Ruprecht-Karls-Universität Heidelberg

Institute of Computer Science

Heidelberg Institute for Geoinformation Technology

Master's Thesis

XGBoost-Based ML Framework for Vandalism Detection in OpenStreetMap

Name	: Sri Pavan Sesha Sai Rallapalli
Matriculation number	: 4730140
Supervisor	: Prof. Dr. Ullrich Köthe
Second Supervisor	: Prof. Dr. Alexander Zipf
Date of submission	: 31.01.2025

Acknowledgement

I would like to express my deepest gratitude to my supervisors, Prof. Dr. Ullrich Köthe, Prof. Dr. Alexander Zipf, and Dr. Benjamin Herfort, for their invaluable guidance, expertise, and unwavering support throughout this thesis. Their constructive feedback and profound insights played a pivotal role in shaping the direction and outcome of my research.

Special thanks are extended to my colleagues and mentors at Heidelberg University, whose support and collaboration have significantly enriched my academic journey. I am particularly grateful to the Heidelberg Institute for Geoinformation Technology (HeiGIT) and the Computer Vision and Learning Lab (IWR) for providing essential resources and infrastructure, which were instrumental in the success of this work.

I am also deeply appreciative of the friendships and connections I have formed during my time as a master's student. The support, motivation, and camaraderie shared with my peers have been a constant source of inspiration and strength.

Finally, I would like to express my heartfelt gratitude to my family for their unwavering love, patience, and encouragement throughout this journey. Your belief in me has been a cornerstone of my perseverance and success. This accomplishment would not have been possible without your enduring support.

Declaration of Independence

Sri Pavan Sesha Sai Rallapalli
Heidelberg, 31.01.2025

Summary

For the German version of this summary, refer to Zusammenfassung section !

OpenStreetMap (OSM) is one of the largest, most influential crowdsourced mapping platforms, providing critical geospatial data to industries such as navigation, urban planning, and disaster response. However, this openness exposes OSM to vandalism—ranging from small-scale tag modifications to large-scale, systematic deletions—that can compromise data integrity. Maintaining OSM's reliability requires robust, scalable methods for identifying and mitigating these harmful edits. This thesis mainly focuses on the development of a machine learning pipeline for detecting vandalism in both OSM contributions and changesets. The overall objective is to create a system capable of accurately identifying vandalism, ensuring reliability in real-world applications.

Problem Statement

Detecting vandalism in OSM is challenging due to the vast number of daily edits and the diverse nature of malicious behaviors. Vandalism may appear as subtle changes closely resembling legitimate edits or as extensive transformations spanning entire regions. Moreover, real-time detection is often essential where timely, accurate map data is required. To address these needs, an automated system must efficiently process large data volumes, adapt to evolving vandalism patterns, and offer strong precision without overwhelming human moderators.

Approach

This thesis proposes a comprehensive machine learning pipeline that targets vandalism at two granularities: *contribution-level* (individual edits) and *changeset-level* (groups of edits constituting a single session).

- **Data Acquisition and Preprocessing**: Large-scale OSM data was gath-

ered from monthly snapshots and historical changeset files, yielding labeled vandalism and non-vandalism datasets. These datasets serve as the foundation for model training and evaluation.

- **Feature Engineering**: A rich feature space was constructed, capturing geometric deltas, user behaviors, temporal patterns, and textual/tag modifications. These features empower the classifier to detect both localized anomalies (e.g., a single offensive tag) and broader patterns (e.g., a surge of suspicious edits across multiple objects).

- **Model Development and Training**: A single XGBoost-based pipeline was implemented to handle both contribution-level and changeset-level data, underscoring the algorithm's flexibility. This unified approach was trained on two distinct labeled datasets—one focusing on individual edits and the other on entire sessions—yet it retained the same underlying model architecture, hyperparameters, and feature extraction principles for robust and efficient vandalism classification.

- **Data Splitting and Evaluation**: Multiple strategies (random, geographic, and temporal splits) ensured that the models were tested under realistic conditions. Evaluation metrics—accuracy, precision, recall, F1-score, and AUC-PR/ROC—were complemented by bootstrapping for confidence intervals and geographic breakdowns for region-specific insights.

- **Visualization and Insights**: Temporal/spatial plots, confusion matrices, and precision-recall curves highlighted the classifier's strengths and weaknesses, including hotspots for vandalism or time periods with heightened malicious activity.

Results

Experimental findings demonstrate that the proposed system achieves high accuracy, precision, and recall across a wide range of test scenarios, effectively distinguishing between vandalism and legitimate edits. Furthermore, bootstrap metrics confirm the model's robustness and stability, indicating its ability to adapt to evolving vandalism patterns over time. Geographic evaluations revealed robust performance in multiple continents, and temporal splits demonstrated the model's resilience to changing vandalism patterns. Overall, these outcomes underscore the efficacy and adaptability of the pipeline in real-world OSM vandalism detection.

Significance

By reinforcing data integrity in OSM, this thesis supports a vital crowdsourced resource relied upon by a global user base. The methods described—encompassing multi-level feature extraction, integrated classification pipelines, and diverse evaluation scenarios—serve as a template not only for OSM but also for other crowdsourced platforms grappling with malicious edits. The emphasis on scalability, precision, and interpretability ensures that both community moderators and automated systems can benefit from rapid, accurate vandalism alerts.

Future Directions

While the pipeline achieves strong performance, several avenues remain for further enhancement:

- **Real-Time Integration**: Incorporating real-time processing to flag vandalism immediately after contributions or changesets are submitted.
- **Advanced Model Architectures**: Investigating transformer-based or deep neural network approaches for more nuanced feature interactions, particularly in complex spatio-temporal edits.

Conclusion

This thesis presents a comprehensive and scalable solution to the problem of vandalism detection in OSM contributions and changesets. By uniting powerful feature engineering, a scalable XGBoost pipeline, and rigorous evaluation strategies, the proposed system effectively preserves the integrity and reliability of OSM—a critical geospatial resource. The methodology and insights herein extend beyond OSM, offering a blueprint for safeguarding other large-scale collaborative platforms against malicious behavior.

Zusammenfassung[1]

OpenStreetMap (OSM) ist eine der größten und einflussreichsten Crowdsourcing-Plattformen für digitale Kartendaten und liefert wichtige Geoinformationen für Bereiche wie Navigation, Stadtplanung und Katastrophenschutz. Diese Offenheit birgt jedoch ein Risiko für Vandalismus—von kleineren Tag-Änderungen bis hin zu großflächigen, systematischen Löschungen—die die Datenintegrität gefährden können. Um die Zuverlässigkeit von OSM zu gewährleisten, sind robuste, skalierbare Methoden zur Erkennung und Eindämmung schädlicher Bearbeitungen erforderlich. Diese Arbeit konzentriert sich auf die Entwicklung einer Machine-Learning-Pipeline zur Erkennung von Vandalismus sowohl in *Contributions* (einzelnen Bearbeitungen) als auch in *Changesets* (Gruppen von Bearbeitungen). Das übergeordnete Ziel ist die Entwicklung eines Systems, das Vandalismus zuverlässig identifizieren kann und sich für reale Anwendungen eignet.

Problemstellung

Die Erkennung von Vandalismus in OSM ist eine Herausforderung, da täglich eine große Anzahl an Bearbeitungen erfolgt und schädliches Verhalten sehr unterschiedlich ausfallen kann. Vandalismus kann subtil sein und seriösen Änderungen ähneln oder umfangreiche Umgestaltungen ganzer Regionen umfassen. Zudem ist oftmals eine Echtzeit-Erkennung entscheidend, wenn aktuelle und präzise Kartendaten benötigt werden. Für diese Anforderungen muss ein automatisiertes System in der Lage sein, große Datenmengen schnell zu verarbeiten, sich an veränderliche Vandalismusmuster anzupassen und eine hohe Genauigkeit aufweisen, ohne Moderatoren durch zu viele Fehlalarme zu überfordern.

[1] *This translation is provided for convenience and is not legally binding. The official version, available in English, remains the authoritative reference in case of any discrepancies.*

Vorgehensweise

In dieser Arbeit wird eine umfassende Machine-Learning-Pipeline vorgestellt, die Vandalismus auf zwei Granularitätsebenen untersucht: *Contributions* (individuelle Bearbeitungen) und *Changesets* (Sitzungen mit mehreren Bearbeitungen).

- **Datenerfassung und -vorverarbeitung**: Umfangreiche OSM-Daten wurden aus monatlichen Snapshots und historischen Changeset-Dateien extrahiert, um Datensätze mit gekennzeichnetem Vandalismus und Nicht-Vandalismus aufzubauen. Diese Datensätze bilden die Grundlage für das Training und die Bewertung der Modelle.

