International Journal of Informatics, Information System and Computer Engineering

XBRL Open Information Model for Risk Based Tax Audit using Machine Learning

Bagas Dwi Suryo Wibowo

University of Glasgow, United Kingdom

E-mail: vox_eu@yahoo.com

ABSTRACTS

Tax audit is an effective instrument for preserving tax compliance, and risk-based tax audit selection can optimize it. Risk-based tax audit selection selectively auditing on high financial risk wealthy taxpayers. In contrast, manually selecting amid the plethora of taxpayer data is difficult, prone to human error, costly and time-consuming. Fortunately, using Extensible Business Report Language (XBRL) as a well-known financial statement reporting standard enables automation. This project proposed software named XAFR as a model for extracting, transforming, and loading the latest XBRL Open Information Model (OIM) 1.0 standard US-SEC dataset and provided it as a data source for risk classification using rule-based risk scoring and Machine Learning. Several thorough testing exposed Random Forest classifier as the best model for Machine Learning risk classification with high accuracy, revealing the excellent collaboration of rule-based risk scoring approach with Machine Learning for risk classification and the importance of XBRL as a transparent but robust report standard that tax authorities can utilize. The excellent system integration resulted in the ability to expose wealthy high-risk taxpayers and high-risk industries and predict risk classification based on two-year financial statements. Moreover, this report introduces the critical importance of RCA (Risk, Current Ratio, Assets) analysis and SIC (Standard Industry Classification) utilization to generate risk classification, rank and explanation. This project utilizes financial indicators in the limited year and leaves the semantic analysis for future works because of time and hardware limitations. The possibility of predicting the possible tax debt prediction are promising Machine Learning future developments.

ARTICLE INFO

Article History:
Submitted/Received 20 Dec 2021
First Revised 04 Feb 2022
Accepted 15 Apr 2022
First Available Online 10 Mar 2022
Publication Date 01 Jun 2022

Keywords:
Audit-Selection, Assets, Current-Ratio, Financial-Risk, Machine-Learning, Open Information-Model, Risk, Risk-Based, Risk-Scoring, Rule-Based, Standard-Industry-Classification, Tax-Audit, XBRL

DOI: https://doi.org/10.34010/injiiscom.v3i1.6891
p-ISSN 2810-0670 e-ISSN 2775-5584

1. INTRODUCTION

Indonesia relies on taxation. Tax contributed to more than three-quarters of total income in 2017-2021 (Central Bureau of Statistics of the Republic of Indonesia, 2021). In contrast, Indonesia's tax to Gross Domestic Product (GDP) ratio is too low and should be elevated. Thus, it is critical to maximize the essential role of tax audits.

Risk management is required to improve compliance by selecting the correct Taxpayer, the high risk but wealthy Taxpayer (Khwaja et al., 2011). The authorities should identify and quantify the chance to choose high-risk taxpayers while eliminating non-compliance amid massive data and various information systems, making the manual selection a hard decision. Fortunately, utilizing the financial reporting standard XBRL enabled comparing and analyzing the corporate disclosure information over time and across entities (SEC 2021, para.2). XBRL technology allows performing automatic risk-based tax audits selection.

This report proposes XAFR (XBRL – Artificial Intelligence Financial Risk Detection), a web-based application built on Python to address the identified problem. XAFR aims to classify taxpayers' financial risk and expose wealthy high financial risk taxpayers based on their financial information data reported to USSEC in XBRL format for tax audit selection. The objective is to extract, transform, validate, and load the dataset to the database. Calculate the risk score using a rule-based approach, utilizing trend and industry level benchmark, resulting in rule-based risk score, classification and explanation. Furthermore, the risk data and statistics were appraised to extract the best features for Machine Learning and predict Taxpayer risk classification. Each risk classification is clustered and ranked by risk-score, current ratio, and total assets (RCA) to reveal the wealthy high financial risk taxpayers. This project contributes by designing a model capable of processing the novel XBRL OIM 1.0 specification datasets to extract essential information for tax authorities.

2. Existing Product, Prior Research and Gathering Requirement

The background research is initiated by collecting information on the related software products and prior research. They are categorized into three:

2.1. XBRL tools and services software

Well-known XBRL tools and services software are Altova, Arelle, Ez-XBRL, Datatracks and Workiva. These software products are listed on the XBRL International Certified Software. They aim to generate and provide a valid XBRL report for single entity XBRL report creation, review, validation, and analysis complying with XBRL 2.1 and Inline XBRL 1.1 specification standards (XBRL International, 2021c).

XBRL 2.1 or Inline XBRL 1.1 rely on Extensible Markup Language (XML) for data transmission. The most crucial feature of XBRL is the precise definitions of taxonomies that provide the meaning and relationships of all reporting terms (XBRL International 2021a, para.15). This technology establishes an appropriate context when reading any words in the XBRL report. Hence, as long as the taxonomy is well regulated, any information can be generated using the correct taxonomies to be exchanged between entities in a functional, practical and accurate digital format.

However, XBRL OIM 1.0 specification, released in October 2021, is prepared for extensive data analysis (XBRL International, 2021b). They are using the JavaScript Object Notation Link Data (JSON-LD), Tab Separated Value (TSV) and HyperText Markup Language (HTML) format (SEC, 2021). The all-in-one context in a single XML format is distributed to JSON-LD for schema, data type and definition used in machine operation, TSV for the data tuples, and HTML for schema, data type and purpose used by the human reader.

2.1.1. XBRL-Artificial Intelligence application

Ashtiani and Raahemi's (2021) study comprehensively compare forty-seven articles studying financial statement fraud detection (FSFD) using machine learning and data mining (ML/DM). The study informs that most essays use classification supervised machine learning model financial ratio structured data sources.

While study to detect financial statement fraud using the hybrid model in China and India found Ensemble Method using RF outperformed others (Hooda et al., 2020; Yao et al., 2018). In conclusion, most prior studies have used SVM and RF as excellent classifiers for financial statement fraud detection, while LR, although used in classification, is more fit for trend prediction.

However, there are no prior financial statement fraud detection studies using the recent XBRL OIM 1.0. Only one previous study found discusses the reasoning and explanation behind fraud detection. Venters and Mikkilineni's (2020) study used unsupervised deep learning to detect anomalies in financial statements, found the lack of transparency of reasoning behind the conclusion, and then provided a sophisticated Deep Reasoning model to mimic human reasoning processes.

2.1.2. Risk-Based Tax Audit Selection Application

Khwaja et al. (2011) study comprehensively summarise data-mining risk-based tax audits implementation in many countries: The United Kingdom, Sweden, The Netherlands, Bulgaria, India, Ukraine, Kazakhstan, Turkey and some other World Bank studies. The summary implies that the risk-scoring technique is the fundament of risk-based tax audit selection. The risk-scoring approach builds a taxpayer profile based on specific attributes and knowledge acquired during the previous audit. Implementing the risk-scoring strategy requires high-quality data, past audit cases, and current taxpayer attributes. This approach required Information Technology (IT) systems to process the data, score, and feed the pointer into audit programming.

2.2. Gathering Requirements

Based on the information collected in the background research above, the project should be able to:

1. Design scalable infrastructure configuration for massive data storage, mining, and machine learning processing. The risk-based audit selection data mining required adequate IT systems (Khwaja et al. 2011).
2. Design a complex general framework for the overall process, then break it into a manageable smaller framework (Jurney, 2017). The manageable smaller framework identified are XBRL, Artificial Intelligent and Web frameworks.
3. Integrate the smaller framework back to the general framework. Extensive unit testing and system integration

testing is required to ensure excellent integration.

4. Design the reliable rule-based risk scoring approach to produce high-quality financial indicators for machine learning features.
5. Design thorough tests and measurements to ensure data validity and integrity, correct taxonomy design to increase the non-zero value extraction and select the best classifier model with high accuracy.
6. Design an informative user interface that provides usability for end-user. Focus more on the data insight, usability, and accuracy interface.

2.2.1. Functional Requirements

User Stories and MoSCoW statements

The User Stories analysis reveals three User Roles with seven User Story cards, allocated using MoSCoW analysis to 4 Must-Have, 2 Should-Have, 1 Could-Have an additional 3 Will-Not-Have cards.

The effort available is 30 ideal days to accomplish 30 User Acceptance Tests. The detail is provided in Appendix A.

2.2.2. Non-Functional Requirements

Hardware and Software Requirement

Hardware and software infrastructure is essential for massive dataset processing. The hardware and software requirements are gathered to build the Application Infrastructure Framework, as shown in Fig. 1.

The software deliveries will be conducted in three stages: development, testing, and deployment using the environment listed in Table 1. The setting is set up in the development stage, and the software is built until the software is ready for testing. The testing stage replicates the developed software and environment into a Virtual Machine (VM). The unit and system integration test will be conducted in the development and testing stages. Finally, the software will be replicated to the cloud service using Vagrant if it passes all user and system integration tests. This project will utilise (AWS) and will take advantage of the AWS Free Tier offer for its easy build full-stack software and infrastructure for web-based Machine-Learning development (Amazon, 2021; Amunategui and Roopaei, 2018; Jurney, 2017).

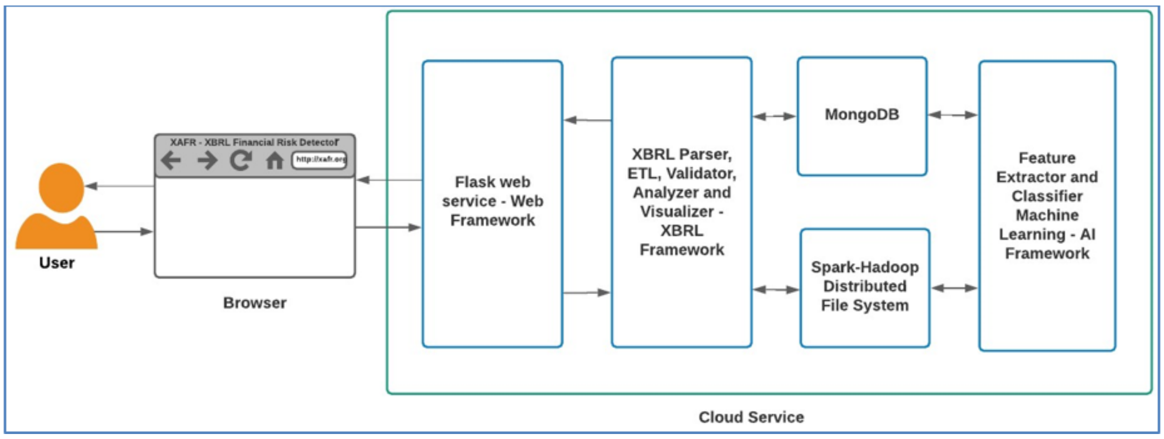

User
XAFR - XBRL Financial Risk Detector
http://xafr.org
Browser
Flask web service - Web Framework
XBRL Parser, ETL, Validator, Analyzer and Visualizer - XBRL Framework
MongoDB
Spark-Hadoop Distributed File System
Feature Extractor and Classifier Machine Learning - AI Framework
Cloud Service

Fig. 1. Application Infrastructure Framework

Table 1. Hardware and software environment in each development stage

Machine	Local Machine – Notebook Computer	Virtual Machine – VirtualBox (Oracle, 2021b)	Amazon Web Service Free Tier – Elastic Compute Cloud (EC2) (Jurney, 2017)
Core Processor	4	2	
Memory	16 GB	12 GB	
Operating System	Windows 10 64bit (Microsoft, 2021a)	Linux Ubuntu 18 LTS 64 bit (Canonical, 2021)	
Reasons	*An excellent user interface, feature-rich, most widely used with extensive supports*	*Resilient, resource-efficient, more resistant to viruses, malware and trojans*	
Container	-	Vagrant (HashiCorp, 2021)	
Reasons	*The final software OS is Linux Ubuntu, not Windows*	*a lightweight, reproducible, and portable VM environment container software* (*Jurney 2017, p. 33*)	

Database Management System (DBMS) is the essential requirement to reduce repeated data processing by storing previous results to improve execution time. Figure 2 shows the differences between the standard form in the relational schema dataset and the denormalised form in the schemaless dataset. The relational schema dataset is vertically scalable, but the schemaless dataset is horizontally scalable. Hence, this project data mining is horizontally scalable.

Relan (2019) mentions that relational databases use Structured Query Language (SQL) for data manipulation and consist of a table of rows and columns, which is excellent for vertically scalable data. While NoSQL stored the data without a structure (schemaless) and denormalised only when necessary. This capability is a superior option for distributed and horizontally scalable systems. Hence, this project will use a NoSQL database model. Mongo-DB is preferred as the document-based NoSQL DBMS (MongoDB, 2021). MongoDB uses the BSON (Binary JSON) format, which is faster for data searching and processing. MongoDB 5 has an Aggregation feature comparable to Hadoop Distributed File System's (HDFS) Map-Reduce.

