**Credit Card Fraud Detection Using Deep Autoencoder**

**Overview**

This project demonstrates the use of a Deep Autoencoder, implemented in Keras, to detect fraudulent credit card transactions. The model is trained on non-fraudulent (normal) transactions and learns to reconstruct their patterns. Fraudulent transactions are then identified as deviations from these patterns based on reconstruction errors.

**Features**

* Preprocessing of credit card transaction data.
* Visualization of data distributions.
* Implementation of a Deep Autoencoder for anomaly detection.
* Evaluation of model performance using metrics like ROC-AUC and Precision-Recall.
* Graphical representation of results, including confusion matrix, ROC curve, and reconstruction errors.

**Requirements**

The following Python libraries are required:

* pandas
* numpy
* matplotlib
* seaborn
* sklearn
* tensorflow
* keras
* scipy

**Dataset**

The dataset used is a credit card transaction dataset stored at: C:\Users\yrosh\Documents\creditcard.csv

**Dataset Columns:**

* **Time**: The time elapsed since the first transaction.
* **Amount**: The transaction amount.
* **Class**: Labels indicating normal (0) or fraudulent (1) transactions.

**Workflow**

1. **Import Libraries** Load essential libraries and set up visualization preferences.
2. **Load and Explore Dataset**
   * Load the dataset using pandas.
   * Check for missing values and inspect the distribution of the Class column.
3. **Data Visualization**
   * Visualize transaction class distribution.
   * Plot histograms for transaction amounts and scatter plots for time vs. amount by class.
4. **Data Preprocessing**
   * Normalize the Amount column using StandardScaler.
   * Drop the Time column.
   * Split the dataset into training (normal transactions) and testing sets.
5. **Model Implementation**
   * Define the Autoencoder architecture with encoding and decoding layers.
   * Use the mean squared error as the loss function.
   * Save the best model using ModelCheckpoint.
6. **Training** Train the Autoencoder on normal transactions and validate it on the test set.
7. **Evaluation**
   * Analyze reconstruction errors to identify fraudulent transactions.
   * Visualize ROC curve, Precision-Recall curve, and reconstruction errors.
   * Calculate the confusion matrix and evaluate performance metrics.

**Results**

* **Model Performance**: The Autoencoder successfully identified fraud based on reconstruction errors.
* **Visual Outputs**:
  + Transaction class distribution.
  + Reconstruction error distributions.
  + ROC curve and Precision-Recall curve.
  + Confusion matrix.

**Key Files**

* **Model Checkpoints**: model.keras
* **Visualizations**:
  + transaction\_class\_distribution.png
  + Amount per transaction by class.png
  + Time of transaction vs Amount by class.png
  + Receiver Operating Characteristic.png
  + Recall vs Precision.png
  + Confusion matrix.png

**Conclusion**

The Autoencoder demonstrates the potential to discriminate fraudulent transactions by modeling patterns of normal transactions. While this implementation is a simple demonstration, further fine-tuning and advanced techniques can enhance its performance.

**Acknowledgments**

This project is inspired by the concept of anomaly detection using Autoencoders. Special thanks to the creators of the dataset.

**How to Run**

1. Clone this repository.
2. Install the required libraries.
3. Update the file path to your dataset in the code.
4. Run the script to train the model and generate visualizations.

**License**

This project is open-source and available under the MIT License.