Adaptive Data Enrichment Pre-Processing System (Adeps) For Duplicate Detection, Outlier Handling, Imputation, And Encoding

R. Nisha[1], G.Dalin[2]

[1]Research Scholar, Hindusthan College of Arts and Science, Coimbatore, Tamilnadu, India.
[2]Professor, Hindusthan College of Arts and Science, Coimbatore, Tamilnadu, India.

Received: 09.04.2024 Revised: 12.05.2024 Accepted: 24.05.2024

ABSTRACT
The Adaptive Data Enrichment Pre-processing System (ADEPS) is a comprehensive and flexible framework designed to optimize data quality for analytical and machine learning tasks. ADEPS integrates four critical preprocessing functions: duplicate detection, outlier handling, imputation, and categorical encoding. Each component is developed to address common data quality issues that can adversely affect model accuracy and reliability. ADEPS's duplicate detection uses advanced similarity algorithms to identify redundant entries, ensuring dataset integrity. Outlier handling leverages clustering and normalization techniques to effectively identify and process anomalies. For missing values, enhanced MICE-based imputation fills gaps using adaptive modeling with error terms, while categorical encoding techniques, such as Target Encoding, transform high-cardinality categorical data for machine compatibility. The ADEPS framework enhances model performance by delivering a high-quality, enriched dataset ready for robust analysis and predictive modeling. Its modular design also allows for adjustments based on data type, resource requirements, and analysis needs, making it suitable for a wide range of applications.

Keywords: Machine learning, duplicate detection, outlier handling, imputation, and categorical encoding

1. INTRODUCTION

Insurance fraud detection is vital globally, as fraudulent activities cause billions of dollars in annual losses and significant financial risk. Fraud takes various forms, including staged accidents, identity theft, and fraudulent claims. Effectively detecting and handling these claims not only protects financial stability but also ensures fair premiums for policyholders. Machine learning (ML) and artificial intelligence (AI) techniques have emerged as essential tools for fraud detection, leveraging data insights to recognize trends in transaction histories and claims data.

Importance of Data Mining and Pre-processing

Fraud detection now heavily relies on data mining, which uncovers hidden patterns and anomalies in large datasets. However, for optimal algorithm performance, data pre-processing cleaning and organizing raw data—is essential. Pre-processing mitigates noise, outliers, and missing values, ensuring that algorithms perform with enhanced accuracy. Techniques such as feature scaling, dimensionality reduction, and class balancing (oversampling and under sampling) are crucial to developing reliable, unbiased models capable of detecting fraud.

Key Components of Pre-processing

1. **Data Cleaning and Imputation:** Addresses errors, missing values, and inconsistencies that can mislead models, ensuring clean datasets for reliable training.
2. **Normalization and Scaling:** Standardizes numerical features, crucial for algorithms sensitive to data magnitude, allowing fraud indicators to surface across variable scales.
3. **Handling Imbalanced Data:** Fraud cases are often sparse. Techniques like oversampling, under sampling, and synthetic data generation balance datasets, supporting model precision in spotting fraud.

Developing a Robust Pre-processing Strategy

Effective fraud detection relies on a comprehensive pre-processing strategy, which strengthens data mining efforts and builds accurate models. Each pre-processing step contributes to a well-prepared

dataset, paving the way for robust fraud detection frameworks. This study emphasizes the intricate role of pre-processing in achieving reliable and adaptable fraud detection systems.

2. LITERATURE REVIEW

2.1 Receiver Operating Characteristic - Area Under the Curve (ROC-AUC)

Abd Halim KN et al. (2020) proposed a data pre-processing algorithm for enhancing neural network binary classification in bank telemarketing. Addressing client targeting challenges, the method involves normalization, imbalance handling, and data cleaning using Missing Common and Tomek Links, with MaxAbsScaler or MinMaxScaler for scaling. Tested on diverse datasets, the approach achieved high AUC scores (0.9129 and 0.9464), demonstrating improved classification performance. Future applications in various fields are suggested.

