2024 IEEE International Conference on Data Mining Workshops (ICDMW) | 979-8-3315-3063-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDMW65004.2024.00091

# Anomaly Detection and Interpretation from Tabular Data Using Transformer Architecture

Hajar Homayouni
*San Diego State University*
hhomayouni@sdsu.edu

Hamed Aghayarzadeh
*Colorado State University*
hamedi@colostate.edu

Indrakshi Ray
*Colorado State University*
iray@colostate.edu

Hossein Shirazi
*San Diego State University*
hshirazi@sdsu.edu

***Abstract*—Anomaly detection in tabular data are often needed in most domains, including finance, medicine, and engineering. However, traditional methods typically require intensive processing and often lack interpretability, which limits their practicality. As the complexity and volume of tabular data increase, there is a need for advanced techniques that can handle diverse data types while offering interpretable results. We introduce a novel approach for anomaly detection of tabular data and its interpretation. The anomaly detector employs a transformer model with a custom embedding layer tailored for tabular data. While the model's attention weights do not directly explain its decision-making process, they offer useful insights when interpreted thoughtfully. Our anomaly interpreter uses these attention weights to identify irregularities by comparing patterns in anomalous data against those in normal data. When the model flags a row as anomalous, the interpreter analyzes its columns and assesses their relationships using a reference association matrix created from the normal dataset. Any deviations from expected associations are flagged as potential rule violations, highlighting unusual relationships between columns. We evaluate our approach against baseline anomaly detection techniques, including Multi-Layer Perceptrons (MLPs), Long Short-Term Memory (LSTM) networks, One-Class Support Vector Machines (OC-SVM), and Deep Support Vector Data Description (Deep-SVDD). Our experiment uses labeled datasets from the Outlier Detection DataSets (ODDS), and KDD datasets, assessing performance using standard metrics alongside a novel mutation analysis technique. Results indicate that our approach achieves accuracy comparable to or surpassing existing methods while also offering interpretable explanations for detected anomalies.**

***Index Terms*—Anomaly Detection, Anomaly Interpretation, Explainable AI, Transformers, Mutation Analysis**

## I. Introduction

Tabular data, which ranges from simple CSV files to complex relational databases and multidimensional data warehouses, plays a crucial role in storing and analyzing information across various domains. Data points are often represented as vectors comprising multiple features [1] with varying data types, including numeric, categorical, and textual. Anomaly detection is needed to identify invalid data to facilitate accurate data analysis. Existing anomaly detection techniques, including statistical methods, decision tree-based approaches, and deep learning models, require extensive preprocessing for categorical and textual data, and the imputation of missing values. Preprocessing tasks can consume up to 80% of data science efforts [2], often increasing data dimensionality. Furthermore, most techniques fail to elucidate the underlying reasons for data invalidity and the specific rules violated by the anomalies—-insights that are essential for domain experts to resolve the root causes.

Transformer-based models have shown significant improvements in anomaly detection by effectively analyzing complex data relationships. These models are well-suited for imbalanced classification tasks, as the self-attention mechanism captures subtle patterns in minority classes by focusing on relevant parts of the input. Additionally, pre-trained transformers can be fine-tuned for anomaly detection, making them particularly useful when the minority class is underrepresented. However, transformer models can be computationally expensive, especially when embedding features separately [1], [3]. Some methods reduce preprocessing by converting data rows into sentence-like structures [4]–[6], but challenges can arise from tokenizer splits in numeric values. Moreover, the interpretability of transformer models in anomaly detection is still underexplored. Existing explainability techniques often rely on directly using attention scores to identify important features, although high attention scores do not always indicate key contributions to model decisions [7].

We present a novel approach to anomaly detection and interpretation in regular (i.e., non-sequential) tabular data using a transformer architecture. Our approach minimizes the required data preprocessing and also provides explanations in the form of violated rules by the anomalous data. Our approach consists of the *Anomaly Detector* and the *Anomaly Interpreter*.

*Anomaly Detector* employs a Transformer encoder, followed by a binary classification layer, to effectively distinguish between normal and anomalous instances. The model is trained using labeled tabular data, with advanced features such as a custom tokenizer and a specialized embedding layer to enhance data processing. We provide an optimal preprocessing methodology that converts table rows of any data type into a sentence-like structure. This ensures that column values and their associated names are treated as single tokens, thereby incorporating the rich context provided by the column names into the embeddings and preventing the splitting of values during tokenization. To retain the quantitative aspects of the data, we implement a computationally efficient continuous number encoding method [8], which replaces numeric values in the input sentences with a special token and multiplies the embedding with the actual value, ensuring the retention of numerical information in the transformer model. Our custom tokenizer effectively handles out-of-vocabulary (OOV) tokens, particularly in numerical data, by identifying the closest known

2375-9259/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDMW65004.2024.00091

tokens, thus preventing the classification of unseen or closely valued data points as anomalies.

*Anomaly Interpreter* analyzes the anomalies detected by the *Anomaly Detector* to identify violated rules. While attention weights do not directly explain model decisions [9]–[12], they provide useful insights. Our method compares attention patterns between anomalous and normal data, highlighting specific token associations that warrant further investigation. This process aids in the interpretability of detected anomalies, making it applicable in contexts where the goal is to identify areas of potential concern rather than fully explain the model's reasoning process. We evaluate our model against MLP [13], LSTM [14], OC-SVM [15], and Deep-SVDD [16] using Outlier Detection DataSets (ODDS [17]) and KDD Cup [18]datasets. Accuracy and F1 scores are used to assess the model's anomaly detection performance. We also propose a mutation analysis technique to demonstrate the model's sensitivity and specificity in identifying and interpreting anomalies. Our implementations are publicly available: https://github.com/hajarhomayouni/Anomaly_Detection_and_Interpretation.git.

