International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Real-time Anomaly Detection in Big Data Streams: Machine Learning Approaches and Performance Evaluation

Aruna Bajpai[1], Samiksha Khule[2], Vijay Prakash Sharma[3], Yogeshkumar Sharma[4], Gaurav Dubey[5]

Submitted: 03/02/2024 **Revised**: 11/03/2024 **Accepted**: 17/03/2024

Abstract: The paper focuses on real-time anomaly detection in big data streams, discussing the machine learning techniques as well as performance evaluation. Different outlier detection methods, including Isolation Forest, Local Outlier Factor, Support Vector Machine and Elliptic Envelope, are considered and analyzed relying on a dataset of financial transactions. The analysis is made up of the visualization of data distributions, correlation matrices and metrics like accuracy and classification reports. The findings reveal varying performance levels among the methods pointing out the significance of choosing appropriate techniques for effective anomaly detection in dynamic data environments. This paper adds to the knowledge of outlier detection in big data streams and provides valuable insights for future studies.

Keywords: *Isolation Forest, Local Outlier Factor, Support Vector Machine, Elliptic Envelope, Anomaly detection, big data, Machine learning, Outlier detection methods, Performance evaluation*

1. Introduction

Big data analytics research is inadequate without real-time inconsistency location, which remains a major impediment. Modern state-of-the-art machine learning strategies to productively and on the spot identify undesirable patterns in enormous, ever-flowing data. The consolidating of expanding amounts of information with the requirement for quick understanding has incited investigation into machine learning strategies that empower this errand. This investigates the scene of real-time inconsistency discovery in expansive data streams. Moreover, investigates a wide extent of machine learning methods, i.e. directed, unsupervised and semi-supervised models. The viability of these strategies is completely assessed through a system execution assessment that incorporates accuracy, precision and computational productivity, among others.

Aim and Objective

Aim

This study intends to investigate and evaluate the machine learning methods for real-time anomaly detection of data streams.

Objective

The main purpose of this study includes exploring different types of supervised, unsupervised, as well as semi-supervised techniques. This assesses their performance metrics and determines their adaptability to dynamic data environments.

2. Related Works

1Clustering-based Real-time Anomaly Detection in Big Data Technologies

AriyaluranHabeeb et al. (2022) significantly improves big data technology. The study presents a one-of-a-kind strategy that employments clustering strategies to rapidly recognize unusual designs in data. The proposed strategy bunches data focuses based on likeness, which permits the detection of exceptions and deviations from the standard. Through viable tests, the authors illustrate the viability of their approach and its capacity to distinguish outliers in a few real data sets. This work gives a new and interesting point of view to the continuous debate approximately real-time anomaly detection utilizing clustering strategies to extend the productivity and accuracy of anomaly detection in big data situations.

[1]*Assistant professor, Computer Science & Engineering Department Institute of Technology & Management, Gwalior, Madhya Pradesh aruna.bajpai@itmgoi.in, arunabajpai20@gmail.com,*
[2]*Assistant Professor, Computer Science & Engineering Department Institute of Technology & Management, Gwalior, Madhya Pradesh*
[3]*Assistant Professor, Computer Science & Engineering Department Institute of Technology & Management, Gwalior, Madhya Pradesh*
[4]*Assistant Professor, Computer Science & Engineering Department Institute of Technology & Management, Gwalior, Madhya Pradesh*
[5]*Assistant Professor, Computer Science & Engineering Department Institute of Technology & Management, Gwalior, Madhya Pradesh*

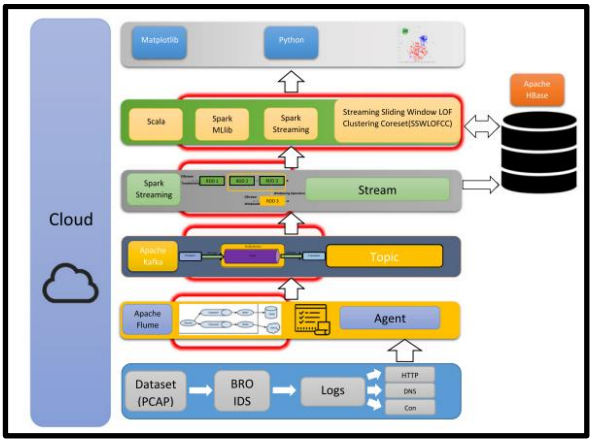


Fig 1: Framework for real-time anomaly detection

These huge assessments highlight the various strategies and methods used to recognize abnormalities in gigantic data streams ceaselessly [1]. From sweeping concentrates that give a total framework of existing ways of managing creative strategies that push the limits of standard anomaly detection methods, the composing offers an overflow of encounters and advances. Expanding on the foundations spread out by these assessments, researchers continue to search for new ways and methods to address the troubles of steady anomaly detection in the time of big data. presents an imaginative gathering-based procedure for consistent anomaly detection in big data technologies. Using clustering algorithms, the philosophy recognizes particular models in a data stream constantly. Gathering data shines considering similarity allows the detection of peculiarities and deviations from conventional approaches to acting. This imaginative strategy is committed to chipping away at the proficiency and exactness of anomaly detection in strong and high-volume data conditions ordinary in big data technologies.

