For cited: *Üzüm, İ., Can, Ö. (2022). Applicability of Machine Learning Models for Detecting Anomalous File Transfer Events in B2B Application Integrations. Journal of Current Research on Engineering, Science and Technology, 8 (1), 1-18.*

Journal of Current Research on Engineering, Science and Technology (JoCREST)
ISSN: 2651-2521

Year: 2022
Volume: 8
Issue: 1

Crossref doi: 10.26579/jocrest.87

Research Article/Araştırma Makalesi

Applicability of Machine Learning Models for Detecting Anomalous File Transfer Events in B2B Application Integrations*

İbrahim ÜZÜM[1] & Özgü CAN[2]

Keywords
anomaly detection; machine learning; file integration; integrity.

Article History
Received
3 Jan, 2022
Accepted
15 Mar, 2022

Abstract
Information systems that are based on real-time file integrations have an essential role to improve the quality of organizations' business process management. File transfers and data integrations between discrete systems have gained great importance. However, network and security issues have emerged due to the integration of file transfer processes and the structure of files. Thus, an effective and self-learning anomaly detection approach is needed for the file integration processes to provide the persistence of integration channels, data integrity, and availability of file transfer processes. A novel anomaly detection approach that focuses on file transfers between discrete systems is proposed in this paper. The proposed anomaly detection approach is a self-learning module for the file transfer processes. For this purpose, anomalies that occur in file transfer processes are detected by applying machine learning techniques. Four clustering-based machine learning approaches are applied to detect anomalies in the file integration processes: Elliptic Envelope, Isolation Forest, Local Outlier Factor, and One-Class Support Vector Machine.

1. Introduction

Cross-system file integrations are one of the most critical components for exchanging data between the information-based systems. These types of data exchanges are the most general method to integrate data between discrete systems. This data exchange allows information systems to create a shared data infrastructure between systems. File integrations provide distributed systems to manage their data types and process flows in a mutual space. On average, two hundred thousand files are exchanged between integration-based distributed systems per day. E-invoice integrations, weather forecast data flow, or exchange rate information systems could be demonstrated as the most popular examples for these types of systems.

---

* Part of M.Sc. thesis of the First Author.

[1] ORCID: 0000-0003-3699-4054. M.Sc., Ege University, Faculty of Engineering, Department of Computer Engineering, ibrahim.uzum@windowslive.com

[2] Corresponding Author. ORCID: 0000-0002-8064-2905. Assoc. Prof. Dr., Ege University, Faculty of Engineering, Department of Computer Engineering, ozgu.can@ege.edu.tr

In file integration systems, the excessive number of transferring files brings an abnormal load into the network infrastructure. These network-based abnormalities can be bandwidth loads, file flow lags, or the discontinuation of the current file transfers. The main cause of these abnormalities is mostly related to the security-based intrusions that threaten the integrity of the transmitted files. Therefore, data integrity which is one of the fundamental components of the security triad is violated. These intrusions are mostly examined as document malfunctions, file extension spoofing, content sniffing, or server-side request forgery intents. These types of attacks are mostly related to disturbances in the structure of the file integration process and are mostly caught by tracing the size of transferred files or by following up the time of the integration process. Further, there are network persistence-related problems in these abnormal situations. If the transferred file's size is anomalously large or the integration process time is longer than a certain threshold value, the integration channels are loaded to a large extent in terms of bandwidth and delay. If anomalous file transfers continue, the bandwidth load will be increased and the integration process could stop. Thereupon, it is important to maintain the continuity, availability, and accessibility of the integration channels. Thus, file integrations must be traced continuously and anomalous files must be detected instantly to provide file management and privacy preservation.

Recent studies on security-based anomaly detection are mostly focused on fields such as intrusion detection systems, web attack and fraud detection systems, real-time data streams, operation systems, and memory usage (Agrawal and Agrawal, 2015). Additionally, most of these studies apply one exact method rather than comparing different models to find an optimal solution. Also, within our knowledge, there exists no study that handles file integrations as a specific application domain. Therefore, the proposed study intends to overcome the described deficiencies of the related domain and presents an efficient machine learning-based anomaly detection approach.

The most common methods that are used to determine the abnormalities in security-based systems are machine learning-based approaches. A self-learning system does not need to define rules to determine the anomaly tolerance values for each file integration (Uzum and Can, 2018a). Also, the system determines these values by training data. Therefore, the system improves itself and minimizes the maintenance cost. The main goal of this study is to find an optimal machine learning-based anomaly detection solution for files transferred in integration-based systems. Data that comes from outside is always suspicious and must be secured. Therefore, an emergency management system must be developed to provide an awareness mechanism for the responsible people of these integration systems (Uzum and Can, 2018a). The proposed approach aims to provide a way to find anomalous integrations and to protect the persistence of integration channels. In terms of maintenance cost reduction, the proposed system will be a self-learning system (Chandola et al., 2009). Therefore, a machine learning-based approach is used to detect anomalies effectively. For this purpose, several anomaly detection models are trained with unlabeled file integration instances. Then, labeled instances are used to test these models until a reasonable accuracy result is

achieved. Finally, the anomaly detection performance of these models is evaluated to determine the optimal solution.

This work aims to create a machine learning-based anomaly detection solution for enterprise cross-system file integrations. The organization of the paper is as follows: Section 2 reviews the related anomaly detection studies in the literature, Section 3 explains the main structure of the enterprise file integration systems and possible data integrity-based file abnormalities that could occur during the file integration process, features of the machine learning-based approaches to detect anomalies are described and experimental results of these approaches are measured, compared and the optimal approach for the case study is discussed in Section 4, Section 5 presents a client application that uses the proposed experimental model, and finally Section 6 concludes the proposed work.