Apache Hadoop 3.3 was selected for DFS because (HDFS) has a large block size, which is excellent for vast capacity data storage (Apache Software Foundation, 2021a). Using the default 128 Megabytes (MB.) HDFS block size is faster for indexing, reading, and writing large files than smaller block sizes in other file systems. For example, NTFS only has a 4 to 8 KB block size (see Fig. 3 for comparison). This project will combine HDFS infrastructure for massive data storage and the Local Machine file system for minor or temporary data storage.

Apache Spark 3.2 (Apache Software Foundation, 2021b) was selected as the Resilient Distributed Dataset (RDD) processor, which runs on top of Apache Hadoop in an RDD-DFS architecture. Flask 2.0 (Pallets, 2021) handles the overall system integration as a lightweight web application server (Relan, 2019).

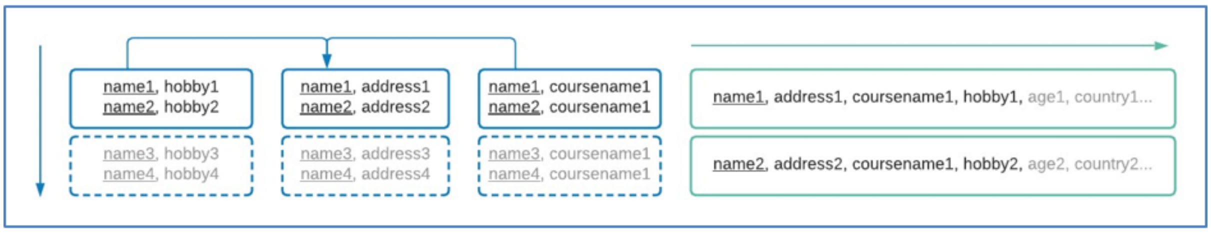


Fig. 2. Normal Form in Relational Schema Dataset (Left – Blue) and Denormalized Form in Schemaless Dataset (Right – Green)

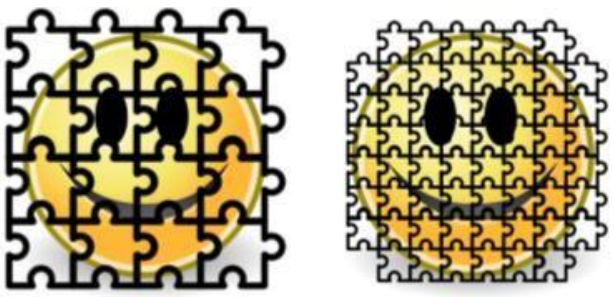


Fig. 3. Block size impact for file indexing, reading, and writing illustrated as puzzles. More extensive block size is faster read/write at the cost of efficiency

Anaconda 3 (Anaconda, 2021) package is selected for its extensive Python library in Machine Learning development, including Jupyter Notebook, ScikitLearn, NumPy, Pandas, Bokeh, Matplotlib library. Microsoft Visual Code 1.62 (Microsoft, 2021b) extensive libraries, paired with the Jupyter Notebook server and Microsoft Powershell 7 terminal, give a robust IDE for developing this Python project in Windows. The last software requirement is Java 11 (Oracle, 2021a) that required to support dependencies between applications and drivers.

2.2.3. User Interface

XAFR is designed for the tax authority with a government employee as the end-user. Although smartphones are sometimes used, government employees mainly work with desktop computers or notebooks. The Bootstrap grid technology (Bootstrap et al., 2021) can be utilized for the responsive grid layout's capability to display correctly on smartphones, tablets, or desktops, but with priority on the desktop environment.

This project aims to present data insight and essential information. Thus, the interface should be minimalistic to avoid distracted users and missing understanding. Likewise, the colour nuance should be minimal but more informative and noticeable. A wrapper or section that could dynamically open and hide the content should be provided for pages that include a large amount of material.

This project will leverage the existing certified XBRL software for its distinctive approach to utilising taxonomy for XBRL 2.1 then adapt it to the XBRL OIM 1.0 development. The XBRL OIM 1.0 provide the data tuples on the TSV files. TSV is more efficient than XML because TSV does not contain context and extra tags, but it is "blind". TSV is "blind" because there is no machine-readable definition or relationship between fields, intra-file or inter-file.

From prior financial statement fraud detection using artificial intelligence studies, this project leverages the importance of features selection to improve Machine Learning prediction accuracy. This project will use the Features Correlation Heatmap analysis and Random Forest features importance score to select the significant elements.

Based on a prior study, the benchmark at the industry level improves any classifier's algorithm's accuracy. Hence, this project will utilize the Standard Industry Classification field to identify the industry cluster of each Taxpayer and compare their financial indicator value with the average in a similar industry. Furthermore, the risk rank will be sorted using risk score, current ratio value and total asset (RCA) to reveal the wealthy high financial risk Taxpayer.

The prior studies mention that supervised machine learning is the most used, with SVM and RF as the best classifiers in different cases. This project will use supervised machine learning and do an iterative test to select the classifier model outperforming the overall average accuracy score. If both classifier models consistently have similar performance, both models will be used for risk classification in future prediction, but only selecting the highest accuracy score for the final result.

3. SCOPE AND IMPLEMENTATION

3.1. General Application Framework Design and Scope

This project's scope is from the XBRL OIM 1.0 documents extraction until the user visualizes the wealthy high financial risk taxpayers (see the blue area in Fig. 4). This project did not implement the complete Tax Audit Case Management System. Furthermore, the General Application Framework is split into smaller frameworks: XBRL, AI, and Web framework to be allocated into the project development roadmap.

3.2. Project Development Roadmap

The project development roadmap divided the project development into stages, sprints and iterations. Gantt chart Timeline map the project development roadmap sprints into a timeline. Both diagrams are attached in Appendix B.

3.3. Constraints

Hardware limitation significantly impacts massive data implementation and makes development processes consume a high amount of time. For example, load the extracted TSV dataset to the database. Pre-optimization loading (HDFS directly to the database) consume 437 minutes to load a part of TSV documents containing four million tuples (148 tuples/second). Post-optimization (file chunk algorithm and HDFS - Local Storage collaboration) speeds up the load time to 7175 documents/second, 27 million tuples in 64 minutes (see Fig. 5) only for data loading, while the total dataset used is around forty million tuples. Concerning the hardware and time limitation, this project utilizes the US-SEC dataset.

It is from 2019-Q1 to 2021-Q3, targeting the 10-K Annual Audited Financial Statement Form. The US-SEC dataset usage purpose in this project is complying with the US-SEC aim to analyze and compare corporate disclosure information from time to time and across entities (SEC, 2021). Likewise, the disclaimer from the US-SEC of this dataset also applies to this project.

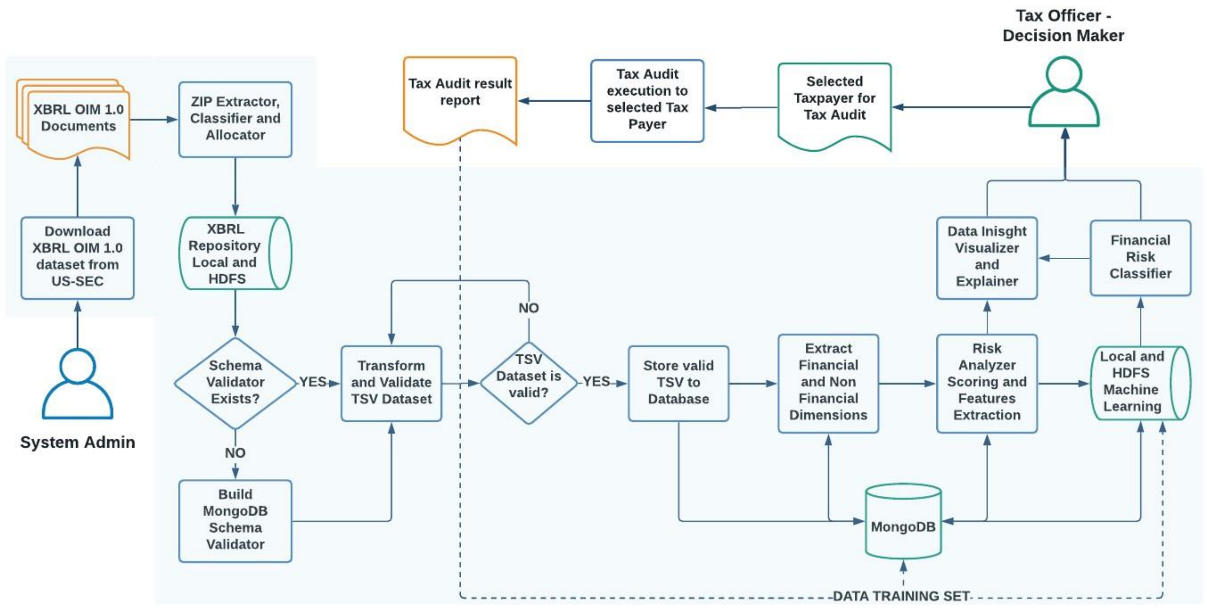
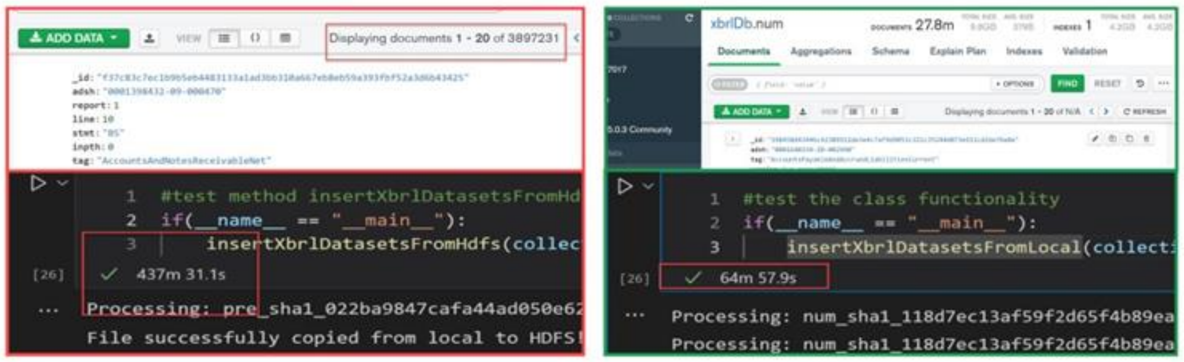

Tax Officer - Decision Maker
XBRL OIM 1.0 Documents
ZIP Extractor, Classifier and Allocator
Tax Audit result report
Tax Audit execution to selected Tax Payer
Selected Taxpayer for Tax Audit
Download XBRL OIM 1.0 dataset from US-SEC
XBRL Repository Local and HDFS
Data Inisght Visualizer and Explainer
Financial Risk Classifier
NO
System Admin
Schema Validator Exists?
YES
Transform and Validate TSV Dataset
TSV Dataset is valid?
YES
Store valid TSV to Database
Extract Financial and Non Financial Dimensions
Risk Analyzer Scoring and Features Extraction
Local and HDFS Machine Learning
NO
Build MongoDB Schema Validator
MongoDB
DATA TRAINING SET

Fig. 4. General Application Framework Diagram and the scope of the project that covered in the blue area



```
_id: "f37c83c7ec1b9b5eb4483113a1ad3bb318a667eb8eb59a393fbf52a3d6b43425"
adsh: "0001398432-09-000470"
report: 1
line: 10
stmt: "BS"
inpth: 0
tag: "AccountsAndNotesReceivableNet"
```

```
#test method insertXbrlDatasetsFromHd
if(__name__ == "__main__"):
    insertXbrlDatasetsFromHdfs(collec
[26] ✓ 437m 31.1s
... Processing: pre_sha1_022ba9847cafa44ad050e62
File successfully copied from local to HDFS!
```

```
#test the class functionality
if(__name__ == "__main__"):
    insertXbrlDatasetsFromLocal(collect
[26] ✓ 64m 57.9s
... Processing: num_sha1_118d7ec13af59f2d65f4b89ea
Processing: num_sha1_118d7ec13af59f2d65f4b89ea
```

Fig. 5. Pre (left) and post (right) optimization data loading comparison

3.4. Software Implementation

3.4.1. XBRL Framework

XBRL Framework is the most prioritized since it provides data sources, split into several features:

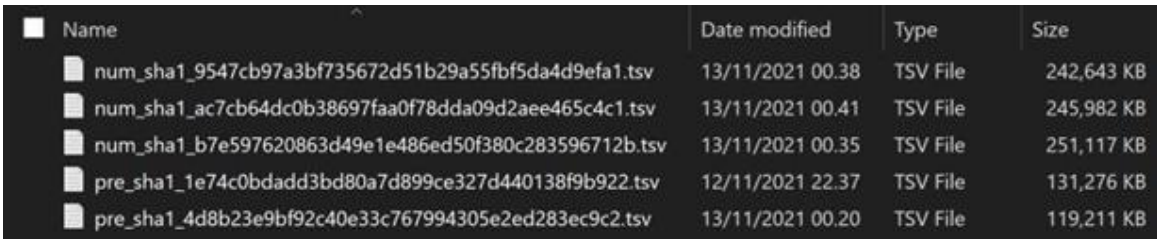
3.4.2. XBRL Repository Feature

HTMLParser.py module collects the XBRL OIM 1.0 documents from the source. Continue with the ZIPParser.py module that extracts the document to the temporary folder and allocate it to Local Storage and HDFS.