## II. Related Work

**Anomaly Detection in Tabular Data.** Anomaly detection in tabular data uses statistical methods [19], machine learning techniques [20]–[22], deep learning models [23], and generative models [24]. Techniques such as Isolation Forests [25] and One-Class SVM [26] are simple and effective for identifying anomalies. However, these methods cannot handle complex, high-dimensional data and offer limited interpretability regarding the nature of the detected anomalies. Deep learning approaches, such as Autoencoders [27], Long-Short Term Memory (LSTM) [28], and Deep-SVDD, can capture high-dimensional data complexities, but struggle with interpretability and require lots of data and resources for training. Transformers and Large Language Models (LLMs) have explored transformers for tabular analysis [1], [3], [29]–[31] including anomaly detection [32].

**Data Preprocessing.** Data preprocessing may result in the loss of information and may increase data dimensionality [4]. Transformer-based models, such as FT-Transformer [1] and TabMT [3], embed numerical and categorical values separately by element-wise multiplication of numeric feature values with a corresponding weight vector and use a lookup table for categorical features. These approaches are computationally complex when the number of features is large. Moreover, they do not include column names, which add important context into the embeddings. Transtab [33] extends the above approaches by including the column names into the embeddings, resulting in a more semantically rich data representation. However, it still separately embeds tabular fields to token-level embeddings which is computationally complex. This issue has been resolved in the proposed encoding technique in [4]–[6], which instead of independently embedding every field, it converts a table row into a sentence like "Occupation is doctor, Gender is female, Age is 34," which requires minimal preprocessing and does not suffer from information loss. Note that this lacks the order inherent in natural language sequences. This also does not consider that the numeric values may be split by the LLM's default tokenizers.

**Anomaly Explanation.** While techniques like SHAP (SHapley Additive exPlanations) [34] and LIME (Local Interpretable Model-agnostic Explanations) [35] have been applied to explain model decisions, their applicability in anomaly detection is limited. These methods provide insights into feature importance but may not fully elucidate the relationships between features that contribute to an anomaly. Approaches XAI for All [36] and HuntGPT [37] use LLMs to transform the complex outputs of machine learning models into easily understandable narratives for end users. These methods employ LLMs post-hoc, attempting to interpret the outputs of models. The LLM lacks interaction during model training and has no knowledge about what the model is predicting. Although transformers and LLMs have been used for anomaly detection [38], their explainability is still unexplored. Existing interpretations of transformer models visualize the attention matrix to show which tokens were most 'attended to' by other tokens during the decision-making process. However, a large attention score does not necessarily imply that a pair of words is important for the model's decision [7].

## III. Proposed Approach

### A. Anomaly Detector

Anomaly Detector comprises a Transformer encoder followed by a binary classification layer. This model, trained with labeled tabular data, predicts whether input rows are anomalous or normal. Figure 1 provides an overview of this component, which includes (1) a custom tokenizer, (2) a custom embedding layer, (3) encoder layers, and (4) a classification layer. Our contributions to the original Transformer model [39] are highlighted in the figure and elaborated below.

*1) Custom Tokenizer:* We develop a custom tokenizer that prepares textual representations of tabular data for processing by transformer models. This includes tokenization, vocabulary mapping, and handling out-of-vocabulary items.

*a) Tokenizing:* By converting each row of a table into a sentence formatted as [column: value], the tokenizer ensures that values and their associated column names are treated as single tokens, thus preventing the splitting of column values during tokenization. The following row shows an example of a tokenized row from the breast cancer dataset [17]: Tokens: ['[ClumpThickness:5]', '[UniformityofCellSize:1]', '[UniformityofCellShape:1]', '[MarginalAdhesion:1]', '[SingleEpithelialCellSize:0]', '[BareNuclei:1]', '[BlandChromatin:3]', '[NormalNucleoli:1]', '[Mitoses:1]']

*b) Vocabulary Mapping:* In this step we construct a vocabulary $VOC$ that is the complete set of unique tokens that the model can recognize and process. Parsing through each row, we ensure that each distinct token is assigned a unique identifier, which facilitates the conversion of textual data into a numerical form suitable for analysis by the transformer model.