2. Related Work

Anomaly detection is the task of finding patterns in data that do not conform to expected behavior (Chandola et al., 2009). Therefore, anomaly detection is an important problem and it has been researched within different application domains and research areas. In the literature, there are many anomaly detection studies and most of them are focused on certain application fields (Chandola et al., 2009; Ahmed et al., 2016). These fields can be listed as fraud detection, intrusion detection systems, memory usage, real-time data streams, and data usage on smartphones (Agrawal and Agrawal, 2015). In addition, anomaly detection is an essential requirement for business and financial web applications (Chandrasekhar and Raghuveer, 2013). As cyber-attacks pose a threat to sensitive network information (Kumari et al., 2016), anomaly detection has also essential importance in network security. In (Callegari et al., 2008), an anomaly detection-based intrusion detection system is proposed. The proposed system aims to detect TCP-based anomalous behaviors, and also to reduce the number and the effect of security-based attacks. In (Chitrakar and Chuanhe, 2012), anomaly-based intrusion detection systems are discussed under the scope of machine learning. As a result, the study states that machine learning-based solutions are more effective. Thus, anomaly detection is defined as a classification problem, and an intrusion detection system based on Support Vector Machines (SVM) and k-medoids clustering are proposed. Similarly, a k-means clustering-based approach is proposed in (Sahoo and Ranjan, 2014) to achieve a higher accuracy percentage. The importance of intrusion detection systems is addressed and a new anomaly detection model is introduced in (Muda et al., 2011). In (Oliveria et al., 2021), machine learning techniques are evaluated for network-based intrusion detection. Similarly, the study presented in (Betarte et al., 2021) investigates the application of machine learning techniques to leverage Web Application Firewalls and proposes an anomaly detection model to detect web attacks. Moreover, a survey on anomaly-based intrusion detection systems is presented in (Daniya et al., 2021).

Anomaly detection in mobile network communication is also important in today's increased mobile data usage. As smartphones keep all the private information of users, they become the main target point for cyber-attacks (Mirsky et al., 2017). Hence, today's cyber-attacks generally aim to access the user's private information without catching up with a detection mechanism. An algorithm based on hybrid

clustering is proposed in (Pawling et al., 2007) for anomaly detection in real-time mobile communication.

In anomaly detection, detecting possible fraud attempts is an important issue (Ahmed et al., 2015). Most of the machine learning-based fraud detection systems are unsupervised because training data is time-consuming and it is impossible to verify all training data (Noto et al., 2012). Also, as stated in (Li et al., 2016), an anomaly is a much more complex concept. As stated in (Ahmad et al., 2017), detecting early anomalies in real-time data streams across several industries is a critical process. Moreover, recent studies focus on detecting and alerting abnormal energy consumption. A neural network architecture is introduced in (Himeur et al., 2020) to analyze and detect energy consumption anomalies. Further, an online abnormal daily energy consumption patterns detection method is proposed in (Zhou et al., 2021) to identify the anomaly energy consumption patterns of central air conditioning systems.

In addition to the existing studies, Semantic Web-based anomaly detection approaches are proposed in the literature. In (Yang et al., 2015), a model to detect anomalies on the blog user comments' is presented. The presented model aims to find intrusion-based semantic patterns in a text. For this purpose, a clustering-based approach is proposed for anomaly detection. Further, anomaly detection algorithms are proposed in (Mahapatra et al., 2012) to find the intrusion-based anomalies in e-mail contexts.

In the literature, there is no machine learning-based anomaly detection approach that considers enterprise file integrations. Ontology-based approaches within the scope of anomalous file integration are presented in (Can et al., 2019; Can and Uzum, 2019). In addition, problems that occur within the scope of information security in file integrations are presented in (Uzum and Can, 2018a; Uzum and Can, 2018b). The proposed study is based on the fundamental concepts that are presented in Uzum and Can, 2018a; Uzum and Can, 2018b).

As a result, the benefits of anomaly detection in the information security field are extremely important. There could be anomalies at any time and these anomalies must be detected and managed. Consequently, self-learning approaches are the best way to handle anomalies. This study proposes an efficient machine learning-based anomaly detection solution approach for enterprise file integrations. The main contribution of the proposed solution is (i) to comprise anomaly detection for enterprise file integrations domain, and (ii) to use different anomaly detection models to find an optimal solution. The current studies in the literature do not try different anomaly detection models to find an optimal solution. Also, to the best of our knowledge, there is no solution to detect anomalies in enterprise file integration systems. Therefore, the proposed study aims to resolve the related problems.

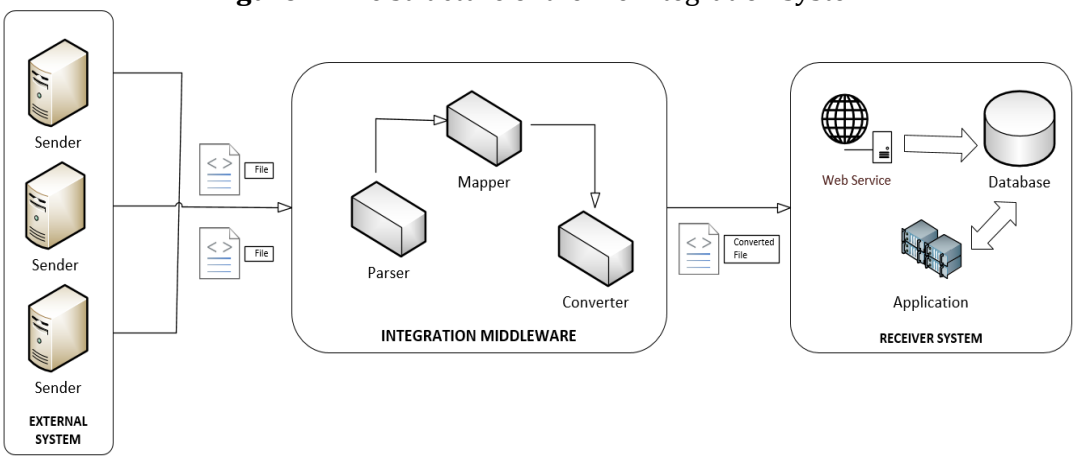
3. Materials and Methods

This section mainly describes the general structure of enterprise file integrations and security-based anomalous file integration symptoms. Each symptom is explained by the most common examples.