2.2 Automated PRE-Processing for Data Mining (APREP-DM)

H. Nagashima et al. (2019) introduced APREP-DM, an automated data pre-processing framework based on CRISP-DM, designed for sensor data analysis in fields like fault prediction, robot autonomy, and customer behavior. APREP-DM addresses key pre-processing tasks such as handling missing data, formatting, and detecting outliers, enhancing analysis reliability and consistency. In a pedestrian trajectory tracking scenario, APREP-DM outperformed alternative frameworks, underscoring the importance of well-defined success criteria and analytical goals. Future research will focus on further APREP-DM implementations.

2.3 Association Rule Mining (ARM)

S. Kareem et al. (2017) proposed a framework utilizing association rule mining to identify fraudulent health insurance claims, addressing the sector's substantial financial losses due to intentional fraud. Given the high volume and complexity of claims, manual detection is impractical. This research demonstrates that association rules effectively identify attribute relationships within claim documents, reducing inconsistencies and enhancing fraud detection in health insurance. Data mining thus offers a promising approach for combating fraud.

2.4 Tabulated Vector Approach (TVA)

Gutierrez RJ et al. (2018) proposed a cyber anomaly detection framework that uses tabulated vectors and embedded analytics to efficiently analyze high-speed internet traffic in large enterprises. By transforming firewall data into meaningful state vectors and applying multivariate methods, including factor analysis and Mahalanobis distance, the study enhances anomaly detection, enabling efficient, repeatable analysis through open-source tools for network intrusion detection.

2.5 Convolutional Neural Network - Gated Recurrent Unit (CNN-GRU)

Ayub N et al. (2020) proposed an electricity theft detection framework utilizing a Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) optimized with the Manta Ray Foraging Optimization (MRFO) algorithm. The approach addresses data imbalances through the Synthetic Minority Over-sampling Technique (SMOTE) and achieves 91.1% accuracy in handling missing data. It outperforms existing methods, including ARM, CNN-GRU, and logistic regression, while focusing on enhancing real-time theft detection in high-incident datasets.

3. Proposed Methodology

Data Collection

https://www.kaggle.com/datasets/mykeysid10/insurance-claims-fraud-detection. The dataset at the link you provided, **Insurance Claims Fraud Detection** on Kaggle, contains:

- **Number of records**: 1,000 records (claims).
- **Number of attributes**: 39 features (columns) are available in the dataset.

These features represent a variety of information related to each claim, including details about the insured person, the nature of the claim, policy details, and incident specifics.

This dataset is structured in a way that enables you to focus on claims analysis, particularly for fraud detection

The Adaptive Data Enrichment Pre-processing System (ADEPS) is a structured and adaptable framework designed to streamline the data preparation process, targeting four essential pre-processing tasks: duplicate detection, outlier handling, imputation, and encoding. Each component of ADEPS is tailored to enhance data quality and suitability for analysis and machine learning, enabling more accurate and reliable model performance.

The four steps in pre-processing the insurance fraud data:

- **Duplicate Rows**: Algorithms like Levenshtein Distance, Jaccard Similarity, and DBSCAN can help in detecting subtle duplicates.
- **Numerical Features**: Algorithms such as PCA and DBSCAN help in identifying outliers and patterns, while normalization techniques like Z-score or min-max scaling prepare the data for machine learning.
- **Missing Values**: KNN Imputation and MICE are effective at filling in missing values using relationships between features.
- **Categorical Features**: Encoding techniques like One-Hot Encoding, Target Encoding, and Binary Encoding transform categorical variables into numerical forms that can be used in models.

1. Duplicate Rows

Algorithms like Levenshtein Distance can help in detecting subtle duplicates.

Levenshtein Distance

Levenshtein Distance (also known as Edit Distance) is a string metric used to measure the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another. This metric is widely used in various applications, including spell-checking, DNA sequencing, and natural language processing.

Given two strings A and B, the Levenshtein distance $d(A,B)$ is the minimum number of operations needed to transform A into B. The allowed operations are:

1. Insertion of a character.
2. Deletion of a character.
3. Substitution of one character with another.

Let $|A|$ and $|B|$ represent the lengths of strings A and B, respectively.