2. Online Anomaly Detection over Big Data Streams

The investigation of Rettig et al. (2019) on online anomaly detection over big data floods of information uncovers significant data and important illustrations for undertakings brought into the world from huge information.The fundamental objective of the venture is to fabricate algorithms that can recognize anomalies in real-time, move to data stream and give real-time realities to chiefs [2]. This strategy utilizes both versatile algorithms and machine learning, extraordinarily intended for online handling, to identify anomalies at a beginning phase, even in exceptionally factor data streams. The review features the job of data proficiency and adaptability in overseeing gigantic measures of data in the present data organizations and exhibits the requirement for lightweight and adaptable anomaly detection techniques.

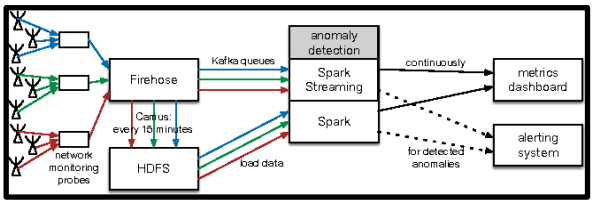


Fig 2: Online Anomaly Detection

The obstacles inherent in the active nature of large-scale data streams such as the usual bunch processing methods often fail due to the timing and mobility requirements, they have been those presented by Rettig et al. (2019) [3]. A new set of methods is introduced in this paper. These methods permit continuous observation and identification of deviations as information streams, which in turn makes it possible to require preventative steps and moderate potential threats in genuine minutes. The authors' methodology appears to be successful in an assortment of areas, such as the healthcare industry, financial administration, and cybersecurity, through the utilization of experimental evaluations and case studies.

3. Methods and Materials

1. Data Collection and Preprocessing

The validity and reusability of the credit card transaction dataset are ensured by a rigorous methodology applied in the “Data Collection and Preprocessing” phase of the review. Data collection begins with gathering the raw transactional data from credible sources adhering to ethical norms and complying with data security laws [4]. The data set also involves very many varying tradeoffs of the actual and counterfeit trades which will therefore create wide research in techniques for detection of disparities. The raw dataset is thoroughly preprocessed after collection to be ready for the subsequent analysis.

The preprocessing stages contain several primary systems for cleaning, refining and the quality and utility of bifurcating the data to process it. The underpinning stage encompasses the defining proof and elimination of duplicate trails within the dataset, relieving literal repetitiveness and boosting computational efficiency. After the disposal of duplicates, the dataset is subjected to a rigorous data-cleaning methodology to deal with any inconsistencies, errors, or missing values which may distort the analysis.

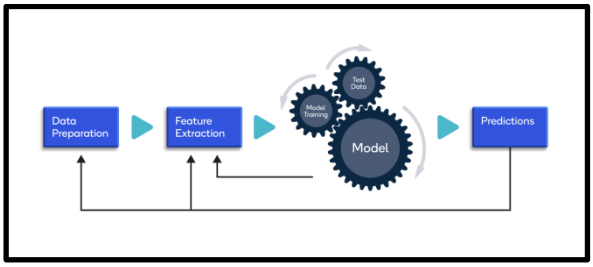


Fig 3: Data Collection and Preprocessing Process

This comprises looking at every quality or element within the dataset and utilizing proper methods to deal with missing or wrong data points. Some attribution methods, e.g. mean, middle or mode replacement can be used to replace the missing values, warranting coherence and completeness in the dataset. Consider feature delineation originates as one significant point in the preprocessing stage, comprising the detection and extraction of those attributes, mainly responsible for the problem of classification [5]. It covers the process of getting experiences through EDA on the loops, correlations, and significances of different features within the dataset.

Highlights considered insignificant or repetitive are pruned from the dataset, smoothing out the investigation and reducing the computational over. All through the information preprocessing organized, extreme quality control measures are executed to preserve the reliability, accuracy, and consistency of the dataset. Thorough endorsement checks are coordinated at each move toward recognizing and changing any anomalies, irregularities, or data uprightness that might think twice about the authenticity of the examination [6]. The dataset is refined into spotless, organized, and standardized, prepared for total examination and irregularity identification by staying to generous preprocessing traditions.

2. Outlier Detection Methods

In this section, various outlier detection methods are employed to identify fraudulent transactions within the dataset. Each method is described below along with its corresponding pseudocode;

Isolation Forest Pseudocode

Isolation Forest is an unsupervised outlier detection 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 [7]. It is effective in isolating anomalies in sparse high-dimensional spaces which make it suitable for detecting outliers in large datasets.

"1. function IsolationForest(X, n_estimators, max_samples, contamination):
2. Initialize a list of isolation trees T = []
3. for i in range(n_estimators):
4. Sample a random subset S from X with size max_samples
5. Build an isolation tree T_i using the samples in S
6. Add T_i to the list of trees T
7. end for
8. Initialize scores array scores = zeros(len(X))
9. for each isolation tree T_i in T:
10. Compute the anomaly score for each data point in X using T_i
11. Update the scores array by adding the anomaly scores
12. end for
13. Normalize the scores by dividing by the number of trees
14. Identify outliers by comparing the normalized scores to the contamination threshold
15. Return a binary array indicating whether each data point is an outlier or not
16. end function
"

Local Outlier Factor (LOF) Pseudocode

Local Outlier Factor (LOF) is a local density-based approach to outlier detection which measures the density discrepancy between a data point and its neighbors [8]. It figures outliers by checking the density of data points in their local neighborhood to the density of their neighbors, so enabling the detection of anomalies in regions of variable densities.