Each document hashed using the MD5 or SHA1 algorithm, and the file name is modified to follow this structure to avoid duplication (see Fig. 6).

originalfiletype_hashtype_hashvalue.original extensionname

MD5 and SHA1 are widely used, fast, relatively secure from collision and brute force attacks, with 128 bit (MD5) and 160 bit (SHA1) binary value length, converted to 32 digits (MD5) and 40 digits (SHA1) in hexadecimal format, relatively suitable for filename length (see Fig. 6).



Name	Date modified	Type	Size
num_sha1_9547cb97a3bf735672d51b29a55fbf5da4d9efa1.tsv	13/11/2021 00.38	TSV File	242,643 KB
num_sha1_ac7cb64dc0b38697faa0f78dda09d2aee465c4c1.tsv	13/11/2021 00.41	TSV File	245,982 KB
num_sha1_b7e597620863d49e1e486ed50f380c283596712b.tsv	13/11/2021 00.35	TSV File	251,117 KB
pre_sha1_1e74c0bdadd3bd80a7d899ce327d440138f9b922.tsv	12/11/2021 22.37	TSV File	131,276 KB
pre_sha1_4d8b23e9bf92c40e33c767994305e2ed283ec9c2.tsv	13/11/2021 00.20	TSV File	119,211 KB

Fig. 6. Hashed filename structure convention for content anti-duplication

If a similar-content ZIP document exists in the repository, the extraction process is halted by default, and the error log is created, as illustrated in Fig. 7. The process can be forced to continue if the forced. Continue parameter is set as True.

The complete ZIP, JSON, HTM and TSV files are stored on the Local Repository, while the big-sized TSV files are stored in HDFS. Finally, the XBRLLog.py module logged all activities and information for future analysis.

3.4.3. XBRL Validator feature

MongoDB validation keys are collected by the HTMLParser.py module parallelly. The JSONParser.py module reads the XBRL JSON file, contains schema, table, and column keys (see Fig. 8), and stores it with the MongoDB validation keys. These reserved-key collections are used as reserved keys for the conversion.

The conversion is essential to build document creation validation in the database. XAFR must validate and clean the XBRL documents at each stage to retain correct data types and formats, eliminate null and infinite values, and balance all categories and numerical variables (see Fig. 9).

This procedure is vital for avoiding Garbage-In and Garbage-Out and preserving Great-In and Great Out dataset subsample (Štěpánek *et al.*, 2021). The MongoDB validation is the primary filter to ensure the loaded data is valid. The multi-step transformation procedure is applied (see the yellow highlights box in Fig. 9), generating multiple MongoDB Validation files, one for each collection, stored in the Local Storage and used to MongoDB as a document creation validator.

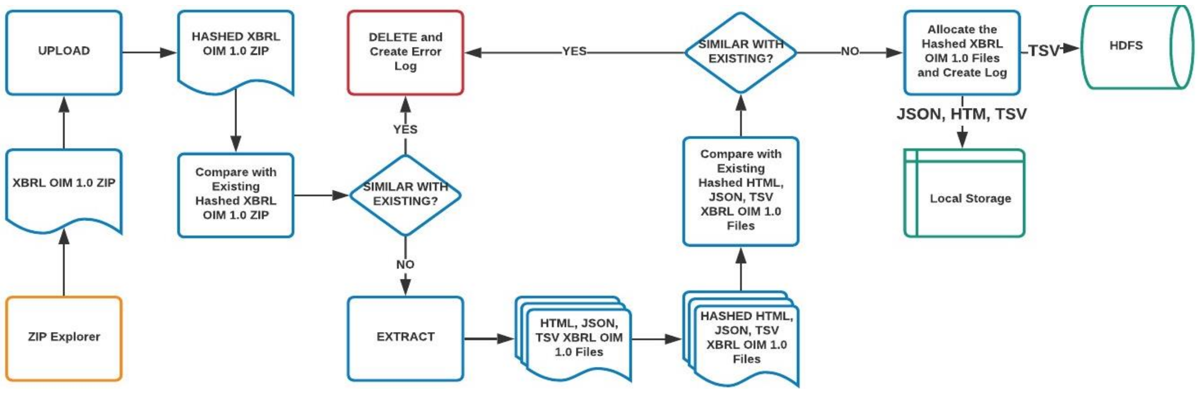
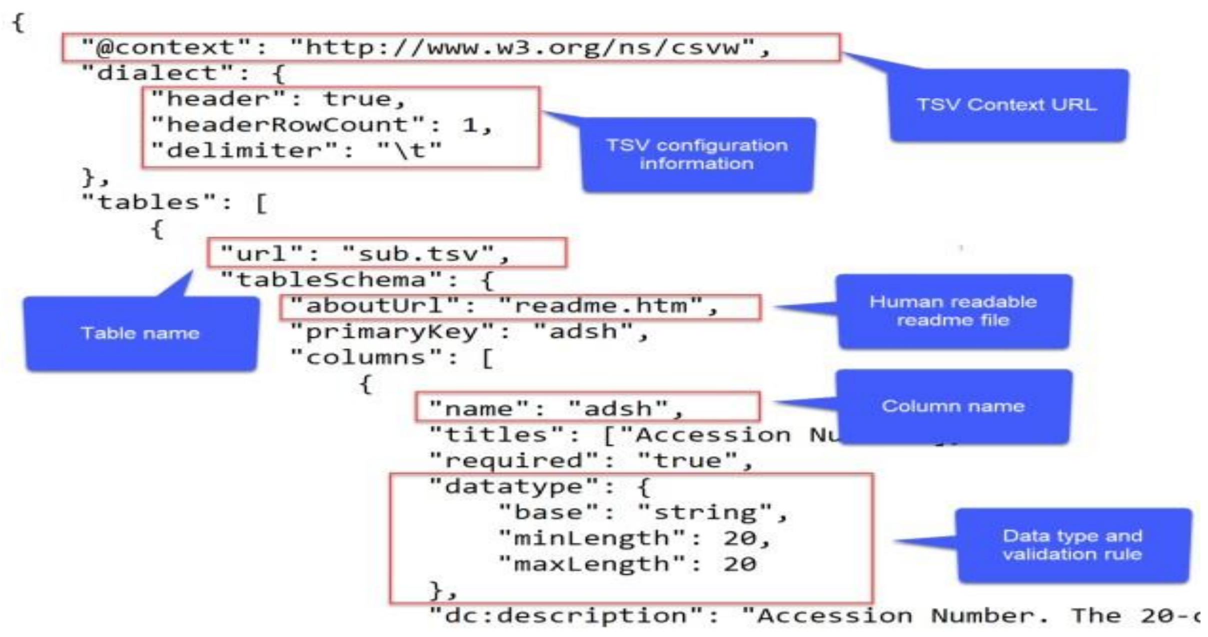

UPLOAD
HASHED XBRL OIM 1.0 ZIP
DELETE and Create Error Log
YES
SIMILAR WITH EXISTING?
NO
Allocate the Hashed XBRL OIM 1.0 Files and Create Log
TSV
HDFS
JSON, HTM, TSV
Local Storage
XBRL OIM 1.0 ZIP
Compare with Existing Hashed XBRL OIM 1.0 ZIP
YES
SIMILAR WITH EXISTING?
NO
Compare with Existing Hashed HTML, JSON, TSV XBRL OIM 1.0 Files
ZIP Explorer
EXTRACT
HTML, JSON, TSV XBRL OIM 1.0 Files
HASHED HTML, JSON, TSV XBRL OIM 1.0 Files

Fig. 7. XBRL Repository feature diagram



```
{
    "@context": "http://www.w3.org/ns/csvw",
    "dialect": {
        "header": true,
        "headerRowCount": 1,
        "delimiter": "\t"
    },
    "tables": [
        {
            "url": "sub.tsv",
            "tableSchema": {
                "aboutUrl": "readme.htm",
                "primaryKey": "adsh",
                "columns": [
                    {
                        "name": "adsh",
                        "titles": ["Accession Nu
                        "required": "true",
                        "datatype": {
                            "base": "string",
                            "minLength": 20,
                            "maxLength": 20
                        },
                        "dc:description": "Accession Number. The 20-c
```

TSV Context URL
TSV configuration information
Table name
Human readable readme file
Column name
Data type and validation rule

Fig. 8. Captured XBRL JSON-LD schema file sample

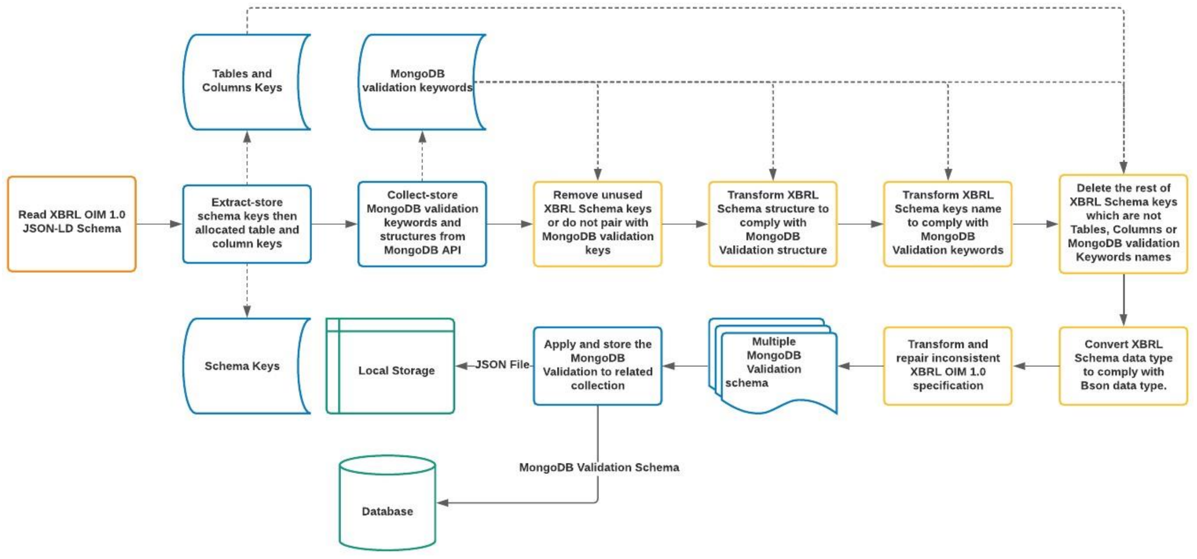

Tables and Columns Keys
MongoDB validation keywords
Read XBRL OIM 1.0 JSON-LD Schema
Extract-store schema keys then allocated table and column keys
Collect-store MongoDB validation keywords and structures from MongoDB API
Remove unused XBRL Schema keys or do not pair with MongoDB validation keys
Transform XBRL Schema structure to comply with MongoDB Validation structure
Transform XBRL Schema keys name to comply with MongoDB Validation keywords
Delete the rest of XBRL Schema keys which are not Tables, Columns or MongoDB validation Keywords names
Schema Keys
Local Storage
JSON File
Apply and store the MongoDB Validation to related collection
Multiple MongoDB Validation schema
Transform and repair inconsistent XBRL OIM 1.0 specification
Convert XBRL Schema data type to comply with Bson data type.
MongoDB Validation Schema
Database

Fig. 9. XBRL Validator feature diagram

3.4.4. XBRL Dataset Loader feature

TSVParser.py module handles the TSV documents to MongoDB loading. The critical step is to chunk the big-sized file into a smaller row (see Fig. 10). The best practice is around 5000-10000 rows per chunk file and avoids total big-sized file row counting ahead. The *TSVparser.py* module also provides documents anti-duplication in the database by hashing each tuple content and storing the hash value as *_id* using the SHA-256 algorithm (see Fig. 11). The *_id* field is a unique indexed document identifier in MongoDB. Finally, *TSVParser.py* stores the dataset using batch mode documents creation, faster than single-mode document creation.