The Levenshtein distance $d(A,B)$ is calculated using a dynamic programming approach where the distance is computed based on the distances between smaller substrings of A and B.

Recurrence Relation

Let $D[i][j]$ represent the Levenshtein distance between the first i characters of string A and the first j characters of string B. The distance $D[i][j]$ can be computed recursively as follows:

$$D[i][j] = \begin{cases} 0 & if\ i = 0\ and\ j = 0 \\ j & if\ i = 0\ and\ j > 0 \\ i & if\ i > 0\ and\ j = 0 \\ D[i-1][j-1] & if\ A[i] = B[j] \\ 1 + min(D[i-1][j], D[i][j-1], D[i-1][j-1]) & if\ A[i] \neq B[j] \end{cases}$$

Where:

$D[i-1][j]$: Deletion cost

$D[i][j-1]$: Insertion cost

$D[i-1][j-1]$: Substitution cost

Algorithm

Step 1: Initialize a matrix D of size $(|A|+1) \times (|B|+1)$, where $|A|$ and $|B|$ are the lengths of strings A and B respectively. Set the first row and the first column of D such that $D[i][0] = i$ for $i = 0$ to $|A|$ and $D[0][j] = j$ for $j = 0$ to $|B|$.

Step 2: For each character pair $A[i-1]$ and $B[j-1]$, calculate the minimum cost of transforming A into B by considering insertion, deletion, or substitution operations. Update the matrix D based on the recurrence relation $D[i][j] = min\{D[i-1][j] + 1\ (deletion),\ D[i][j-1] + 1)\ (insertion), D[i-1][j-1] + cost\ (substitution))$, where the cost is 0 if $A[i-1] = B[j-1]$, otherwise the cost is 1.

Step 3: The Levenshtein distance between strings A and B will be the value at $D[|A|][|B|]$, representing the minimum number of edits required to transform A into B.

Let's compute the Levenshtein distance between two strings:

$A = "kitten"$

$B = "sitting"$

	0	s	i	t	t	i	n	g
0	0	1	2	3	4	5	6	7
k	1	1	2	3	4	5	6	7
i	2	2	1	2	3	4	5	6
t	3	3	2	1	2	3	4	5
t	4	4	3	2	1	2	3	4
e	5	5	4	3	2	2	3	4
n	6	6	5	4	3	3	2	3

The Levenshtein distance is $D[6][7] = 3$. Therefore, "kitten" can be transformed into "sitting" with 3 edits (substitute 'k' with 's', substitute 'e' with 'i', and insert 'g').
The Levenshtein Distance is a robust and intuitive way to measure the similarity between strings, providing a clear path to transforming one string into another through well-defined operations. Its dynamic programming implementation ensures that it can be computed efficiently, even for relatively large strings.

2. Numerical Features:
Algorithms DBSCAN help in identifying outliers and patterns, while normalization techniques like Z-score or min-max scaling prepare the data for machine learning.

DBSCAN
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular clustering algorithm used primarily for numerical data. Unlike algorithms such as k-means that require specifying the number of clusters in advance, DBSCAN identifies clusters based on the density of data points and can also detect noise (outliers).

1. **Epsilon (ε)**: The maximum distance between two data points for them to be considered as part of the same neighbourhood.
2. **MinPts**: The minimum number of points required to form a dense region (i.e., a cluster).
3. **Core Point**: A point is a core point if it has at least MinPts neighbours within distance ε.
4. **Border Point**: A point that is not a core point but is within the neighbourhood of a core point.
5. **Noise Point**: A point that is neither a core point nor a border point.