"1. function LocalOutlierFactor(X, n_neighbors, algorithm, leaf_size, contamination):
2. Initialize a list of LOF scores for each data point
3. for each data point x_i in X:
4. Find the n_neighbors nearest neighbors of x_i using algorithm and leaf_size
5. Compute the local reachability density (lrd) of x_i based on its neighbors
6. Compute the local outlier factor (LOF) of x_i based on lrd values of its neighbors
7. Add the LOF score of x_i to the list of

LOF scores
8. end for
9. Normalize the LOF scores to range between 0 and 1
10. Identify outliers by comparing the normalized LOF scores to the contamination threshold
11. Return a binary array indicating whether each data point is an outlier or not
12. end function
"

Support Vector Machine (SVM) Pseudocode

Support Vector Machine is a supervised outlier detection method which differentiates data points in separate categories using a hyperplane [9]. In SVM, outlier detection means finding data points situated most distant from the separating hyperplane; such points are classed as outliers.

"1. function OneClassSVM(X, kernel, nu, gamma):
2. Initialize the SVM model with parameters kernel, nu, and gamma
3. Fit the SVM model on the input data X
4. Predict the labels for the input data X
5. Return the predicted labels, where 1 indicates an inlier and -1 indicates an outlier
6. end function

3. Evaluation Metrics and Analysis

The different measurements within the "Evaluation Metrics and Analysis" section are utilized to assess the execution of exception location strategies against the capacity to distinguish false exchanges [11]. These measurements deliver experiences into the effectiveness and constancy of the strategies in recognizing the inconsistencies inside the dataset. Key evaluation measurements incorporate;

Outlier Detection Method	Type	Key Characteristics
Isolation Forest	Unsupervised	Randomly selects features and split values.
		Isolates anomalies in sparse high-dimensional spaces.
		Suitable for detecting outliers in large datasets.
Local Outlier Factor (LOF)	Density-based	Measures local density deviation of data points.

"

Elliptic Envelope Pseudocode

The Elliptic Envelope algorithm fits an ellipse to the central data points, assuming that the majority of the data points are from the same Gaussian distribution [10]. Outliers are identified as data points that fall outside the elliptic envelope, making this method effective for detecting outliers in multivariate data with a Gaussian distribution.

"1. function EllipticEnvelope(X, contamination):
2. Initialize the Elliptic Envelope model with the contamination parameter
3. Fit the Elliptic Envelope model on the input data X
4. Compute the Mahalanobis distances of each data point in X
5. Identify outliers based on the Mahalanobis distances and the contamination threshold
6. Return a binary array indicating whether each data point is an outlier or not
7. end function"

		Compares density to neighbors for outlier detection.
		Effective in identifying anomalies in varying densities.
Support Vector Machine (SVM)	Supervised	Separates data points using a hyperplane.
		Identifies outliers farthest from the separating hyperplane.
		Can handle high-dimensional data.
Elliptic Envelope	Parametric	Fits an ellipse to central data points.
		Assumes the majority of data follows Gaussian distribution.
		Effective for multivariate data with Gaussian distribution.

Accuracy

Accuracy may be a principal metric that measures the generally rightness of the outlier detection demonstrates [12]. It is calculated as the proportion of accurately recognized exceptions to the whole number of information focuses. The accuracy condition is communicated as

Accuracy= (True Positives + True Negatives)/Total Data Points

Precision

Precision measures the proportion of correctly identified outliers among all data points labeled as outliers by the model [13]. It focuses on the correctness of positive predictions. Precision is calculated as

Precision = True Positives / (True Positives + False Positives

Recall (Sensitivity)

Recall, also known as sensitivity, quantifies the ability of the model to correctly identify all actual outliers in the dataset. It measures the completeness of positive predictions. Recall is computed as

Recall = True Positives = (True Positives + False Negatives)

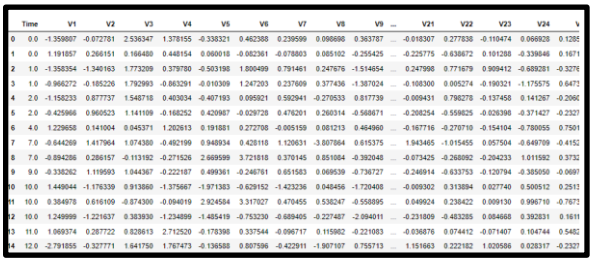
F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balanced assessment of the model's performance [14]. It combines precision and recall into a single metric, making it suitable for evaluating models with imbalanced classes. The F1 score is given by;

