УДК: 004.896

СПОСІБ ІНТЕРПРЕТАЦІЇ РЕЗУЛЬТАТІВ АУДИТІВ

Трочун Є. В.

e-mail: zheniatrochun@gmail.com

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна, Київ

Аудити програмного забезпечення – чудовий спосіб перевірки та підтримки системи в повністю функціональному стані. Та зважаючи на постійне збільшення розмірів як систем, так і збільшення об'ємів даних, що проходять через системи, проблема підтримки всіх їх частин в коректному стані є дуже важливою, та аудити програмного забезпечення допомагають у вирішенні цієї проблеми. Зі збільшенням розмірів систем, збільшуються об'єми даних, що їх продукують аудити та ускладнюється процес виявлення помилок в системах на основі їх аудиту. Проте, підчас дослідження літератури, не було знайдено загального алгоритму інтерпретації результатів аудитів, що може бути використаний для роботи з різними системами. Тому в даній статті розглянута задача інтерпретації результатів аудитів на прикладі технічних аудитів веб-сайтів. Запропоновано алгоритм вирішення даної проблеми за допомогою методів машинного навчання.

Ключові слова: машинне навчання, аудити програмного забезпечення, аналіз даних, нейронні мережі, глибоке навчання.

Трочун Е. В. Способ интерпретации результатов аудитов / Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Украина, Киев

Аудиты программного обеспечения - отличный способ проверки и поддержания системы в полностью функциональном состоянии. И

учитывая постоянное увеличение размеров как систем, так и увеличение объемов данных, проходящих через системы, проблема поддержки всех их частей в корректном состоянии очень важна, и аудиты программного обеспечения помогают в решении этой проблемы. С увеличением размеров систем, увеличиваются объемы данных, которые производят аудиты и усложняется процесс выявления ошибок в системах на основе их аудита. Однако, во время исследования литературы, не было найдено общего алгоритма интерпретации результатов аудитов, который может быть использован для работы с различными системами. Поэтому в данной статье рассмотрена задача интерпретации результатов аудитов на примере технических аудитов сайтов. Предложен алгоритм решения данной проблемы с помощью методов машинного обучения.

Ключевые слова: машинное обучение, аудит программного обеспечения, анализ данных, нейронные сети, глубокое обучение.

Y. V. Trochun Way of Audit Results Interpretation / National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, Kyiv

Software audits are a great way to test and keep your system fully functional. But given the ever-increasing size of both systems and the amount of data that passes through systems, the problem of keeping all of their parts in good condition is very important, and software audits help solve this problem. As systems grow, the amount of data produced by audits increases and the process of detecting errors in systems based on their audit becomes more complicated. However, during the literature study, no general algorithm for interpreting audit results was found that could be used to work with different systems. Therefore, this article discusses the problem of interpreting the results of audits on the example of technical audits of websites. Proposed an algorithm for solving this problem using machine learning techniques.

Key words: machine learning, software audit, data analysis, neural networks, deep learning.

Вступ. В умовах бурхливого розвитку інформаційних систем та грандіозного збільшення об’ємів даних, що проходять через системи обробки даних, особливо важливим стає питання стабільності та точності роботи систем масового обслуговування, адже навіть найменші помилки в їх роботі можуть мати дуже великий негативний вплив на якість продукту чи послуги, що отримує користувач а як наслідок — сильно вплинути на утримання аудиторії користувачів та відповідно на дохідність бізнесу. Один з розповсюджених методів контролю стабільної роботи та виявлення помилок в системах — це їх регулярні аудити. Аудити дозволяють збирати велику кількість даних з систем, але постає питання в інтерпретації зібраних даних та аналізі результатів. У більшості випадків, аналіз проводиться в неавтоматизованому або напівавтоматизованому режимі, що у більшості випадків зводиться до попередньої обробки даних перед проведенням експертного аналізу. Це значно збільшує вартість проведення регулярних аудитів, потребує експертів високої кваліфікації для отримання якісних результатів та допускає можливість виникнення людської помилки. Тому виникає проблема проведення автоматизованого аналізу результатів аудитів систем.

