.00 | 1.00 | 1.00 | 1.00 |

**Interpretation**:

* All three models achieved perfect scores on both the training and testing sets.
* This indicates that the models were able to perfectly distinguish between normal and anomalous transactions in the synthetic dataset.

## Visualizations

### Histograms

Histograms were plotted for the numerical features to understand their distributions:

1. **Amount**:
   * The distribution is centered around the mean value of 10,000 with a standard deviation of 2,000.
   * Anomalies are visible as extreme values due to the multiplication by 0.1 or 10.
2. **TradeVolume**:
   * The distribution is centered around the mean value of 100 with a standard deviation of 20.
   * Anomalies appear as outliers.
3. **TransactionTime**:
   * Uniformly distributed between 0 and 24.
   * Anomalies are introduced at extreme times (0 or 23.99).

### Correlation Heatmap

A correlation heatmap was generated to visualize the relationships between features:

* **Amount and TradeVolume**: Moderate positive correlation.
* **Anomaly Feature**:
  + High correlation with Amount and TradeVolume, indicating that anomalies significantly affect these features.

### 3D PCA Anomaly Plots

Principal Component Analysis (PCA) was used to reduce the dimensionality of the data to three components for visualization:

* **Isolation Forest**:
  + Normal transactions cluster together, while anomalies are scattered in different regions.
* **One-Class SVM**:
  + Similar separation between normal data and anomalies.
* **Autoencoder**:
  + Anomalies are well separated from normal transactions in the PCA space.

## Conclusion

* **Effectiveness of Models**:
  + All three models effectively detected anomalies in the synthetic trade data.
  + The perfect scores indicate the models' suitability for this type of data.
* **Synthetic Data Limitations**:
  + The dataset's synthetic nature and the method of injecting anomalies might have made it easier for the models to detect anomalies.
  + In real-world scenarios, anomalies may be more subtle.
* **Future Work**:
  + Apply the models to real trade datasets to evaluate performance.
  + Experiment with other anomaly detection techniques.
  + Fine-tune model parameters for improved generalization.

## References

1. **Scikit-learn Documentation**: Isolation Forest
2. **Scikit-learn Documentation**: One-Class SVM
3. **PyTorch Documentation**: Autoencoder Tutorial

## Acknowledgments

This project was completed as part of the coursework at **Thapar University**. We would like to thank our professors and peers for their support and guidance throughout this project.

**Authors**:

* **Aditya Jain**
* **Malhar Vhatkar**

**Note**: This report summarizes the steps and findings of our anomaly detection project. All visualizations and models were created using Python, leveraging libraries such as pandas, NumPy, scikit-learn, PyTorch, matplotlib, and seaborn.

We hope this report provides a comprehensive understanding of our project. Please feel free to reach out for any further information or clarifications.

Disclaimer: This is a text-based report. While we are unable to provide a PDF format here, the content can be converted into a PDF document using any text editor or word processing software.