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

4. Experiments

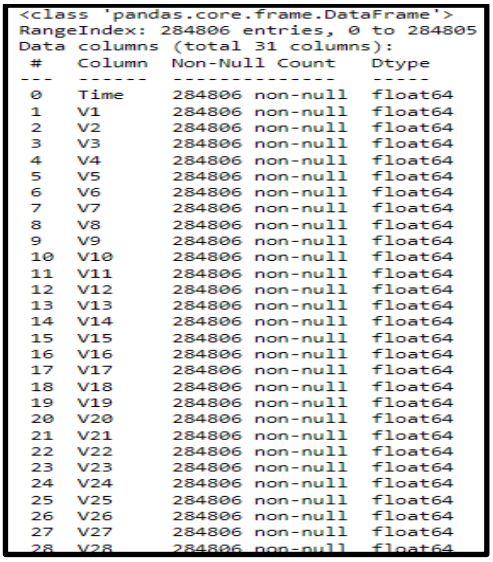
Results and Analysis



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.011592	0.373
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.385050	-0.069
10	10.0	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	...	-0.009302	0.313894	0.027740	0.500512	0.251
11	10.0	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	...	0.049924	0.238422	0.009130	0.996710	-0.767
12	10.0	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-2.094011	...	-0.231809	-0.483285	0.084668	0.392831	0.161
13	11.0	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	...	-0.036876	0.074412	-0.071407	0.104744	0.548
14	12.0	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	...	1.151663	0.222182	1.020586	0.028317	-0.232

Fig 4: Display Dataset

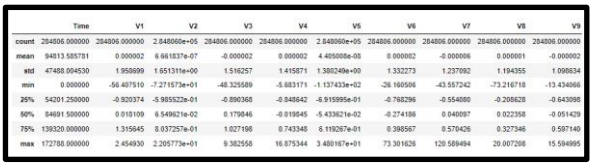
The above figure appears to show containing rows and columns of numbers. There are 14 columns and 11 rows of data. The far left and top rows contain labels for the data.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284806 entries, 0 to 284805
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284806 non-null  float64
 1   V1      284806 non-null  float64
 2   V2      284806 non-null  float64
 3   V3      284806 non-null  float64
 4   V4      284806 non-null  float64
 5   V5      284806 non-null  float64
 6   V6      284806 non-null  float64
 7   V7      284806 non-null  float64
 8   V8      284806 non-null  float64
 9   V9      284806 non-null  float64
 10  V10     284806 non-null  float64
 11  V11     284806 non-null  float64
 12  V12     284806 non-null  float64
 13  V13     284806 non-null  float64
 14  V14     284806 non-null  float64
 15  V15     284806 non-null  float64
 16  V16     284806 non-null  float64
 17  V17     284806 non-null  float64
 18  V18     284806 non-null  float64
 19  V19     284806 non-null  float64
 20  V20     284806 non-null  float64
 21  V21     284806 non-null  float64
 22  V22     284806 non-null  float64
 23  V23     284806 non-null  float64
 24  V24     284806 non-null  float64
 25  V25     284806 non-null  float64
 26  V26     284806 non-null  float64
 27  V27     284806 non-null  float64
 28  V28     284806 non-null  float64
```

Fig 5: Dataset Information

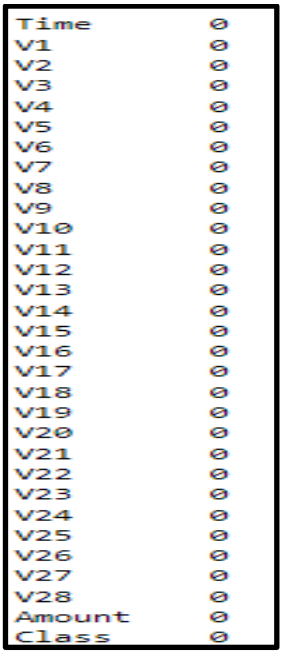
A table summarizing a dataset by the function of "data.info ()" which is shown in the above figure. It shows 284,806 entries with 31 features (columns) such as V1, V2, V3 and so on.



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284806.000000	284806.000000	2.848060e+05	284806.000000	284806.000000	2.848060e+05	284806.000000	284806.000000	284806.000000	284806.000000
mean	94813.585781	0.000002	6.661837e-07	-0.000002	0.000002	4.405008e-08	0.000002	-0.000006	0.000001	-0.000007
std	47488.004530	1.958699	1.651311e+00	1.516257	1.415871	1.380249e+00	1.332273	1.237092	1.194355	1.098634
min	0.000000	-56.407510	-7.271573e+01	-48.325589	-5.683171	-1.137433e+02	-26.160506	-43.557242	-73.216718	-13.434066
25%	54201.250000	-0.920374	-5.985522e-01	-0.890368	-0.848642	-6.915995e-01	-0.768296	-0.554080	-0.208628	-0.643098
50%	84691.500000	0.018109	6.549621e-02	0.179846	-0.019845	-5.433621e-02	-0.274186	0.040097	0.022358	-0.051429
75%	139320.000000	1.315645	8.037257e-01	1.027198	0.743348	6.119267e-01	0.398567	0.570426	0.327346	0.597140
max	172788.000000	2.454930	2.205773e+01	9.382558	16.875344	3.480167e+01	73.301626	120.589494	20.007208	15.594995

Fig 6: Dataset Description

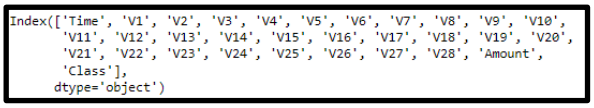
The table shows descriptive statistics by the function of "data.describe" for a dataset with 284,806 entries and 31 features which are shown in the above figure.