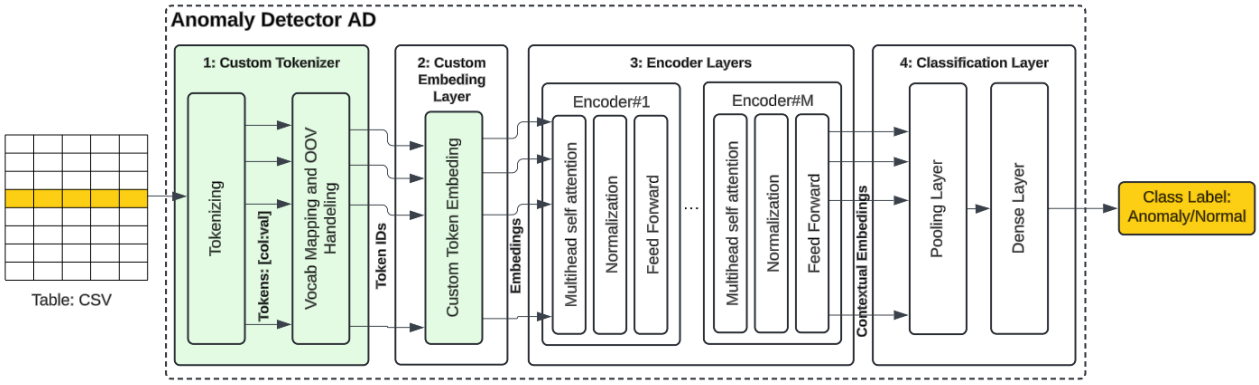


Fig. 1. An Overview of Anomaly Detector Component

*c) Continuous Number Encoding:* To handle numeric values efficiently, we implement a method for continuous number encoding [8]. This involves replacing numeric values in the sentence with a [NUM] token (in the first step in Figure 1) and then multiplying the embedding of the [NUM] token with the actual numeric value during the embedding phase (in the second step in Figure 1). This ensures that numerical information is retained in a form that the transformer model can process effectively. Moreover, by scaling the embedding with the actual numeric value, we ensure that the quantitative aspect of the data is retained.

*d) Out-of-Vocabulary Handling:* The inclusion of out-of-vocabulary (OOV) tokens is crucial for handling values that were not present in the training dataset, ensuring the model can process any input row. For numeric tabular data, a specific method is developed to manage OOV tokens by assessing the "closeness" of numeric values associated with column names. If an OOV token is numeric, its closeness to known tokens is calculated by locating the nearest numeric value for that column in the training data and using that token's ID. A numeric closeness function, which operates based on a threshold value for closeness set based on the dataset, aids in mapping an OOV token to the nearest known token. If no value meets the threshold, the function returns a special OOV token. Suppose token $T_1 = [\text{height} : 120]$ exists in training data, and we encounter token $T_2 = [\text{height} : 120.001]$ in testing data, then the function finds the closest number to this value, checks their difference against a threshold value (e.g., 0.5), and if the difference is less than the threshold ($|120 - 120.001| < 0.5$), then $T_1$'s token ID is assigned to $T_2$.

*2) Custom Embedding Layer:* This layer converts each token from its tokenized form (i.e., discrete IDs) into a vector of fixed size (i.e., continuous vector representations). Unlike natural language, where the order of tokens matters, there is no preferred order of tokens for tabular columns. Therefore, we excluded position embedding from our custom embedding layer implementation, unlike the standard transformer embedding layer. The self-attention mechanism in Transformers does not inherently consider the order of tokens.

*a) Custom Token Embedding:* We develop a custom token embedding layer that initializes the embedding lookup table based on our vocabulary size and the embedding dimension. For the model architecture we used, the embedding dimension is predefined to be 768. We created a tensor of custom embeddings, containing randomly initialized weights.

*3) Encoder Layers:* Encoder layers aim to develop a deep understanding of context, semantics, and associations between multiple columns. We used a standard implementation of encoder transformer layers. Each layer processes input embeddings to build increasingly complex representations of these associations. The encoder layer refines the representation of the input row by integrating contextual information from the entire row using multi-head self-attention, transforming these representations through a dense feed-forward network, and stabilizing the learning process with normalization.

*a) Multi-Head Self-Attention:* This component allows the model to weigh the influence of different tokens (i.e., [column: value]) within the input row, regardless of their position. It helps the model capture syntactic and semantic relationships. The self-attention mechanism computes attention scores between all pairs of input tokens. Multi-head attention splits this process across multiple 'heads', allowing the model to simultaneously attend to information from different representation subspaces at different positions.

*b) Normalization and Feed Forward:* This one normalizes the input across the features for stability and faster training. Moreover, each encoder layer includes a feed-forward network, which applies the same neural network to each token separately and identically. These networks introduce non-linearity and facilitates learning complex relationships among the data columns.

*4) Classification Layer:* The classification layer comprises a pooling and dense layer to aid in categorizing rows as either normal or anomalous.

*a) Pooling Layer.:* The output of the last encoder layer (corresponding to each token) is pooled to create a fixed-size output. The [CLS] token's embedding is chosen specifically for its role in capturing the contextualized information from the entire row. This token stands for "classification" and is

appended to the beginning of the input row. During training, the output representation corresponding to the [CLS] token is used as the input to a classifier to make predictions about the entire row. The model is trained to predict the correct class label (i.e., anomalous or normal) based on this [CLS] token representation.

*b) Dense Layer.:* This is a fully connected neural network layer that maps the high-dimensional output of previous layer to two classes in our binary classification problem. A sigmoid activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$) is used at the output layer to estimate the probability of the positive (anomalous) class.

### B. Anomaly Interpreter

This component aims to interpret anomalies in tabular data using the trained transformer model. After training, this component processes a test table row to identify any rules it violates based on token pairs. The anomaly interpretation method is valid and applicable in conditions where the normal dataset is representative, the anomalies disrupt typical patterns, and the goal is to gain insights into potential issues rather than providing a complete explanation of the model's decisions.

