

Credit Fraud Detection Using Various Anomaly Detection Algorithms

Time Series Anomaly Detection Research Intern | IIT Kharagpur

Shanmukh Garapati | 22MA10025

Sumeet D Dubey | 21AR10034

Mohd Amaan | 21EE10049

Abstract :

Credit fraud detection is a critical task for financial institutions due to its significant financial and operational impacts. This report explores various anomaly detection algorithms to identify fraudulent transactions in credit card datasets and learnings[1]. The algorithms evaluated include Isolation Forest, Local Outlier Factor (LOF), Support Vector Machines (SVM), and AutoEncoders. We analyze their performance on two different datasets with varying degrees of class imbalance.

1. Introduction

Credit fraud detection is essential for several critical reasons. It plays a crucial role in protecting consumers from financial loss and maintaining their trust in financial systems. Effective fraud detection ensures that consumers feel secure when using credit cards and digital payment methods. Furthermore, it helps financial institutions comply with stringent regulations, such as the PCI DSS, which mandate robust fraud prevention measures to avoid fines and legal consequences.[2]

Improving operational efficiency is another significant benefit of effective fraud detection. By reducing the number of fraudulent transactions, financial institutions can decrease the operational burden and costs associated with investigations, chargebacks, and customer service, allowing them to focus on their core activities. Additionally, institutions that excel in fraud detection can gain a competitive edge by offering secure and trustworthy services, thereby attracting and retaining customers.

Preventing identity theft is a crucial aspect of credit fraud detection, as it protects personal information and prevents further criminal activities. This, in turn, contributes to overall economic stability. Widespread fraud can destabilize financial systems and increase costs for goods and services, but effective fraud detection ensures secure transactions, thereby contributing to economic stability. [3]

2. Datasets

Here is the merged information:

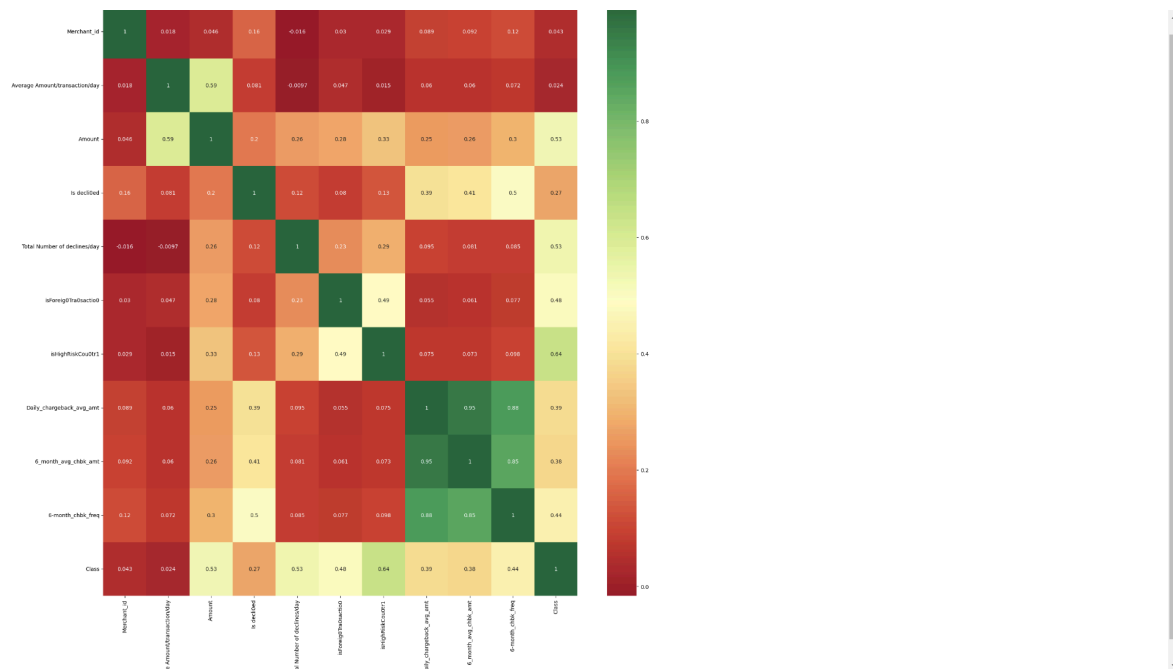
Features in the dataset:

1. Merchant_id: Identifier for the merchant.
2. Average Amount/transaction/day: The average amount per transaction per day.
3. Transaction_amount: The amount of the current transaction.
4. Is declined: Indicator if the transaction was declined.
5. Total Number of declines/day: Total number of declined transactions per day.
6. Is Foreign Transaction: Indicator if the transaction is foreign.
7. Is High Risk Country: Indicator if the transaction is from a high-risk country.
8. Daily chargeback average amount: The average chargeback amount per day.
9. 6_month_avg_chbk_amt: The average chargeback amount over the past 6 months.
10. 6-month chargeback freq: The frequency of chargebacks over the past 6 months.
11. Is Fraudulent: Indicator if the transaction is fraudulent.