```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
```

Fig 7: Null value check

This figure shows the checks for missing values in a dataset named "data" and show that there are no missing values in the dataset.



```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

Fig 8: Column Name

This figure shows the column name which is present in the dataset called "data".

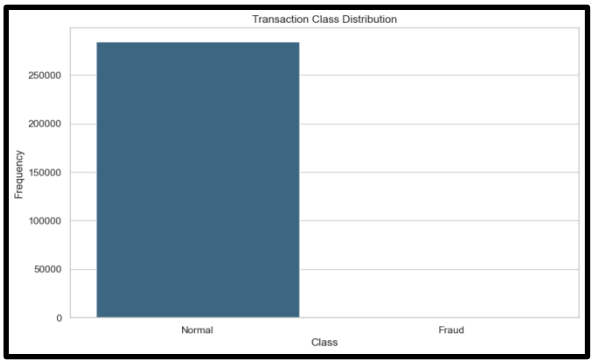


Fig 9: Transaction Class Distribution

The graph shows the distribution of transaction classes, with many more normal transactions than fraudulent ones which is shown in the above figure.

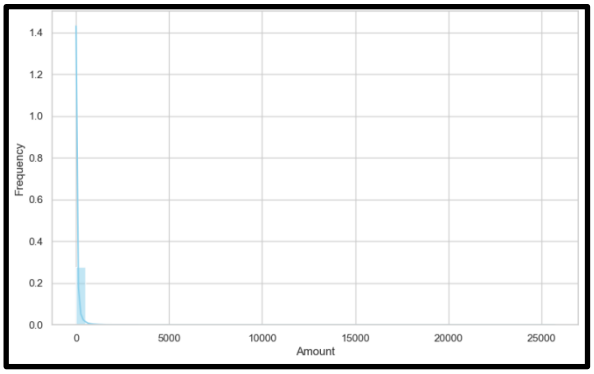


Fig 10: Histogram of Transaction Amounts

The figure shows a histogram of transaction amounts, with most transactions being between $0 and $5,000. There are fewer transactions as the amount increases where the x-axis contains the amount and the y-axis contains the frequency.

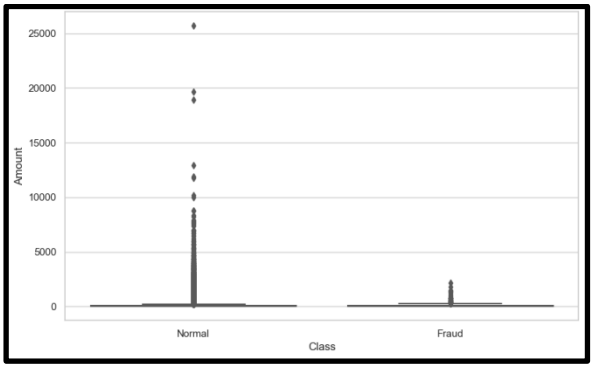


Fig 11: Transaction Amounts by Class

The boxplot shows transaction amounts are higher in the "Fraud" class than in the "Normal" class where the x-axis represent the class and the y-axis represents the amount.

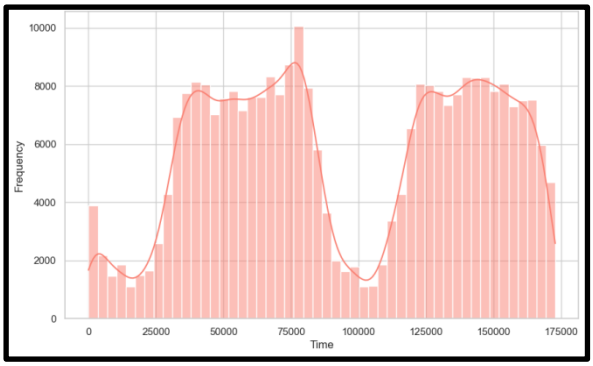


Fig 12: Time Distribution of Transactions

The graph shows a peak in transaction frequency around 4 PM, with fewer transactions earlier and later in the day.

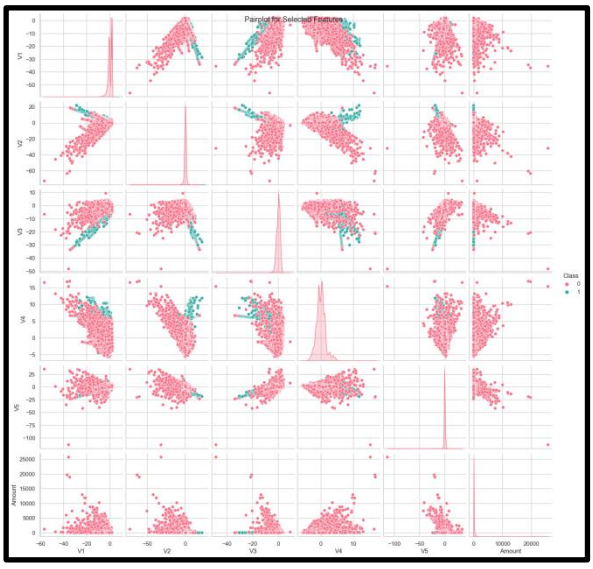


Fig 13: Pair plot for Features

The pair plot visualizes relationships between several features, with "Amount" colored by class.