The logic of anomaly interpretation operates as follows: If the trained model labels a test row as anomalous, the component identifies related tokens within that row based on attention weights extracted from the model. Subsequently, the component assesses whether these linked tokens exhibit a similar association level in the normal data from the training dataset. To facilitate this analysis, we construct an association matrix, which encapsulates the average token associations across all normal training data. This matrix serves as a benchmark for comparing associations in the test row. If an associated token pair from the test row deviates from the patterns observed in the association matrix, this discrepancy is flagged as a violation of an association rule. Such violations, highlighting divergent token relationships from the established norms, are reported as outputs of the component.

Figure 2 shows an overview of the step-wise modules of this component. It takes a test row $X$ with $d$ tokens $X_i, ..., X_d$ as input, where $X_i$ is in form of $[column_i : value_i]$, and uses the trained anomaly detector model $AD$ and the normal association matrix $M$ to return the violation rule set $VR$ in the form of token pairs $(X_i, X_j)$. An example is a $VR$ output for a patient dataset that is represented as ([gender:'Male'], [pregnant:'True']). Another example is ([drug:'Diphenhydramine'], [dosage/day: 5200]), indicating that these pairs of columns and values are potentially the reasons behind the invalidity of an anomalous row. We outline the process of reporting rule violations from a test row below.

*1) Create Normal Association Matrix:* The normal association matrix $M$, used as a reference for detecting anomalies, is constructed from a representative sample of normal data to capture typical patterns and relationships. $M$ represents the average attention between all token pairs in the normal dataset. This process involves iterating over normal training rows, extracting attention weights between token pairs, and averaging these weights across all rows to form $M$ (Equation 1).

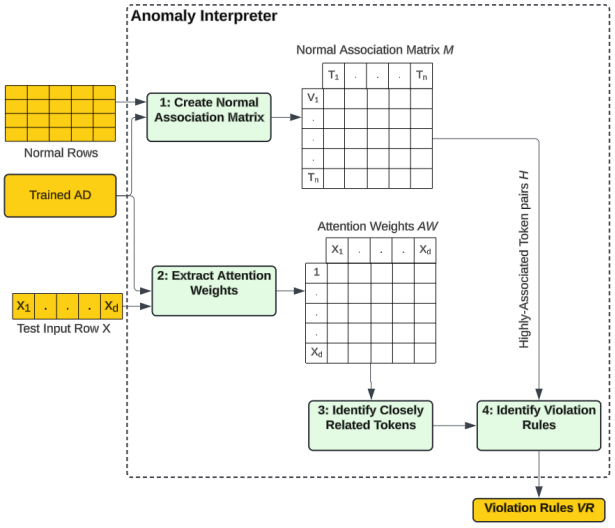


Fig. 2. An Overview of Anomaly Interpreter Component

$$M = \{M[T_i, T_j] = \text{average}(\text{normal\_attention}(T_i, T_j)) \mid 1 < i, j < n, T_i, T_j \in \text{VOC}\} \quad (1)$$

The score $M[T_i, T_j]$ indicates the average association between tokens $T_i$ and $T_j$ with respect to the normal data.

*2) Extract Attention Weights:* This module is designed to extract attention weights from the trained transformer model for a given test row. The module starts by tokenizing the input row using our custom tokenizer. Next, an attention mask is created for informing the model which parts of the input are actual data versus padding, ensuring that attention calculations are only performed on meaningful data. The module returns the predicted class, the probability of that class, and the attention weights for the input row. The attention weights is used by the next module to interpret the anomalies. This module assumes that the last layer's attention is the most representative of the model's final decision-making process. In transformer-based models, the intuition is that earlier layers capture more about syntax and lower-level features, while later layers capture more complex semantics and task-specific features. The module gets the attentions for the last layer and mean across all heads resulting in an attention weight matrix $AW_{d*d}$, where $d$ is the number of columns in input data (Equation 2).

$$AW = \{AW[X_i, X_j] = \text{attention\_score}(X_i, X_j) \mid 1 < i, j < d, X_i, X_j \in \text{row}\} \quad (2)$$

$AW[X_i, X_j]$ is the attention score between two tokens $X_i$ and $X_j$ representing how much token $X_i$ is attending to or considering token $X_j$ during the processing of input row by that last layer of the model. A higher value of score implies stronger attention or focus from token $X_i$ to token $X_j$.

*3) Identify Related Tokens:* This module aims to identify pairs of tokens in an input row that receive non-zero attention weights, indicating associations according to the model's attention mechanism. It will analyze the attention matrix from the previous module to find these pairs and return token pairs with attention weights greater than zero. The output of this module is a set of associated token pairs $H$:

$$H : \{(X_i, X_j) | 1 < i, j < d,\ AW[X_i, X_j] > 0\} \quad (3)$$

*4) Identify Violation Rules:* After identifying attended pairs in a test row as anomalous, this module compares these pairs against the association matrix $M$ to identify rule violations based on a threshold value. This module compares the attention scores between each token pair $(X_i, X_j)$ in the associated pairs set $H$ with the scores for the same token pairs in the normal association matrix $M$. Deviations from the norm, identified by a predefined threshold, are reported in a violation rule set $VR$, which is defined as follows:

$$\begin{aligned} VR : \{(X_i, X_j) \mid 1 < i, j < d, \\ (X_i, X_j) \in H, M[X_i, X_j] < t\} \end{aligned} \quad (4)$$

where $t$ is a threshold and set to the $mean(M)$.