3.1. General Structure

The proposed model is based on the file integration system as the case study. Figure 1 presents the basic structure of a file integration system. The figure shows the process of getting a file from an external system to the internal system and submitting the file to the database. Also, a summary of the general structure is presented in (Uzum and Can, 2018a).

Figure 1. The structure of the file integration system.



As seen in Figure 1, the sender systems could also be specified as the integrators and these systems could be mainly capsulated as external systems. The main role of the sender systems is to send the generated file to the integration middleware of the receiver system. An e-invoice file in a specified format is an example of this scenario. The related file is created by an invoice integrator and sent to the receiver taxpayer's integration system. The integration middleware has the featured role in the proposed system. In this intermediate structure, files from the external system that have a certain standard are parsed, mapped, and converted into a structure that the system could use. At first, the parser parses the incoming file and determines if the file is in the correct structure. Then, the mapper maps the data of the incoming file to a new XML (eXtensible Markup Language) (W3C, 2022) file that the incoming system will accept. Finally, the converter transforms the data types in the XML document. An example of this scenario is the conversion of the incoming billing data. This conversion is from UBL (Universal Business Language) standard (Bengtsson, 2022) into a separate XML standard. During the conversion, only the system requirements are available, such as the billing number, invoice amount, and billing date. The middleware determines the transfer start time when the file comes to the middleware and determines the transfer end time when the parsing and conversion end. The converted files are transferred to services that will use these files in the receiver system. These services can be web services or executable scripts. Finally, the related services in the receiver system submit the contents of the file to the inner database and the integration process finishes.

3.2. General Structure

There can be information security-based violations, problems, or requirements during the file transfers in integration-based work structures. Data that is coming from outside is always suspicious and needs to be controlled to detect present

anomalies. Security-based symptoms are generally related to anomalies in the syntactic and semantic content of XML documents that are introduced into integration channels (OWAS Cheat Sheet Series, 2022). These symptoms are mainly listed as content sniffing, malformed documents, invalid documents, and server-side request forgery:

- *Content Sniffing*: Exposing an executable file command extension as an XML file and transferring it to the integration channel is an example of content modification violations. Thus, the attacker can execute the attack commands that are sent by showing it as an XML file on the integration middleware. This type of violation is usually performed on completely untrusted file senders. Such executable files are abnormally large from an actual XML file. Furthermore, the processing of such commands will take a long time to be distinguished from the conversion time of a normal file. As a result, it is possible to detect the anomaly in size and time.
- *Malformed Documents*: Malformed XML files are defined as files that have syntactically noticeable defects and do not conform to the W3C's XML specification (W3C, 2022). These defects are the most common data integrity violation in the file integrations. This type of violation disrupts the XSD element definition structure in the file. Incorrect ordering of the element tags and the presence of the unidentified or unauthorized characters in elements are some examples for this type of symptoms. In the file integrations, the processing of such malformed files takes much longer than usual and results in fatal errors. It is possible to detect this anomaly in terms of transfer time.
- *Invalid Documents*: Invalid XML files can be represented as files that contain non-validated data or contain elements with an infinitesimal number of sub-element sequences. In this type of violation, an attacker can try to slow down the file transfer process or increase the file size by trying the following scenario. XML entity expansion, coercive parsing, recursive entity references and quadratic blowup attack are examples of the invalid documents. In these types of attacks, the transfer time will be noticeably longer and it is possible to detect the anomaly with the size of the file.
- *Server-Side Request Forgery (SSRF)*: The most up-to-date example of the server-side request forgeries is the issuance of a web address or executable command that can execute an attack scenario against XML root information. Port scanning and external connection violations are examples of these types of attacks. In such cases, the operation of the commands results in an anomalous time according to the normal transfer time. These attacks bring a noticeable difference in time to the file integrations.

Besides the data integrity-based issues, there might be data persistence and availability-based security problems. These problems are detected with the incoming file's size or the integration process time. The main symptom of this type of attack is the file transfer queue obstruction and the loss of files that is occurred during the integration process. As a scenario, the incoming files are inserted into a queue during the integration. The first file in the queue becomes the first entry in

the integration channel according to the First-In-First-Out (FIFO) structure. The rest of the files are also integrated into the system. There could be abnormalities in the transfer time or file size during the process. If the transfer time is anomalous, the processing of other files is delayed. Thus, delay condition occurs. If the file size is anomalous, the resources on the server are overused. In this case, the files in the sequence become unworkable and the integration queue, which will increase continuously, becomes obstructed. In such cases, the integration server must be restarted.

3.3. Experimental Work

The experimental work of the study consists of three sub-sections. In the first section, the dataset used for the training and test phase, and the feature extraction process are described. The proposed anomaly detection models are Elliptic Envelope, Isolation Forest, Local Outlier Factor, and One-Class Support Vector Machine. The related models are described in the second section. Finally, the experimental results of these models are measured, compared, and discussed in the third section.

3.3.1. Dataset and Feature Extraction

In the experimental study that is conducted within the scope of anomaly detection, a file integration log dataset is created. The dataset is a collection of the incoming data that is transferred into a production system and each record represents a file integration log. A file integration log record consists of the following properties:

- *File Size*: Represents the total size of the file that comes to the integration channel in kilobytes. As the file size is one of the important factors to detect anomalies, it is one of the main features for the analysis.
- *Integration Start Time*: Represents the start time of the integration parsing phase. Also, integration start time represents the time of the file's entrance into the integration middleware.
- *Integration Finish Time*: Represents the end of the integration parsing phase. Also, integration finish time represents the finish time of the file conversion and the file's entrance into the receiver system.
- *Integration Status*: Gives information about the success or failure of the integration process. If the file is successfully parsed and written into the database, the integration is succeeded. If an error occurs during the parsing, mapping, or conversion, the integration is erroneous.
- *Integration Process Time*: Gives the file integration processing time between the start time of the parsing and the registration of the system in seconds. As the integration process time is one of the important factors to detect anomalies, it is one of the main features for the analysis.
- *Re-Transmission Count*: Gives the information that the integration has been retriggered. Unsuccessful integrations could be triggered manually. Re-transmission count is given explicitly for each file integration.