About the datasets used

1. Dataset 1: Highly imbalanced with fraudulent cases comprising less than 1% of the total transactions.
2. Dataset 2: Moderately imbalanced with 16% fraudulent cases.

These datasets were selected to evaluate the algorithms' performance in different class distribution scenarios.



3. Algorithms and Methods

3.1 Isolation Forest

Isolation Forest is an ensemble learning method primarily used for anomaly detection. It works by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The key idea is that anomalies are few and different, and thus they are easier to isolate. [4]-[5]

Mathematics:

The average path length of unsuccessful search in a Binary Search Tree can be used to isolate points.

For a forest of binary trees, anomalies are isolated closer to the root of the tree.

Parameters Used:

``n_estimators=100``: The number of base estimators (trees) in the ensemble.

``max_samples=len(X)``: The number of samples to draw from X to train each base estimator. It is set to the length of X, meaning all samples are used.

``contamination=outlier_fraction``: The proportion of outliers in the dataset.

``random_state=state``: Ensures reproducibility by setting the seed for random number generation.

``verbose=0``: Controls the verbosity of the output. A value of 0 means no output.

Results:

Accuracy: 93%

High precision and recall on both datasets.

Isolation Forest: 19

Accuracy Score :

0.9383116883116883

Classification Report :

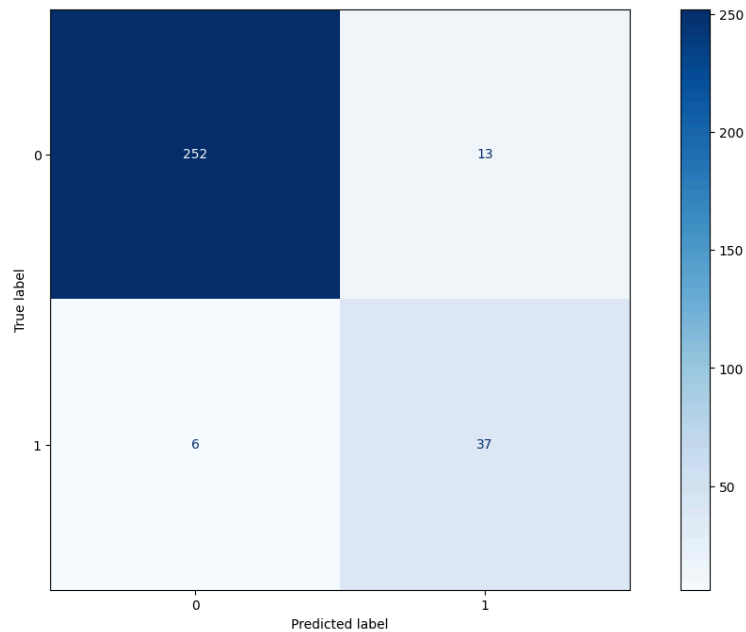
Precision recall f1-score support

0	0.980	.950	.96	265
1	0.740	.860	.80	43

accuracy 0.94 308

macro avg 0.860.910.88 308

weighted avg 0.940.940.94 308



3.2 Local Outlier Factor (LOF)

LOF is a density based anomaly detection algorithm that identifies anomalies by comparing the local density of a point to that of its neighbors. Points with a significantly lower density than their neighbors are considered anomalies. [6]

Mathematics:

Local Reachability Density (LRD): Measures how isolated an object is with respect to the density of its neighborhood.

LOF score: The ratio of the average LRD of the object's k nearest neighbors to the object's LRD.

Parameters Used:

``n_neighbors=20``: The number of neighbors to use for computing the local density.

``algorithm='auto'``: The algorithm used to compute the nearest neighbors. 'Auto' selects the appropriate algorithm based on the input data.

``leaf_size=30``: The leaf size parameter for the tree based algorithms.

