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 des dysfonctionnements survenus sur les applications**

\

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. 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. 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. En outre, comme il s'agit d'une approche axée sur l'apprentissage, il est possible de mettre à jour le modèle de manière incrémentielle afin qu'il puisse s'adapter aux nouveaux modèles de journaux qui apparaissent au fil du temps.

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 see later 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. The entire process has been established from the original log to the real-time reliability measurement. New methods were adopted to improve the accuracy, recall and F1 value of anomaly detection. In addition, because it is a learning approach, the model can be incrementally updated to accommodate new log patterns that emerge over time.

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. 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. Un exemple d’anomalie qu’on pourrait citer dans une application pourrait être une faille de sécurité, ou bien une panne de service fournit par un tiers comme le cas de Wizzall Money. Les implémentations récentes de la détection des anomalies sont principalement basées sur des méthodes d'auto-apprentissage.

L'application des méthodologies d'apprentissage machine ML [1] aux opérations informatiques (AIOps) [2], grâce au nombre croissant de frameworks disponibles, souvent open-source, devient omniprésente dans de nombreux domaines. De plus, l'énorme disponibilité de données d'entraînement, due à la grande capacité de stockage des centres de données et aux systèmes modernes de collecte de données, permet de développer des modèles d'apprentissage automatique plus précis et moins sujets aux erreurs. La diffusion de l'apprentissage automatique correspond à un nombre croissant d'utilisateurs et de systèmes qui accèdent aux prédictions fournies par le modèle d'apprentissage automatique. Cela pose de nouveaux défis, car le modèle ML développé doit être placé dans un environnement dit opérationnel. Les environnements opérationnels ne sont généralement pas sous le contrôle des développeurs, contrairement aux environnements de développement, qui sont entièrement sous contrôle. La principale différence est qu'il est difficile de prévoir l'ampleur des données que le système pourrait être amené à traiter et que ces valeurs peuvent changer à tout moment. En outre, les systèmes matériels et logiciels peuvent tomber en panne à tout moment, mais aucun temps d'arrêt ne peut être toléré dans les environnements opérationnels, car les utilisateurs et les systèmes réels dépendent des services fournis par l'infrastructure ML. Pour résoudre ces deux problèmes, l'infrastructure qui alimente le modèle ML doit être mise à l'échelle et répliquée en conséquence.

Un autre défi des environnements opérationnels est que nous sommes soumis à des exigences temporelles strictes, souvent en temps réel, pour éviter les goulots d'étranglement dans le système supporté par le modèle ML. La plupart du temps, l'application des modèles ML n'est qu'un élément d'une chaîne de traitement plus large, appelée pipeline logiciel. Les pipelines sont utilisés pour traiter les flux de données en continu, l'échelle de l'infrastructure résout en partie ces problèmes, il convient également d'exécuter les modèles d'apprentissage automatique sur des logiciels spécialisés qui permettent une prédiction distribuée à faible latence sur de grandes quantités de données. La surveillance des performances et de l'état des logiciels dans le pipeline devient fondamentale pour répondre aux exigences strictes imposées par la nature de l'environnement opérationnel : les administrateurs doivent être en mesure de détecter les applications défectueuses et de les remplacer rapidement, de reconnaître rapidement les éventuels goulets d'étranglement et de les résoudre dès que possible. La surveillance nécessite une application spécialisée et un stockage dédié pour prendre en charge la collecte des métriques [1], ainsi qu'une interface dédiée pour analyser les données collectées.

Un autre aspect important des environnements opérationnels est la gestion de la sécurité, un problème souvent négligé dans les phases de développement. Les attaques de pirates sur les systèmes opérationnels sont de plus en plus fréquentes, ce qui fait de la sécurité l'une des préoccupations les plus importantes pour les environnements opérationnels où des informations sensibles sont traitées. Nous devrions être en mesure de garantir ce que l'on appelle la triade de la CIA [2] : La confidentialité, c'est-à-dire que toutes les communications entre les étapes du traitement ne doivent pas être lues ou mises à la disposition d'entités non autorisées, l'intégrité, les données doivent être conservées et ne pas être modifiées par des utilisateurs malveillants et la disponibilité, c'est-à-dire la capacité du système à fonctionner en permanence. Il est essentiel de garantir cette propriété dans un environnement opérationnel afin d'éviter les fuites de données potentielles ou la mise hors service du système par un attaquant. La sécurisation de la pile d'applications doit donc être effectuée dès le début du processus de développement en appliquant les meilleures pratiques de sécurité telles que le cryptage et l'authentification entre les parties. Le logiciel choisi doit être bien connues pour avoir été développées avec la sécurité comme première préoccupation, ce que l'on appelle la sécurité dès la conception. Les applications à code source ouvert ont un plus grand avantage que les applications à code source fermé en ce qui concerne la sécurité : étant gratuites, elles sont largement adoptées et testées contre les vulnérabilités et les bogues, et comme leur code est accessible au public, il est généralement examiné par des experts en sécurité.