- *Error Count*: Represents the number of errors that are received in the parsing process if the integration is failed. These errors could appear on the parsing, mapping, and converting phase in the middleware, or the database submission phase in the receiver.
- *System Transformation Count*: Gives the number of system conversions in the integration. There could be multiple mapping and conversion operations for the incoming file. For example, the file might be mapped into an intermediate structure, then mapped again to a third final XML structure.
- *File Name*: Represents the name of the file that comes to the integration channel. File names are given in the sender systems.
- *Responsible Name*: Gives the name of the person that is responsible for the integration in the organization. If an anomalous integration log occurs, the responsible person will be informed or alarmed.

The features that are used in the training of the models are initially defined as the duration of the integration and the file size. However, the number of these features is increased during the experimental study in order to get more accurate results. Therefore, the count of re-transmissions, the count of errors, the count of system conversion, and the integration status metrics are also included as key features to train anomaly detection models. The main cause of these inclusions is that these metrics must also be thought of as assistant features to detect anomalous integration logs.

The raw training data consists of 150.000 records. During the training phase of the models, 20.000 integration records are randomly used in this dataset. There is no information that the integration records on the training data are anomalous or normal. This information is generated according to the results of the models in the training phase. A minimal eight entry-sized subset of the file integration log train dataset is shown in Table 1.

After the creation of the training dataset, a set of test data is created to test the model results. It is precisely known that the records in the test dataset are anomalous or normal, and the data is labeled according to this knowledge. In the test dataset, an "Anomaly" field is added to hold this information. The test data consists of 1.000 integration records. After training the models, all of these records are used to measure the false positive and accuracy rates. A minimal eight entry-sized subset of the file integration log test dataset is given in Table 2.

Table 1. Instances of the file integration log train dataset.

File Size (kB)	Status	Process Time (msec)	Re-transmission	Error	System Transform
115.836	1	3159	0	0	0
94.937	1	4774	0	1	0
94.851	1	2466	0	0	2
160.208	1	3343	1	0	0
205.601	1	2795	0	1	1
206.398	0	4775	2	1	0
121.515	1	2622	0	0	0
205.596	1	7389	0	0	1

Table 2. Instances of the file integration log test dataset.

File Size (kB)	Status	Process Time (msec)	Re-transmission	Error	System Transform
99.999	1	3159	0	0	1
123.000	1	4774	0	0	1
2.000	1	2466	0	2	-1
2.500	1	3343	1	0	-1
122	1	2795	0	1	1
126.890	1	9002	0	0	1
144.845	0	4798	0	0	-1
197.306	0	4197	0	1	-1

Table 2 represents the labeled data for the file integration logs. In the test dataset, it is precisely known which integration log is anomalous and which is not. If the value of the Anomaly field equals "-1", it means that the integration log is normal, else if the value of the Anomaly field equals "1", it means that the integration log is anomalous.

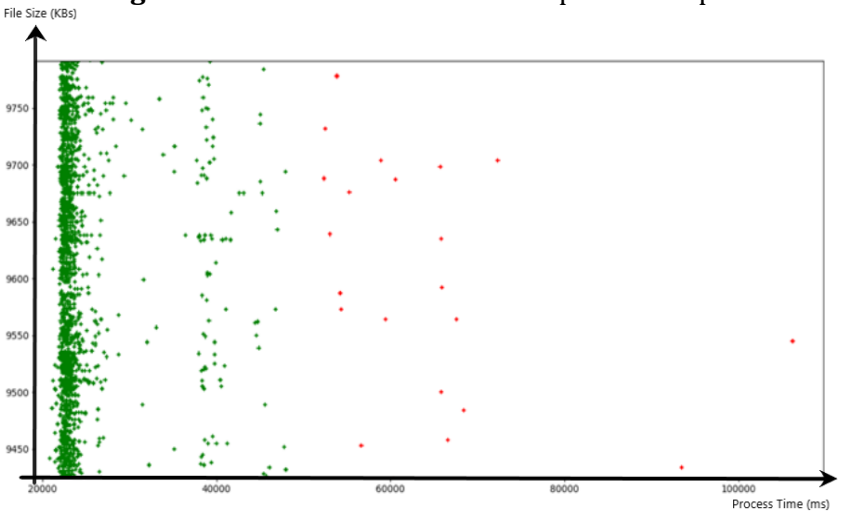
3.4. Anomaly Detection Models

Four anomaly detection models are used to detect anomalous records in the dataset. These models are Elliptic Envelope, Isolation Forest, Local Outlier Factor, and One-Class Support Vector Machine. There are some reasons to select these approaches. First of all, outlier or anomaly detection is mostly known as unsupervised anomaly detection. In the context of anomaly detection, the anomalies cannot form a dense cluster as available estimators assume that anomalies are located in low-density regions (Scikit-Learn, 2022). One-Class Support Vector Machine estimator is best suited for anomaly detection when the training set is not contaminated by outliers. The Elliptic Envelope method has a very robust mechanism for outliers in ellipse-shaped data distributions. Isolation Forest has an advantage on multi-modal datasets. Besides, the Local Outlier Factor method is useful on high-dimensional datasets. Therefore, these four models are compared on the dataset to obtain the optimal anomaly results.