DBSCAN Algorithm
Let's denote the dataset as $D = \{x_1, x_2, \dots, x_n\}$, where x_i is a point in the feature space, typically $\mathbb{R}^m$.
Step 1: Initialize the parameters ϵ (epsilon) and MinPts, setting all points in the dataset D as unvisited.
Step 2: Select an arbitrary unvisited point p from the dataset D.
Step 3: For the selected point p, find all points within an ϵ -radius using a distance metric (such as Euclidean distance). This set of points is called the neighborhood $N_{epsilonn}(p)$ and is defined as $N_\epsilon(p) = \{ q \in D \mid d(p,q) \leq \epsilon \}$, where $d(p,q)$ is the distance metric.
Step 4: Check the core point conditions by determining if the size of $N_\epsilon(p)$ is greater than or equal to MinPts. If $|N_\epsilon(p)| \geq$ MinPts, classify p as a core point; otherwise, mark p as noise, with the possibility of later reclassifying it as a border point.
Step 5: If p is identified as a core point, create a new cluster and recursively expand it by including all points in $N_\epsilon(p)$ that are reachable from p and meet the density condition. For each point q in $N_\epsilon(p)$, if q is a core point, add its neighborhood to the cluster; if q is a border point, simply add it to the cluster.

3. Missing Values:
Enhanced MICE are effective at filling in missing values using relationships between features.

Enhncement of MICE for Missing Values
Multiple Imputations by Chained Equations (MICE) is a popular method for handling missing data in datasets. The basic concept is to impute missing values multiple times by generating several complete datasets, then combining the results to account for the uncertainty of missing data. However, to improve the accuracy and robustness of MICE, some enhancements can be applied. These include improving the prediction model for imputation, handling multicollinearity, and improving efficiency in large datasets.

MICE Algorithm

Step 1: Initialize missing values for each feature using a basic imputation method, such as the mean, median, or random sampling, and let the initial complete dataset be $X^{(0)}$.

Step 2: Define an iterative imputation model by selecting an appropriate predictor model for each feature X_j with missing values, based on data type (e.g., linear regression for continuous data, logistic regression for binary data).

Step 3: For iterative imputation using chained equations, for each feature X_j with missing values, denote $X_{(-j)}$ as all other features except X_j. Use the current imputed values of $X_{(-j)}$ to predict missing values in X_j.

Step 4: For each feature X_j, fit the regression model to predict X_j using available data in $X_{(-j)}$ and update imputed values for missing entries in X_j with the model prediction plus an error term ϵ_j to account for variability: $\hat{X}_j^{(t+1)} = f_j(X_{\{-j\}}^{(t)}) + \epsilon_j$.

Step 5: To enhance regularization and prevent overfitting in regression-based models, if necessary, apply a regularization technique to the model, aiming to minimize $\| X_j - X_{-j}\beta_j \|_2^2 + \lambda \| \beta_j \|_2^2)$, where λ the regularization parameter is.

Step 6: Repeat this iterative imputation for each feature X_j with missing values, iterating through all features until the imputed values stabilize, and indicating convergence.

Step 7: Check for convergence, continuing the iteration until the change in imputed values across all features is below a threshold (δ), such that ($||X^{(t+1)} - X^{(t)}|| < \delta$).

Step 8: To account for uncertainty, repeat the imputation process M times to generate multiple complete datasets ($X_1^*, X_2^*, \dots, X_M^*$), with each iteration yielding slightly different imputed values.

Step 9: Pool the results from multiple imputed datasets by applying statistical methods such as Rubin's rules, calculating the pooled estimate $\bar{\theta} = \frac{1}{M}\sum_{m=1}^{M}\hat{\theta}_m$, where $\hat{\theta}_m$ is the parameter estimate from the $m^{t⍰}$ dataset, and compute the total variance $T = W + (1 + \frac{1}{M})B$ where W is within-imputation variance and B is between-imputation variance.

Step 10: Use one of the M imputed datasets or the pooled results for final analysis, providing a completed dataset for subsequent modeling or analysis.

Model-based imputation with tailored models and an error term to account for data-type-specific needs and add variability, regularization to prevent overfitting and enhance stability in model predictions, multiple imputation and pooling to account for uncertainty and provide robust aggregated results, and convergence thresholding for efficient, high-quality imputation.