``metric='minkowski'``: The distance metric to use for the tree. 'Minkowski' is a generalization of the Euclidean distance.

``p=2``: The power parameter for the Minkowski metric. $p=2$ corresponds to the Euclidean distance.

``metric_params=None``: Additional keyword arguments for the metric function.

``contamination=outlier_fraction``: The proportion of outliers in the dataset.

Results:

Consistent performance with high precision and recall.

Local Outlier Factor: 75

Accuracy Score :

0.7564935064935064

Classification Report :

Precision recall f1-score support

0 0.870 .850 .86 265

1 0.180 .210 .1943

accuracy 0.76 308

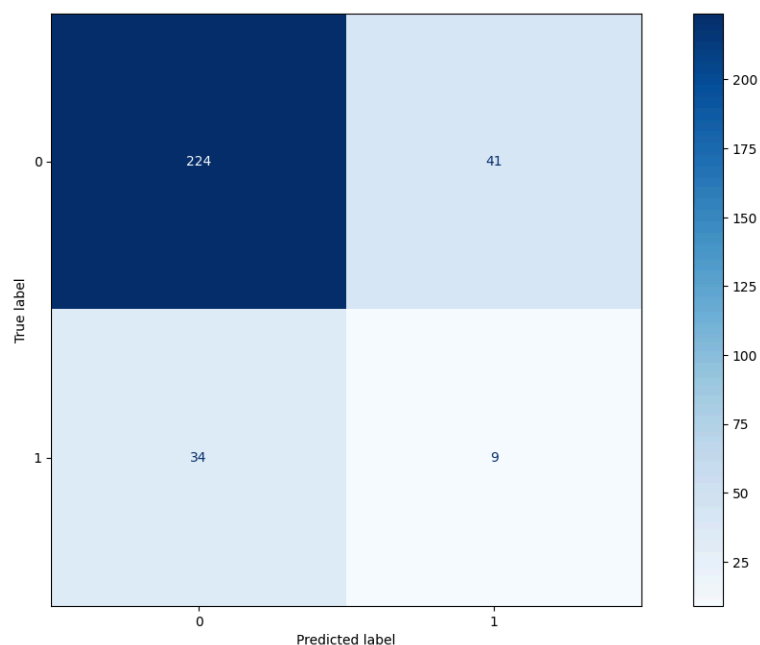
macro avg 0.520.530.53 308

weighted avg 0.770.760.76 308

Confusion Matrix:

[[22441]

[34 9]]



3.3 Support Vector Machines (SVM)

SVMs are supervised learning models that analyze data for classification and regression analysis. For anomaly detection, we use a one-class SVM, which learns a decision function for anomaly detection. [7]

Mathematics:

Constructs a hyperplane or set of hyperplanes in a high dimensional space.

The goal is to find the maximum margin between the hyperplane and the closest data points (support vectors).

Parameters Used:

`kernel='rbf': The kernel type to be used in the algorithm. 'RBF' (Radial Basis Function) maps samples into a higher dimensional space.

`degree=3`: The degree of the polynomial kernel function (if 'poly' is chosen as the kernel). For 'RBF', this parameter is ignored.

`gamma=0.1`: The kernel coefficient for 'RBF'.

`nu=0.05`: An upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors.

`max_iter=1`: The maximum number of iterations. A value of 1 means no limit.

Results:

Classification Report : Most cases classified as negatives,

Precision recall f1-score support

0	0.900	.230	.37 265
---	-------	------	---------

1	0.150	.840	.2543
---	-------	------	-------

accuracy 0.31 308

macro avg 0.520.530.31 308

weighted avg 0.790.310.35 308

3.4 AutoEncoders

AutoEncoders are neural networks used for unsupervised learning of efficient coding. The aim is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.[8]

Mathematics:

Consists of an encoder that compresses the input and a decoder that reconstructs the input from the compressed version.

Minimizes the reconstruction error.

Parameters Used:

Here is a summary of the technical aspects of the autoencoder model from the provided code snippet:

Model Architecture:

1. Input Layer:

- `input_layer = Input(shape=(input_dim,))`
- The input layer takes data with the shape `(input_dim,)`.

2. Encoder:

- First Encoding Layer:

- `encoder = Dense(encoding_dim, activation="tanh", activity_regularizer=regularizers.l1(10e-5))(input_layer)`

- A dense layer with `encoding_dim` units, using the `'tanh'` activation function, and L1 activity regularization.