3.4.1. Elliptic Envelope

Elliptic Envelope is the first model to detect anomalous file integration instances. The model is based on the Gaussian distribution. In this method, the points outside the central mode are ignored and determined as external values (Rousseeuw and Van Driessen, 1999). The distribution in the central region is determined as an ellipse. Areas outside this region are determined as anomalous values. The result of the Elliptic Envelope distribution is shown in Figure 2. In Figure 2, the normal integration records are shown in green, and anomalous records are shown in red.

Figure 2. The distribution of the Elliptic Envelope.

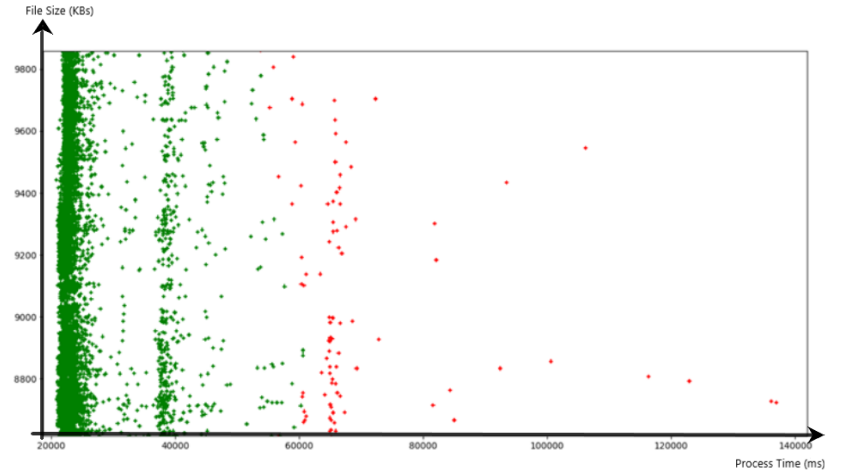


As seen in Figure 2, normal findings are collected in an elliptical cluster along the file size column. Registries outside of this cluster are identified as anomalous. It is observed that the model gives a good result especially in the integration period. Also, the integration records that exceed a certain integration period are labeled as abnormal.

3.4.2. Isolation Forest

The Isolation Forest method is used as the second method to perform outlier detection in the file integration dataset. In this method, observations are monitored by randomly selecting a value and determining a random discrimination value between the minimum and maximum values of the selected characteristics (Liu et al., 2008). Random partitioning produces considerably shorter paths for anomalies. For this reason, it is likely to be an anomaly when a random tree forest produces shorter path lengths for certain specimens collectively. The result of the Isolation Forest method is shown in Figure 3. In Figure 3, normal integration records are shown in green, and anomalous records are shown in red.

Figure 3. The distribution of the Isolation Forest.

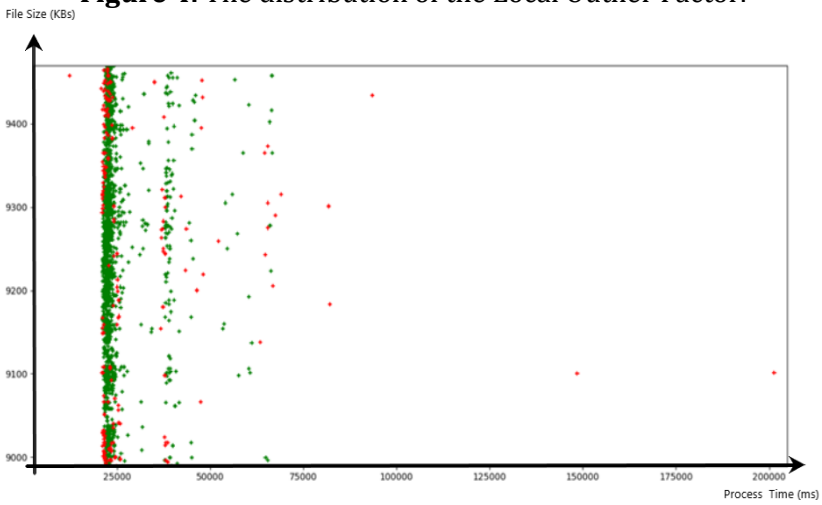


As seen in Figure 3, normal findings are collected in an elliptical cluster along the file size column. Here, unlike the Elliptic Envelope approach, it is understood that records after a given file size column threshold are also observed as an anomaly. Hence, it is observed that the model gives a good result based on the file size anomaly as well as the integration duration. Also, it is seen that the integration records that exceed a certain integration duration and file size threshold are labeled as abnormal.

3.4.3. Local Outlier Factor

The Local Outlier Factor (LOF) method is used as the third method to perform outlier detection on the file integration dataset. In this method, the local density is obtained from the k-closest neighbors (Breunig et al., 2000). An observation of LOF score is equal to the ratio of the average local intensity of the k-closest neighbors to its local intensity. It is expected that a normal sample would have a local density similar to that of its neighbors, whereas the anomalous one would have a much smaller local density. The result of the LOF distribution is shown in Figure 4. Similar to the former figures, normal integration records are shown in green, and anomalous records are shown in red.

Figure 4. The distribution of the Local Outlier Factor.