*5) Output:* Table I presents a sample output for the Vertebral [40] dataset, which contains values for six biomechanical features used to classify orthopedic patients into three classes (normal, disk hernia, or spondylolisthesis) or two classes (normal or abnormal). The input row is fed to the trained model to check for anomaly status. The returned output indicates an anomalous status, along with additional information about the violated rules by this row. As shown in the table, the occurrence of Pelvic Incidence (PI) equal to 44.31 and Pelvic Tilt (PT) equal to 12.53 simultaneously in this row could be a potential cause of invalidity. Domain experts suggest that the anomaly in this row may indicate structural imbalance. Normally, a higher Pelvic Incidence (PI) corresponds to an increase in Pelvic Tilt (PT) to maintain posture. However, the moderate PI and low PT combination observed here could signify spinal misalignment, potentially leading to conditions like lumbar lordosis or spondylolisthesis. Another anomaly involves a Pelvic Radius (PR) of 124.11 and a Degree of Spondylolisthesis (DS) of 5.41. Typically, a larger PR provides pelvic stability, reducing vertebral slippage. The combination of a large PR with a noticeable DS suggests vertebral instability, potentially caused by weakened ligaments or degenerative changes.

TABLE I
SAMPLE EXPLAINABLE OUTPUT FOR VERTEBRAL DATASET

| Component | Details |
|---|---|
| Input | PI:44.31, PT:12.53, LL:36.10, SS:31.78, PR:124.11, DS:5.41 |
| Output Status | Anomalous |
| Violations of Association Rules | [PI:44.31, PT:12.53], [PR:124.11, DS:5.41] |

## IV. EVALUATIONS

We now describe the datasets, evaluation setup, objectives, metrics, results, and observations.

**Datasets.** We conduct the evaluation using diverse datasets from the Outlier Detection DataSets (ODDS) [17], and KDD [18] datasets. Table II details the characteristics of these datasets. Each row corresponds to a distinct dataset, arranged in descending order based on the number of rows ranging from 214 to 25192. The number of columns across these datasets ranges from 7 to 42, including a mix of numerical and categorical data types. Additionally, the last column provides a description of each dataset's domain, highlighting a variety that spans cellular biology, medical, network security, microbiology, and material science. The last column displays the proportion of anomalies in each dataset, with values ranging from as low as 2.5% to as high as 35% of anomalous data. In these datasets, the minority class is considered the anomalous class. This class may identify a specific or rare type of disease or cells in biomedical datasets, or a specific type of material in material science datasets.

TABLE II
DATASETS USED IN THE EVALUATION.

| Dataset | Instances | Features | Feature Types | Anomaly Rate |
|---|---|---|---|---|
| KDD Cup | 25,192 | 42 | Categorical, Numerical | 20.0% |
| Thyroid | 3,772 | 21 | Numerical | 2.5% |
| Yeast | 1,484 | 10 | Categorical, Numerical | 4.7% |
| Breast Cancer | 683 | 11 | Numerical | 35.0% |
| Ecoli | 336 | 9 | Categorical, Numerical | 2.6% |
| Vertebral | 310 | 7 | Categorical, Numerical | 30.0% |
| Glass | 214 | 11 | Numerical | 4.2% |

**Evaluation Setup.** We implemented the approach using Jupyter Notebook with Python 3.10 on a system running AlmaLinux. The computing environment is powered by a 12th Gen Intel® Core™ i7-12700K processor, which has 20 CPU cores (12 physical cores, each with a single thread). It operates at a base frequency of 3.6 GHz and can reach a maximum turbo frequency of 5.0 GHz. The system is equipped with 64 GiB of RAM and a 1.9 TB NVMe SSD for storage. The processor features an L1d cache of 48K, an L1i cache of 32K, an L2 cache of 1280K, and an L3 cache of 25600K, along with VT-x virtualization technology[1].

**Objectives.** (1) We evaluate the anomaly detection effectiveness and efficiency by comparing to standard models and (2) also evaluate the anomaly interpretation effectiveness and efficiency through mutation analysis.

### *A. Evaluate the Anomaly Detection Effectiveness and Efficiency by Comparing to Standard Models*

Our study compares our model against standard anomaly detection methods, including MLP, LSTM, OC-SVM, and Deep-SVDD. This comparison aimed to benchmark the effectiveness of transformer-based models in recognizing anomalies

[1]The code is available publicly: https://github.com/hajarhomayouni/Anomaly_Detection_and_Interpretation.git.

compared to more conventional approaches that have been extensively used in the field.

**Metrics.** To compare the models for their effectiveness and efficiency, we employed three metrics. **Accuracy** measures the proportion of true results (both true positives and true negatives) among the total number of cases examined. **F1 Score** is the average of precision and recall, providing a balance between these two aspects. It is particularly useful when the class distribution is uneven, which is a common scenario in anomaly detection problems. **Training Time** denotes the duration required for the models to learn from the datasets during the training phase.

**Results.** Table III presents the evaluation results for the first objective. Each row displays the outcomes of applying five different models to the datasets. The metrics measured are recorded in the cells of the table. The best values of metrics for each dataset is highlighted in bold text. For each dataset, we tested five variations of the BERT model, each customized at the embedding layer, including BERT Base [39], BERT Large [39], DistilBERT [41], RoBERTa [42], and ALBERT [43], and we documented the optimal results for the corresponding dataset. BERT Base and BERT Large are foundational models introduced by Google, pre-trained on the BookCorpus and English Wikipedia, utilizing 110 million and 340 million parameters respectively. DistilBERT, a streamlined version of BERT, retains 95% of BERT's performance on the Stanford Question Answering Dataset (SQuAD) with 40% fewer parameters. RoBERTa, developed by Facebook, omits BERT's next-sentence pretraining objective and is trained on a larger corpus that includes additional datasets like CommonCrawl News. ALBERT reduces parameter count drastically via parameter sharing, offering BERT-like performance with less memory usage and faster training times.