- **Feature Engineering**: Ein umfangreicher Feature-Satz wurde entwickelt, der geometrische Änderungen, Nutzerverhalten, zeitliche Muster und Text-/Tag-Modifikationen erfasst. Mithilfe dieser Merkmale lassen sich sowohl lokal begrenzte Anomalien (z.B. ein einzelnes beleidigendes Tag) als auch umfassendere Auffälligkeiten (z.B. eine Welle verdächtiger Bearbeitungen über mehrere Objekte) erkennen.

- **Modellentwicklung und Training**: Eine einzige XGBoost-basierte Pipeline wurde implementiert, die sowohl für Contribution- als auch für Changeset-Daten genutzt werden kann, was die Flexibilität des Algorithmus unterstreicht. Dieser einheitliche Ansatz wurde auf zwei unterschiedlichen gekennzeichneten Datensätzen trainiert—einer mit Fokus auf einzelne Edits und der andere auf gesamte Sitzungsvorgänge. Dennoch behielt er dieselbe Modellarchitektur, Hyperparameter und Prinzipien der Merkmalsextraktion bei, um eine robuste und effiziente Klassifizierung von Vandalismus zu gewährleisten.

- **Datenaufteilung und Bewertung**: Verschiedene Strategien (zufällige, geografische und zeitliche Aufteilungen) stellten sicher, dass die Modelle unter realistischen Bedingungen getestet wurden. Die Leistungskennzahlen—Genauigkeit, Präzision, Recall, F1-Score und AUC-PR/ROC—wurden durch Bootstrapping für Konfidenzintervalle und geografische Auswertungen zur Untersuchung regionsspezifischer Unterschiede ergänzt.

- **Visualisierung und Erkenntnisse**: Zeitliche/räumliche Diagramme, Konfusionsmatrizen und Precision-Recall-Kurven veranschaulichten die Stärken und Schwächen der Klassifikatoren, einschließlich potenzieller Vandalismus-Hotspots oder Zeiträume mit vermehrtem böswilligen Verhalten.

Ergebnisse

Die Experimente zeigen, dass das vorgestellte System eine hohe Genauigkeit, Präzision und Recall in vielfältigen Testszenarien erreicht und so Vandalismus effektiv von legitimen Bearbeitungen unterscheidet. Zudem bestätigen Bootstrap-Ergebnisse die Robustheit und Stabilität des Modells und belegen seine Anpassungsfähigkeit an sich wandelnde Vandalismusmuster im Zeitverlauf. Geografische Auswertungen weisen auf eine starke Leistung auf mehreren Kontinenten hin, und zeitliche Splits verdeutlichen die Widerstandsfähigkeit des Modells gegenüber veränderten Vandalismusstrategien. Insgesamt unterstreichen diese Resultate die Effizienz und Flexibilität der Pipeline bei der praktischen Erkennung von OSM-Vandalismus.

Bedeutung

Durch die Stärkung der Datenintegrität in OSM unterstützt diese Arbeit eine wesentliche Crowdsourcing-Ressource, die weltweit genutzt wird. Die vorgestellten Methoden—darunter mehrstufige Feature-Extraktion, integrierte Klassifikationspipelines und breit gefächerte Bewertungsstrategien—dienen nicht nur OSM, sondern können auch auf andere Plattformen mit offenem Bearbeitungsmodell angewendet werden, die mit schädlichen Beiträgen zu kämpfen haben. Die Fokussierung auf Skalierbarkeit, Präzision und Interpretierbarkeit gewährleistet, dass sowohl Community-Moderatoren als auch automatisierte Systeme von zügigen, präzisen Warnmeldungen profitieren können.

Zukünftige Ansätze

Obwohl die Pipeline bereits starke Leistungen zeigt, bestehen noch folgende Erweiterungsmöglichkeiten:

- **Echtzeit-Integration**: Implementierung eines Echtzeitprozesses, um Vandalismus unmittelbar nach dem Einreichen von Contributions oder Changesets zu erkennen.
- **Erweiterte Modellarchitekturen**: Erforschung von transformerbasierten oder tiefen neuronalen Netzwerken für komplexere Feature-Interaktionen, insbesondere bei anspruchsvollen raum-zeitlichen Bearbeitungen.

Fazit

Diese Arbeit stellt eine umfassende und skalierbare Lösung zur Erkennung von Vandalismus in OSM-Contributions und -Changesets vor. Durch den Einsatz leistungsfähiger Feature-Engineering-Methoden, einer skalierbaren XGBoost-Pipeline sowie einer rigorosen Evaluationsstrategie trägt das System entscheidend dazu bei, die Integrität und Zuverlässigkeit von OSM als essenzielle Geodatenquelle zu sichern. Die hier vorgestellte Methodik und gewonnenen Erkenntnisse reichen über OSM hinaus und bieten eine Orientierung für den Schutz anderer groß angelegter kollaborativer Plattformen vor böswilligem Verhalten.

Abstract

OpenStreetMap (OSM) is a globally crowdsourced mapping project that empowers volunteers to create and update geographic information. While this open collaboration accelerates the growth and coverage of map data, it also introduces vulnerabilities to vandalism, encompassing both unintentional errors and deliberate malicious edits. The reliability and trustworthiness of OSM data therefore hinge on automated vandalism detection mechanisms that can efficiently and accurately flag suspect edits.

This thesis develops a comprehensive machine learning pipeline for detecting vandalism in OSM, leveraging XGBoost for classification at both *contribution-level* and *changeset-level* data. The methodology is modular, comprising data preprocessing, feature engineering, and model training and evaluation. Spatial, temporal, and user-centric features are extracted to uncover subtle vandalism cues that might otherwise elude simpler heuristic checks. Real-world deployment conditions are simulated through evaluation sets with realistic vandalism ratios, while extensive experiments verify that the pipeline consistently achieves high accuracy, recall, and precision across diverse scenarios. The approach is further supported by temporal and geographic analyses, illustrating how the system pinpoints malicious behavior at multiple scales.

Beyond robustness and predictive performance, the pipeline underscores scalability: it efficiently processes millions of OSM entries each month. Flexibility is also a key strength; the system can seamlessly handle fine-grained (*contribution*) or session-level (*changeset*) detection tasks. Taken together, these results advance the state of the art in OSM vandalism detection, offering a solution that is both rigorous and operationally feasible. Future enhancements may include real-time processing capabilities, deeper integration of neural network models, and more extensive community-driven validation frameworks. By ensuring high-quality map data, this work bolsters the integrity and utility of OSM for a broad spectrum of applications.

Contents

1 Introduction

OpenStreetMap (OSM) [?] is a globally collaborative platform where volunteers contribute to creating and maintaining a free and openly accessible map of the world [?, ?]. It has experienced immense growth since its inception, fueled by a community of contributors who collectively add, edit, and curate geospatial information. Today, OSM data is extensively used in applications ranging from navigation and routing (e.g., in smartphone apps) to humanitarian and disaster response efforts. The open and inclusive nature of OSM enables anyone with internet access to contribute map data about roads, buildings, public facilities, and natural features. This diversity of inputs transforms OSM into a highly dynamic environment, ensuring that map information remains up-to-date and continuously expanding.

However, the very openness that drives OSM's success also presents vulnerabilities. One of the most pressing challenges is vandalism [?]—the deliberate or accidental submission of erroneous or malicious edits that can undermine data integrity and trust. Vandalism can appear in countless forms: from adding fictitious roads that do not exist on the ground, to completely removing landmarks critical to local navigation. While some vandalism is benign or arises from well-meaning but inexperienced contributors, other acts can be maliciously intended to mislead or disrupt the platform. Beyond obvious malicious acts, subtler forms of vandalism can also have significant repercussions on OSM's quality. For instance, an experienced vandal could modify map geometry slightly or alter textual metadata in ways that go unnoticed by human reviewers for extended periods.

Given that OSM is utilized globally by a vast ecosystem of researchers, developers, corporations, and governmental agencies, maintaining high data quality is of paramount importance. Public and private sectors rely on OSM for tasks ranging from route planning to epidemiological studies. Errors injected by vandalism risk damaging user confidence and may have real-world consequences, especially in critical situations such as emergency response. Manual review processes, while helpful, cannot keep pace with the enormous inflow of daily edits, which numbers in the millions. As a result, automated techniques—particularly those rooted in machine learning (ML)—are increasingly integral to the ongoing stewardship of OSM data.

1.1 Motivation and Significance

The motivation behind this thesis stems from the widespread and escalating demand for accurate, timely identification of harmful OSM edits. Early detection of vandalism is crucial for preventing the proliferation of erroneous information across the numerous services and applications that depend on OSM data. Once inaccurate edits propagate into navigation software or analytics tools, they can create cascading effects, such as misrouting, resource misallocation, or even compromised safety in critical operations. By catching and rectifying vandalism early in the data pipeline, the reliability of OSM can be greatly enhanced. Automated detection methods provide a compelling solution. They can continuously analyze large volumes of OSM edits in near real-time, generating alerts for human moderators. Modern machine learning techniques—especially ensemble-based algorithms like XGBoost [?, ?]—are well-suited to handling structured data at scale. These methods can parse complex interactions among features, ranging from contributor histories to geometric attributes of map features, to identify suspicious patterns. When designed efficiently, ML-driven systems complement the volunteer OSM community, creating a more robust and resilient mapping infrastructure.

