Ministère de l’Enseignement Supérieur, de la Recherche et de l’Innovation

**ECOLE POLYTECHNIQUE DE THIES**



**DEPARTEMENT GENIE INFORMATIQUE ET TÉLÉCOMMUNICATIONS**

**Projet de Fin d’Études en vue de l’obtention du Diplôme d’Ingénieur de Conception**

Présenté par : Papa Omar DIOP

**Sujet :**

**Conception et Réalisation d'un système de détection et de gestion automatisé des anomalies basé sur l’AIOps**

\

Soutenu publiquement le ……. devant le jury composé de :

|  |  |  |  |
| --- | --- | --- | --- |
| Président |  |  |  |
| Examinateurs |  |  |  |
|  |  |  |
| Encadrant |  |  |  |
|  |  |  |

**PRÉFACE**

**DEDICACES**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**REMERCIEMENTS**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Résumé**

Quand il s’agit de la détection d’anomalies, combien de fois on a pu entendre un collègue dire si tu rencontres tel ou tel bug fais ceci ou bien il y a seulement le lead qui connait comment résoudre le problème. Ne pas documenter, normaliser ou partager la façon d'analyser les données pour détecter les intrusions potentielles dans un réseau est plus courant qu’on ne le pense, surtout lorsqu’une équipe est très diverse du point de vue technique et de l'expertise. Cela n'affecte pas seulement les stratégies de détection mais aussi la dynamique de l’équipe ainsi que le bon avancement des projets.

Maintenant, combien de fois avons-nous également pensé à un moyen plus efficace, intuitif ou créatif d'analyser les événements de sécurité que notre organisation collecte, mais nous nous sentons limité aux capacités d'une barre de recherche dépendant d'une seule langue ?

Dans le développement de logiciels, il est absolument nécessaire de s'assurer qu'un système, une fois développé, fonctionne au mieux pendant toute sa durée de vie. Avec le temps, il génère des ressources de données à grande échelle impliquant une grande quantité d'informations de logs, dont le contenu est très significatif. Les données des journaux d'applications (logs) sont essentielles au maintien des performances des applications. Les techniques d'analyse, de compréhension et de détection des anomalies dans les journaux d'applications sont donc essentielles pour garantir l'efficacité du développement logiciel. Bien qu'initialement entravées par un matériel limité et un manque d'ensembles de données de qualité, les techniques de détection d'anomalies ont récemment bénéficié d'un regain d'intérêt grâce aux progrès de la technologie d'apprentissage automatique et notamment des réseaux neuronaux.

Dans cette étude, nous explorons la détection d'anomalies, les techniques historiques de détection d'anomalies et les progrès récents des méthodes d’apprentissage, qui promettent de révolutionner la détection d'anomalies dans les données des journaux d'applications. En outre, nous analysons les techniques de détection d'anomalies les plus prometteuses et proposons un modèle basé principalement sur les méthodes d’apprentissage automatique qui améliore les techniques existantes.

Nous verrons par la suite que les résultats de l'évaluation sont implémentés à travers un système d’alertes qui répond aux besoins de nos applications actuellement en production à TeamX. Ce qui améliore considérablement la précision de la fiabilité en temps réel et fournit ainsi une base de données précise et une base de localisation des anomalies pour l'intelligence artificielle pour les opérations informatiques. Dans ce document, nous avons apporté les innovations et les contributions suivantes à la détection des anomalies et à la mesure de la fiabilité. L'ensemble du processus a été établi depuis le journal original jusqu'à la mesure de la fiabilité en temps réel. De nouvelles méthodes ont été adoptées pour améliorer la précision, le rappel et la valeur F1 de la détection des anomalies.

Termes clés – **AIOps**, **Détection d'anomalies, journaux d'applications, apprentissage automatique.**

**Abstract**

When it comes to anomaly detection, how many times have we heard a colleague say if you encounter such and such a bug do this or only the leader knows how to fix the problem. Not documenting, standardizing, or sharing how to analyze data to detect potential network intrusions is more common than you might think, especially when a team is very diverse in terms of technical expertise. This affects not only detection strategies but also team dynamics and project progress.