Мета статті. Метою цієї роботи є побудова та обґрунтування алгоритму автоматизованої інтерпретації результатів проведення аудитів програмного забезпечення на прикладі технічних аудитів веб-сайтів.

Виклад основного матеріалу. Задача аналізу результатів аудитів у більшій мірі відноситься до категорії навчання без учителя. Оскільки наявні наступні вимоги та обмеження:

- Дані історичних аудитів, що доступні для аналізу та для

підготовки мережі не розмічені, тобто дані аномальних сторінок не позначені як аномальні.

- Об’єм тренувальних даних доволі великий — дані близько половини мільйона сторінок знайдених на близько десятка сайтів. Такий об’єм даних не дозволяє розмітити їх вручну для всього набору.
- Тренувальні дані зібрані з близько десятка різних сайтів, тому навіть вручну визначити аномальні дані доволі важко, адже у кожного користувацького сайту своя специфіка імплементації та роботи.

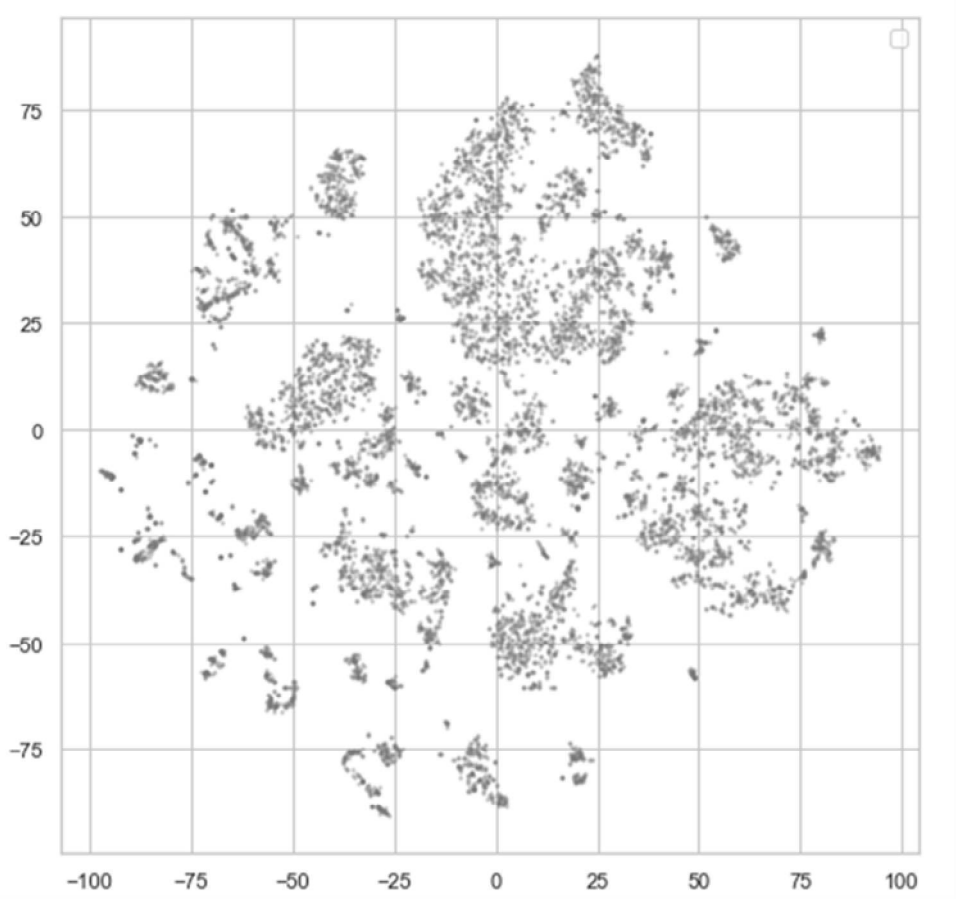

Рисунок 1. T-SNE візуалізація частити тренувального набору даних

Отже є можливість в ручному режимі розмітити лише відносно невелику частину доступного набору даних, але аж ніяк не весь набір тренувальних даних. Тому розмічену частину даних варто

використовувати виключно для верифікації та аналізу результатів роботи побудованої та вже навченої моделі.