A prime example of institutional commitment to OSM data quality is the Heidelberg Institute for Geoinformation Technology (HeiGIT)[?], which focuses on developing advanced geoinformation services and open-source solutions. Many of HeiGIT's tools and research initiatives revolve around OSM, including the analysis of changesets for data quality. However, existing methods primarily target vandalism detection at the changeset level [?, ?]. While changeset-based detection is vital, it may overlook granular anomalies that appear in *individual* contributions. Contributions represent single edits to OpenStreetMap data, encompassing the addition or modification of geographic features, metadata, relations, or other map-related elements. By contrast, *changesets* [?] encapsulate multiple contributions grouped together in a single upload. Although some prior studies have outlined changeset-level strategies, there is a noticeable gap in comprehensive *contribution-level* vandalism detection.

This thesis addresses this gap by proposing a dual-focus pipeline—one that handles detection for contributions while still supporting the changeset-level perspective—to ensure that both large-scale and fine-grained vandalism patterns are identified. By bridging these two granularities, the system aims to catch subtle edits that might slip under changeset-level scrutiny, as well as wider malicious behavior affecting multiple map features within a single session. In doing so, the approach aspires to bolster OSM data integrity for crucial real-world applications, from advanced routing systems to crisis management tasks, where the accuracy of map data can be a decisive factor in outcomes.

1.2 Overview of the Research Problem

Malicious edits in OSM may be intentionally disruptive (e.g., deleting large swaths of roads) or subtly misleading (e.g., altering a place name by a small but impactful degree), whereas unintentional errors arise from well-meaning contributors with limited expertise. This wide spectrum of vandalism types, compounded by OSM's global scale, makes automated detection both crucial and difficult. Several factors underscore the complexity of vandalism detection. First, the volume of OSM edits is immense, with millions of contributions and changesets generated monthly.

Attempting to monitor such a continuous torrent of data manually or using naive methods is infeasible. Second, vandalism can materialize in diverse ways: from single-tag manipulation to multi-faceted changeset alterations. Generic heuristic-based rules often fail to capture novel or evolving attack patterns, especially those that mimic legitimate modifications.

From a computational standpoint, traditional techniques that scan the dataset in a single pass tend to become intractable at scales involving hundreds of millions of records. Efficient chunk-based processing, with parallelized feature extraction and classification, is essential for real-time or near real-time decision-making. A robust system must therefore balance high accuracy with performance requirements. If the approach becomes too slow or resource-intensive, it risks impracticality in real-world deployment scenarios where OSM data are continuously updated across diverse global regions.

In summary, the research problem centers on designing an automated, scalable framework capable of identifying a multitude of vandalism types in OSM's ever-growing data. Such a system should handle both fine-grained edits at the *contribution* level and larger-scale patterns discernible at the *changeset* level, bridging the gap between localized anomalies and session-wide disruptions.

1.3 Scope of the Thesis

This thesis tackles the specific task of building and evaluating a scalable machine learning pipeline for vandalism detection in OSM, addressing both *contribution-level* and *changeset-level* analyses. While the broader realm of OSM data quality includes numerous challenges (e.g., incomplete tagging, schema inconsistencies, or map validation), this work focuses on malicious or unintentional edits that significantly degrade spatial or attribute integrity. Within this defined scope, the thesis encompasses four main areas of investigation:

- **Feature Engineering:** Identify and extract diverse features spanning *ge-*

ometric, *textual*, *user-history*, *temporal*, *spatial*, *map-based*, and *OSM element–focused* domains. These include bounding-box changes, tag modifications, user edit frequencies, time-of-day patterns, map feature indicators (e.g., roads, buildings), and historical context of edited objects. Such a rich feature set ensures that the model can detect vandalism signals at multiple levels of granularity, from small metadata changes to large-scale deletions.

- **Pipeline Architecture:** Devise a robust, end-to-end workflow capable of handling massive OSM datasets through chunk-based data loading and parallelized feature extraction. The architecture should accommodate both contribution-level and changeset-level data, ensuring that it can process large volumes of edits in a scalable manner. Techniques such as distributed model inference and memory-efficient data handling are employed to sustain high throughput without sacrificing predictive accuracy.

- **Model Training and Evaluation:** Develop and train an XGBoost classifier, experimenting with hyperparameter tuning to optimize performance. The pipeline is validated against labeled data (contribution-level and changeset-level) for training and validation, while also facilitating large-scale inference on unlabeled data. Metrics of interest include precision, recall, F1-score, and computational efficiency, with a focus on achieving balanced performance suitable for real-time or near real-time vandalism detection.

- **Performance Assessment:** Conduct comprehensive benchmarks to compare accuracy and resource usage in various scenarios. Special attention is given to user-history signals and contextual features that can influence detection performance, along with analyzing how chunk-based processing affects throughput. The evaluation explores whether the pipeline can effectively detect subtle single-edit vandalism and more obvious multi-edit disruptions at the changeset level.

By restricting the thesis to these focal points, the research provides a specialized yet adaptable framework that can be integrated into existing OSM oversight tools. The methodology does not address every nuance of OSM data quality—such as complex semantic tagging validation or duplication detection—but instead aims to produce a high-impact solution for automated vandalism identification. Ultimately, the combination of detailed feature engineering, a scalable pipeline, and a tailored ML model positions this thesis to offer tangible advancements in OSM data integrity for real-world scenarios.

1.4 Objectives and Research Questions

The primary aim of this thesis is to develop a robust and scalable machine learning framework for detecting vandalism in OpenStreetMap (OSM) contributions and changesets. To achieve this, the research is guided by the following key objectives and research questions.

1.4.1 Research Objectives

The objectives of this research are:

- **Feature Engineering for Vandalism Detection:** Develop a comprehensive set of features capturing spatial, temporal, textual, and user-specific signals indicative of vandalism. This involves leveraging diverse data characteristics to enhance the model's ability to distinguish between legitimate contributions and malicious edits.
- **Scalable Machine Learning Pipeline:** Design and implement a highly scalable machine learning pipeline capable of handling the extensive volume of OSM data with minimal memory overhead. Computational efficiency is critical for processing millions of contributions generated monthly.
- **Optimizing Predictive Performance:** Achieve high predictive performance, measured through accuracy, precision, and recall, while minimizing

false positives that could overwhelm human reviewers. The solution must strike a balance between predictive power and practical usability.

- **Investigating the Role of Contextual Features:** Examine how historical features, user editing behavior, and other contextual cues contribute to improving vandalism detection. This includes evaluating the impact of these features under different real-world scenarios.

1.4.2 Research Questions

To address the challenges associated with vandalism detection in OSM, this thesis explores the following research questions:

- **Feature Effectiveness:** Which features are most effective in distinguishing vandalism from legitimate contributions, and how do different feature types (e.g., user behavior, spatial edits) contribute to model performance?

- **Scalability and Efficiency:** What techniques can be employed to process large-scale OSM datasets efficiently without compromising prediction accuracy?

- **Model Trade-offs:** What are the trade-offs between model complexity and usability in real-world deployments? How can the model achieve high detection accuracy while maintaining operational feasibility?

- **Integration into OSM Workflows:** How can the proposed solution be seamlessly integrated into existing OSM moderation workflows to enhance real-time vandalism detection?

By addressing these objectives and research questions, this thesis ensures a comprehensive exploration of both the technical and practical dimensions of vandalism detection, contributing to a scalable and effective solution for safeguarding OSM data integrity.

1.5 Thesis Outline

To comprehensively address the research problem, the thesis is organized into the following chapters:

- **Chapter 1: Introduction** – Provides an overview of OpenStreetMap (OSM) vandalism, the motivation for machine learning-based detection, and the core objectives and scope of this research.
- **Chapter 2: Background and Related Work** – Reviews existing literature on vandalism detection in collaborative mapping platforms, tracing the evolution from heuristic methods to machine learning approaches. It also compares different ML models for tabular OSM data and discusses the challenges in this domain.
- **Chapter 3: Methods** – Details the methodological approach, including data preparation, feature engineering, data splitting strategies, and model architecture. It introduces the XGBoost-based classification model and the enhancements applied to changeset-level detection, such as the hyper-classifier and meta-classifier.
- **Chapter 4: Model Performance** – Evaluates the effectiveness of the proposed models, comparing contribution-level and changeset-level detection results. This chapter presents performance metrics, visual analyses (ROC, PR curves), geographical evaluations, and feature importance studies.
- **Chapter ??: Application: Vandalism in OpenStreetMap** – Explores real-world applications of the trained models, analyzing key vandalism trends, seasonal variations, and the potential for real-time monitoring. Includes visualizations such as heatmaps to highlight high-risk areas.
- **Chapter ??: Discussion and Future Work** – Reflects on the research findings, discusses their implications for OSM integrity, and highlights limi-

tations. This chapter also outlines future research directions, including improvements in model interpretability and integration into real-time OSM workflows.