Now, how many times have we also thought of a more efficient, intuitive, or creative way to analyze the security events our organization collects, but we feel limited to the capabilities of a single language-dependent search bar?

In software development, it is absolutely necessary to ensure that a system, once developed, works at its best throughout its life. Over time, it generates large-scale data resources involving a large amount of log information, the content of which is very significant. Application log data is critical to maintaining application performance. Techniques for analyzing, understanding, and detecting anomalies in application logs are therefore critical to ensuring effective software development. Although initially hampered by limited hardware and a lack of quality datasets, anomaly detection techniques have recently received renewed interest due to advances in machine learning technology, particularly neural networks.

In this paper, we explore anomaly detection, historical anomaly detection techniques, and recent advances in learning methods, which promise to revolutionize anomaly detection in application log data. In addition, we analyze the most promising anomaly detection techniques and propose a model based primarily on machine learning methods that improve on existing techniques.

We will later see that the results of the evaluation are implemented through an alerting system that meets the needs of our applications currently in production at TeamX. This significantly improves the accuracy of real-time reliability and thus provides an accurate database and anomaly location basis for artificial intelligence for IT operations. In this paper, we have made the following innovations and contributions to anomaly detection and reliability measurement. The whole process has been established from the original log to the real-time reliability measurement. New methods have been adopted to improve the precision, recall, and F1 value of anomaly detection.

Index Terms – **AIOps, Anomaly Detection, Application logs, Machine Learning.**

**TABLE DES MATIERES**

**Résumé**

**Abstract**

**Liste des Figures**

**Liste des Tableaux**

**Liste des Sigles et Abréviations**

1. **Introduction générale**
   1. Présentation de la structure d’accueil ………………………………………………

1.1.1. Structure d’accueil ……………………………………………………………

1.1.2. Contexte de l’immersion ………………………………………………………

* 1. Problématique …………………………………………………………………………
  2. Objectifs du projet ……………………………………………………………………
  3. Plan du mémoire ………………………………………………………………………

1. **Généralités sur la détection d’anomalies**
   1. Mise en contexte …………………………………………………………………………

1.1.1. Logs ……………………………………………………………………………….

1.1.2. Anomalies …………………………………………………………………………

* 1. Les types d’anomalies ……………………………………………………………………

1.1.1. Les anomalies ponctuelles…………………………………………………………

1.1.2. Les anomalies contextuelles ………………………………………………………

1.1.3. Les anomalies collectives …………………………………………………………

* 1. Les challenges ……………………………………………………………………………

1.1.1. Données non structurées ……………………………………………………………

1.1.2. Informations d'exécution redondantes ………………………………………………

1.1.3. Données déséquilibrées …………………………………………………………….

1. **Etude de solution**
   1. Évolution des techniques de détection d’anomalies ………………………………………

3.1.1. Détection d'anomalies basée sur la statistique ……………………………………… 3.1.2. Détection d'anomalies basée sur la profondeur …………………………………… 3.1.3. Détection d'anomalies basée sur le regroupement (apprentissage non supervisé)

3.1.4. Détection d'anomalies basée sur la distance ……………………………………

3.1.5. Détection d'anomalies basée sur la densité (apprentissage non supervisé) ………

3.1.6. Décomposition spectrale …………………………………………………………

3.1.7. Détection d'anomalies basée sur l'apprentissage automatique supervisé …………

3.1.8. Détection d'anomalies par réseau neuronal classique ……………………………

* 1. Analyse des données fournies ……………………………………………………………

3.2.1. Nature des données ……………………………………………………………

3.2.2. Types d’anomalies ………………………………………………………………

3.2.3. Etiquettes des données ……………………………………………………………

3.2.4. Sortie de la détection d'anomalies ………………………………………………

* 1. Discussion sur la meillleure approche pour ces données …………………………………

(Discussion expliquer le pourquoi on choisit telle méthode d’apprentissage par rapport à un autre ect regarder dans tes notes)