- Second Encoding Layer:

- `encoder = Dense(int(encoding_dim / 2), activation="relu")(encoder)`

- A dense layer with half the units of the first encoding layer, using the `'relu'` activation function.

3. Decoder:

- First Decoding Layer:

- `decoder = Dense(int(encoding_dim / 2), activation='tanh')(encoder)`

- A dense layer with the same number of units as the second encoding layer, using the `'tanh'` activation function.

- Second Decoding Layer:

- `decoder = Dense(input_dim, activation='relu')(decoder)`

- A dense layer with the same number of units as the input layer, using the `'relu'` activation function.

4. Autoencoder Model:

- `autoencoder = Model(inputs=input_layer, outputs=decoder)`

- The autoencoder model is created with the defined encoder and decoder layers.

Training Configuration:

1. Epochs and Batch Size:

- `nb_epoch = 100`

- `batch_size = 32`

- The model is set to train for 100 epochs with a batch size of 32.

2. Compilation:

- `autoencoder.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])`

- The model is compiled with the Adam optimizer, mean squared error loss function, and accuracy as a metric.

3. Callbacks:

- Model Checkpoint:

- `checkpointer = ModelCheckpoint(filepath="model.h5", verbose=0, save_best_only=True)`

- Saves the best model based on validation performance to "model.h5".

- TensorBoard:
- ``tensorboard = TensorBoard(log_dir='./logs', histogram_freq=0, write_graph=True, write_images=True)``
- Configures TensorBoard for logging.

Training Process:

- ``history = autoencoder.fit(X_train, X_train, epochs=nb_epoch, batch_size=batch_size, shuffle=True, validation_data=(X_test, X_test), verbose=1, callbacks=[checkpointer,tensorboard]).history``
- The model is trained using the training data `X_train` as both input and target (unsupervised learning), with the validation data `X_test` used similarly.
- Training is done with shuffling, and progress is reported verbosely.
- The training history is saved in `history`.

Results:

Poor performance with less than 50% accuracy.

High error and failure to decrease error with increased epochs, likely due to NaN values and highly biased data.

4. Results and Discussion

Isolation Forest: Achieved the best results with 93% accuracy and high precision and recall across both datasets.

LOF and SVM: Also performed well with consistent high precision and recall.

AutoEncoders: Underperformed, highlighting the challenges of dealing with imbalanced and noisy data.

5. Conclusion



Isolation Forest, LOF are effective for credit fraud detection in varying degrees of class imbalance. AutoEncoders,SVM require further tuning and handling of missing values to be viable. Future work will focus on improving the preprocessing steps and exploring additional anomaly detection methods.

References :


- [1] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on Outlier/Anomaly Detection in Time Series data," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–33, Apr. 2021, doi: 10.1145/3444690.
- [2] K. Chaudhary, J. Yadav, B. Mallick, and GCET, Greater Noida, INDIA, "A review of Fraud Detection Techniques: Credit Card," May 2012.
- [3] "Credit card fraud detection with a neural-network," *IEEE Conference Publication | IEEE Xplore*, 1994. <https://ieeexplore.ieee.org/abstract/document/323314>
- [4] M. Belgiu and L. Drăguț, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, Apr. 2016, doi: 10.1016/j.isprsjprs.2016.01.011.
- [5] A. Parmar, R. Katariya, and V. Patel, "A review on Random Forest: An ensemble classifier," *Lecture Notes on Data Engineering and Communications Technologies*, pp. 758–763, Dec. 2018, doi: 10.1007/978-3-030-03146-6_86.
- [6] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, "A review of local outlier factor Algorithms for outlier detection in big data streams," *Big Data and Cognitive Computing*, vol. 5, no. 1, p. 1, Dec. 2020, doi: 10.3390/bdcc5010001.
- [7] "The research of the fast SVM classifier method," *IEEE Conference Publication | IEEE Xplore*, Dec. 01, 2015. <https://ieeexplore.ieee.org/abstract/document/7493959/>
- [8] "Unsupervised anomaly detection in time series using LSTM-Based autoencoders," *IEEE Conference Publication | IEEE Xplore*, Dec. 2019, [Online].

Notebooks and Datasets :

Initial report :  Report_Deakin

Results :  Results_Anomaly ,:  Results_Autoencoders

Datasets Used :  Datasets

Notebooks :  Anamoly Detection.ipynb

<https://drive.google.com/file/d/1oqWs3GCACQHmdj2r704NglAa9mLTxvT2/view?usp=sharing>