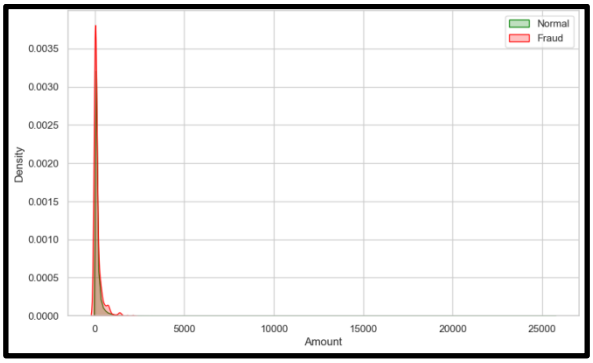


Fig 14: Kernel Density Estimation (KDE) Plot of Transaction Amounts by Class

The graph shows the probability density of transaction amounts by class, with more normal transactions than fraudulent ones. Here the x-axis reflects the amount and the y-axis reflects the Density.

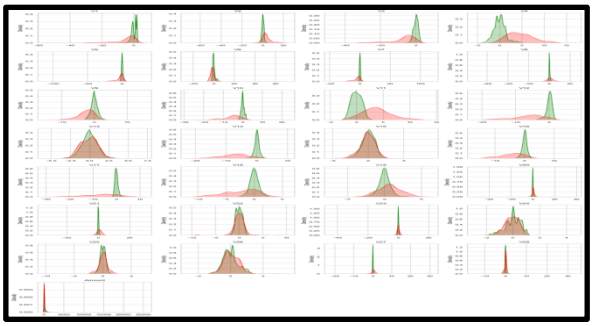


Fig 15: Distribution of Features for Normal vs. Fraudulent Transactions

The above figure shows the distribution of features dataset with 284,806 entries and 14 features, possibly representing financial transactions.

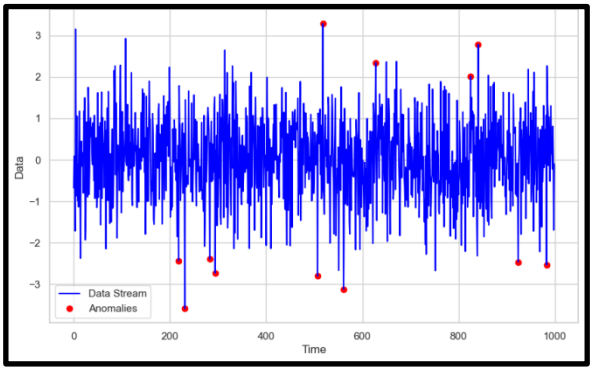


Fig 16: Real-time Anomaly Detection

A real-time anomaly detection graph. It shows the number of anomaly events detected over time, with a horizontal axis representing time and a vertical axis representing the number of anomalies. The graph also includes a dashed line that seems to represent a threshold for anomaly detection.

Anomalies are likely flagged when the number of events exceeds the threshold.

```
Statistics for Normal Transactions:
count    284314.000000
mean         88.290570
std         250.105416
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Fig 17: Statistics for Normal Transactions

According to the above figure shows descriptive statistics for transaction amounts, with most transactions under $100 and a few very large ones.

```
Statistics for Fraudulent Transactions:
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Fig 18: Statistics for Fraudulent Transactions

The above figure shows summary statistics for fraudulent transactions, with an average amount of $122 and a maximum of $2125.87.

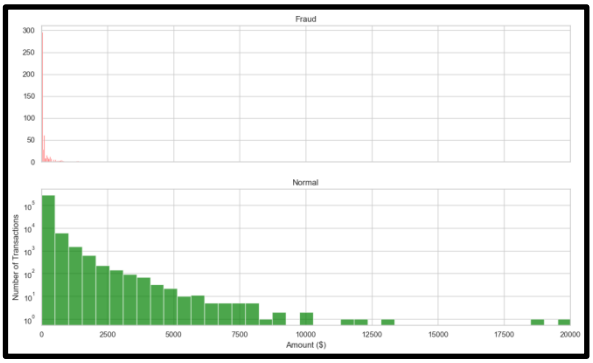


Fig 19: Amount per transaction by class

The above graph shows the average transaction amount by class, with fraud transactions much higher than normal transactions.

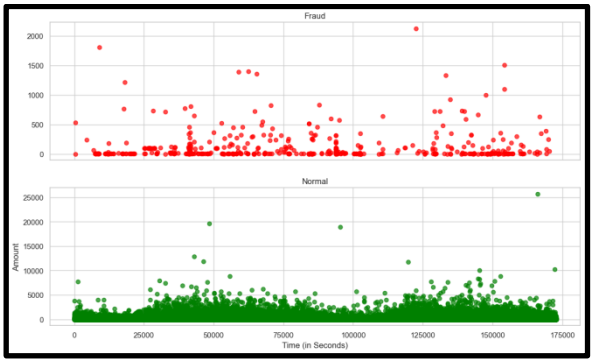


Fig 20: Time of transaction vs Amount by class

This graph shows the distribution of transaction amounts, with fraudulent transactions being more likely for higher amounts.