Pour incorporer les exigences susmentionnées dans notre cadre, nous proposons une architecture logicielle en pipeline supportant l'application de modèles ML dans un environnement opérationnel d'une manière évolutive et tolérante aux pannes, et permettant le traitement en temps quasi réel de données en continu. La conception proposée peut prendre en charge jusqu'à des milliers de sources de données dans un environnement réel et imprévisible et est construite à l'aide de modules logiciels de pointe et de sources ouvertes qui sont enchaînés pour former un pipeline logiciel. Tous les composants prennent en charge le déploiement distribué et sont conformes aux meilleures pratiques de sécurité : toutes les applications sont conçues pour fonctionner dans un cluster afin d'effectuer la distribution du travail et le traitement parallèle, et elles incluent le cryptage des données échangées et l'authentification entre les parties. L'architecture de pipeline proposée est composée de quatre parties :

* Ingestion de données : Applications qui prennent en charge l'ingestion des données dans le système. Elle comporte quatre phases : mise en mémoire tampon, prétraitement et stockage. C'est le point d'entrée des sources de données en continu.
* Application de modèle d'apprentissage automatique : les données stockées sont traitées par lots avec le modèle d'apprentissage automatique et les résultats sont stockés en retour.
* Visualisation des données : permet d'examiner les données d'entrée et les résultats du traitement des données ainsi que la prédiction dans une interface.
* Surveillance de la santé : surveille l'état des applications du pipeline afin de détecter les défaillances logicielles et d'y réagir rapidement.

Afin de réaliser une implémentation de référence et d'avoir un cas d'utilisation réel pour tester le pipeline développé, nous considérons l'application de la ML pour la gestion des anomalies dans le projet DER [4] actuellement en développement à TeamX et utilisons un modèle ML prédéveloppé qui va effectuer la prédiction d’une panne potentielle dans l’application.

Ainsi, nous avons structuré le document en quatre grandes parties. La première partie de ce mémoire portera sur la présentation la structure d’accueil. Ensuite, la deuxième partie va décrire la réalisation d’une étude de l’état de l’art technique dans ce domaine. Dans la troisième on va la proposer d’une architecture appropriée pour le système. Ensuite la quatrième partie traitera la mise en œuvre agile de la solution allant de l’implémentation jusqu’au déploiement et enfin proposer des pistes d’évolution pour la solution.

## PRÉSENTATION DE LA STRUCTURE D’ACCUEIL

## 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). Thèse de doctorat, University of Texas at Dallas, Disponible: <https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1640&context=etd_projects>
6. Machine learning on Elastic Search using Apache Spark and ES-Hadoop [En Ligne]. Disponible: <https://excelerate.systems/machine-learning-elasticsearch/>
7. Apache Spark Tutorial [En ligne]. Disponible: <https://www.tutorialkart.com/apache-spark>
8. User Agent Anomaly Detection [En Ligne] Disponible: <https://slideplayer.com/slide/13693768/>
9. <https://posts.specterops.io/threat-hunting-with-jupyter-notebooks-part-1-your-first-notebook-9a99a781fde7>
10. <https://opensource.com/article/19/5/log-data-apache-spark>
11. Spark MLib [En Ligne] Disponible: <https://spark.apache.org/docs/latest/ml-features>
12. Machine Learning Pipeline [En Ligne] Disponible: ​​<https://valohai.com/machine-learning-pipeline/>
13. Machine Learning Model [En Ligne] Disponible:   
    <https://www.youtube.com/watch?v=Om7G2qRBSVw>
14. Embed a Spark ML Model [En Ligne] Disponible:   
    <https://towardsdatascience.com/how-to-embed-a-spark-ml-model-as-a-kafka-real-time-streaming-application-for-production-deployment-933aecb79f3f>
15. Complete Data Science Project Template  (Tres utile pour la presentation) [En Ligne] Disponible:   
    <https://towardsdatascience.com/complete-data-science-project-template-with-mlflow-for-non-dummies-d082165559eb>
16. XGBoost [En Ligne] Disponible:   
    <https://towardsdatascience.com/pyspark-and-xgboost-integration-tested-on-the-kaggle-titanic-dataset-4e75a568bdb>
17. Reason to embed model in kafka istead of remote [En Ligne] Disponible: <https://www.kai-waehner.de/blog/2020/10/27/streaming-machine-learning-kafka-native-model-server-deployment-rpc-embedded-streams/>
18. Creating Apache Kafka Connector [En Ligne] Disponible: <https://www.confluent.io/blog/create-dynamic-kafka-connect-source-connectors/>
19. Send data from kafka to elasticsearch [En Ligne] Disponible: <https://selectfrom.dev/how-to-send-data-from-a-kafka-topic-to-elasticsearch-c623fba84a84>