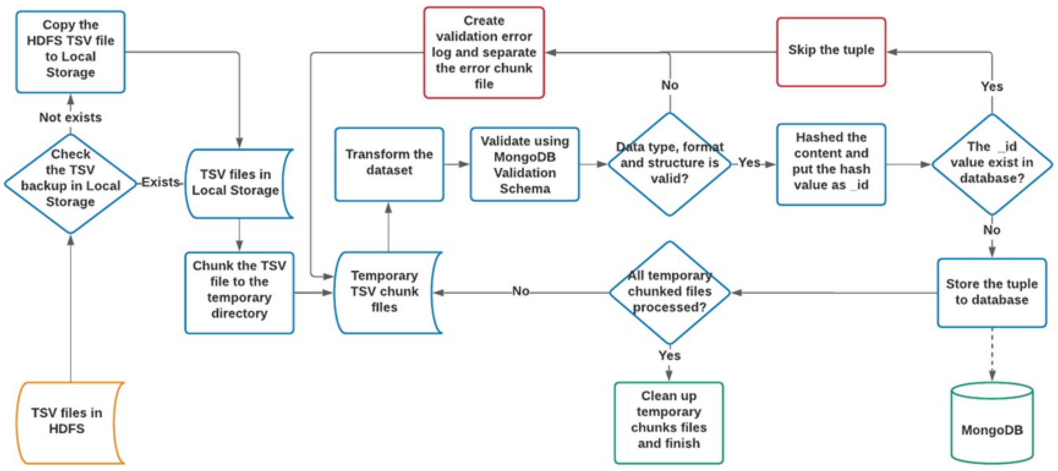
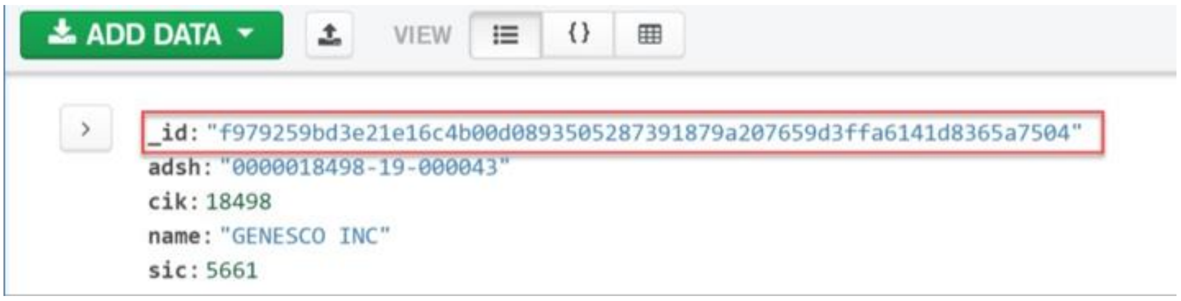

Copy the HDFS TSV file to Local Storage
Not exists
Check the TSV backup in Local Storage
Exists
TSV files in Local Storage
Chunk the TSV file to the temporary directory
TSV files in HDFS
Temporary TSV chunk files
Transform the dataset
Validate using MongoDB Validation Schema
Create validation error log and separate the error chunk file
No
Data type, format and structure is valid?
Yes
Hashed the content and put the hash value as _id
Skip the tuple
Yes
The _id value exist in database?
No
Store the tuple to database
MongoDB
All temporary chunked files processed?
No
Yes
Clean up temporary chunks files and finish

Fig. 10. XBRL Dataset Loader Feature Diagram



```
_id: "f979259bd3e21e16c4b00d0893505287391879a207659d3ffa6141d8365a7504"
adsh: "0000018498-19-000043"
cik: 18498
name: "GENESCO INC"
sic: 5661
```

Fig. 11. Store the content SHA-256 hash value *as _id* for anti-duplication

3.4.5. XBRL Denormalizer feature

The XBRL dataset consists of eight collections (tables): sub for the XBRL report submission and filer identity, tag for all tag used in the request, dim for dimensional tags, num for numeric XBRL facts presented on the financial statements, txt for each non-numeric XBRL plain-text fact, ren for the filing a summary, pre for each line item text, assigned by the filer, in the financial statements, and cal for relationships among the tags in a filing.

The table relationship diagram in Fig. 12 shows that sub, tag, dim or ren is candidates for the pivot point depending on the requirement. XAFR uses Taxpayer as a target module. Thus, the sub is selected as the pivot point to denormalise the others because only the sub contains the filer entity information.

Complete denormalisation is heavy resources consumption process, while XAFR only focuses on the financial indicator, not semantic analysis. Hence, based on thorough research, the collection reduction strategy used only essential collections: *sub*, *tag*, *num*, and *pre*.

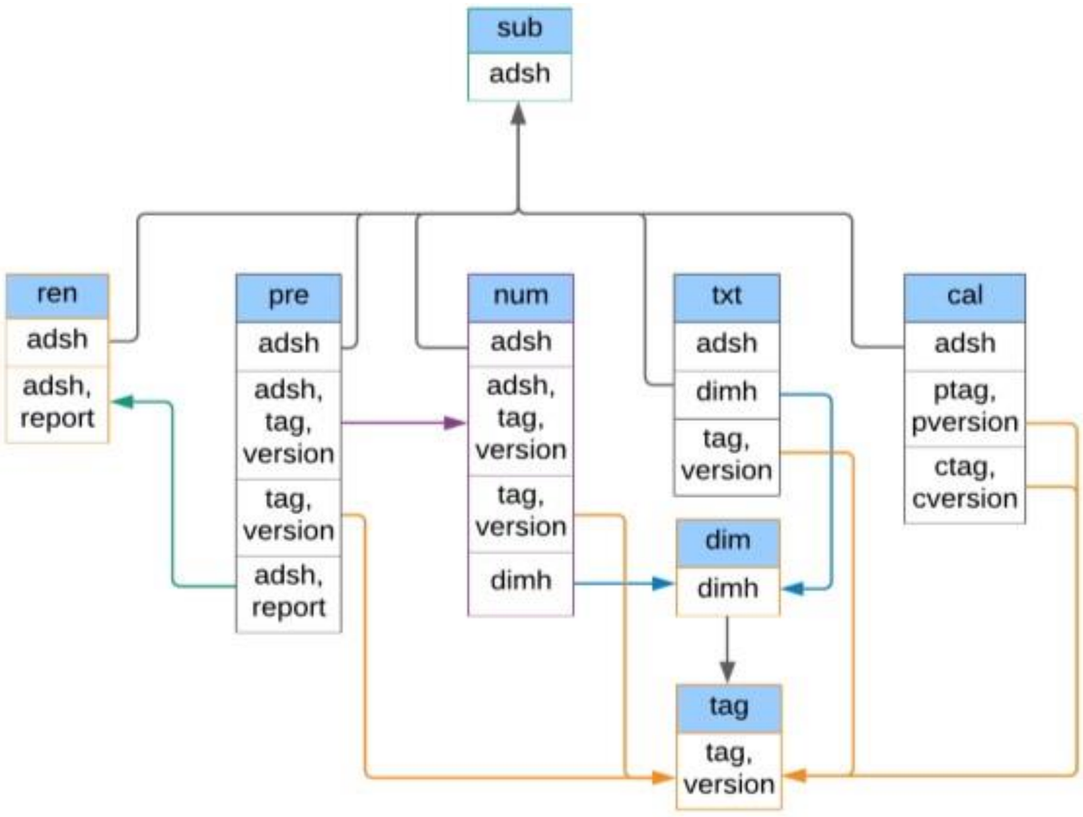


Fig. 12. Table Relationship and Primary Key diagram based on the readme.htm and JSON schema

From the *sub*, Central Index Key (CIK) is extracted. CIK is the SEC filing unique identifier, used as an entity identifier key to collect and denormalise data from the *tag, num* and *pre* using the correct primary key (see Fig. 13). *XBRLDenormalizer.py* module processed it entity by an entity, stored it in the memory, and worked together with *XBRLDimension.py* module to store the denormalised data into the database until all entities were denormalised (see Fig. 13). This procedure ensures the system memory is not overwhelmed and can denormalise entire entities successfully in a faster execution time.

The *XBRLDenormalizer.py* module collaborates with the *XBRLDimension.py* module as one comprehensive unity. *XBRLDenormalizer.py* gathered all CIK, created a single entity as a single object, denormalised other attributes and temporarily stored the data in memory. *XBRLDimension.py* catch and clean the denormalised data from *null*, duplicate tuples (a consequence of the denormalisation process), calculate the financial ratios and trends, collect the nonfinancial values, horizontally merge and store them in the database (see Fig. 14). The financial indicator used in XAFR was the most used in a prior study representing the Taxpayer profitability, liquidity, solvability and efficiency performance. *XBRLDimension.py* module calculates the logarithmic trend of financial values, ratios and trends from the specified and previous year. The logarithmic trend is selected because of its "honesty" in representing the value fluctuation (see Table 2).

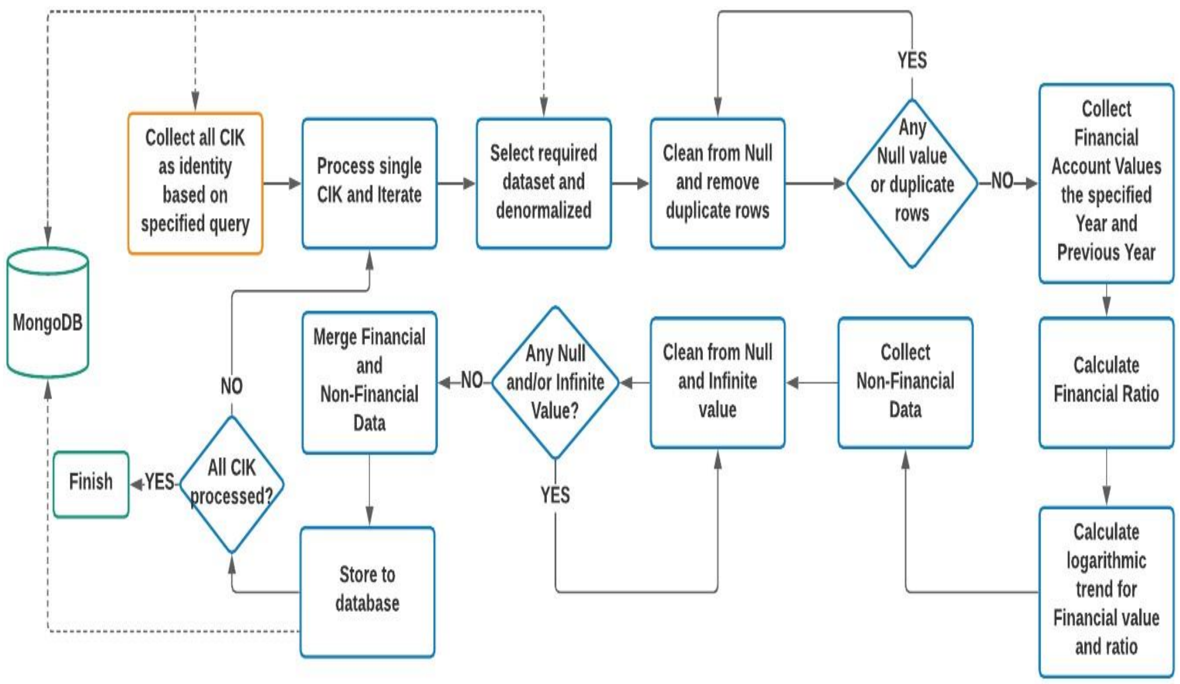

Collect all CIK as identity based on specified query
Process single CIK and Iterate
Select required dataset and denormalized
Clean from Null and remove duplicate rows
YES
Any Null value or duplicate rows
NO
Collect Financial Account Values the specified Year and Previous Year
MongoDB
Merge Financial and Non-Financial Data
NO
Any Null and/or Infinite Value?
Clean from Null and Infinite value
Collect Non-Financial Data
Calculate Financial Ratio
NO
Finish
YES
All CIK processed?
YES
Store to database
Calculate logarithmic trend for Financial value and ratio

Fig. 13. XBRL Denormalizer feature diagram

Table 2. Arithmetic and Logarithmic Trend Comparison. The logarithmic trend is more "honest" in trend fluctuation magnitude and summarisation

Year	Value	Arithmetic Trend	Logarithmic Trend
2018	10000	-	-
2019	1000	-90.00%	-230.26%
2020	500	-50%	-69.31%
2021	1000	100%	69.31%

3.4.6. XBRL Analyzer feature

XBRLAnalyzer.py module collaborates with AI Features-Extraction feature in AI Framework. The *XBRLAnalyzer.py* module build aggregates pipeline and the view from the stored denormalised entities (see Fig. 14). XBRL Analyzer feature enables financial indicator value and ratio benchmarking by comparing and calculating deviation with its competitor average statistic in a similar industry.

3.4.7. XBRL Visualizer feature

The XBRLVisualizer.py module visualizes the data insight into a chart, diagram, plot and table presented for thorough analysis using an interactive graph from Bokeh (Ven, 2021) and a rich feature table from Data tables (SpryMedia, 2021). Figure 15 illustrates how the interactive RCA analysis can help the Tax Officer reveal the high-risk Taxpayer.