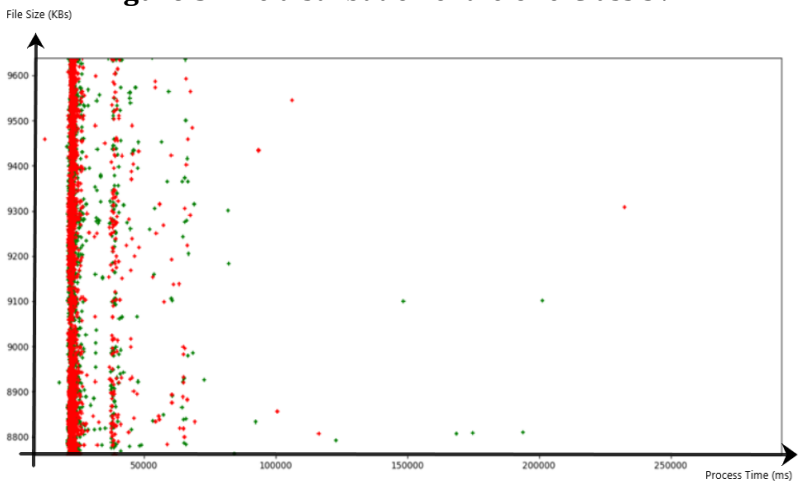
As seen in Figure 4, the normal findings appear as a small cluster in the middle, and the clusters of anomalous and normal findings are scattered irregularly. Therefore, anomalous and normal integrations cannot be fully decomposed. As a result, it could be determined that the Local Outlier Factor approach fails to detect the anomaly in the file integration domain. This result is unfortunately unsatisfactory and does not provide the exact detection of anomalous file integrations.

3.4.4. One Class Support Vector Machine

One-Class Support Vector Machine (SVM) is the last approach used within the scope of this study. This approach is essentially an anomaly detection mechanism. Therefore, it is not an external value finding approach (Schölkopf et al., 2001). However, in the scope of this study, the method is used to compare the findings of the experimental study. One-Class SVM approach generally requires selecting a kernel and a scalar parameter be to define the boundary. The results of the One-

Class SVM method are shown in Figure 5. In Figure 5, normal integration records are shown in green, and anomalous records are shown in red.

Figure 5. The distribution of the One-Class SVM.



As seen in Figure 5, the normal findings appear as a small cluster in the middle, and anomalous and normal integrations are very dispersed. As a result, the One-Class SVM approach fails to detect the anomaly within the training data. This result is unfortunately inadequate and does not provide the exact detection of anomalous file integrations.

4. Results and Discussion

The related models that are given in Section 3 are tested for their performance and training success to detect anomalies in file integrations. Also, the accuracy percentages are measured. In the scope of the test, 1.000 test data that is explained in Section 4.1 are used. In this test data, it is known exactly which integration log record is anomalous and which is normal. Experimental results are based on the accuracy rate and false-positive rate. The results are presented in Table 3 and Table 4, respectively.

Table 3. The accuracy rates of the models.

Model	Accuracy Rate
Elliptic Envelope	85,71%
Isolation Forest	86,22%
Local Outlier Factor	14,28%
One-Class SVM	19,25%

Table 4. The false positive rates of the models.

Model	False Positive
Elliptic Envelope	18,72%
Isolation Forest	15,13%
Local Outlier Factor	60,72%
One-Class SVM	67,13%

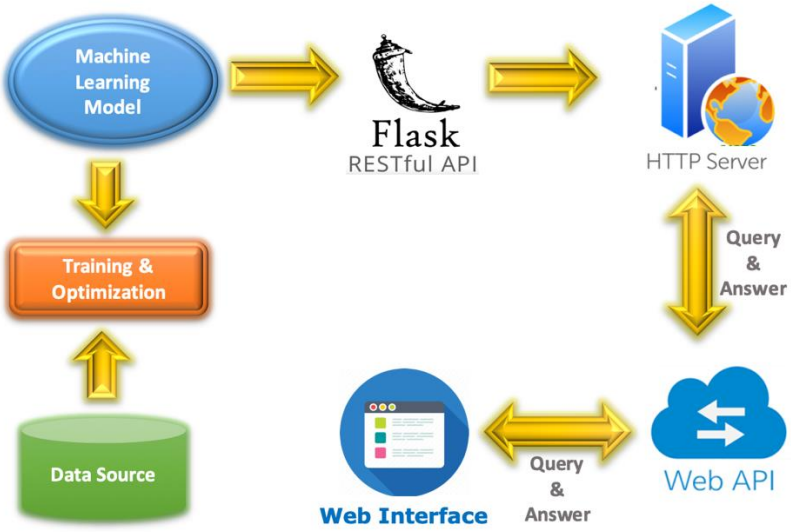
Table 3 shows that Elliptic Envelope and Isolation Forest approaches are better suited to detect anomalies in file integrations. These two approaches exceed the 85% threshold value. On the other hand, the other two approaches are quite inadequate in terms of work and probing. The power of the first two approaches is

also seen in Table 4. The false-positive rates of the Elliptic Envelope and Isolation Forest methods are below the 20% threshold value which is a satisfying value. Contrarily, Local Outlier Factor and One-Class SVM performances do not give satisfactory results in false-positive rates.

Experimental results show that Elliptic Envelope and Isolation Forest methods are optimal solutions to detect anomalies in this type of data distribution. However, this does not mean that these two approaches are appropriate for all types of data dispersion. In this experimental dataset, train data are clustered mostly in a horizontal elliptic or linear area. This type of data distribution corresponds to a standard Gaussian distribution or a random tree. Therefore, the main reason for this experimental power of Gaussian-based Elliptic Envelope and random tree-based Isolation Forest methods could be examined. On the other hand, if the training dataset is distributed like several circular clusters, high-dimensional dataset distribution based One-Class Support Vector Machine and Local Outlier Factor approaches may have resulted in a better way. Also, the experimental results show that the number of training data and the main distribution structure of datasets is the main key factor in anomaly detection. There must be a unique optimal method for each file integration type because of the variation of the dataset distribution.