1. **Etude conceptuelle de la solution**

4.1. Analyse des besoins ………………………………………………………………

4.1.1. Les besoins fonctionnels ………………………………………………………

4.1.2. Les besoins non fonctionnels ………………………………………………………

4.2. Modélisation UML ………………………………………………………………

4.2.1. Diagramme des cas d’utilisation (Uses Cases) ……………………………………

4.2.3. Diagramme des classes ……………………………………………………………

4.3. Architecture du projet ………………………………………………………………

4.4. Workflow global du système ……………………………………………………………

4.5. Collecte de données ………………………………………………………………

4.6. Description du jeu de données ……………………………………………………………

4.7. Workflow de la détection ………………………………………………………………

4.8. Conclusion ………………………………………………………………

1. **Implémentation et déploiement**

5.1. Environnement de développement et technologies

5.1.1. Machine Learning Pipeline ………………………………………………………

5.1.1. Docker ……………………………………………………………………………

5.1.1.1 Containers and Virtual Machines ………………………………………

5.1.1.2 Images …………………………………………………………………

5.1.1.3 Containers Lifecycle …………………………………………………

5.1.1.4 Networking ……………………………………………………………

5.1.1.5 Docker Compose …………………………………………………………

5.1.2. Apache Kafka …………………………………………………

5.1.2.1 Kafka architecture …………………………………………………

5.1.2.2 Topic structure …………………………………………………

5.1.2.3 Clustered deployment …………………………………………………

5.1.2.4 Parallel processing of the topic ………………………………………

5.1.3.5 Zookeeper (talk about it) ……

5.1.3. Logstash …………………………………………………

5.1.3.1 Data extraction ……………………………………………

5.1.3.2 Data transformation …………………………………………………

5.1.3.3 Data load …………………………………………………

5.1.3.4 Parallel processing …………………………………………………

5.1.3.5. Grok (talk about it) …….

5.1.4. Elasticsearch ………………………………………………

5.1.4.1 Elasticsearch architecture …………………………………………………

5.1.4.2 Difference from a database ………………………………………………

5.1.4.3 Clustered deployment …………………………………………………

5.1.4.4 Search and aggregation ………………………………………………

5.1.5. Kibana ………………………………………………

5.1.5.1 Widget and Dashboard …………………………………………………

5.1.5.2 Discover ………………………………………

5.1.5.3 ELK stack monitor and management ………………………………………

5.1.6. Beats ………………………………………………

5.1.7.1 Filebeat and Metricbeat…………………………………………

5.1.7. Apache Spark ………………………………………………

5.1.7.1 Challenges in big data processing …………………………………………

5.1.7.2 Spark cluster …………………………………………

5.1.7.3 Handling distributed data …………………………………………

5.1.7.4 Broadcast variable …………………………………………

5.1.7.5 Low-level unstructured Spark API ………………………………………

5.1.7.6 Structured Spark API …………………………………………

5.1.7.7 Distributed execution of jobs …………………………………………

5.1.7.8 Mleap (talk about it) ……

5.1.8. Jupyter Notebook ………………………………………………

5.1.9. Summary ………………………………………………

5.2. Présentation de la solution ……………………………………………………………

5.3. Analyse et évaluation des résultats obtenus ………………………………………………

5.4. Déploiement de la solution ………………………………………………………………

1. **Conclusion générale**

6.1. Synthèse des travaux

6.2. Apport du stage

6.3. Difficultés rencontrées

6.4. Perspectives

**Références**

## LISTE DES FIGURES

**FIGURES**

## LISTE DES TABLEAUX

**TABLEAU**

## LISTES DES ABBRÉVIATIONS

**ABBREV**

## INTRODUCTION GÉNÉRALE

La détection des anomalies est l'une des tâches les plus importantes pour construire un système fiable et sûr. L'objectif de la détection d'anomalies est de détecter une déviation significative du comportement du système par rapport au comportement normal. Cette approche est largement utilisée sur des données statiques, par exemple sur des vidages de données de journaux. La plupart des systèmes nécessitent une détection en temps réel des anomalies dans le but spécifique de réduire les dommages qui peuvent être causés par l'ignorance