3.4.8. Artificial Intelligence (AI) Framework

AI Framework is the core of XAFR Machine Learning classification. AI Framework consists of two features:

3.4.9. AI Features-Extractor feature

The *AIFeatures.py* module executes the rule-based risk scoring. Financial indicators values, trends, and deviations from the average in a similar industry or known standard are used to calculate the risk score (see Fig. 16).

3.4.10. AI Machine-Learning feature

The extracted features by *AIFeatures.py* were treated to eliminate the insignificant features that could reduce the Machine Learning performance handled by *the AIMachine.py* module.

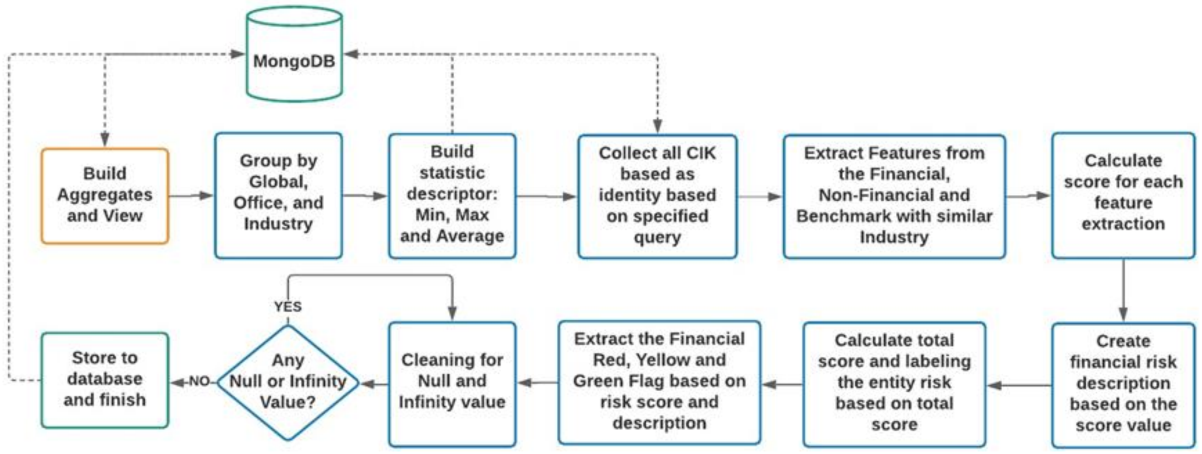
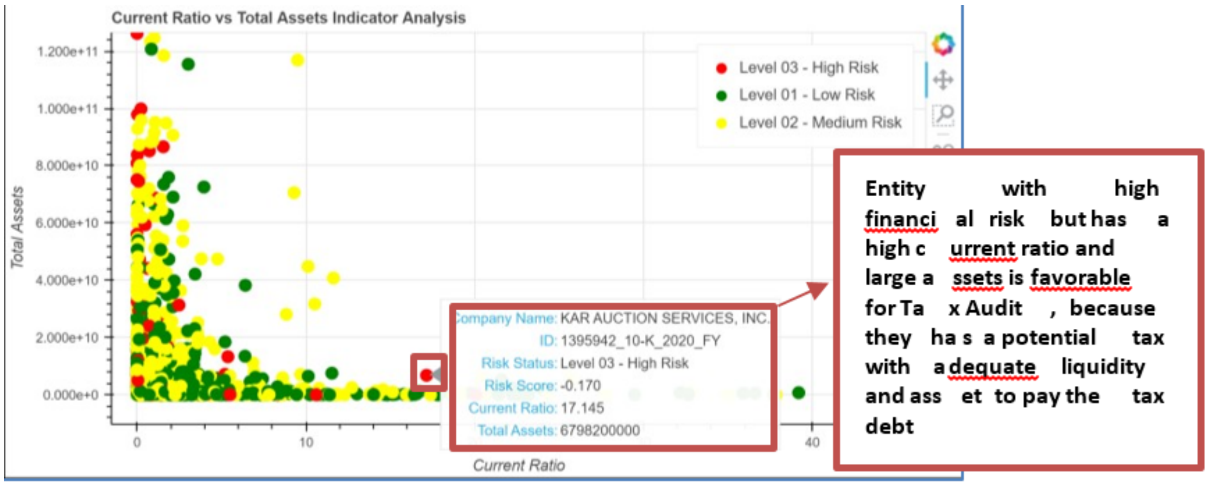

MongoDB
Build Aggregates and View
Group by Global, Office, and Industry
Build statistic descriptor: Min, Max and Average
Collect all CIK based as identity based on specified query
Extract Features from the Financial, Non-Financial and Benchmark with similar Industry
Calculate score for each feature extraction
Create financial risk description based on the score value
Calculate total score and labeling the entity risk based on total score
Extract the Financial Red, Yellow and Green Flag based on risk score and description
Cleaning for Null and Infinity value
Any Null or Infinity Value?
YES
NO
Store to database and finish

Fig. 14. XBRL Analyzer and AI Features-Extraction feature



Company Name: KAR AUCTION SERVICES, INC.
ID: 1395942_10-K_2020_FY
Risk Status: Level 03 - High Risk
Risk Score: -0.170
Current Ratio: 17.145
Total Assets: 6798200000

Entity with high financi al risk but has a high c urrent ratio and large a ssets is favorable for Ta x Audit , because they ha s a potential tax with a dequate liquidity and ass et to pay the tax debt

Fig. 15. XBRL Visualizer feature sample: Interactive Risk Status in Current Ratio vs Total Assets (RCA) Indicator Analysis Scatter Plot

3.4.11. Web Framework

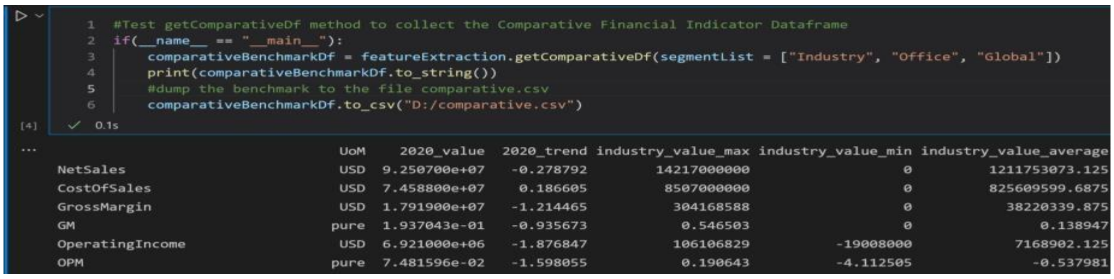
Finally, the calculation and classification result from XBRL and AI Framework is presented in the web application handled by Web Framework. XAFR implements Model View Controller (MVC) pattern for web framework presentation by utilizing Flask for web service and MongoDB for the data model provider. The *WebIndex.py* define all the routes and logic controller function, while the view function would be presented on HTML files in the *templates* folder. XAFR implement a responsive web view to adapt to any device display dynamically.

4. TESTING

4.1. Unit Testing

Unit testing is essential to ensure each method, class, module, and package works as expected. This report conducts unit tests in two approaches. First, use internal iterative testing in *xbrl*, *ai*, and *web* packages in each module's *primary* scope function (see Fig. 16).

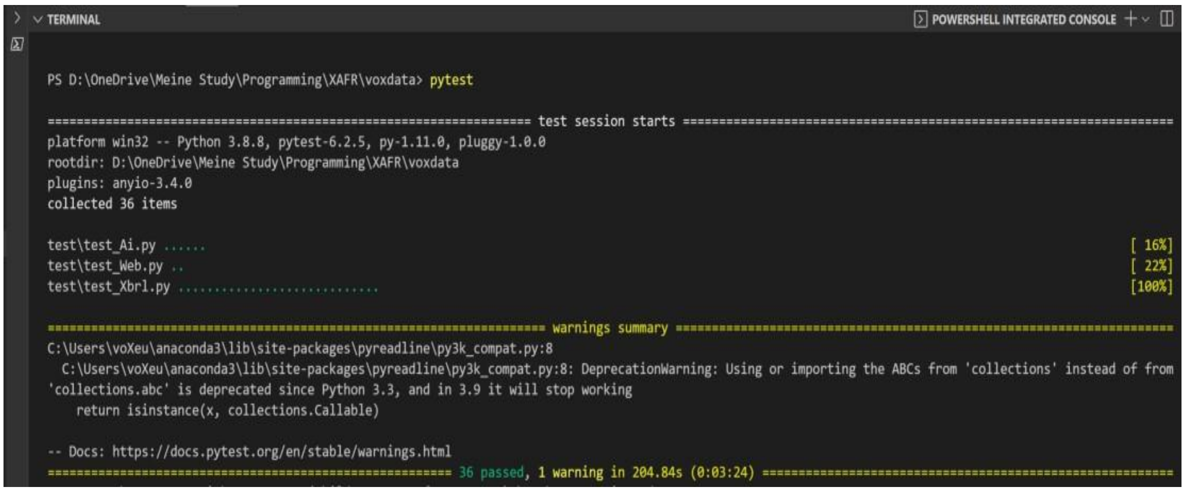
Second, use the *pytest* 6.25 (Krekel, 2021) library and create three-unit test modules: *test_Ai.py*, *test_Xbrl.py* and *test_Web.py* (see Fig. 17) to develop critical checkpoints test for each necessary module functionality.



```
#Test getComparativeDf method to collect the Comparative Financial Indicator Dataframe
if(__name__ == "__main__"):
    comparativeBenchmarkDf = featureExtraction.getComparativeDf(segmentList = ["Industry", "Office", "Global"])
    print(comparativeBenchmarkDf.to_string())
    #dump the benchmark to the file comparative.csv
    comparativeBenchmarkDf.to_csv("D:/comparative.csv")
[4]  ✓  0.1s
```

```
                     UoM     2020_value  2020_trend  industry_value_max  industry_value_min  industry_value_average
NetSales             USD   9.250700e+07   -0.278792         14217000000                   0         1211753073.125
CostOfSales          USD   7.458800e+07    0.186605          8507000000                   0          825609599.6875
GrossMargin          USD   1.791900e+07   -1.214465           304168588                   0           38220339.875
GM                  pure   1.937043e-01   -0.935673            0.546503                   0               0.138947
OperatingIncome      USD   6.921000e+06   -1.876847           106106829           -19008000            7168902.125
OPM                 pure   7.481596e-02   -1.598055            0.190643           -4.112505              -0.537981
```

Fig. 16. Unit testing in *AIFeatures.ipynb* module's *leading* scope example



```
PS D:\OneDrive\Meine Study\Programming\XAFR\voxdata> pytest

================================================================= test session starts ==================================================================
platform win32 -- Python 3.8.8, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: D:\OneDrive\Meine Study\Programming\XAFR\voxdata
plugins: anyio-3.4.0
collected 36 items

test\test_Ai.py ......                                                                                                                            [ 16%]
test\test_Web.py ..                                                                                                                               [ 22%]
test\test_Xbrl.py ............................                                                                                                    [100%]

=================================================================== warnings summary ===================================================================
C:\Users\voXeu\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8
  C:\Users\voXeu\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from
'collections.abc' is deprecated since Python 3.3, and in 3.9 it will stop working
    return isinstance(x, collections.Callable)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
====================================================== 36 passed, 1 warning in 204.84s (0:03:24) =======================================================
```

Fig. 17. Unit testing using pytest

4.2. User Acceptance Test

The User Acceptance Test (UAT) is designed with the Story Cards and MoSCoW analysis. There are thirty UAT for seven-story cards designed. This report conducts UAT in two approaches. First, use a tickmark element checking to ensure the thirty UAT elements exist and working correctly. The detail is in Appendix A.

Second, using the User Survey approach. The survey conducting by demonstrating XAFR to the participant for 10 to 15 minutes then each participant was asked to answer the questionnaire for around 15 minutes. The survey used Google Form (Google, 2021a) from 15 participants. The participant is come from various backgrounds and experiences but is still related to tax, finance, and IT with multiple ranges of age, gender and experience in using Financial Analytical Tools. The participant demography detail is presented in Fig. 18.

Based on the user perspective, in Likert Scale from 1 – Highly Unuseful to 7 – Highly Useful, XAFR usability is 6.24 or helpful in overall features and from 1 – Highly Inaccurate to 7 – Highly Accurate, XAFR accuracy is 5.87 or slightly accurate to accurate in general parts, see table 3 for more detail per feature score. While for the awareness survey, with the scale of 1 for aware, 0 for not sure and -1 for not familiar, the average value is 0.65. From the user perspective, they are 65,6% aware of overall XAFR features.

4.3. System Integration Test

The system integration test is conducted to ensure data integrity is preserved, extracted wholly and correctly, extracted in the correct taxonomy context resulting in a high Non-Zero value, thus generating a high-quality risk score, explanation, benchmark, and rule-based risk classification.