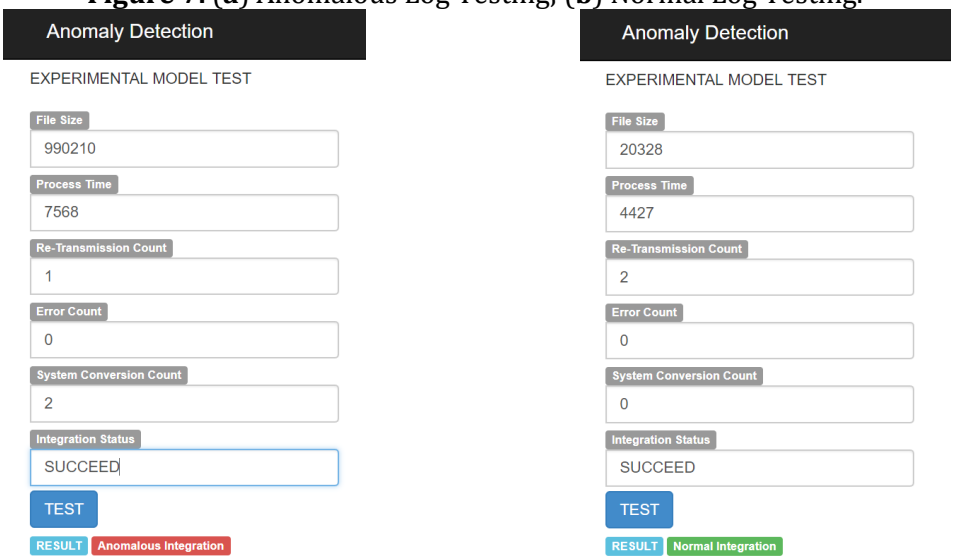
The architecture of the proposed anomaly detection is shown in Figure 6. As a final step, the experimental model is used in a client application. The client application is developed to test the experimental model's reaction and its performance by using sample integration log inputs. Therefore, a web application that consists of integration log inputs and a post button is developed. There are five inputs in the web form to represent the file size, integration process time, re-transmission count, error count, system transformation count, and integration status. As mentioned before, all of these inputs are extracted as key features in the experimental model creation phase.

Figure 6. The architecture of the proposed anomaly detection approach.



The mobile browser view of the client application with sample anomalous and normal integration logs are shown in Figure 7(a) and Figure 7(b), respectively. When the test button is pressed, the client calls the model, and the result that determines whether the entered integration log is anomalous or normal is generated. Figure 7(a) and Figure 7(b) represent the user interface of the client application. In the backend of the system, there is a service that calls the experimental model with integration log input parameters. When the test button is clicked, the service works and the experimental model is triggered. The model produces and returns the result. Finally, this result is shown in the frontend client.

Figure 7. (**a**) Anomalous Log Testing; (**b**) Normal Log Testing.



(**a**) (**b**)

5. Conclusions

In information systems, anomaly detection mechanism for enterprise file integrations has significant importance. However, anomalies arise during these enterprise file integrations. These anomalies must be detected on time to ensure the determine the possible malfunctions in the file content, to indicate an alarm for possible data integrity intrusions, and to provide persistence and availability of the integration process and network channels. Moreover, the related alarm must be sent to the administrator to provide the authentication. Every anomalous movement is recorded as a log to trace and to prevent non-repudiation. In addition, anomaly detection must be self-disciplined to reduce the high maintenance cost for new anomalies. The main goal of this study is to provide a high accuracy rate to detect file integration anomalies and to detect these anomalies on time. The proposed study is a self-learning anomaly detection solution for enterprise file integration systems. In the literature, there is no anomaly detection solution for enterprise integration systems. Additionally, most of the existing studies apply one exact method rather than comparing different models to find an optimal solution. The proposed study aims to overcome these deficiencies of literature and presents a novel machine learning-based anomaly detection solution for file integration systems.

In the process of ascertaining the anomaly in the file integration, four probabilistic approaches are executed and results are obtained to assist in decision making. In the experiments, tests and training datasets with extensive, ideal distribution and ideal dimensions are used. The number of key features of the related models that are used in education is increased to provide a healthier anomalous cluster detection. Also, the performance of the training and the percentage of outcomes of the tests are measured. Results of the training and testing of the models show that the Elliptic Envelope and Isolation Forest approaches based on the central distribution respond better. Finally, a client application is developed to represent the experimental model with integration log inputs and to test the responses of the proposed model. There is no connection between paragraphs.

Acknowledgments: This work has been conducted during the Master degree thesis study of Ibrahim Uzum under the supervision of Ozgu Can at Ege University Computer Engineering Department.

References

Agrawal, S., and Agrawal, J. (2015) Survey on Anomaly Detection using Data Mining Techniques. Procedia Computer Science, Volume 60, pp.708-713.

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. (2017) Unsupervised Real-Time Anomaly Detection for Streaming Data. Neurocomputing, Volume 262, pp.134-147.

Ahmed, M., Mahmood, A.N., and Islam, M. R. (2015) A Survey of Anomaly Detection Techniques in Financial Domain. Future Generation Computer Systems, Volume 55, pp.278-288.

Ahmed, M., Mahmood, A.N., and Hu., J. (2016) A Survey of Network Anomaly Detection Techniques. Journal of Network and Computer Applications, Volume 60, Issue C, pp. 19-31.

Bengtsson, K. (2022) OASIS Universal Business Language (UBL) TC. Available online: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl (Accessed on 12 January 2022).

Betarte, G., Gimenez, E., Martinez, R., and Pardo, A. (2018) Improving Web Application Firewalls through Anomaly Detection. 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, Florida, USA, 17-20 Dec 2018, IEEE, pp. 779-784.

Breunig, M.M., Kriegel, H-P., Ng, R.T., and Sander, J. (2000) LOF: Identifying Density-Based Local Outliers. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, 15-18 May 2000; ACM: New York, NY, USA, ACM SIGMOD Record, Volume 29, Issue 3, pp. 93-104.

Callegari, C., Vaton, S., and Pagano, M. (2008) A New Statistical Approach to Network Anomaly Detection. Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunications

Systems (SPECTS 2008), Edinburgh, UK, 16-18 June 2008, IEEE, pp. 441-447.

Can, Ö., Ünalır, M.O., and Üzüm, İ. (2019) An Ontology Development for Anomaly Detection in File Integration Domain (Dosya Entegrasyonu Etki Alanında Anomali Tespiti İçin Bir Ontoloji Geliştirimi). Journal of Information Technologies (Bilişim Teknolojileri Dergisi), Volume 12, Issue 3, pp. 239-252.