На рисунку 1 наведений T-SNE (t-distributed stochastic neighbor embedding) [1] графік частини тренувального набору даних, що відноситься до аудиту одного веб-сайту. З рисунку видно, що на наборі даних можна виділити близько 15-ти окремих кластерів, та певну кількість точок, що знаходяться поза межами будь-якого кластеру. Такі точки з високою ймовірністю є аномальними сторінками, оскільки дані сторінки не відповідають загальним шаблонам, що їм відповідають інші сторінки. В загальному випадку, кожен окремий кластер на T-SNE графіку відповідає окремій групі подібних сторінок на сайті. Проте є і винятки — деякі одиничні сторінки, що не схожі на більшість інших не є дійсно аномальними, оскільки трапляються виключення з правил. Гарним прикладом є унікальні сторінки сайту: стартові сторінки, сторінка корзини на сайтах інтернет-магазинів. Це приклади одиничних сторінок, які не є автоматично згенерованими. Якщо вважати такі сторінки аномальними — це доволі груба помилка, адже користувачу, що проводить регулярні аудити свого веб сайту потрібно буде кожного разу в ручному режимі відділяти подібні негативно-позитивні результати, що значно знижує цінність автоматизації інтерпретації результатів.

Через вищезгадану проблему, а також через те, що у кожної окремої системи наявна різна кількість сформованих груп сторінок, використання алгоритмів виявлення аномалій на основі кластерного аналізу [2] таких як DBSCAN [3] не є найкращим підходом до розв’язання поставленої задачі, оскільки дані алгоритми потребують або вказання очікуваної кількості кластерів, або ж вони ігнорують точки, що не відносяться до якогось певного кластеру, що є неприпустимо зважаючи на обмеження задачі.

Іншим підходом до виявлення аномалій є використання нейронних мереж. Одна з найбільш поширених архітектур нейронних мереж для виявлення аномалій з навчанням без учителя — мережі автоенкодери [4]. Розглянемо структуру мережі автоенкодера.