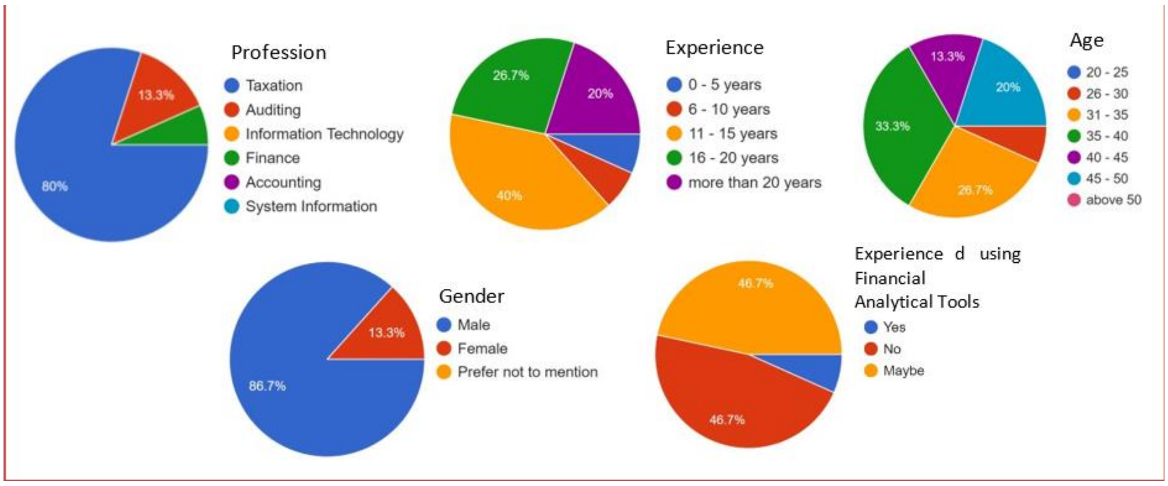


Fig. 18. Participants Demography Statistic

Table 3. Usability and Accuracy Survey Statistic Result

Criteria	1	2	3	4	5	6	7
Usability - Likert Scale from Highly Unuseful (1) to Highly Useful (7)							
Ten Companies diagram Risk				**6.7%**	**13.3%**	**40%**	**40%**
Ten Industries diagram Risk				**6.7%**	**6.7%**	**60%**	**26.7%**
Risk Distribution Diagram			**7.1%**	**7.1%**	**7.1%**	**50%**	**28.6%**
RCA (Risk, Current Ratio and Asset) Diagram				**7.1%**	**14.3%**	**42.9%**	**35.7%**
Taxpayer List				**6.7%**	**13.3%**	**40%**	**40%**
Taxpayer Detail				**6.7%**	**13.3%**	**46.7%**	**33.3%**
XBRL Report - EDGAR Archive			**6.7%**	**13.3%**	**13.3%**	**46.7%**	**20%**
Benchmark Financial Indicator Diagram		**7.1%**		**7.1%**	**14.3%**	**35.7%**	**35.7%**
Benchmark Financial Indicator Table				**6.7%**	**20%**	**33.3%**	**40%**
Risk Explanation			**6.7%**	**6.7%**	**6.7%**	**46.7%**	**33.3%**
Financial Indicator Sorting				**6.7%**	**20%**	**40%**	**33.3%**
Machine Learning Prediction				**13.3%**	**33.3%**	**20%**	**33.3%**
Average Score		**7.1%**	**6.8%**	**7.9%**	**14.6%**	**41.8%**	**33.3%**
Weighted Total Score for overall Features							**6.24**
Accuracy – Likert Scale from Highly Inaccurate (1) to Highly Accurate (7)							
Risk Classification of the Blue Chips Taxpayers			**6.7%**	**26.7%**	**20%**	**33.3%**	**13.3%**
Risk Explanation			**6.7%**	**13.3%**	**13.3%**	**33.3%**	**33.3%**
Industries List				**6.7%**	**13.3%**	**46.7%**	**33.3%**
Industries Risk				**6.7%**	**33.3%**	**26.7%**	**33.3%**
Financial Indicator Sorting				**6.7%**	**13.3%**	**40%**	**40%**
Machine Learning Prediction				**26.7%**	**20%**	**33.3%**	**20%**
Average Score			**6.7%**	**14.4%**	**18.8%**	**35.5%**	**28.8%**
Weighted Total Score for overall Features							**5.87**

4.4. XBRL ETL Succession Rate Test

This test measures the completeness of data extracted, transformed and loaded to the database. The *integrationTest.py - testXbrlEtlSuccessionRate()* method handle this test by comparing the total tuples from each collection TSV with the entire valid document stored in the database. The succession rate is 100% for *num*, *pre* and *sub*, but the rate of *tag* collection is only 91.01% (see Fig. 19).

The tag collection can duplicate because it contains the available XBRL taxonomy tag, and when two or more entities have the same event context, they will use an equal tag. Hence, the 91.01% ETL succession rate is acceptable (see Fig. 20).

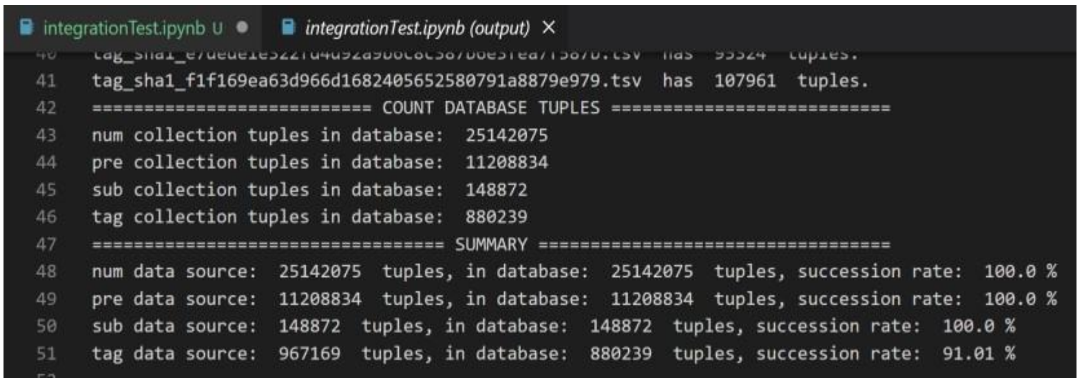
4.5. Non-Zero Value (NZV) Extraction Rate Test

This test measures how many non-zero extracted financial indicators are for each Taxpayer. If the number of extracted features is 31, the non-zero value rate varies between 0 to 31.

The origin of the zero value can come from the objective 0 value or the *null* value converted to 0. The *null* value was generated because the algorithm failed to read the taxonomy context. Pre-optimization, XAFR had a poor NZV extraction rate, see Fig. 21.

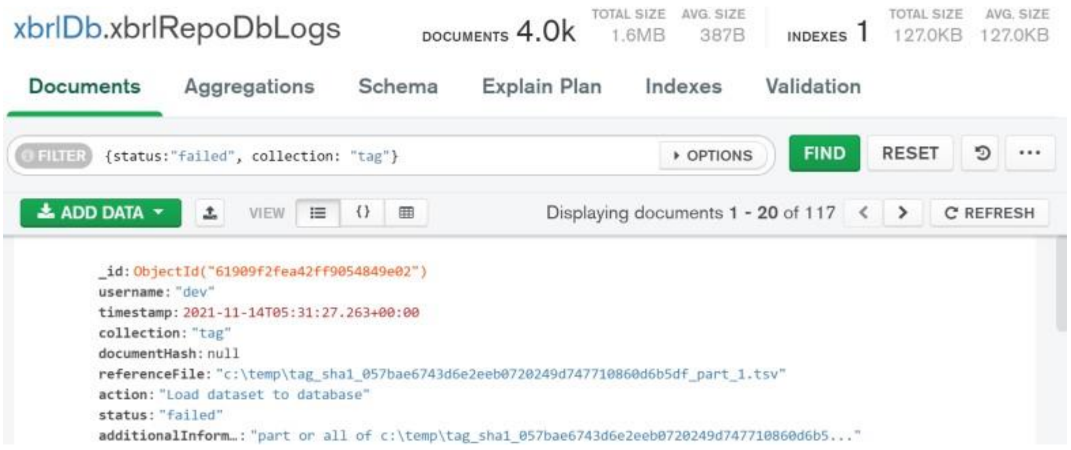
There are three essential fields to read XBRL taxonomy context: *stmt, crdr* and *tag* fields. For example, locating operating income is started by querying the stmt value with "IS" for Income Statement. The *crdr* value with "C" for Credit (normal balance for operating income) and *tag* with regex keyword value indicate operating income.

An in-depth evaluation was conducted to improve the rate by comparing the Taxpayer that used the reserved accounting standard taxonomy with the custom one and creating a dictionary to capture the keyword for custom taxonomy context optimisation. Finally, the NZV Extraction rate testing significantly increased using the custom taxonomy context optimisation (see Fig. 20 and Table 4).



```
tag_sha1_f1f169ea63d966d1682405652580791a8879e979.tsv  has  107961  tuples.
=========================== COUNT DATABASE TUPLES ===========================
num collection tuples in database:  25142075
pre collection tuples in database:  11208834
sub collection tuples in database:  148872
tag collection tuples in database:  880239
================================== SUMMARY ==================================
num data source:  25142075  tuples, in database:  25142075  tuples, succession rate:  100.0 %
pre data source:  11208834  tuples, in database:  11208834  tuples, succession rate:  100.0 %
sub data source:  148872  tuples, in database:  148872  tuples, succession rate:  100.0 %
tag data source:  967169  tuples, in database:  880239  tuples, succession rate:  91.01 %
```

Fig. 19. XBRL ETL Succession Rate test



```
{status:"failed", collection: "tag"}
Displaying documents 1 - 20 of 117

_id: ObjectId("61909f2fea42ff9054849e02")
username: "dev"
timestamp: 2021-11-14T05:31:27.263+00:00
collection: "tag"
documentHash: null
referenceFile: "c:\temp\tag_sha1_057bae6743d6e2eeb0720249d747710860d6b5df_part_1.tsv"
action: "Load dataset to database"
status: "failed"
additionalInform…: "part or all of c:\temp\tag_sha1_057bae6743d6e2eeb0720249d747710860d6b5..."
```

Fig. 20. XBRL Dataset to Database Load Process Log

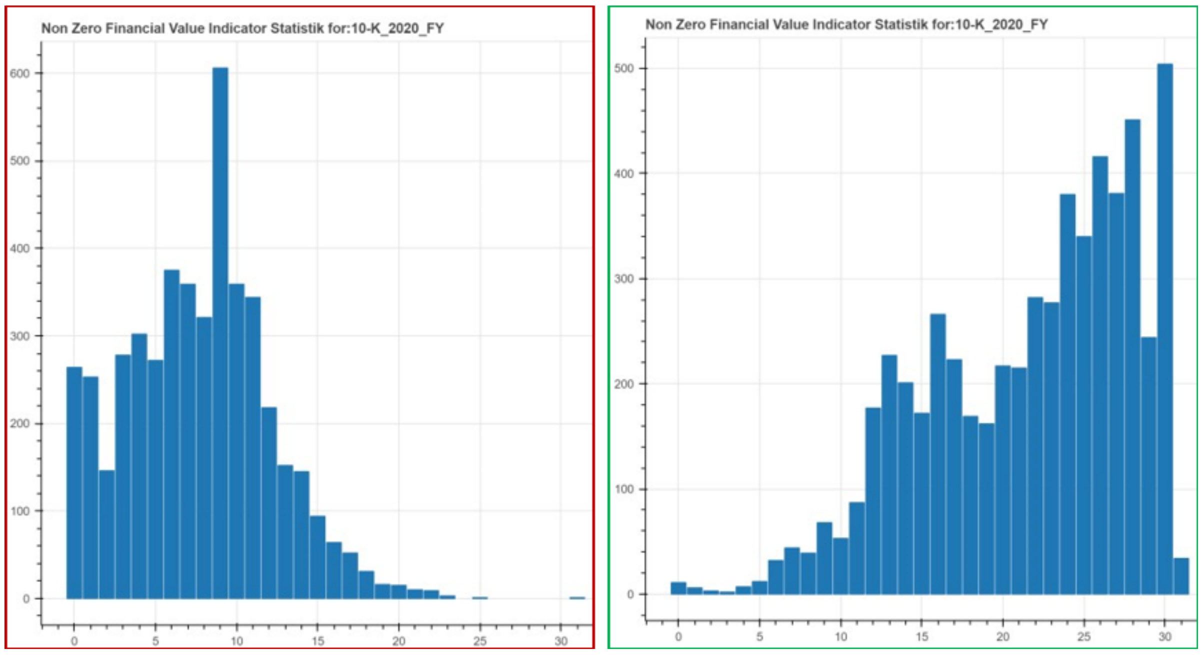


Fig. 21. Non-Zero-Value Distribution Comparison, before custom dictionary keywords optimisation (left-red border) and after optimisation (right-green border)