By systematically exploring these chapters, readers will gain both theoretical and practical insights into how machine learning can effectively combat vandalism in large-scale, dynamic geospatial datasets such as OpenStreetMap.

2 Background and Related Work

OpenStreetMap (OSM) has evolved from a modest, community-driven project into a globally influential mapping platform, hosting millions of user contributions each month. This chapter provides a broad overview of vandalism detection research within OSM and similar collaborative environments, reflecting on the transition from simple rule-based approaches to advanced machine learning (ML) methods. It also reviews and compares state-of-the-art models applicable to tabular data, along with the challenges these methods face when scaled to the large, dynamic OSM ecosystem.

2.1 Introduction to Vandalism Detection in Collaborative Platforms

Vandalism—defined as intentional or harmful edits that degrade the integrity of shared data—reoccurs in numerous crowdsourced systems. In such open-edit platforms, the tension between openness and data quality is ever-present. Rapid detection of vandalism is paramount because erroneous or malicious edits can undermine user trust and, in critical domains such as navigation or disaster response, lead to severe consequences.

Although this thesis concentrates on OSM-specific solutions, insights from other platforms remain highly relevant. For instance, Wikipedia and Wikidata also confront malicious edits despite harnessing vast volunteer communities. Growing interest in vandalism detection tools for public knowledge bases has produced a number of studies on various online collaborative projects over the years, including OpenStreetMap [?, ?, ?], Wikidata [?, ?, ?, ?, ?], and Wikipedia [?, ?, ?, ?, ?, ?, ?, ?].

For Wikipedia, vandalism detection has evolved from rule-based systems to sophisticated machine learning approaches. WikiTrust, introduced by Adler et al. (2010), used author reputations and edit histories to detect vandalism [?]. Later studies integrated metadata, natural language processing (NLP), and user behavior analysis to enhance detection accuracy [?, ?, ?, ?]. Wikidata has seen similar developments, with vandalism detection tasks at the WSDM Cup 2017 fostering various models, including large-scale linear classifiers [?], production-oriented approaches [?], and automated systems leveraging structured data patterns [?, ?].

Deep learning strategies further improved detection performance. Martinez-Rico et al. (2019) explored deep learning-based vandalism detection for Wikipedia [?], while Yuan et al. (2017) proposed embedding-based early warning systems [?]. These studies underscore the importance of high-quality labeled data, adaptable architectures, and ongoing model retraining to keep pace with evolving vandalism tactics. The challenges faced in Wikipedia and Wikidata are analogous to those in OSM, reinforcing the need for robust and scalable vandalism detection models.

2.2 Evolution of Vandalism Detection in OSM

OpenStreetMap, as a map of the world collaboratively built by volunteers, faces unique challenges in preserving data quality due to the diverse range of edits—ranging from minor tag adjustments to comprehensive alterations of geographic features. Over time, the community has experimented with numerous strategies to maintain the integrity of this enormous dataset.

2.2.1 Early Manual and Heuristic Defenses

In the early stages of OSM, contributors relied on manual inspection, mailing lists, and forum discussions to identify questionable edits. However, as the volume of contributions grew, these manual methods became increasingly impractical. To address this, early rule-based tools such as OSMCha [?] were introduced, designed to flag large bounding-box expansions and bulk deletions for closer review. While these heuristics effectively detected blatant vandalism, they also generated numerous false positives, particularly in cases of legitimate large-scale changes, and struggled to identify more subtle manipulations.

Recognizing these limitations, Alexander Zipf et al. [?] conducted an in-depth analysis of vandalism in OSM and developed a rule-based system for automated detection. Their study underscored the escalating challenge of ensuring data reliability in crowd-sourced platforms and highlighted the necessity for robust, automated solutions to counteract diverse forms of vandalism. Despite their contributions, heuristic-based methods lacked adaptability, prompting the shift towards more flexible, data-driven approaches capable of learning from evolving editing behaviors.

2.2.2 Unsupervised Clustering for Anomaly Detection

Beyond early heuristics, certain researchers explored unsupervised methods to pinpoint anomalous edits without extensive labeled data. Truong et al. (2018) proposed clustering techniques that grouped edits by similarity (e.g., spatial proximity, metadata), then flagged clusters with outliers as suspicious [?]. While this approach captured region-specific vandalism patterns, it also yielded significant false positives—merging legitimate bulk edits with anomalies. These challenges underscored the limitations of purely unsupervised systems for real-time detection in a globally distributed map.

2.2.3 Early Supervised Learning for Vandalism Detection

A major transition from heuristics and clustering-based detection methods to fully supervised learning occurred with the introduction of models trained on labeled vandalism corpora. These models incorporated a variety of predictive signals, such as:

- Edit metadata (e.g., changeset size, frequency of edits)
- User reputation metrics
- Geometric changes to map elements
- Natural language processing (NLP) features extracted from changeset comments

One of the earliest such systems was **OSMWatchman**, introduced by Truong et al. (2020), which employed a **Random Forest classifier** to detect vandalism by combining user statistics, changeset metadata, and spatial context [?]. This work demonstrated the benefits of moving beyond static heuristics, significantly improving accuracy over clustering-based methods. However, it also revealed key challenges—such as the **limited availability of labeled vandalism cases**—that constrained model generalization.

Building upon these early supervised models, researchers began incorporating more advanced representations of user behavior and edit history, leading to **deep learning-driven approaches** such as embedding-based and attention-based models.

2.2.4 Advanced Machine Learning: Embedding and Attention-Based Models

To further improve vandalism detection, researchers explored sophisticated machine learning techniques, particularly **embedding-based** and **attention-based** models. These methods aimed to extract richer information from user behaviors, textual content, and geometric patterns.

User Embedding-Based Approaches. Li et al. (2021) introduced an embedding-based framework where user behaviors were mapped into high-dimensional vector representations [?]. By analyzing past contribution frequency, geographic focus, and tagging patterns, these embeddings captured nuanced editing behaviors indicative of malicious intent. The model then integrated these embeddings into a Gradient Boosted Decision Tree (GBDT) classifier, significantly improving precision and recall compared to previous rule-based approaches. However, this method had key limitations:

- Dependence on historical data, making it ineffective for new or low-activity users.
- High computational cost for generating and maintaining embeddings in real time.

Attention-Based Deep Learning. Tempelmeier et al. (2022) proposed an alternative approach leveraging deep learning with **multi-head attention mechanisms** [?]. Their model dynamically prioritized the most relevant features within changeset metadata, textual attributes, and geometric changes, making it particularly effective at detecting complex and multi-faceted vandalism patterns. Despite achieving state-of-the-art accuracy, this approach faced challenges such as high computational requirements, necessitating significant GPU resources.

The various approaches explored in vandalism detection—embedding-based, attention-driven, clustering-based, and hybrid models—offer unique advantages while facing notable limitations. These challenges emphasize the need for scalable solutions that balance model performance with practical efficiency. This thesis builds upon these advancements by employing a gradient-boosted model (XGBoost), which maintains high predictive power while remaining computationally efficient for large-scale OSM data.

2.3 Comparison of ML Models for Tabular OSM Data

While OSM data may appear heterogeneous—encompassing geometric, textual, temporal, and user-centric attributes—it often ultimately resides in tabular form. For such structured data, three notable machine learning models warrant detailed consideration: XGBoost, TabNet, and FT-Transformers.

2.3.1 XGBoost: Extreme Gradient Boosting

XGBoost is a gradient-boosted decision tree (GBDT) model recognized for its high performance and computational efficiency [?]. By constructing multiple decision trees and optimizing their combination through gradient boosting, XGBoost produces robust predictions for binary classification tasks such as vandalism detection in OSM. The model excels in adaptability, effectively handling mixed data types, including numerical and categorical features. This makes it particularly suitable for OSM data, which integrates diverse attributes such as changeset metadata, user behavior metrics, and geometric properties. Additionally, XGBoost is computationally efficient, offering fast training and inference times, and its feature importance rankings provide interpretability, allowing insights into the most predictive attributes. Its scalability is another advantage, supporting parallel computation to handle large datasets efficiently. However, XGBoost does have limitations: it heavily relies on well-engineered features, requiring significant preprocessing and domain knowledge, and it may struggle with extremely high-dimensional or sparse data compared to deep learning models like FT-Transformers.