**Observations.** Based on Table III, our model can achieve results comparable to standard anomaly detection techniques without the need for data preprocessing. The performance of our Transformer-based model varies across different datasets compared to other models. On the KDD Cup dataset, our model excels, achieving perfect accuracy of 1.0 and a high F1 score of 0.99. It also performs well on the Thyroid and Breast Cancer datasets. However, for Yeast, Ecoli, and Vertebral, the MLP model shows better results. OC-SVM and Deep-SVDD models generally perform worse than others, except for the Glass dataset where they achieve the best results. On the Yeast dataset, we observe the largest performance gap, with the OC-SVM achieving a higher accuracy of 0.91 compared to our Transformer model's 0.71. For Ecoli, the MLP outperforms our model with an accuracy of 0.97 versus our 0.91. Vertebral dataset also shows a notable difference, with the MLP scoring 0.87 while our model scores 0.74.

Despite not consistently outperforming all other models across every dataset, our Transformer-based approach has advantages. It requires less extensive preprocessing, which is particularly useful for complex, high-dimensional datasets like KDD Cup. Additionally, our model offers explanations for detected anomalies—something traditional models like MLP, LSTM, OC-SVM, or Deep-SVDD do not provide. This interpretability, combined with strong performance on larger and more complex datasets, makes our model valuable for anomaly detection when understanding the reasons behind anomalies is essential.

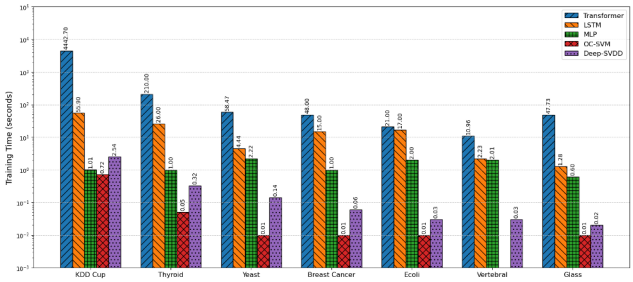


Fig. 3. Training Time Comparison Across Different Models and Datasets (logarithmic scale)

Training time depends on the dataset size and the complexity of the models. Figure 3 illustrates how the training time changes for datasets of different sizes. X-axis is sorted by dataset size, calculated as $size = number\_of\_rows \times number\_of\_columns$, which shows an almost increasing trend with dataset size. MLP exhibits consistent training times across all datasets. For the LSTM model, it appears that dataset size does not significantly affect training time; instead, other factors such as data types might be influencing its training duration. Our model is the slowest among the five for most datasets, with training times ranging from 10 seconds to nearly 74 minutes. However, since anomaly detection tasks can be performed through offline training and online prediction, training time is not a major concern. Additionally, this time can be significantly reduced using GPU-accelerated systems.

Our results did not reveal any correlation between the proportion of anomalies and the performance of the five models under test. We anticipated better performance in datasets with a larger number of anomalous data due to sufficient positive samples. While the algorithms performed exceptionally well with the breast cancer dataset, which possesses this characteristic, they did not perform as well with the vertebral dataset, containing 30% anomalous data.

The results show that the best-performing BERT model depends on the dataset's characteristics and domain, as different models have been trained on varying corpora and have different architectures. For medical and biological datasets such as Thyroid, Breast Cancer, and Yeast, ALBERT and RoBERTa models were the top performers. This may be due to their design effectively capturing domain-specific language nuances, and their comprehensive pretraining on diverse corpora, essential for datasets where precise terminology is critical. The DistilBERT model excelled with the Glass dataset, potentially due to its streamlined architecture being well-suited for datasets with simpler language structures, such as those found in material science descriptions.

TABLE III
COMPARISON OF MODEL PERFORMANCE ACROSS DIFFERENT DATASETS WHERE BOLD VALUES INDICATE THE BEST PERFORMANCE.

| Dataset | Transformer | | LSTM | | MLP | | OC-SVM | | Deep-SVDD | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | F1 Score | Accuracy | F1 Score | Accuracy | F1 Score | Accuracy | F1 Score | Accuracy | F1 Score |
| KDD Cup | **1.0** | **0.99** | 0.96 | 0.96 | 0.99 | 0.99 | 0.92 | 0.92 | 0.58 | 0.25 |
| Thyroid | **0.99** | **0.85** | 0.98 | 0.80 | 0.98 | 0.77 | 0.76 | 0.00 | 0.88 | 0.00 |
| Yeast | 0.71 | 0.71 | 0.67 | 0.67 | 0.73 | 0.73 | **0.91** | 0.00 | 0.90 | 0.00 |
| Breast Cancer | **1.00** | **1.00** | 0.94 | 0.92 | 0.97 | 0.96 | 0.94 | 0.94 | 0.67 | 0.38 |
| Ecoli | 0.91 | 0.91 | 0.91 | 0.91 | **0.97** | **0.97** | 0.91 | 0.00 | 0.88 | 0.00 |
| Vertebral | 0.74 | 0.74 | 0.74 | 0.74 | **0.87** | **0.87** | 0.58 | 0.24 | 0.65 | 0.00 |
| Glass | 0.68 | **0.68** | 0.63 | 0.63 | 0.63 | 0.63 | 0.86 | 0.57 | **0.91** | 0.67 |