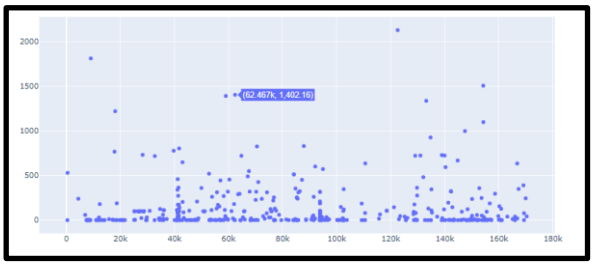


Fig 21: Scatter plot of trace

The scatter plot shows a positive correlation between amount and time , with higher income associated with older amount which is shown in the above figure

```
0.0016529506928325245
Fraud Cases : 47
Valid Cases : 28434
```

Fig 22: Outlier fraction and no of Fraud and Valid Transaction cases

The above figure shows the outlier fraction and counts fraud and valid transactions in a dataset. Here outlier fraction is 0.00165, the count of fraud cases is 47 and the valid case is 28434.

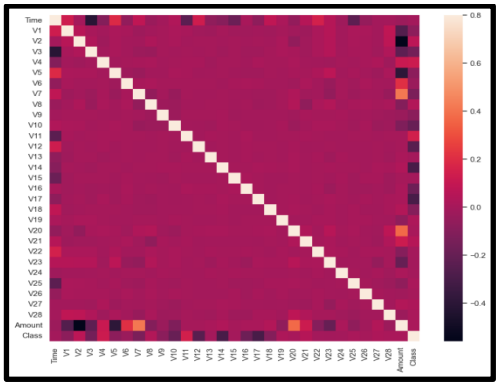


Fig 23: Correlation Matrix

This figure shows the correlation matrix of the outlier function and transactions for anomaly detection in the big data stream.

```
Isolation Forest: 274
Accuracy Score:
0.9903795512798006
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     28434
           1       0.10      0.62      0.17        47

    accuracy                           0.99     28481
   macro avg       0.55      0.80      0.58     28481
weighted avg       1.00      0.99      0.99     28481

Local Outlier Factor: 328
```

Fig 24: Accuracy of Isolation Forest

This figure shows the accuracy of the isolation forest scoring 274 and the accuracy score is 99% approximately.

```
Local Outlier Factor: 328
Accuracy Score:
0.9884835504371335
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     28434
           1       0.01      0.04      0.01        47

    accuracy                           0.99     28481
   macro avg       0.50      0.52      0.50     28481
weighted avg       1.00      0.99      0.99     28481
```

Fig 25: Accuracy of Local Outlier Factor

This figure shows the metrics detail of the LOF and the accuracy score of the LOF is 98% approximately.

```
Support Vector Machine: 8411
Accuracy Score:
0.7046803131912504
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.71      0.83     28434
           1       0.00      0.34      0.00        47

    accuracy                           0.70     28481
   macro avg       0.50      0.52      0.42     28481
weighted avg       1.00      0.70      0.83     28481
```

Fig 26: Accuracy of SVM

This figure shows the accuracy of the SVM model and the accuracy score is 70% approximately.

```
Elliptic Envelope: 278
Accuracy Score:
0.9902391067729364
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     28434
           1       0.09      0.57      0.16        47

    accuracy                           0.99     28481
   macro avg       0.55      0.78      0.58     28481
weighted avg       1.00      0.99      0.99     28481
```

Fig 27: Accuracy of Elliptic Envelope

This figure shows the accuracy of the elliptic envelope and the whole classification report. Hence the accuracy score of the envelope is 99% in approx.

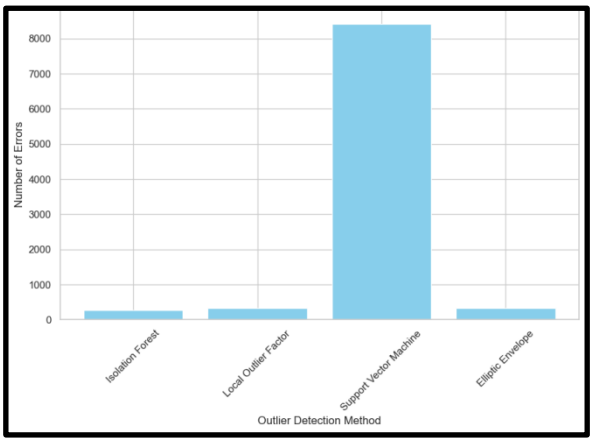


Fig 28: Comparison of Outlier Detection Methods

This figure shows the comparison of the outlier detections based on the number of errors with several types of models used here.

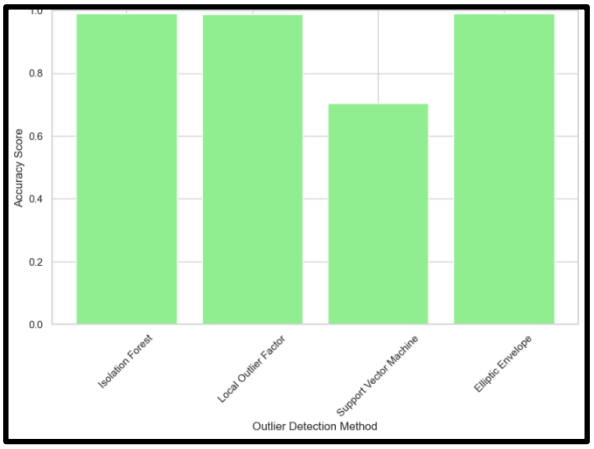


Fig 29: Comparison of Outlier Detection Methods - Accuracy

This figure shows the outlier decision comparisons in terms of accuracy with the bar plot based on the different types of models used here.