4. Categorical Features:

Encoding techniques like One-Hot Encoding, Target Encoding, and Binary Encoding transform categorical variables into numerical forms that can be used in models.

Target Encoding for Categorical Features

Target encoding is a technique used to convert categorical features into numerical features by using the relationship between the feature and the target variable. This is particularly useful in cases where the dataset contains categorical variables with many unique levels (high cardinality), which may not be well-handled by one-hot encoding. Target encoding is especially effective for fraud detection, where categorical variables like "Transaction Type," "Location," or "Customer ID" may have specific interactions with the target variable (fraud or no fraud).

1. Target Encoding Overview

In target encoding, the categorical values are replaced with a summary statistic (usually the mean) of the target variable (fraud or not fraud) for each category. For fraud detection, the target variable y would typically be binary (fraud = 1, no fraud = 0).

Equation for Target Encoding

Let X_i be a categorical feature, and y be the target variable. For each category $c \in X_i$, the target encoding is calculated as:

$$TE(c) = \frac{\sum_{\{j \in \{X_i = c\}\}} y_j}{|\{X_i = c\}|}$$

Where:

- $TE(c)$ is the target encoding for category c of feature X_i,

- $\sum_{\{j \in \{X_i=c\}\}} y_j$ is the sum of target values where $X_i = c$,
- $|\{X_i = c\}|$ is the number of occurrences of category c in feature X_i.

2. Smoothing the Encoding

Target encoding can lead to overfitting, especially when categories have few samples. To avoid this, smoothing is applied by incorporating the global mean of the target variable μ.

The smoothed target encoding can be calculated as:

$$TE_{smoot\ \square}(c) = \frac{nc \cdot TE(c) + k \cdot \mu}{n_c + k}$$

Where:

- $TE_{smoot\ \square}(c)$ is the smoothed target encoding for category c,
- n_c is the number of occurrences of category c,
- μ is the global mean of the target variable (i.e., the overall fraud rate),
- k is a smoothing factor that controls the trade-off between the category mean and the global mean.

3. Handling New Categories

For unseen categories during inference, the global mean μ can be used as a fallback. This ensures that the model can still make predictions when new categories are encountered in production.

$$TE_{new} = \mu$$

Algorithm for Target Encoding

Input:

- X: Categorical feature to be encoded
- y: Target variable (fraud or no fraud)
- k: Smoothing parameter

Output:

- Encoded feature $X_{encoded}$

Step 1: Initialize by calculating the global mean of the target variable $\mu = \frac{1}{n}\sum_{i=1}^{n} y_i$, where n is the total number of samples.

Step 2: For each unique category c in the categorical feature X, calculate n_c, the number of occurrences of category c, and compute the target encoding for c using the formula $TE(c) = \frac{\sum_{\{j \in \{X_i=c\}\}} y_j}{n_c}$.

Step 3: Apply smoothing by calculating the smoothed encoding for each category c as $TE_{smoot\ \square}(c) = \frac{n_c \cdot TE(c) + k \cdot \mu}{n_c + k}$, where k is a smoothing parameter to balance category-level and global estimates.

Step 4: Assign encodings by replacing each instance of category c in X with its corresponding smoothed target encoding $TE_{smoot\ \square}(c)$.

Step 5: For any new category c_{new} that appears in the test set but not in the training set, assign the global mean μ as the encoding.

Step 6: Return the encoded feature $X_{encoded}$, where each categorical value is replaced by its corresponding target encoding.

Target encoding is a powerful technique for dealing with categorical variables in fraud detection tasks, especially when the dataset contains high-cardinality features. It captures the relationship between categories and the target variable, improving the model's predictive power. Applying smoothing helps mitigate overfitting, making the encoded values more reliable when category frequency is low.