2.3.2 TabNet: A Deep Learning Model for Tabular Data

TabNet, a neural network explicitly designed for tabular data, leverages an attention mechanism to dynamically focus on the most relevant features during training [?]. This feature reduces the need for extensive feature engineering while retaining interpretability through sparse attention mechanisms that highlight feature importance for individual predictions. Additionally, TabNet is adept at handling hierarchical feature dependencies, uncovering intricate relationships in data. Despite these strengths, TabNet requires longer training times compared to XGBoost. Its higher computational resource demands during both training and inference further limit its practicality for medium to large-sized datasets like those in OSM.

2.3.3 FT-Transformers: Transformer Models for Tabular Data

FT-Transformers extend the transformer architecture, originally designed for natural language processing (NLP), to tabular data by tokenizing features and employing self-attention mechanisms [?]. These models excel in capturing complex feature interactions across high-dimensional or sparse datasets, reducing dependence on manual feature engineering. FT-Transformers are particularly well-suited for datasets with intricate relationships between features, enabling end-to-end learning. However, these models are computationally intensive, with significant resource requirements for training and inference. Additionally, as a relatively new approach, FT-Transformers lack the mature tooling and documentation available for XGBoost.

2.3.4 Model Selection: Why XGBoost?

In the context of vandalism detection, XGBoost emerges as the most suitable model for OSM data due to its balance of accuracy, interpretability, and computational efficiency. Studies have demonstrated that XGBoost consistently matches or outperforms deep learning–based models like TabNet or FT-Transformers for structured datasets [?, ?]. Its ability to adapt to diverse feature types and provide interpretable feature importance rankings aligns well with the requirements of detecting vandalism in OSM. Moreover, the computational efficiency of XGBoost, makes it a practical choice for real-world implementation. As a result, the pipeline proposed in this thesis incorporates XGBoost to accommodate the diverse attributes of OSM edits effectively.

2.4 Current Challenges in Vandalism Detection Research

Despite progress, OSM vandalism detection still faces key hurdles:

Label Scarcity and Quality. High-quality labeled datasets for vandalism remain limited. Studies like Li et al. (2021) [?] introduce corpora, but these are not always updated or comprehensive, restricting the model's ability to adapt to evolving malicious behaviors. Crowdsourced labeling or semi-supervised learning could partially alleviate this shortfall.

Scalability Demands. Given OSM's rapid editing pace, handling potentially millions of updates daily can overburden even sophisticated ML systems. Efficient data chunking, parallel processing, and incremental model updates are required, as highlighted by Tempelmeier et al. (2022) [?], to keep pace with near real-time ingestion.

Adaptability to Evolving Vandalism. Attack vectors change over time, whether through new editing tools or emergent malicious patterns. ML solutions must be retrained periodically on fresh data, or employ online learning approaches, to remain robust against newly crafted disruptions.

2.5 Summary

In summary, OSM vandalism detection has transitioned from simple rule-based alerts to data-intensive, ML-based solutions. Key insights from Wikidata and Wikipedia underscore the broad utility of supervised learning for collaborative platform vandalism, while embedding-based and attention-driven methods exemplify cutting-edge OSM research. However, large-scale deployments require trade-offs in model selection, leaning toward robust yet efficient algorithms like XGBoost, and well-crafted features spanning textual, geometric, user-centric, temporal, and additional map-based domains.

This thesis builds on the strengths of these approaches by adopting a balanced strategy. Specifically, it leverages rich features from OSM data and integrates them into scalable tree-based machine learning models, such as XGBoost. This approach aims to combine the interpretability and efficiency of traditional models with the robustness of feature-rich representations, thereby addressing the practical limitations identified in embedding-based and attention-driven methods.

The following chapters present the methodology for constructing a scalable and efficient XGBoost-based vandalism detection pipeline. This includes data preparation, feature engineering, and model design tailored for both contribution-level and changeset-level detection. Additionally, the thesis explores enhancements such as the hyper-classifier and meta-classifier to improve detection accuracy by leveraging aggregated signals from individual edits.

3 Methodology

This chapter presents the methodological framework designed to detect vandalism across OpenStreetMap (OSM) contributions and changesets. Building on the insights from the background chapter (Chapter 2), the methodology addresses two core demands of OSM data: handling its substantial volume and diversity, and capturing the multitude of vandalism patterns that may arise in small, individual edits or large, session-wide changes. The approach hinges on a configurable pipeline that can be executed independently on **contribution-level** data (fine-grained edits) or **changeset-level** data (aggregated sessions). By retaining a similar structure but separate executions for each perspective, the pipeline captures unique signals essential for robust vandalism detection without conflating their distinct data structures.

3.1 Introduction to the Methodology

The central challenge in OSM vandalism detection is twofold:

(1) effectively managing a high-throughput data stream (often tens of millions of edits each month), and

(2) accurately detecting a broad spectrum of malicious behaviors, ranging from subtle, localized edits to extensive, orchestrated attacks within a single changeset session.

In response, the proposed pipeline unites parallel processes for contribution-level data and changeset-level data, powered by a core machine learning workflow centered on **XGBoost**. Drawing on chunk-based feature extraction, hyperparameter optimization, and rigorous evaluation strategies, this chapter demonstrates how the pipeline copes with OSM's dynamic nature while retaining interpretability and efficiency. Furthermore, it covers strategies to enhance changeset-level detection using insights from contribution-level predictions, ensuring minimal gaps in coverage.

3.1.1 Purpose and Goals

The overarching purpose of this methods chapter is to detail how the framework confronts OSM's colossal data scale and diverse vandalism patterns. Specifically, the key objectives include:

- **High-Volume Data Throughput**: Designing a chunk-based pipeline to process millions of monthly edits within hours, keeping computational and memory overhead in check.
- **Dual Data Perspectives**: Facilitating separate pipeline executions for *contribution-level* (isolated edits) and *changeset-level* (session-wide batches), allowing each vantage to expose particular vandalism indicators.
- **Core Classification Model (XGBoost)**: Exploiting XGBoost's capability to handle structured tabular data with high accuracy and interpretability. Hyperparameter tuning is incorporated to optimize predictive performance.
- **Ensuring Strong Predictive Metrics**: Maintaining precision, recall, F1-score, AUC-PR, and ROC-AUC at levels sufficient to minimize both missed vandalism and false alarms.
- **Real-Time or Near Real-Time Feasibility**: Achieving inference speeds that suit short-window detection (e.g., every 5 minutes) and large-file batch processing (monthly updates in 1–2 hours). This aligns with OSM's evolving data streams and the operational needs of platforms like HeiGIT.

- **Feature Diversity and Scalability**: Unifying user behaviors, OSM element metadata, temporal and geometric signals, and textual attributes, all in a scalable manner to detect subtle or overt malicious edits.
- **Enhancing Changeset-Level Detection with Per-Edit Insights**: Integrating aggregated per-contribution predictions (hyper-classifier) and fusing them with a normal changeset model (meta-classifier) when needed, improving session-wide detection of mass or systematic vandalism.

In meeting these goals, the pipeline provides a holistic solution that pinpoints vandalism early—protecting OSM's data integrity and mitigating the broader risks of malicious edits in downstream applications such as navigation or disaster response.

3.1.2 Data Perspectives

One of the most distinctive aspects of our methodology is the deliberate treatment of OSM data at two complementary levels: contribution-level and changeset-level. In many crowdsourced environments, including OSM, the simplest unit of data is often an individual edit or contribution—such as adding a building tag or updating a highway's geometry. However, edits in OSM are commonly submitted in groups known as changesets, each of which may contain numerous contributions made during a single upload session. Vandalism can thus occur in subtle, isolated edits or in large clusters of malicious activity spanning many map elements. Although the pipeline remains consistent in structure, it can be run independently on two data perspectives, each addressing different vandalism scenarios:

Contribution-Level Data. Within this perspective, each record corresponds to a single updated feature, such as a node or way. By analyzing contributions at this level, the system can detect localized anomalies—an abrupt coordinate shift, the removal of a crucial tag, or the introduction of nonsensical text. This fine-grained view is essential because vandalism often arises in small but impactful edits. A single outlier edit could relocate a landmark inappropriately, or rename a well-known city with offensive text, and it is the contribution-level analysis that is best poised to flag such granular behavior.

Changeset-Level Data. While the contribution-level approach captures fine detail, a supplementary view emerges when edits are grouped by the changeset in which they were committed. In this changeset-level perspective, the entire batch of edits made by a user during one editing session is evaluated collectively. This viewpoint is valuable because it can detect broad or systematic attacks—for example, a user who deletes hundreds of roads across disparate locations in one changeset. Even if individual edits look innocuous by themselves, the aggregate pattern might reveal extensive vandalism. By examining changeset-wide metrics like the ratio of deleted features, the number of continents affected, or the time-of-day distribution of edits, one can capture higher-level signals that may go undetected in an exclusively contribution oriented approach.