Table 4. NZV Extraction Rate Comparison before and after optimisation

Condition	NZV Above Threshold Entities Counts	Total Entities	NZV Extraction Rate
Before Optimization	39	5178	0.6%
After Optimization	3217	5178	62.1%

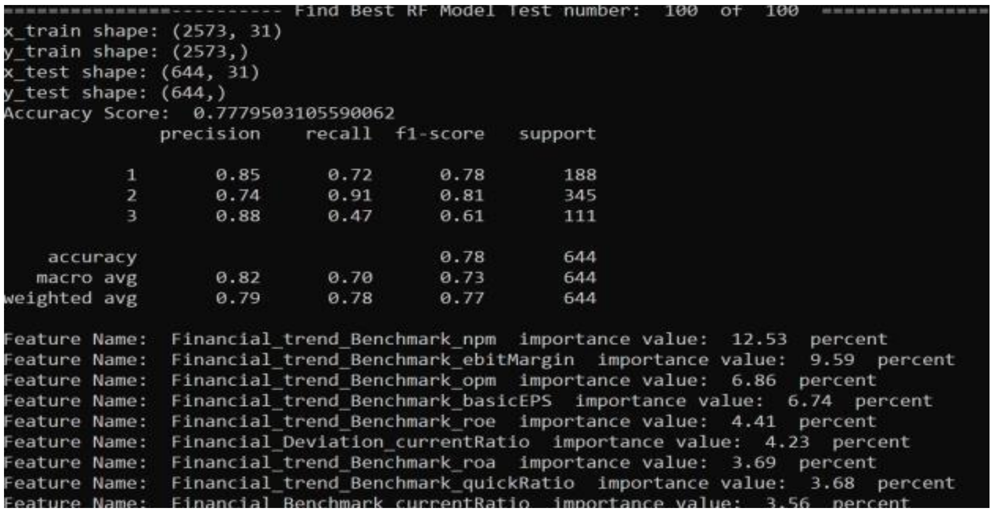
4.6. Feature Reduction Rate Test

XAFR calculate the matrix correlation and display the information in a heatmap diagram. The threshold is ninety per cent. If features correlate equal to or greater than ninety per cent, only one part is preserved (see Table 5).

Table 6 informs that mostly the industry-level benchmark's features remain. Besides the prior study that reveals the importance of the industry level benchmark for machine learning performance, the other rationale is the Random Forest Feature Importance test. While performing 100 iterations to find the best Random Forest model, the Features Importance test was also executed. From 100 iterations, it was found that the benchmark features are consistently in the top ten ranks of the most crucial part (see Fig. 22).

Table 5. High Correlation Features List, feature with red colour is eliminated

Feature	Feature Pair	Correlation
Financial_trend_opm	Financial_trend_gm_opm	94%
Financial_trend_roa	Financial_trend_assetTurnover	99%
Financial_benchmark_quickRatio	Financial_deviation_quickRatio	92%
Financial_benchmark_der	Financial_deviation_der	94%
Financial_trend_benchmark_ebitMargin	Financial_trend_ebitMargin	96%
Financial_trend_benchmark_roa	Financial_trend_benchmark_assetTurnover	97%
Financial_trend_benchmark_opm	Financial_trend_gm_opm	90%
Financial_trend_benchmark_gm	Financial_trend_gm	94%
Financial_trend_benchmark_opm	Financial_trend_opm	95%
Financial_trend_benchmark_ebitMargin	Financial_trend_ebitMargin	96%
Financial_trend_benchmark_npm	Financial_trend_npm	96%
Financial_trend_benchmark_basicEPS	Financial_trend_basicEPS	95%
Financial_trend_benchmark_quickRatio	Financial_trend_quickRatio	93%
Financial_trend_benchmark_currentRatio	Financial_trend_currentRatio	94%
Financial_trend_benchmark_roa	Financial_trend_roa	92%
Financial_trend_benchmark_roe	Financial_trend_roe	95%
Financial_trend_benchmark_der	Financial_trend_der	95%
Financial_trend_benchmark_roa	Financial_trend_assetTurnover	91%
Financial_trend_benchmark_assetTurnover	Financial_trend_roa	93%
Financial_trend_benchmark_assetTurnover	Financial_trend_assetTurnover	94%



```
======================---------- Find Best RF Model Test number:  100  of  100  ===============
x_train shape: (2573, 31)
y_train shape: (2573,)
x_test shape: (644, 31)
y_test shape: (644,)
Accuracy Score:  0.7779503105590062
              precision    recall  f1-score   support

           1       0.85      0.72      0.78       188
           2       0.74      0.91      0.81       345
           3       0.88      0.47      0.61       111

    accuracy                           0.78       644
   macro avg       0.82      0.70      0.73       644
weighted avg       0.79      0.78      0.77       644

Feature Name:  Financial_trend_Benchmark_npm  importance value:  12.53  percent
Feature Name:  Financial_trend_Benchmark_ebitMargin  importance value:  9.59  percent
Feature Name:  Financial_trend_Benchmark_opm  importance value:  6.86  percent
Feature Name:  Financial_trend_Benchmark_basicEPS  importance value:  6.74  percent
Feature Name:  Financial_trend_Benchmark_roe  importance value:  4.41  percent
Feature Name:  Financial_Deviation_currentRatio  importance value:  4.23  percent
Feature Name:  Financial_trend_Benchmark_roa  importance value:  3.69  percent
Feature Name:  Financial_trend_Benchmark_quickRatio  importance value:  3.68  percent
Feature Name:  Financial_Benchmark_currentRatio  importance value:  3.56  percent
```

Fig. 22. Random Forest Feature Importance Test screenshot

4.7. Target Class Distribution Test

XAFR classifies the Taxpayers into three classes: Low Risk labelled as 1, Medium Risk labelled as two and Low Risk marked as 1. The class name and label is defined based on the risk score calculation and threshold.

The risk score threshold should classify the class to mimic the actual distribution in reality as close as possible. After conducting several tests, it was found that the best threshold is the 15% Medium Risk threshold from -0.15 to 0.15 (see Table 6).

The 15% Medium Risk threshold will classify every indicator as a medium risk if the standard or average industry level benchmark deviation is between -15% to 15%.

For example, if the average Net Profit Margin in a similar industry is 20%, the medium risk is located between 5% to 35%. If less than 5%, then classified as High Risk and if greater than 35%, classified as Low Risk. This formula is the XAFR rule-based risk scoring approach (see Fig. 23).

4.8. Accuracy Score Test

Finally, the accuracy score test is the ultimate test to measure Machine Learning performance. XAFR uses z-score normalisation and Hyperparameter to select the best SVM and RF model parameters to improve the Machine Learning accuracy score. The result of the Hyperrparameter test is in Fig. 24 and applied in each classifier model parameter.

Table 6. Rule-based risk scoring technique

Risk Score Range	Risk Status	Risk Label	Explanation Flag
-1 <= x < -0.15	High Risk	3	Red Flag
-0.15 <= x <= 0.15	Medium Risk	2	Yellow Flag
0.15 < x <= 1	Low Risk	1	Green Flag

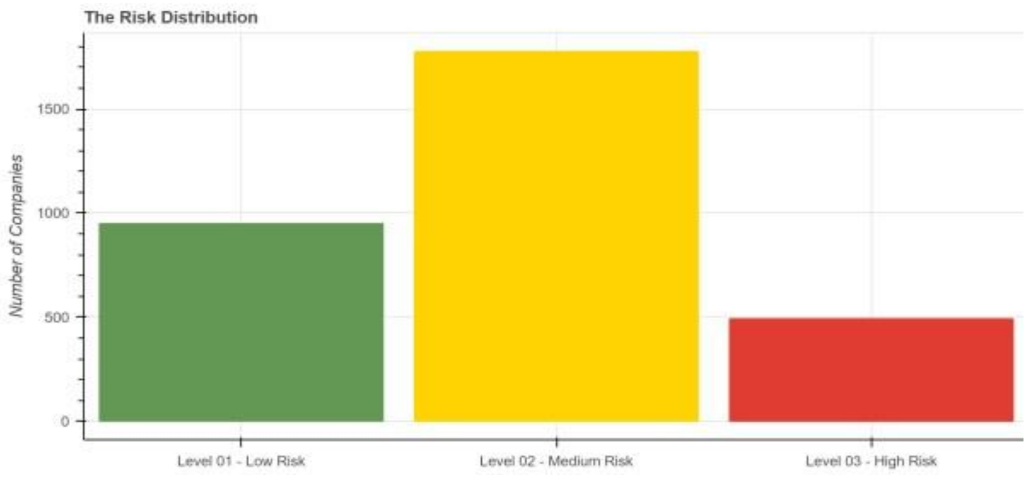
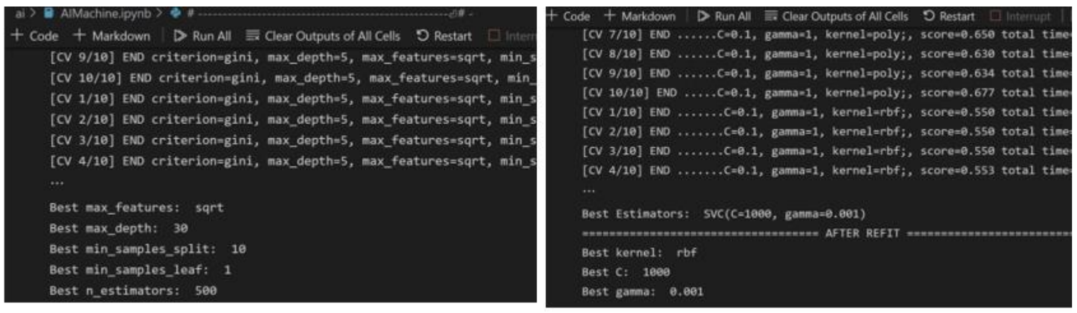


Fig. 23. Target class distribution



```
[CV 9/10] END criterion=gini, max_depth=5, max_features=sqrt, min_s
[CV 10/10] END criterion=gini, max_depth=5, max_features=sqrt, min_
[CV 1/10] END criterion=gini, max_depth=5, max_features=sqrt, min_s
[CV 2/10] END criterion=gini, max_depth=5, max_features=sqrt, min_s
[CV 3/10] END criterion=gini, max_depth=5, max_features=sqrt, min_s
[CV 4/10] END criterion=gini, max_depth=5, max_features=sqrt, min_s
...

Best max_features:  sqrt
Best max_depth:  30
Best min_samples_split:  10
Best min_samples_leaf:  1
Best n_estimators:  500
```

```
[CV 7/10] END ......C=0.1, gamma=1, kernel=poly;, score=0.650 total time
[CV 8/10] END ......C=0.1, gamma=1, kernel=poly;, score=0.630 total time
[CV 9/10] END ......C=0.1, gamma=1, kernel=poly;, score=0.634 total time
[CV 10/10] END .....C=0.1, gamma=1, kernel=poly;, score=0.677 total time
[CV 1/10] END .......C=0.1, gamma=1, kernel=rbf;, score=0.550 total time
[CV 2/10] END .......C=0.1, gamma=1, kernel=rbf;, score=0.550 total time
[CV 3/10] END .......C=0.1, gamma=1, kernel=rbf;, score=0.550 total time
[CV 4/10] END .......C=0.1, gamma=1, kernel=rbf;, score=0.553 total time
...

Best Estimators:  SVC(C=1000, gamma=0.001)
================================ AFTER REFIT ========================
Best kernel:  rbf
Best C:  1000
Best gamma:  0.001
```

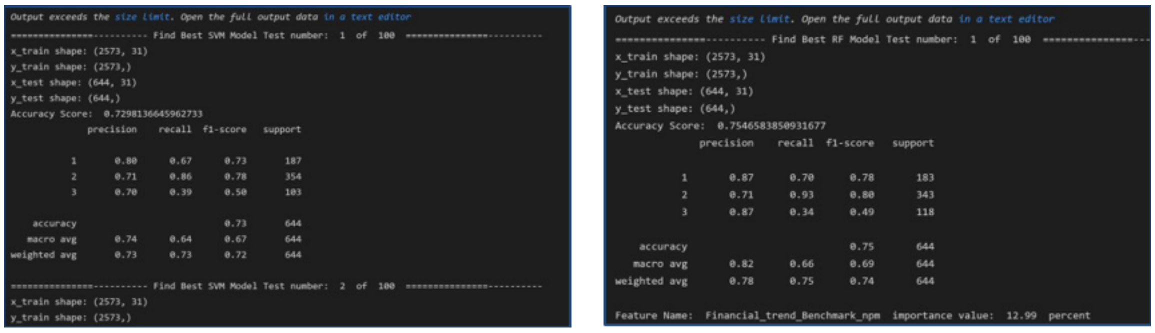
Fig. 24. Hyperparameter Test Result. Random Forest model (Left) and Support Vector Machine (Right)

Furthermore, the test to find the best classifier model is conducted by iterating a hundred times for both SVM and RF models (see Fig. 25).