### *B. Evaluating the Effectiveness and Efficiency of Anomaly Interpretation through Mutation Testing*

Due to the absence of ground truth data about anomalies, we use mutation analysis to introduce a set of controlled anomalies into the data. This allows us to determine which columns, when modified, form anomalies. To perform mutation analysis on the datasets, we selected 5% of the normal data (as this is the average proportion of anomalies in the ODDS datasets [17]), randomly mutated two column values per row, and then checked if these mutations led to the data being classified as anomalous and whether the mutated pairs were in the set of violations. The random values injected into each pair of columns were within the similar range of the standardized column values.

**Metrics.** Like traditional mutation analysis used in software testing [44], we calculated the mutation score, which is the ratio of mutants killed (those that result in anomalies with violations) to the total number of mutants. We define two different scores, one for anomaly detection evaluation and one for anomaly interpretation evaluation:

*Mutation Score 1:* Percentage of total mutants identified as anomalous.

*Mutation Score 2:* Percentage of mutants identified as anomalous and accurately reported in the violation rules.

Table IV shows a sample output from our mutation module. The first row displays a normal row selected at random to be mutated. The second row shows the mutated row, in which two column values are modified to random values. Our anomaly detection model successfully identified this mutated row as anomalous. A set of violated rules, in the form of invalid column pairs, is also reported by our anomaly interpreter model. As shown in the table, the two mutated columns, when paired with other columns, are identified as potential reasons for the row being classified as anomalous.

**Observations.** The results show the effectiveness in identifying violated rules as potential causes of anomaly. Mutation scores 1 and 2 are almost identical, demonstrating that our model was able to identify rule violations for all detected anomalies. As random values injected into each pair of columns were within a similar range to the original column values, lower values of Mutation Score 1 might result from the data not deviating too much from the norm. Most time taken by the anomaly interpreter is spent in building matrix $M$. Note that the time to build $M$ ranges from 5.76 seconds to 474.60 seconds, increasing with dataset size.

TABLE IV
SAMPLE EXPLAINABLE OUTPUT OF MUTATION MODULE FOR VERTEBRAL DATASET

| Component | Details |
|---|---|
| Normal Row | [PI:46.42], [PT:6.62], [LL:48.09], [SS:39.80], [PR:130.35], [DS:2.44] |
| Mutated Row | [PI:46.42], [PT:6.62], [LL:48.09], [SS:39.80], [**PR:1**], [**DS:2**] |
| Output Status | Anomalous |
| Violations of Association Rules | {[PI:46.42], [PR:1]}, {[PI:46.42], [DS:2.00]}, {[PT:6.62], [PR:1]}, {[PT:6.62], [DS:2.00]}, {[LL:48.09], [PR:1]}, {[LL:48.09], [DS:2.00]}, {[SS:39.80], [PR:1]}, {[SS:39.80], [DS:2.00]}, {[PR:1], [DS:2.00]} |

TABLE V
MUTATION ANALYSIS RESULTS

| Dataset | Mutation Score 1 (%) | Mutation Score 2 (%) | Time to Build M (sec) |
|---|---|---|---|
| KDD Cup | 75.45 | 73.81 | 474.60 |
| Thyroid | 80.98 | 79.35 | 166.00 |
| Yeast | 82.35 | 82.35 | 47.73 |
| Breast Cancer | 81.82 | 81.82 | 23.00 |
| Ecoli | 90.00 | 90.00 | 10.00 |
| Vertebral | 100.00 | 100.00 | 5.76 |
| Glass | 71.43 | 71.43 | 6.13 |

### *C. Limitations*

Our approach has some limitations for practical datasets: 1) We rely on labeled anomaly data. 2) We assume that tables are consistent in structure, with each row and column representing uniform data types and formats.

## V. CONCLUSIONS

We introduced a Transformer-based approach for anomaly detection and interpretation from tabular data using a transformer architecture, which reduces the need for extensive data preprocessing and provides explanatory insights through violated rules identified by anomalous data. Our future work includes evaluating the scalability of the model when used with

larger and high-dimensional datasets. Additionally, we will implement a systematic approach to vocabulary mapping for numeric data, and use separate embedding layers for numeric and categorical data.

## VI. Acknowledgement

This work was partly supported by NSF Grant Nos. CNS 1822118, CNS 2226232, DMS 2123761, AMI, NewPush, Cyber Risk Research, NIST and ARL, the State of Colorado #SB 18-086, Colorado State University, and by NIST Grant No. 60NANB23D152.

## References

[1] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Proc. of NeurIPS*, 2021, pp. 18 932–18 943.

[2] S. Lau, J. Gonzalez, and D. Nolan, *Learning Data Science*. " O'Reilly Media, Inc.", 2023.

[3] M. Gulati and P. Roysdon, "TabMT: Generating tabular data with masked transformers," in *Proc. of NeurIPS*, 2023, pp. 46 245–46 254.

[4] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," *arXiv preprint arXiv:2210.06280*, 2022.