4. Experimental result

4.1 Accuracy

Accuracy is the degree of closeness between a measurement and its true value. The formula for accuracy is:

$$\boldsymbol{Accuracy} = \frac{(\boldsymbol{true\ value} - \boldsymbol{measured\ value})}{\boldsymbol{true\ value}} * \mathbf{100}$$

Table 1. Comparison Table of Accuracy

Dataset	ARM	TVA	Proposed ADEPS
100	80	81	92
200	73	84	94

300	83	67	96
400	88	86	98
500	95	79	100

The Comparison table of Accuracy demonstrates the different values of existing ARM, TVA and Proposed ADEPS. While comparing the Existing algorithm and Proposed ADEPS, provides the better results. The existing algorithm values start from 73 to 95, 67 to 86 and Proposed ADEPS values starts from 90 to 99. The proposed method provides the great results.

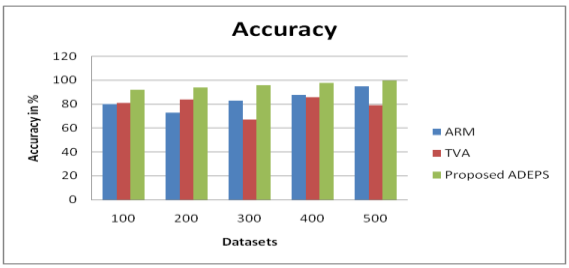

Figure 1. Comparison Chart of Accuracy

The Figure 1 Shows the comparison chart of Accuracy demonstrates the existing ARM, TVA and Proposed ADEPS. X axis denote the Dataset and y axis denotes the Accuracy. The Proposed ADEPS values are better than the existing algorithm. The existing algorithm values start from 73 to 95, 67 to 86 and Proposed ADEPS values starts from 90 to 99. The proposed method provides the great results.

4.2 Precision

Precision is a measure of how well a model can predict a value based on a given input.

$$\boldsymbol{Precision} = \frac{\boldsymbol{true\ positive}}{(\boldsymbol{true\ positive}\ +\ \boldsymbol{false\ positive})}$$

Table 2. Comparison Table of Precision

Dataset	ARM	TVA	Proposed ADEPS
100	82.12	80.37	94.67
200	84.69	92.82	95.26
300	81.62	90.54	96.21
400	76.55	82.63	93.58
500	73.94	77.72	89.87

The Comparison table 2 of Precision demonstrates the different values of existing ARM, TVA and Proposed ADEPS. While comparing the Existing algorithm and Proposed ADEPS, provides the better results. The existing algorithm values start from 73.94 to 84.69, 77.72 to 92.82 and Proposed ADEPS values starts from 89.87 to 96.21. The proposed method provides the great results.

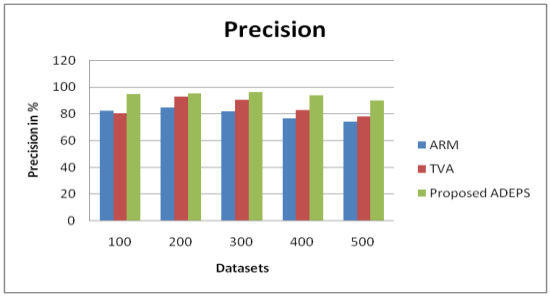

Figure 2. Comparison Chart of Precision

The Figure 2 Shows the comparison chart of Precision demonstrates the existing ARM, TVA and Proposed ADEPS. X axis denote the Dataset and y axis denotes the Precision ratio. The Proposed ADEPS values are better than the existing algorithm. The existing algorithm values start from 73.94 to 84.69, 77.72 to 92.82 and Proposed ADEPS values starts from 89.87 to 96.21. The proposed method provides the great results.

4.3 Recall

Recall is a measure of a model's ability to correctly identify positive examples from the test set:

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positives}}{(\boldsymbol{True\ Positives} + \boldsymbol{False\ Negatives})}$$

Table 3. Comparison Table of Recall

Dataset	ARM	TVA	Proposed ADEPS
100	0.75	0.83	0.88
200	0.77	0.79	0.95
300	0.83	0.68	0.96
400	0.86	0.78	0.97
500	0.87	0.73	0.99

The Comparison table 3 of Recall demonstrates the different values of existing ARM, TVA and Proposed ADEPS. While comparing the Existing algorithm and Proposed ADEPS, provides the better results. The existing algorithm values start from 0.75 to 0.87, 0.68 to 0.83 and Proposed ADEPS values starts from 0.88 to 0.99. The proposed method provides the great results.