d'une anomalie ou d'une détection ultérieure. Les implémentations récentes de la détection des anomalies sont principalement basées sur des méthodes d'auto-apprentissage. L'apprentissage automatique a entraîné une transformation importante dans le domaine de la détection des anomalies. L'une des méthodologies de détection des anomalies repose sur des algorithmes de regroupement. L'implémentation discutée dans cet article utilise une approche d'évaluation des séries temporelles pour la détection des anomalies. L'article explique le pipeline construit pour la détection des anomalies et la visualisation des résultats.

L'application des méthodologies d'apprentissage machine (ML) dans divers domaines de la vie quotidienne est devenue importante ces dernières années. Les modèles ML sont utilisés dans des environnements opérationnels pour effectuer de nombreuses tâches difficiles, telles que la classification du trafic, la détection automatique des pannes, la reconnaissance d'images, qui nécessitent traditionnellement une expertise humaine spécialisée et des efforts pour être accomplies. Les modèles ML deviennent de plus en plus précis, moins sujets aux erreurs et capables de mettre en évidence des modèles cachés dans les données que même un œil expert ne peut trouver. L'un des objectifs les plus difficiles lors du déploiement d'algorithmes ML, outre le développement des modèles ML, est de faire en sorte que ces modèles ML fonctionnent efficacement dans un environnement opérationnel, par exemple, lorsque l'échelle des données à traiter augmente avec le nombre d'utilisateurs et qu'aucune interruption de service ne peut être tolérée. Dans de tels cas, l'infrastructure matérielle/logicielle qui supporte les modèles ML doit pouvoir évoluer et être résiliente aux pannes logicielles ou matérielles. Dans ce travail, nous proposons un pipeline logiciel distribué, évolutif et tolérant aux pannes qui prend en charge l'ingestion de données et l'application de modèles ML à des données en continu. Le pipeline proposé peut prendre en charge des milliers de sources de données dans un environnement réel et est construit à l'aide de logiciels de pointe et de sources ouvertes qui sont enchaînés pour former un pipeline logiciel qui réalise l'application en temps quasi réel du modèle d'apprentissage automatique. Comme environnement d'application du pipeline d'ingestion de données développé, nous considérons le cas de la gestion des pannes dans les réseaux à micro-ondes, où l'objectif des modèles ML est de détecter et de classer les pannes dans l'équipement à micro-ondes en regardant seulement les journaux de travail du matériel qui rapporte la mesure de puissance sur le lien. Enfin, nous présentons les résultats expérimentaux de ce cas d'utilisation, en nous concentrant sur la latence introduite par les composants du pipeline sur un nombre différent de flux de données actifs.

## RÉFÉRENCES

1. Anomaly Detection [En Ligne] Disponible : <https://www.gavstech.com/anomaly-detection-in-aiops/>
2. What is Elasticsearch? [En Ligne] Disponible: <https://www.elastic.co/guide/en/elasticsearch/reference/8.1/elasticsearch-intro.html>
3. Elastic Machine Learning [En Ligne] Disponible: <https://www.elastic.co/guide/en/machine-learning/current/machine-learning-intro.html#machine-learning-intro>
4. Nginx [En Ligne]. Disponible : <http://nginx.org/en/docs/http/ngx_http_log_module.html#log_format>
5. Mohiuddin Solaimani, “Anomaly Detection for Application Log Data” (2015). Mémoires de maîtrise et recherches de troisième cycle, University of Texas at Dallas, Disponible : <https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1640&context=etd_projects>