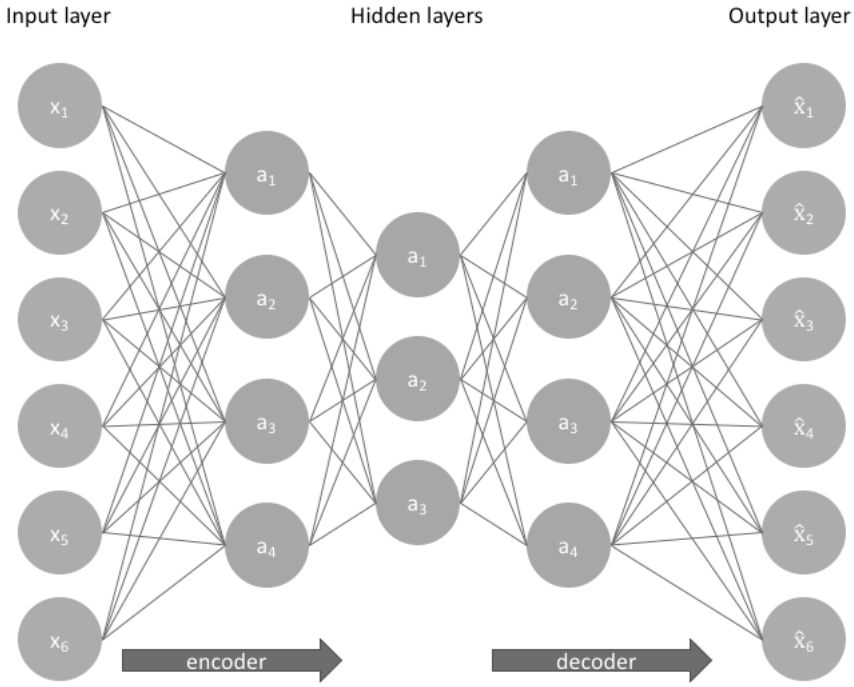

Рисунок 2. Приклад мережі автоенкодера

На рисунку 2 зображено приклад стандартної архітектури глибокої нейронної мережі автоенкодера. Кількість метрик, що збираються з сторінок систем, на яких проводяться аудити — 26. Тому вхідний та вихідний шар нейронної мережі має містити саме 26 нейронів. Також є сенс у внесенні незначного шуму у дані, що допомагає при кожній ітерації по тренувальному наборі даних подавати в мережу трішки різні дані для кожної з сторінок, що допомагає мережі краще узагальнити дані та вивчати не конкретні значення, а шаблони, яким відповідають дані. Це досягається за рахунок того, що значення кожної з метрик сторінки кожного разу, коли поступає на вхід нейронної мережі, має незначні відхилення від оригінального значення. Для досягнення цього ефекту, другий шар мережі має складатись з 26 нейронів, що зашумлюють дані у межах 5%. Тобто до

кожного значення конкретної метрики буде випадковим чином додано випадкове значення в межах [-x * 0.05; x * 0.05], де x — значення конкретної метрики. Перший та другий шар нейронної мережі не повинні бути повністю зв'язні з огляду на те, що роль другого шару мережі — зашумлення даних, тому кількість зв'язків першого шару та другого — 26 зв'язків, оскільки кожен нейрон вхідного рівня пов'язаний лише з одним нейроном другого рівня, а кожен нейрон другого рівня пов'язаний лише з одним нейроном першого рівня. Ці зв'язки не повинні бути оновлені в процесі тренування, оскільки все, що вони роблять — доставляють дані в наступний рівень та не повинні ніяк модифікувати дані та повинні бути мати значення рівне 1.

Після штучного зашумлення, дані потрапляють в першу частину мережі, що піддається тренуванню — шифрувальник (encoder). Шифрувальник складається з трьох повністю сполучених шарів по 20, 15 та 10 нейронів відповідно. Після шифрувальника йде один шар "коду" (code), що складається з 3 нейронів. Після коду знаходяться 3 шари дешифрувальника (decoder), що дзеркалять шифрувальник — три повністю сполучені шари: 10 нейронів, 15 нейронів, 20 нейронів. І останній шар — шар виводу, що теж має 26 нейронів, як і вхідний шар, оскільки кількість вхідних параметрів, що доступні про сторінку — теж 26.

Оскільки у шарі коду знаходиться значно менша кількість нейронів, ніж кількість вхідних параметрів, мережа змушена навчатись шифрувати та відтворювати з "шифру" вхідні параметри, вона не може просто завчити всі тренувальні дані, вона змушена знаходити шаблони у тренувальних даних та відтворювати оригінал з знайдених шаблонів. За рівнем успішності відтворення дешифрувальником вхідних даних можна визначити рівень аномальності вхідних даних. Метрики сторінок, схожі до яких були

присутні в тренувальному наборі даних, мережа буде відтворювати доволі точно, оскільки вона зможе знайти у вхідних даних шаблони, які були присутні на сторінках в тренувальному наборі даних. А для нетипових сторінок, для яких мережа не зможе знайти шаблону з сторінок тренувального набору даних, рівень точності відтворення буде відносно низьким. Тому чим більша різниця між вхідними даними та відтвореними — тим більш аномальною є сторінка.