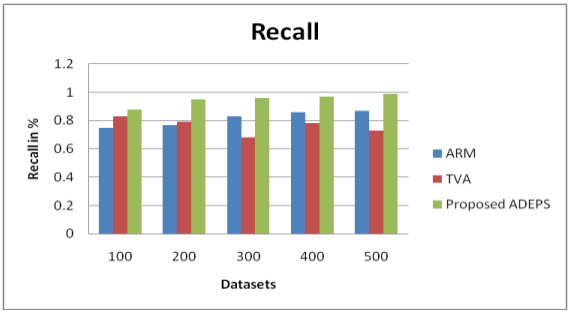

Figure 3. Comparison Chart of Recall

The Figure 3 Shows the comparison chart of Recall demonstrates the existing ARM, TVA and Proposed ADEPS. X axis denote the Dataset and y axis denotes the Recall ratio. The Proposed ADEPS values are better than the existing algorithm. The existing algorithm values start from 0.75 to 0.87, 0.68 to 0.83 and Proposed ADEPS values starts from 0.88 to 0.99. The proposed method provides the great results.

4.4 F -Measure

F1-measure is a test's accuracy that combines precision and recall. It is calculated by taking the harmonic mean of precision and recall.

$$\boldsymbol{F1 - Measure} = \frac{(\boldsymbol{2 * Precision * Recall})}{(\boldsymbol{Precision + Recall})}$$

Table 4. Comparison Table of F -Measure

Dataset	ARM	TVA	Proposed ADEPS
100	0.93	0.84	0.98
200	0.95	0.82	0.99
300	0.88	0.75	0.90
400	0.97	0.73	0.96
500	0.86	0.74	0.94

The Comparison table 4 of F -Measure Values explains the different values of existing ARM, TVA and Proposed ADEPS. While comparing the Existing algorithm and Proposed ADEPS, provides the better results. The existing algorithm values start from 0.86 to 0.97, 0.73 to 0.84 and Proposed ADEPS values starts from 0.90 to 0.99. The proposed method provides the great results.

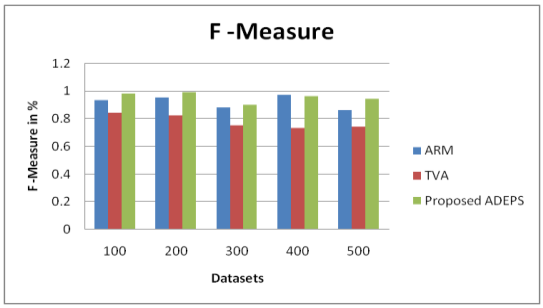

Figure 4. Comparison Chart of F -Measure

The Figure 4 Shows the comparison chart of F -Measure demonstrates the existing ARM, TVA and Proposed ADEPS. X axis denote the Dataset and y axis denotes the F -Measure ratio. The Proposed ADEPS values are better than the existing algorithm. The existing algorithm values start from 0.86 to 0.97, 0.73 to 0.84 and Proposed ADEPS values starts from 0.90 to 0.99. The proposed method provides the great results.

CONCLUSION

The ADEPS framework demonstrates a highly effective approach to data preprocessing, combining adaptability with precision across multiple critical steps: duplicate detection, outlier handling, imputation, and encoding. Through advanced techniques in each module, ADEPS ensures that datasets are cleansed, enriched, and transformed in a way that optimally supports analytical and machine learning outcomes. By using adaptive models and regularization, ADEPS minimizes common issues like overfitting and bias, providing a reliable basis for training models. The modular design not only improves data quality but also offers flexibility, enabling users to adapt the system to various datasets and analytical needs. Ultimately, ADEPS represents a significant advancement in data preprocessing, providing a streamlined solution that addresses common data quality challenges, improves model reliability, and enhances analytical accuracy.