The model is dumped into the *pickle* file at the end of each iteration. While the accuracy score is put to the end of the filename using this structure:

classifiertype_ModelPreedictor_accuracyscore.pkl

At the end of the test, the best model can be carried out by selecting the *pickle* file with the highest accuracy score number and using that model for Machine Learning risk classification prediction (see Fig. 26).



```
Output exceeds the size limit. Open the full output data in a text editor
===============---------- Find Best SVM Model Test number:  1  of  100  ===============----------
x_train shape: (2573, 31)
y_train shape: (2573,)
x_test shape: (644, 31)
y_test shape: (644,)
Accuracy Score:  0.7298136645962733
              precision    recall  f1-score   support

           1       0.80      0.67      0.73       187
           2       0.71      0.86      0.78       354
           3       0.70      0.39      0.50       103

    accuracy                           0.73       644
   macro avg       0.74      0.64      0.67       644
weighted avg       0.73      0.73      0.72       644

===============---------- Find Best SVM Model Test number:  2  of  100  ===============----------
x_train shape: (2573, 31)
y_train shape: (2573,)
```

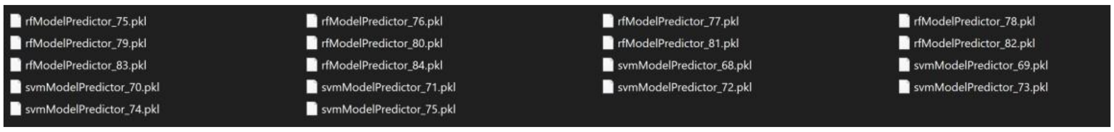
```
Output exceeds the size limit. Open the full output data in a text editor
===============---------- Find Best RF Model Test number:  1  of  100  ===============---
x_train shape: (2573, 31)
y_train shape: (2573,)
x_test shape: (644, 31)
y_test shape: (644,)
Accuracy Score:  0.7546583850931677
              precision    recall  f1-score   support

           1       0.87      0.70      0.78       183
           2       0.71      0.93      0.80       343
           3       0.87      0.34      0.49       118

    accuracy                           0.75       644
   macro avg       0.82      0.66      0.69       644
weighted avg       0.78      0.75      0.74       644

Feature Name:  Financial_trend_Benchmark_npm  importance value:  12.99  percent
```

Fig. 25. Find best classifier model test



```
rfModelPredictor_75.pkl     rfModelPredictor_76.pkl     rfModelPredictor_77.pkl     rfModelPredictor_78.pkl
rfModelPredictor_79.pkl     rfModelPredictor_80.pkl     rfModelPredictor_81.pkl     rfModelPredictor_82.pkl
rfModelPredictor_83.pkl     rfModelPredictor_84.pkl     svmModelPredictor_68.pkl    svmModelPredictor_69.pkl
svmModelPredictor_70.pkl    svmModelPredictor_71.pkl    svmModelPredictor_72.pkl    svmModelPredictor_73.pkl
svmModelPredictor_74.pkl    svmModelPredictor_75.pkl
```

Fig. 26. Selecting The Best Classifier Model

3. EVALUATION

XAFR passes all Unit Test using two approaches, completed all the User Acceptance Test and from the survey participant perspective, XAFR features is proper and slightly accurate to accurate. XAFR features had a high awareness from the survey participant. System integration tests show that XAFR has a high XBRL ETL Succession Rate Test, and significantly improves the Non-Zero Value (NZV) Extraction Rate Test and found the root cause of the problem.

The high Feature Reduction Rate Test shows that XAFR can recognize and eliminate insignificant features and reduce multicollinearity. The Target Class Distribution Test can reveal the best threshold to produce a risk distribution class that mimics the reality as possible, and The Accuracy Test shows Random Forest as the best classifier model.

XAFR can significantly improve the overall performance by enriching the Taxonomy Context Dictionary to recognize more financial indicators, especially for Taxpayer that uses custom taxonomy.

4. CONCLUSION

XAFR successfully expose the high financial risk taxpayers capable of paying their tax using the RCA analysis and SIC utilization using rule-based risk scoring and machine learning risk classification and can explain the risk classification for Risk-Based Tax Audit Selection. It can model the latest XBRL OIM 1.0 specification processing to generate meaningful information for Tax Authorities. XAFR can process forty million tuples of data to produce insightful information in a minimalist but informative, responsive and interactive display.

XBRL OIM 1.0 dataset contain rich structured and semi-structured meaningful data. It is a treasure for semantic analysis using the Natural Language Processing (NLP) model, providing better risk explanation and risk classification performance. For example, the financial indicator might calculate as medium risk. However, the NLP report analysis found a significant number of sentiment negative words: "fraud", "loss" which located near words: "exposed", "mitigated", "recognized", "followed", and "year "then it could be classified as high-risk semantically.

Furthermore, predicting the range of possible tax debt paid from the specified XBRL financial statements using the regression predictor model is significant for the future development of Machine Learning in this area. For example, at every start of the year, based on the last Taxpayer's financial indicator, the Tax Officer could predict using Machine Learning how much is the tax debt for the recent year if everything is in similar condition. Then, at the end of the year, the tax debt prediction is compared with the prediction using the actual Financial

Statement indicator and the actual tax paid. If the difference is significant, it will trigger a warning alarm sent to Taxpayer and Tax Authority. If there is no additional tax payment or clarification from the Taxpayer in any regulated period, then the Taxpayer is automatically selected for Tax Audit.

REFERENCES

Abbasi, A., Albrecht, C., Vance, A., & Hansen, J. (2012). Metafraud: a meta-learning framework for detecting financial fraud. *Mis Quarterly*, 1293-1327.

Amazon (2021) *Cloud Services - Amazon Web Services (AWS)*, *Amazon Web Services, Inc.*

Amunategui, M., & Roopaei, M. (2018). Displaying Predictions with Google Maps on Azure. In *Monetizing Machine Learning* (pp. 195-235). Apress, Berkeley, CA.

Anaconda (2021) *Anaconda with Python 3 on 64-bit Windows — Anaconda documentation*.

Anggia, P. (2019). Achieving of income tax with awareness of taxation in Indonesia's tax law system. *Yustisia Jurnal Hukum*, *8*(2), 292-308.

Apache Software Foundation (2021a) *Apache Hadoop*. Available at: https://hadoop.apache.org/ (Accessed: 31 October 2021).

Apache Software Foundation (2021b) *Apache Spark™ - Unified Engine for largescale data analytics*. Available at: https://spark.apache.org/ (Accessed: 31 October 2021).

Ashtiani, M. N., & Raahemi, B. (2021). Intelligent fraud detection in financial statements using machine learning and data mining: a systematic literature review. *IEEE Access*.

Bootstrap *et al.* (2021) *Bootstrap*. Available at: https://getbootstrap.com/ (Accessed: 31 October 2021).

Canonical (2021) *Enterprise Open Source and Linux*, *Ubuntu*. Available at: https://ubuntu.com/ (Accessed: 31 October 2021).

Carroll, J., & Morris, D. (2015). *Agile project management in easy steps*. In Easy Steps.

Central Bureau of Statistics of the Republic of Indonesia (2021) *Economic and Finance Publication*. Central Bureau of Statistics of the Republic of Indonesia. Available at: https://www.bps.go.id/site/resultTab (Accessed: 30 August 2021).

El-Bannany, M., Dehghan, A. H., & Khedr, A. M. (2021, March). Prediction of Financial Statement Fraud using Machine Learning Techniques in UAE. In *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)* (pp. 649-654). IEEE.

GmbH (2021) *The QR Code Generator*, *The QR Code Generator*. Available at: https://www.the-qrcode-generator.com/ (Accessed: 17 December 2021).

Gomaa, M. I., Markelevich, A., & Shaw, L. (2011). Introducing XBRL through a financial statement analysis project. *Journal of Accounting Education*, *29*(2-3), 153-173.

DOI: https://doi.org/10.34010/injiiscom.v3i1.6891
p-ISSN 2810-0670 e-ISSN 2775-5584

Google (2021a) *Google Forms*. Available at: https://docs.google.com/forms/ (Accessed: 17 December 2021).

Google (2021b) *YouTube*. Available at: https://www.youtube.com/ (Accessed: 17 December 2021).

HashiCorp (2021) *Vagrant by HashiCorp, Vagrant by HashiCorp*. Available at: https://www.vagrantup.com/ (Accessed: 31 October 2021).

Hidayattullah, S., Surjandari, I., & Laoh, E. (2020, October). Financial Statement Fraud Detection in Indonesia Listed Companies using Machine Learning based on Meta-Heuristic Optimization. In *2020 International Workshop on Big Data and Information Security (IWBIS)* (pp. 79-84). IEEE.

Hooda, N., Bawa, S., & Rana, P. S. (2020). Optimizing fraudulent firm prediction using ensemble machine learning: a case study of an external audit. *Applied Artificial Intelligence*, *34*(1), 20-30.

Joblib (2021) *Joblib: running Python functions as pipeline jobs — joblib 1.2.0.dev0 documentation*. Available at: https://joblib.readthedocs.io/en/latest/ (Accessed: 31 October 2021).

Jurney, R. (2017). *Agile data science 2.0: Building full-stack data analytics applications with Spark*. " O'Reilly Media, Inc.".

Khwaja, M. S., Awasthi, R., & Loeprick, J. (Eds.). (2011). *Risk-based tax audits: Approaches and country experiences*. World Bank Publications.

Kotsiantis, S., & Kanellopoulos, D. (2008, November). Multi-instance learning for predicting fraudulent financial statements. In *2008 Third International Conference on Convergence and Hybrid Information Technology* (1), 448-452. IEEE.

Krekel, H. (2021) *pytest: helps you write better programs — pytest documentation*. Available at: https://docs.pytest.org/en/6.2.x/ (Accessed: 31 October 2021).

Microsoft (2021a) *Explore Windows 11 OS, Computers, Apps, & More Microsoft, Windows*. Available at: https://www.microsoft.com/en-gb/windows (Accessed: 31 October 2021).

Microsoft (2021b) *Visual Studio Code - Code Editing. Redefined*. Available at: https://code.visualstudio.com/ (Accessed: 31 October 2021).

MongoDB (2021) *MongoDB: the application data platform, MongoDB*. Available at: https://www.mongodb.com (Accessed: 31 October 2021).

Oracle (2021a) *JDK 11*. Available at: https://openjdk.java.net/projects/jdk/11/ (Accessed: 31 October 2021).

Oracle (2021b) *Oracle VM VirtualBox*. Available at: https://www.virtualbox.org/ (Accessed: 31 October 2021).

Pallets (2021) *Welcome to Flask — Flask Documentation (2.0.x)*. Available at: https://flask.palletsprojects.com/en/2.0.x/ (Accessed: 31 October 2021).

Relan, K. (2019). Beginning with flask. In *Building REST APIs with Flask* (pp. 1-26). Apress, Berkeley, CA.

SEC, S. (2021) *SEC, Financial Statement Data Sets*. Available at: https://www.sec.gov/dera/data/financial-statement-data-sets.html (Accessed: 25 November 2021).

Singh, P. (2018). *Machine Learning with PySpark: With Natural Language Processing and Recommender Systems*. Apress.

SpryMedia (2021) *DataTables Table plug-in for jQuery*. Available at: https://datatables.net/ (Accessed: 31 October 2021).

Štěpánek, L., Habarta, F., Malá, I., & Marek, L. (2021, July). "Great in, great out" is the new "garbage in, garbage out": subsampling from data with no response variable using various approaches, including unsupervised learning. In *2021 International Conference on Computing, Computational Modelling and Applications (ICCMA)* (pp. 122-129). IEEE.

Ven, B.V. de (2021) *Bokeh*. Available at: https://bokeh.org/ (Accessed: 31 October 2021).

Venters, C., & Mikkilineni, R. (2020, September). Representation and Evolution of Knowledge Structures to Detect Anomalies in Financial Statements. In *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 58-63). IEEE.

XBRL International (2021a) *An Introduction to XBRL, An Introduction to XBRL*. Available at: https://www.xbrl.org/the-standard/what/an-introduction-to-xbrl/ (Accessed: 7 December 2021).

XBRL International (2021b) *XBRL & Big Data, XBRL & Big Data*. Available at: https://specifications.xbrl.org/big-data.html (Accessed: 28 November 2021).

XBRL International (2021c) *XBRL Certified Software, XBRL Certified Software*. Available at: https://software.xbrl.org/ (Accessed: 28 November 2021).

Yao, J., Zhang, J., & Wang, L. (2018, May). A financial statement fraud detection model based on hybrid data mining methods. In *2018 international conference on artificial intelligence and big data (ICAIBD)* (pp. 57-61). IEEE.