Separate Execution, Shared Architecture. Although both data perspectives exploit similar methodologies—feature extraction, classification, bootstrapping, etc.—they remain logically distinct in execution. This separation ensures:

- *Data Integrity*: Each pipeline inherits data fields and labels relevant to its granularity (per-edit vs. per-session).
- *Tailored Feature Engineering*: Contribution pipelines focus on geometry deltas, tag changes, or user edit frequencies for each individual record; changeset pipelines emphasize session-level aggregates like total deletions or bounding box expansions.
- *Focused Evaluation*: Metrics are computed independently, reflecting the different nature of vandalism detection at each scale.

In essence, the pipeline's flexibility accommodates the distinct vantage points OSM data offers—either isolating suspicious single edits or unmasking broad patterns in changesets. The subsequent sections further illustrate how the pipeline ingests, cleans, and labels these datasets (§3.2.1), how features are built out (§3.2.2), and how the XGBoost-based classifier is optimized and validated.

3.2 Data Preparation and Feature Engineering

This section describes how raw OpenStreetMap (OSM) data—spanning both **contribution-level** and **changeset-level** perspectives—is transformed into a comprehensive set of features suitable for vandalism detection. By systematically integrating labeled edits, refining their schemas, and engineering domain-relevant attributes, we lay the groundwork for robust machine learning models capable of discerning malicious behavior across different scales of editing activity.

3.2.1 Data Sources and Preprocessing

The data collection process targets two principal datasets: **Contributions** (individual edits) and **Changesets** (session-wide edits). Though they differ in granularity, each dataset captures unique signals pertinent to vandalism detection. The following subsections detail how these data are gathered, labeled, and prepared for downstream tasks.

Contribution Data

To build the *contribution-level* dataset, we first identified all vandalism-labeled changesets from an external reference corpus [1]. Each edit within these flagged changesets was marked as vandalism at the contribution level. For each contribution record, essential attributes include user-related information (e.g., `user_id`,

[1] *Vandalism labels for OSM are derived from the corpus referenced in [?],* https://github.com/NicolasTe/Ovid

editing frequency), OSM element metadata (e.g., bounding box, tags, geometry type), and time-based indicators (e.g., timestamps of creation or modification). These fields are parsed from monthly OSM files provided by the Heidelberg Institute for Geoinformation Technology (HeiGIT) and enriched with any relevant user metrics.

Next, to ensure class balance, random non-vandalism edits were sampled from the same monthly OSM files provided by the HeiGIT. These monthly snapshots record every contribution made during the respective month, capturing diverse editing behaviors from routine user updates. By merging the vandalism-labeled edits with an appropriate sample of non-vandalism edits, the resulting contribution dataset reflects both malicious and benign editing patterns.

Changeset Data

To assemble the *changeset-level* based dataset, we again drew on the same external reference corpus of vandalism-labeled changesets, identifying which changesets were deemed malicious or legitimate. Subsequently, we retrieved the pertinent metadata for each changeset ID by parsing the OSM changeset history files [2]. This history encompasses broad information about each changeset, including timestamps of creation and closure, user details, bounding box coordinates, comments, other changeset metadata.

3.2.2 Feature Construction

Having prepared and labeled the datasets (§3.2.1), the next step is to derive a comprehensive set of features that capture spatial, temporal, user-based, and content-related signals indicative of malicious edits. This section distinguishes between *contribution-level* and *changeset-level* feature engineering, reflecting the distinct nature of individual edits versus aggregated sessions. While many of the concep-

[2] *Changeset history files* `https://planet.openstreetmap.org/planet/full-history/`

tual categories (e.g., user features, geometric features) overlap across data types, each dataset emphasizes different aspects depending on granularity and available metadata.

Overview of the Feature Extraction Workflow. Given that OSM data often comprises millions of rows, we employ a parallel, chunk-based mechanism to compute features. Records are split into manageable batches, processed concurrently on multiple CPU cores, and recombined into a unified feature matrix. This approach both decreases processing time and ensures the pipeline can handle large-scale datasets without exhausting memory resources. Depending on pipeline configurations, user-related attributes and OSM element statistics may be selectively included or excluded to test various model assumptions.

3.2.2.1 Feature Engineering for Contribution Data

Contribution-level features revolve around individual edits—such as modifying a building tag or shifting a highway node. These signals target fine-grained anomalies, where even small changes can disrupt map accuracy or introduce offensive content.

1. **User Features.**

   - **Total number of edits** (`n_edits`): Indicates how active a user is overall. Though many edits suggest experience, they may also indicate prolific vandalism in rare cases.
   - **Cumulative statistics** (`user_n_changesets_cum`, `user_n_edits_cum`, `user_n_edit_days_cum`): Track a contributor's broader history, including how many changesets and edit days they have logged. These metrics differentiate stable, consistent mappers from users with bursts of erratic activity.
   - **Timestamps and time since previous edits** (`user_previous_edit_timestamp`, `user_time_since_previous_edit`): Reveal large temporal gaps or sudden

surges. For instance, if a user reappears after months of inactivity and makes thousands of edits in a short window, the pipeline may boost the vandalism likelihood.

These user-based indicators inform the model whether an edit aligns with typical contributor behavior or exhibits sudden irregularities, thus helping flag potential vandalism.

2. OSM Element Features.

- **Cumulative edits on an element** (`element_n_users_cum`, `element_n_versions`): Record how many distinct users or total versions a feature has accumulated. High version counts can mean repeated controversies or "edit wars."
- **Time and interval since last edit** (`element_previous_edit_timestamp`, `element_time_since_previous_edit`): Frequent, repeated edits on the same entity in short succession can signal suspicious conflict or revert battles.

By highlighting contentious objects or frequently altered elements, these attributes complement user features in revealing unusual edit behaviors.

3. Geometric Features.

- **Geometry size and deltas** (`area`, `length`, `area_delta`, `length_delta`): Quantify the magnitude of an edit. For instance, an edit that suddenly enlarges a polygon's area by 500% is suspicious.
- **Geometry type** (`geometry_type`): Distinguishes between point, line, or polygon. Small coordinate nudges on a node may differ in impact from major polygon reshaping.
- **Bounding box** (`xmin`, `xmax`, `ymin`, `ymax`, `bounding_box_size`): Summarize the edited feature's footprint. Comparing bounding box size before and after an edit can flag abrupt geometry shifts.

Large or unnatural geometry changes often indicate malicious manipulations, making geometric features a cornerstone of vandalism detection.

4. Temporal Features.

- **Time of day (`time_of_day`):** Broad intervals (morning, afternoon, evening, night) derived from timestamps. While not universally suspicious, off-peak editing hours can correlate with reduced community scrutiny.
- **Day of week (`day_of_week`), weekend flag (`is_weekend`):** Differentiate between weekday and weekend editing patterns. Some studies note increased vandalism when fewer moderators are active.
- **Creation date (`date_created`):** Used for temporal splits or trend analysis, enabling chronological evaluations of the model's performance over time.

Capturing when a contribution occurs helps the model identify outlier behaviors against typical daily or weekly rhythms.

5. Content Features.

- **Tag modifications (`tags_added`, `tags_removed`, `tags_modified`, `tags_changed`, `total_tags`):** Reflect key-value changes in the OSM database. Large-scale tag deletions or nonsensical additions can signal vandalism.
- **Key tags (e.g., `building`, `highway`, `landuse`):** Boolean flags that determine whether a crucial attribute was introduced, removed, or updated. Systematic removal of `building` tags might indicate sabotage.

Because OSM tags are central to map semantics, analyzing them reveals malicious attempts to alter fundamental attributes or introduce inflammatory text.

6. Spatial Features.

- **Centroid and bounding box range** (`centroid_x`, `centroid_y`, `bbox_x_range`, `bbox_y_range`): Simplify geometric footprints into coordinate-based summaries.
- **Grid cell ID** (`grid_cell_id`): Assigns each edit to a spatial index for clustering. Concentrated vandalism can emerge in certain hotspots.
- **Countries and continents** (`countries`, `continents`): Identifies administrative boundaries relevant to the edit's location, enabling region-specific analysis.

7. Derived Features.

- **Tag density** (`tag_density`): Ratio of total tags to area (often log-scaled to mitigate skew). Outliers might denote over- or under-documented objects.
- **Relative area change** (`relative_area_change`): Compares a new area to the previous one, surfacing extreme expansions or contractions.

These aggregated indicators compress raw signals into more discriminative forms, enhancing the model's predictive power.

8. Changeset Features. Though primarily relevant to session-level analysis, certain changeset-level attributes (e.g., `changeset_comment_length`) can appear in contributions data if the pipeline is configured to attach session metadata to individual edits. This approach helps unify certain changeset-wide signals with the single-edit viewpoint.

9. Clustering Feature. Clustering is introduced to capture spatial editing patterns, leveraging the `centroid_x` and `centroid_y` coordinates to assign a `cluster_label` to each edit. This transformation simplifies geographic data into discrete clusters, enabling the model to identify localized editing behaviors and detect region-specific vandalism.