[5] A. V. Solatorio and O. Dupriez, "Realtabformer: Generating realistic relational and tabular data using transformers," *arXiv preprint arXiv:2302.02041*, 2023.

[6] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag, "TabLLM: Few-shot classification of tabular data with large language models," in *Proc. of AISTAT*, 2023, pp. 5549–5581.

[7] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *Proc. of AAAI*, 2021, pp. 12 963–12 971.

[8] S. Golkar, M. Pettee, M. Eickenberg, A. Bietti, M. Cranmer, G. Krawezik, F. Lanusse, M. McCabe, R. Ohana, L. Parker *et al.*, "xVal: A continuous number encoding for large language models," *arXiv preprint arXiv:2310.02989*, 2023.

[9] S. Jain and B. C. Wallace, "Is Attention Interpretable?" in *Proc. of NAACL-HLT*, 2019, pp. 4193–4202.

[10] S. Serrano and N. A. Smith, "Attention is not Explanation," in *Proc. of NAACL-HLT*, 2019, pp. 3543–3556.

[11] S. Wiegreffe and Y. Pinter, "Attention is not not Explanation," in *Proc. of EMNLP/IJCNLP*, 2019, pp. 11–20.

[12] A. Rogers, O. Kovaleva, and A. Rumshisky, "A Primer in BERTology: What We Know About How BERT Works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020.

[13] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller, *SVM and boosting: One class*. GMD-Forschungszentrum Informationstechnik, 2000.

[16] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, pp. 45–66, 2004.

[17] "Outlier Detection Datasets," http://odds.cs.stonybrook.edu/ (Accessed 2024-04-25).

[18] "KDD cup 1999 data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999, accessed: 2024-04-25.

[19] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[20] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *IEEE Access*, vol. 9, pp. 78 658–78 700, 2021.

[21] H. Lim, S. Park, M. Kim, J. Lee, S. Lim, and N. Park, "MadSGM: Multivariate Anomaly Detection with Score-based Generative Models," in *Proc. of CIKM*, 2023, p. 1411–1420.

[22] A. Lee, Y. Zhang, H. M. Gomes, A. Bifet, and B. Pfahringer, "Look at me, no replay! surprisenet: Anomaly detection inspired class incremental learning," in *Proc. of CIKM*, 2023, p. 4038–4042.

[23] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[24] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, and N. Ding, "Gan-based anomaly detection: A review," *Neurocomputing*, vol. 493, pp. 497–535, 2022.

[25] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[26] Y. Qiao, K. Wu, and P. Jin, "Efficient anomaly detection for high-dimensional sensing data with one-class support vector machine," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 404–417, 2021.

[27] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. of KDD*, 2017, pp. 665–674.

[28] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proc. of Q2SWinet*, 2020, pp. 37–45.

[29] Y. Gorishniy, I. Rubachev, and A. Babenko, "On Embeddings for Numerical Features in Tabular Deep Learning," in *Proc. of NeurIPS*, 2022, pp. 24 991–25 004.

[30] D. Zhang, L. Wang, X. Dai, S. Jain, J. Wang, Y. Fan, C.-C. M. Yeh, Y. Zheng, Z. Zhuang, and W. Zhang, "FATA-Trans: Field And Time-Aware Transformer for Sequential Tabular Data," in *Proc. of CIKM*, 2023, p. 3247–3256.

[31] Z. Wang and J. Sun, "TransTab: Learning Transferable Tabular Transformers Across Tables," in *Proc. of NeurIPS*, 2022, pp. 2902–2915.

[32] H. Thimonier, F. Popineau, A. Rimmel, and B.-L. Doan, "Making parametric anomaly detection on tabular data non-parametric again," *arXiv preprint arXiv:2401.17052*, 2024.

[33] Z. Wang and J. Sun, "Transtab: Learning transferable tabular transformers across tables," in *Proc. of NeurIPS*, 2022, pp. 2902–2915.

[34] I. Giurgiu and A. Schumann, "Additive Explanations for Anomalies Detected from Multivariate Temporal Data," in *Proc. of CIKM*, 2019, p. 2245–2248.

[35] S. Kiefer and G. Pesch, "Unsupervised anomaly detection for financial auditing with model-agnostic explanations," in *Proc. of KI*, 2021, pp. 291–308.

[36] P. Mavrepis, G. Makridis, G. Fatouros, V. Koukos, M. M. Separdani, and D. Kyriazis, "Xai for all: Can large language models simplify explainable ai?" *arXiv preprint arXiv:2401.13110*, 2024.

[37] T. Ali and P. Kostakos, "Huntgpt: Integrating machine learning-based anomaly detection and explainable ai with large language models (llms)," *arXiv preprint arXiv:2309.16021*, 2023.

[38] J. Su, C. Jiang, X. Jin, Y. Qiao, T. Xiao, H. Ma, R. Wei, Z. Jing, J. Xu, and J. Lin, "Large language models for forecasting and anomaly detection: A systematic literature review," *arXiv preprint arXiv:2402.10350*, 2024.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[40] G. Barreto and A. Neto, "Vertebral Column," UCI Machine Learning Repository, 2011, DOI: https://doi.org/10.24432/C5K89B.

[41] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[42] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[43] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[44] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. L. Traon, and M. Harman, "Mutation Testing Advances: An Analysis and Survey," *Advances in Computers*, vol. 112, pp. 275–378, 2017.