REFERENCES

[1] Abd Halim KN, Jaya AS, Fadzil AF. Data pre-processing algorithm for neural network binary classification model in bank tele-Marketing. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2020; 9:272-7.

[2] C. Hines and A. Youssef, "Class Balancing for Fraud Detection in Point Of Sale Systems," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 4730-4739, doi: 10.1109/BigData47090.2019.9006040.

[3] C. Lawrencia and W. Ce, "Fraud Detection Decision Support System for Indonesian Financial Institution," 2019 International Conference on Information Management and Technology (ICIMTech), Jakarta/Bali, Indonesia, 2019, pp. 389-394, doi: 10.1109/ICIMTech.2019.8843719.

[4] G. F. Monkam, M. J. D. Lucia and N. D. Bastian, "Preprocessing Network Traffic using Topological Data Analysis for Data Poisoning Detection," 2023 IEEE Conference on Dependable and Secure Computing (DSC), Tampa, FL, USA, 2023, pp. 1-8, doi: 10.1109/DSC61021.2023.10354143.

[5] Gutierrez RJ, Bauer KW, Boehmke BC, Saie CM, Bihl TJ. Cyber anomaly detection: Using tabulated vectors and embedded analytics for efficient data mining. Journal of Algorithms & Computational Technology. 2018 Dec; 12(4):293-310.

[6] H. Luo, Y. Zheng, K. Chen and S. Zhao, "Probabilistic Temporal Fusion Transformers for Large-Scale KPI Anomaly Detection," in IEEE Access, doi: 10.1109/ACCESS.2024.3353201.

[7] H. Nagashima and Y. Kato, "APREP-DM: a Framework for Automating the Pre-Processing of a Sensor Data Analysis based on CRISP-DM," 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 2019, pp. 555-560, doi: 10.1109/PERCOMW.2019.8730785.

[8] J. Lu, A. Hales, D. Rew and M. Keech, "Timeline and episode-structured clinical data: Pre-processing for Data Mining and analytics," 2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW), Helsinki, Finland, 2016, pp. 64-67, doi: 10.1109/ICDEW.2016.7495618.

[9] J. Saikam and K. Ch, "EESNN: Hybrid Deep Learning Empowered Spatial-Temporal Features for Network Intrusion Detection System," in IEEE Access, doi: 10.1109/ACCESS.2024.3350197.

[10] J. Vimala Devi and K. S. Kavitha, "Fraud Detection in Credit Card Transactions by using Classification Algorithms," 2017 International Conference on Current Trends in Computer, Electrical, Electronics

and Communication (CTCEEC), Mysore, India, 2017, pp. 125-131, doi: 10.1109/CTCEEC.2017.8455091.

[11] N. Ayub, K. Aurangzeb, M. Awais and U. Ali, "Electricity Theft Detection using CNN-GRU and Manta Ray Foraging Optimization Algorithm," 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318196.

[12] N. Choudhry, J. Abawajy, S. Huda and I. Rao, "A Comprehensive Survey of Machine Learning Methods for Surveillance Videos Anomaly Detection," in IEEE Access, vol. 11, pp. 114680-114713, 2023, doi: 10.1109/ACCESS.2023.3321800.

[13] N. Prabha and S. Manimekalai, "Imbalanced data Classification in Credit Card Fraudulent Activities Detection using Multi-Class Neural Network," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 131-138, doi: 10.1109/ICAIS53314.2022.9742878.

[14] R. Purohit, J. P. Verma, R. Jain and M. Bhavsar, "WePaMaDM-Outlier Detection: Weighted Outlier Detection using Pattern Approaches for Mass Data Mining," 2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT), Gharuan, India, 2023, pp. 1-6, doi: 10.1109/InCACCT57535.2023.10141778.

[15] S. Goyal, S. Rawat and A. G, "Credit Card Fraud Detection using Logistic Regression and Decision Tree," 2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22), Nagpur, India, 2022, pp. 1-5, doi: 10.1109/ICETET-SIP-2254415.2022.9791743.