KMeans clustering [?, ?, ?] is employed for this purpose, dynamically adjusting the number of clusters (`n_clusters`) based on the dataset size. The algorithm is trained on the spatial centroids of the training data, and the resulting cluster labels are propagated to the validation and test datasets, ensuring spatial consistency across all splits. By grouping geographically adjacent edits, the clustering feature encapsulates spatial context, making it easier for the model to discern regional patterns and anomalies.

3.2.2.2 Feature Engineering for Changeset Data

When the pipeline operates on *changeset-level* data, each record corresponds to an entire changeset. Here, the feature extraction process aggregates information that characterizes the session's scope, content, and timing, rather than focusing on individual edits.

1. Spatial Bounding Box.

- **min_lon, max_lon, min_lat, max_lat**: Summarize the session's geographical footprint. Missing values default to 0, ensuring consistent computations and preventing dropped rows.
- **Centroid computation** (`centroid_x`, `centroid_y`): Calculated as the average of bounding box corners, approximating where the user predominantly edited in that changeset.

A large bounding box could indicate extensive map edits within one session—potentially suspicious if the user historically focuses on a single locality.

2. Clustering Features.

- **cluster_label**: As discussed in the earlier section, clustering is performed using KMeans on spatial centroids (`centroid_x`, `centroid_y`), and the resulting cluster labels are assigned to each changeset.

3. Comment-Based Attributes.

- **changeset_comment_length**: The length of the user's description. Detailed comments may correlate with more legitimate sessions, whereas minimal or nonsensical remarks can raise suspicion.
- **source_used**, **source_reliability**: Capture whether the user relied on known high-quality imagery (e.g., Bing, Maxar). Sessions lacking recognized sources may be prone to errors or intentional distortions.

4. Temporal Metrics.

- **date_created**: Records when the changeset began, facilitating chronological splits or drift analyses over time.

This combination of features allows the pipeline to analyze changeset-level data comprehensively, incorporating spatial, textual, and temporal dimensions. The addition of clustering features further enhances the model's ability to identify patterns that may signify anomalous or malicious activity within specific geographic regions.

3.2.3 Summary and Key Takeaways

This section has detailed the comprehensive process of preparing OSM data and constructing meaningful features for vandalism detection. By integrating user behavior patterns, spatial and geometric transformations, content modifications, and changeset-level characteristics, the feature engineering pipeline captures a wide spectrum of vandalism signals. This multi-faceted representation ensures the model can effectively detect both localized, subtle manipulations and broad, systematic attacks on OSM data.

The carefully designed feature space not only enhances model interpretability but also plays a crucial role in maintaining scalability—enabling the efficient processing of millions of edits each month. The next sections build upon this foundation, detailing the data splitting strategies and classification models that leverage these features to distinguish between benign and malicious contributions with high precision.

3.3 Data Splitting Strategies

To ensure robust evaluation of the vandalism detection model, data is partitioned into training, validation, and test subsets using three primary splitting strategies: *random*, *geographic*, and *temporal*. Each strategy addresses distinct real-world scenarios, ranging from general model performance to geographic and temporal adaptability.

3.3.1 Random Splits.

The random split strategy is the foundation for most evaluations, designed to mimic the diverse yet unstructured nature of OpenStreetMap (OSM) edits. Separate configurations are applied for *contribution-level* and *changeset-level* datasets to reflect their unique characteristics:

Contribution-Level Data Splits. The contribution dataset is divided as follows:

- **Training set:** 400,000 samples, accounting for 72.73% of the total data.
- **Validation set:** 50,000 samples, or 9.09% of the total data.
- **Test set:** 150,000 samples, comprising 27.27% of the total data.

To emulate real-world conditions, the validation and test sets are created with an adjusted vandalism ratio that facilitates effective model evaluation. This proportion, while higher than the actual occurrence of vandalism in OSM, ensures sufficient representation of the minority class for robust performance assessment.

- Validation set: 10,000 vandalism examples and 40,000 non-vandalism examples.
- Test set: 30,000 vandalism examples and 120,000 non-vandalism examples.

The training set is generated through random sampling, resulting in an imbalanced dataset where the vandalism ratio deviates from real-world distributions. To address this, `scale_pos_weight` is applied during model training. This parameter adjusts the weight of the vandalism class in the loss function, compensating for the imbalance and preventing the model from disproportionately favoring the majority class (non-vandalism).

Extensive experiments were conducted with various split ratios, including balanced vandalism and non-vandalism distributions. However, preserving the real-world imbalance in the validation and test sets consistently produced superior results, as detailed in Chapter 4. Metrics from the random split, including precision, recall, and AUC-PR, are summarized in Table 4.1, and the corresponding precision-recall and ROC curves are presented in Figure 4.2.

Changeset-Level Data Splits. For changeset-level data, a distinct partitioning strategy is adopted to facilitate the evaluation of multiple classifiers: the normal changeset model, the hyper-classifier, and the meta-classifier. The data is split as follows:

- **Training set:** 60%.
- **Validation set:** 5%.
- **Test set:** 15%.

- **Meta-test set:** 20%.

These splits ensure that training, validation, and testing processes are conducted independently and provide robust evaluations of all classifiers. The training set is used for both the normal changeset model and the hyper-classifier, while the meta-test set is exclusively utilized to assess the performance of the meta-classifier.

Further details about the hyper-classifier and meta-classifier, including their roles and integration into the pipeline, are discussed in Section 3.5.

3.3.2 Geographic Splits.

Geographic splits evaluate the model's ability to generalize across different regions, reflecting the global scope of OSM contributions. Data is partitioned based on the `GEOGRAPHIC_SPLIT_KEY`, which may correspond to continents or countries. For example:

- Training data: Edits from Europe and South America.
- Validation data: Edits from Africa.
- Test data: Edits from Asia and North America.

This strategy ensures that a model trained in one region can accurately detect vandalism in unfamiliar geographic contexts. Results, including performance metrics by region, are presented in Table 4.1.

3.3.3 Temporal Splits.

Temporal splits simulate real-world scenarios where newer vandalism patterns emerge over time. Data is partitioned chronologically based on the `date_created` field:

- Training data: Edits from 2018–2019.

- Validation data: Edits from earlier years, such as 2015.
- Test data: Edits from an intervening year, such as 2017.

This approach evaluates the model's ability to generalize to edits occurring outside its training window, a crucial capability for deployment in a constantly evolving platform like OSM. Temporal robustness metrics, such as AUC-PR and F1-score, are discussed in Chapter 4.

Conclusion. The data splitting strategies outlined in this section establish a rigorous and well-rounded evaluation framework for assessing the vandalism detection models. By incorporating **random, geographic, and temporal splits**, the thesis ensures a thorough examination of the model's ability to generalize across diverse real-world scenarios—whether encountering edits from different regions, handling temporal shifts in vandalism patterns, or maintaining performance under varying data distributions.

These strategies not only enhance the reliability of the model's evaluation but also simulate practical deployment conditions, ensuring adaptability to OSM's ever-evolving editing landscape. The effectiveness of these splits is further validated in the upcoming model performance analysis (chapter 4), where we assess how well the classifier maintains accuracy and robustness across different testing conditions.

3.4 Core Classification Model

The detection of vandalism in OpenStreetMap (OSM) relies on the capacity of a learning algorithm to aggregate and interpret diverse feature signals—ranging from individual user behavior and geometric deltas to session-wide changeset patterns. In this work, the primary classifier employed is an XGBoost model, chosen for its ability to handle large-scale, structured data efficiently. Through an ensemble of weak decision-tree learners, XGBoost balances interpretability, speed, and flexibility in dealing with the heterogeneous nature of OSM attributes.

3.4.1 XGBoost: Rationale and Architecture

XGBoost (*eXtreme Gradient Boosting*) underpins the core classification mechanism in our vandalism detection pipeline, chosen for its high efficiency, scalability, and strong performance on structured data [?]. This section begins with an overview of gradient boosting and decision tree ensembles, then highlights the design elements that make XGBoost particularly well-suited for large-scale OSM data. We conclude by illustrating how XGBoost is integrated into our final model training and inference workflow.

3.4.1.1 Gradient Boosting and Decision Tree Ensembles

Gradient boosting is an ensemble technique that iteratively refines its predictions by adding "weak learners," typically decision trees, in a sequential manner [?]. Each new tree focuses on the residual errors from the previous ensemble model, collectively minimizing a specified loss function. Conceptually, the process proceeds as follows:

1. *Initialize*: Start with a simple model (often a constant prediction).
2. *Compute Residuals*: Calculate the difference between current predictions and the true labels.
3. *Train Next Tree*: Fit a small decision tree to these residuals, thereby learning how to correct the existing model's errors.
4. *Update Ensemble*: Add the new tree to the model, typically scaled by a learning rate to prevent over-correction.
5. *Repeat*: Continue until convergence or until a maximum number of iterations is reached.