Comparison to Related Work

Correlation with related works features significant similarities and contrasts in methodology and execution. Although past studies might have utilized comparable outlier detection methods such as isolation forest and local outlier, contrasts in dataset characteristics and preprocessing methods might prompt various outcomes [15]. Moreover, propels in ML algorithms and information handling methods can further develop precision contrasted with past exploration. Generally, the correlation features the significance of considering setting explicit factors and

advancing systems while assessing and deciphering the consequences of anomaly detection studies.

5. Conclusion and Discussion

The exhaustive analysis and assessment of anomaly detection methods with regard to real-time anomaly detection in enormous data streams gives important data about their viability and productivity. This correlation of various methodologies, including isolation forests, local bias, support vector machine, and elliptic envelope, features their assets and impediments in identifying oddities in financial transaction data. While certain methods show high exactness and power, others might battle with specific sorts of irregularities or have lower execution. The conversation features the significance of picking proper anomaly detection methods in light of the particular characteristics of the dataset and the ideal trade-offs between precision, computational intricacy and interpretability. Future examination could investigate mixture draws near or new techniques to additionally further develop anomaly detection in unique and high-layered data conditions.

References

[1] Habeeb, R.A.A., Nasaruddin, F., Gani, A., Hashem, I.A.T., Ahmed, E. and Imran, M., 2019. Real-time big data processing for anomaly detection: A survey. International Journal of Information Management, 45, pp.289-307.

[2] Rettig, L., Khayati, M., Cudré-Mauroux, P. and Piórkowski, M., 2019. Online anomaly detection over big data streams. Applied Data Science: Lessons Learned for the Data-Driven Business, pp.289-312.

[3] AriyaluranHabeeb, R.A., Nasaruddin, F., Gani, A., Amanullah, M.A., AbakerTargioHashem, I., Ahmed, E. and Imran, M., 2022. Clustering-based real-time anomaly detection A breakthrough in big data technologies. Transactions on Emerging Telecommunications Technologies, 33(8), p.e3647.

[4] Viegas, E., Santin, A., Bessani, A. and Neves, N., 2019. BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. Future Generation Computer Systems, 93, pp.473-485.

[5] Al-amri, R., Murugesan, R.K., Man, M., Abdulateef, A.F., Al-Sharafi, M.A. and Alkahtani, A.A., 2021. A review of machine learning and deep learning techniques for anomaly detection in IoT data. Applied Sciences, 11(12), p.5320.

[6] Ahmed, C.M., MR, G.R. and Mathur, A.P., 2020, October. Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. In Proceedings of the 6th ACM on cyber-physical system security workshop (pp. 23-29).

[7] Schmidl, S., Wenig, P. and Papenbrock, T., 2022. Anomaly detection in time series: a comprehensive evaluation. Proceedings of the VLDB Endowment, 15(9), pp.1779-1797.

[8] Ahmed, A., Sajan, K.S., Srivastava, A. and Wu, Y., 2021. Anomaly detection, localization and classification using drifting synchrophasor data streams. IEEE Transactions on Smart Grid, 12(4), pp.3570-3580.

[9] Ahmed, A., Sajan, K.S., Srivastava, A. and Wu, Y., 2021. Anomaly detection, localization and classification using drifting synchrophasor data streams. IEEE Transactions on Smart Grid, 12(4), pp.3570-3580.

[10] Ullah, W., Ullah, A., Hussain, T., Muhammad, K., Heidari, A.A., Del Ser, J., Baik, S.W. and De Albuquerque, V.H.C., 2022. Artificial Intelligence of Things-assisted two-stream neural network for anomaly detection in surveillance Big Video Data. Future Generation Computer Systems, 129, pp.286-297.

[11] Rezaee, K., Rezakhani, S.M., Khosravi, M.R. and Moghimi, M.K., 2021. A survey on deep learning-based real-time crowd anomaly detection for secure distributed video surveillance. Personal and Ubiquitous Computing, pp.1-17.

[12] Mokhtari, S., Abbaspour, A., Yen, K.K. and Sargolzaei, A., 2021. A machine learning approach for anomaly detection in industrial control systems based on measurement data. Electronics, 10(4), p.407.

[13] Nassif, A.B., Talib, M.A., Nasir, Q. and Dakalbab, F.M., 2021. Machine learning for anomaly detection: A systematic review. Ieee Access, 9, pp.78658-78700.

[14] Dong, Y., Wang, R. and He, J., 2019, October. Real-time network intrusion detection system based on deep learning. In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS) (pp. 1-4). IEEE.

[15] Granat, J., Batalla, J.M., Mavromoustakis, C.X. and Mastorakis, G., 2019. Big data analytics for event detection in the IoT-multicriteria approach. IEEE Internet of Things Journal, 7(5), pp.4418-4430.