Результат роботи нейронної мережі автоенкодера — відтворені вхідні дані. У даному випадку — відтворені дані сторінки. Підчас тренування мережа навчається відтворювати сторінки з тренувального набору даних, вивчає певні залежності між різними метриками сторінок з тренувальних аудитів. Тому при використанні натренованої мережі на даних нового аудиту, сторінки, що схожі на ті, що були присутні в тренувальному наборі даних відтворюються добре та мають низьке значення функції втрат. Аномальні ж сторінки відтворюються значно гірше, адже вони не відповідають шаблонам та співвідношенням, що були знайдені мережею на тренувальних даних, та які вона навчилась знаходити та відтворювати. Тому аномальні сторінки матимуть значно більші значення функції втрат ніж звичайні. Зважаючи на це, знаючи значення функції втрат на останніх ітераціях навчання мережі, можна визначити певне порогове значення. І всі сторінки, значення функції втрат для яких перевищують дане порогове значення, можна вважати аномальними. Це порогове значення по своїй суті є одним з гіперпараметрів побудованої моделі та може корегуватись залежно від бажаної точності та бажаного максимального числа аномальних даних, що можу бути повідомлене користувачам, що запускали аудити та які є зацікавленими в їх результатах. Дане порогове значення має бути пораховане один раз для кожної з нейронних мереж. Один з способів визначення даного

порогового значення — домноження середнього значення функції втрат на останній значимій ітерації навчання мережі на певний коефіцієнт, що має бути визначений експериментально та може бути конфігурований відповідно до відгуків та побажань користувачів. Для першої ітерації даного алгоритму, було встановлено значення даного коефіцієнту, що дорівнює 3.5.

Висновки. В даній роботі було розглянуто спосіб інтерпретації результатів технічних аудитів веб-сайтів, що базується на використанні нейронних мереж для винесення рішення про аномальність сторінки. Розроблений алгоритм використовує глибоку нейронну мережу автоенкодер для відтворення метрик сторінок веб-сайту на якому проходить аудит. Нейронна мережа складається з 10 шарів. Навчання мережі відбувається на сторінках з попередніх аудитів веб-сайтів, таким чином мережа знаходить та вивчає шаблони та закони, яким підкоряються різні групи сторінок веб-сайту. Навчена мережа перевіряє сторінки нових аудитів веб-сайту на відповідність вивченим шаблонам. Сторінки, що мають високий рівень схожості з сторінками з попередніх аудитів, відтворюються нейронною мережею дуже добре, проте сторінки, що є аномальними та не схожими на сторінки з тренувального набору даних, відтворюються з низькою точністю, тому значення функції втрат для них буде доволі високе. Також було обрано порогове значення, при перевищенні якого функцією втрат сторінка буде вважатись аномальною.

Розроблена модель має точність відтворення вхідних даних близько 92% та знаходить близько 80% аномальних сторінок з тестового набору даних. 20% аномальних даних було пропущено у зв’язку з тим, що тренувальний набір даних теж містив аномальні сторінки, що дещо знижує ефективність роботи алгоритму. Проте з часом, при наступних ітераціях перетренування алгоритму, його

точність буде поступово рости у зв’язку з тим, що тренувальний набір даних буде все більше і більше очищуватись від аномалій знайдених моделлю. Я вважаю, що це зможе довести точність виявлення аномалій до значень близьких до 90%.

Зважаючи на отримані результати, точність роботи розробленого алгоритму є задовільною та алгоритм може бути використаний для виконання автоматичного аналізу результатів технічних аудитів веб-сайтів. Що може зменшити час, що витрачається на проведення аналізу результатів експертами, що в свою чергу знижує вартість проведення регулярних технічних аудитів та зменшує час, що затрачується на їх проведення.

Література:

1. T-distributed stochastic neighbor embedding. <https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding> (2021, квітень, 21).

2. Cluster analysis <https://en.wikipedia.org/wiki/Cluster_analysis> (2021, травень, 4).

3. DBSCAN Clustering Algorithm in Machine Learning <https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html> (2020, квітень).

4. Introduction to autoencoders <https://www.jeremyjordan.me/autoencoders/> (2018, березень, 19).

References:

1. T-distributed stochastic neighbor embedding. Retrieved from https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding [in English]. (2021, April, 21).

2. Cluster analysis. Retrieved from https://en.wikipedia.org/wiki/Cluster_analysis [in English]. (2021, May, 4).

3. DBSCAN Clustering Algorithm in Machine Learning. Retrieved from https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html [in English]. (2020, April).

4. Introduction to autoencoders. Retrieved from https://www.jeremyjordan.me/autoencoders/ [in English]. (2018, March, 19).