Can, O. and Uzum, I. (2019) Ontology Based Anomaly Detection for File Integration. The 13th International Conference on Metadata and Semantic Research (MTSR 2019), Rome, Italy, Rome, Italy, 28-31 October, 2019, Springer, Cham, Volume 1057, pp. 194-199.

Chandola, V., Banerjee, A., and Kumar, V. (2009) Anomaly Detection: A Survey. ACM Computing Surveys, Volume 41, No. 3, Article 15, pp. 1-58.

Chandrasekhar, A.M., and Raghuveer, K. (2013) Intrusion Detection Technique by Using K-Means Fuzzy Neural Network and SVM Classifiers. Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 4-6 January 2013, IEEE, pp. 1-7.

Chitrakar, R., and Chuanhe, H. (2012) Anomaly Detection using Support Vector Machine Classification with K-Medoids Clustering. Proceedings of The Third Asian Himalayas International Conference on Internet (AH-ICI), Kathmandu, Nepal, 23-25 November 2012, IEEE, pp. 1-5.

Daniya, T., Suresh Kumar, K., Santhosh Kumar, B., and Sekhar Kolli, C. (2021) A survey on anomaly based intrusion detection system. Materials Today: Proceedings. DOI: 10.1016/j.matpr.2021.03.353.

Himeur, Y., Alsalemi, A., Bensaali, F., and Amira, A. (2020) A Novel Approach for Detecting Anomalous Energy Consumption Based on Micro-Moments and Deep Neural Networks. Cognitive Computation, Volume 12, pp. 1381–1401.

Kumari, R., Sheetanshu, Singh, M.K., Jha, R., and Singh, N.K. (2016) Anomaly Detection in Network Traffic using K-Mean Clustering. Proceedings of the 3rd International Conference on Recent Advances in Information Technology (RAIT 2016), Dhanbad, India, 3-5 March 2016, IEEE, pp. 387-393.

Li, F., Zheng, D., Zhao, T., and Pedrycz, W. (2016) A Novel Approach for Anomaly Detection in Data Streams: Fuzzy-Statistical Detection Model. Journal of Intelligent & Fuzzy Systems, Volume 30, No. 5, pp. 2611-2622.

Liu, F.T., Ting, K.M., and Zhou, Z-H. (2008) Isolation Forest. Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15-19 December 2008, IEEE, pp. 413-422.

Mahapatra, A., Srivastava, N., and Srivastava, J. (2012) Contextual Anomaly Detection in Text Data. Algorithms, Volume 5, Issue 4, pp.469-489.

Mirsky, Y., Shabtai, A., Shapira, B., Elovici, Y., and Rokach, L. (2017) Anomaly Detection for Smartphone Data Streams. Pervasive and Mobile Computing, Volume 35, pp.83-107.

Muda, Z., Yassin, W., Sulaiman, M., and Udzir, N.I. (2011) Intrusion Detection based on K-Means Clustering and Naive-Bayes Classification. Proceedings of the 7th International Conference on IT in Asia (CITA), Kuching, Sarawak, Malaysia, 12-13 July 2011, IEEE, pp. 1-6.

Noto, K., Bradley, C., and Slonim, D. (2012) FRaC: A Feature-Modeling Approach for Semi-Supervised and Unsupervised Anomaly Detection. Data Mining and Knowledge Discovery, Volume 25, pp.109-133.

Oliveira, N., Praça, I., Maia, E., and Sousa, O. (2021) Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems. Applied Sciences, 11(4), 1674.

OWASP Cheat Sheet Series. XML Security Cheat Sheet. (2022) Available online: https://www.owasp.org/index.php/XML_Security_Cheat_Sheet (Accessed on 12 April 2022).

Pawling, A., Chawla, N.V., and Madey, G. (2007) Anomaly Detection in a Mobile Communication Network. Computational and Mathematical Organization Theory volume, Volume 13, Issue 4, pp.407-422.

Rousseeuw, P.J. and Van Driessen, K. (1999) A fast algorithm for the minimum covariance determinant estimator. Technometrics, Volume 41, Issue 3, pp. 212-223.

Sahoo, G., and Ranjan, R. (2014) A New Clustering Approach for Anomaly Intrusion Detection. International Journal of Data Mining & Knowledge Management Process (IJDKP), Volume 4, No. 2, pp. 29-38.

Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., and Williamson, R.C. (2001) Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, *Volume 13, No. 7*, pp.1443-1471.

Scikit-Learn. (2022) Novelty and Outlier Detection. Available online: https://scikit-learn.org/stable/modules/ outlier_detection.html (Accessed on 12 April 2022).

Uzum, I., and Can, O. (2018a) An Anomaly Detection Approach for Enterprise File Integration. Proceedings of the 6th International Symposium on Digital Forensic and Security (ISDFS 2018), Antalya, Turkey, 22-25 March 2018, IEEE, pp. 336-339.

Uzum, I. and Can, O. (2018b) An anomaly detection system proposal to ensure information security for file integrations. The 26th Signal Processing and Communications Applications Conference (SIU 2018), Izmir, Turkey, 2-5 May, 2018, IEEE, pp. 1-4.

W3C. Extensible Markup Language (XML). (2022) Available online: https://www.w3.org/XML/ (Accessed on 12 April 2022).

Yang, W., Shen, G., Wang, W. et al. (2015) Anomaly Detection in Microblogging via Co-Clustering. Journal of Computer Science and Technology, Volume 30, Issue 5, pp.1097-1108.

Zhou, X., Yang, T., Liang, L., Zi, X., Yan, J., and Pan, D. (2021) Anomaly detection method of daily energy consumption patterns for central air conditioning systems. Journal of Building Engineering, Volume 38, 102179.