Because each tree attempts to fix the shortcomings of its predecessor, this ensemble gradually hones in on accurate predictions. Employing decision trees—well-suited to non-linear relationships and mixed feature types—further enhances this flexibility, making gradient boosting (and, by extension, XGBoost) highly effective for OSM's complex tabular data.

3.4.1.2 Enhancements in XGBoost

While classical gradient boosting offers a robust foundation, **XGBoost** refines it via multiple architectural innovations [**?**]:

- **Regularization**: Introduces $L1$ and $L2$ penalties to curb overfitting by limiting tree complexity, thus promoting better generalization.
- **Efficient Split Finding**: Uses histogram-based methods to swiftly detect optimal split thresholds, reducing computational overhead and memory usage—vital for large-scale OSM data.
- **Tree Pruning**: Adopts a depth-first tree-building approach with post-hoc pruning. Nodes yielding minimal gain can be pruned to focus on meaningful splits.
- **Handling Missing Data and Sparsity**: Automatically routes missing values in a way that maximizes training gain, while sparsity-aware algorithms accommodate real-world data with zeros or nulls.
- **Parallelization and Scalability**: Parallelizes gradient computations and tree construction, enabling substantial speed-ups during training. This is particularly advantageous when processing monthly OSM files containing millions of edits.
- **Out-of-Core Computation**: Facilitates disk-based training for datasets exceeding main memory, an important feature when dealing with large geospatial data.

These mechanisms differentiate XGBoost from earlier gradient boosting implementations, making it particularly apt for tasks requiring high accuracy and interpretability on large tabular datasets.

3.4.1.3 System Architecture and Design

XGBoost's internal architecture centers on efficiency and parallel operations:

- **Block Structure Storage**: Optimizes memory usage and cache performance by storing features in column blocks, enabling faster access during gradient computations.
- **Column Block Design for Parallel Learning**: Allows simultaneous processing of multiple features, significantly accelerating split finding.

Figure 3.1 illustrates these architectural concepts. By leveraging columnar storage and parallel computations, XGBoost can effectively scale to the data demands typical of OSM vandalism detection.

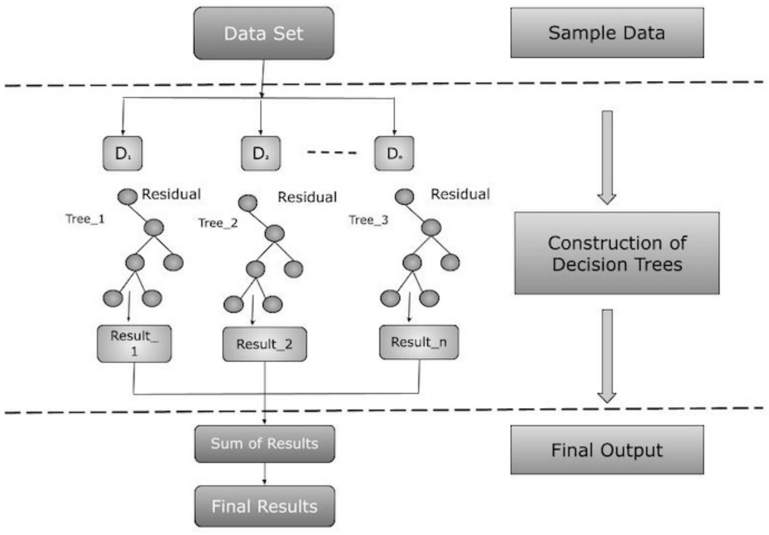


Figure 3.1: High-level schematic of the XGBoost architecture, highlighting parallelization and columnar data blocks. [?]

In our pipeline, XGBoost not only handles the individual feature vectors derived from user, geometric, temporal, and textual signals, but also robustly manages partial sparsity and imbalances in the class labels—key aspects in vandalism detection.

By blending gradient boosting fundamentals with XGBoost's engineering optimizations, the resulting classifier consistently yields high precision and recall in vandalism detection, whether at the *contribution* or *changeset* scale.

3.4.2 Hyperparameter Tuning

Although XGBoost provides strong baseline performance, careful tuning of its hyperparameters can substantially improve recall and precision in vandalism detection. To efficiently explore the hyperparameter space, we employ **RandomizedSearchCV** [?], a technique that selects random combinations of parameters and evaluates their performance through cross-validation. To this end, the model is tested against a search space of key parameters, including the *learning rate*, which regulates the contribution of each new tree, and the *max depth*, which limits how many splits a tree can perform. Additional parameters like *subsample* (row sampling) and *colsample_bytree* (feature sampling) control stochasticity, whereas *gamma*, *alpha*, and *lambda* impose regularization and prevent overfitting by restricting or penalizing overly complex splits.

Table 3.1 summarizes the search space explored during hyperparameter tuning. Each parameter is optimized through a structured search process to achieve a balance between **model complexity and generalization ability.**

Table 3.1: Hyperparameter Search Space of Ovid

Parameter	**Description**	**Search Space**
`learning_rate`	Controls step size in boosting	{0.01, 0.05, 0.1, 0.2, 0.3}
`max_depth`	Maximum depth of trees	{3, 5, 7, 9}
`subsample`	Training data per boosting round	{0.6, 0.7, 0.8, 0.9, 1.0}
`colsample_bytree`	Fraction of features per tree	{0.6, 0.7, 0.8, 0.9, 1.0}
`lambda`	L2 regularization strength	{0, 1, 3, 5, 10}
`alpha`	L1 regularization strength	{0, 1, 2, 3, 5}
`min_child_weight`	Min sum of instance weights in a leaf	{1, 3, 5, 7, 10}
`gamma`	Min loss reduction for splits	{0, 0.1, 0.3, 0.5, 1, 2, 3}
`n_estimators`	Number of boosting rounds	{50, 60, 80, 100}
`scale_pos_weight`	Class imbalance weight	{0.5, 1, 1.5, 2.5, 3, 4.5, 10}

The tuning process often relies on cross-validation [?]. For instance, a five-fold setup divides the training set into five parts, cycling through each portion as a validation fold while the remaining four folds serve as the training data. Performance metrics—commonly the area under the ROC curve (AUC-ROC) or the area under the precision-recall curve (AUC-PR)—are tracked across all folds. Because vandalism in OSM tends to be rarer than benign edits, AUC-PR can be more informative, capturing how well the model preserves recall without sacrificing precision. Once the model achieves strong cross-validation scores, it is retrained on the entire training set using the best hyperparameters found.

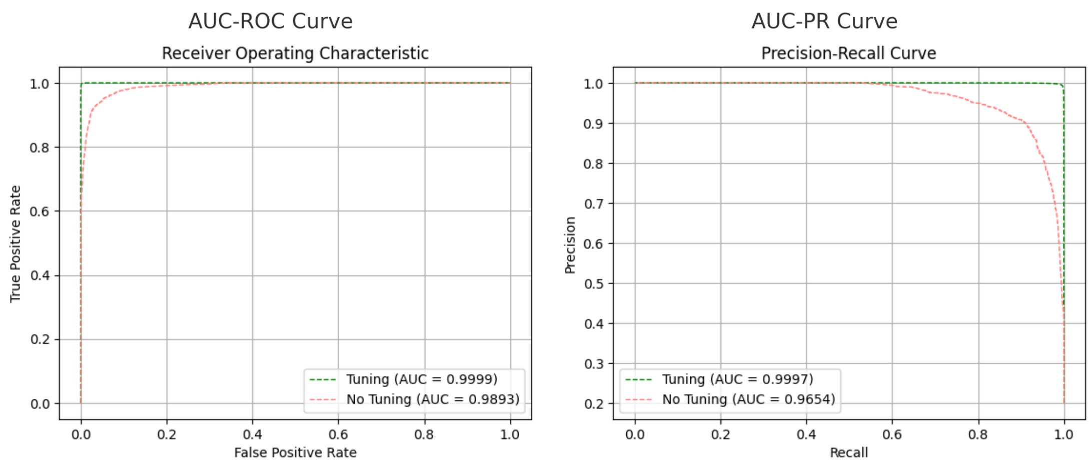


Figure 3.2: Comparison of AUC-ROC and AUC-PR curves for the model with and without hyperparameter tuning. The green dashed line represents the model with hyperparameter tuning, achieving superior performance with an AUC-ROC of 0.9999 and an AUC-PR of 0.9997. In contrast, the red dashed line shows the model without tuning, which achieves lower AUC-ROC (0.9893) and AUC-PR (0.9654) scores. The results highlight the effectiveness of tuning in improving the model's ability to distinguish between classes and handle imbalanced datasets.

Figure 3.2 illustrates the significant impact of hyperparameter tuning on model performance. The green curves, representing the tuned model, demonstrate higher area under both the ROC and PR curves, indicating better discrimination between vandalism and non-vandalism edits. Hyperparameter tuning allows for optimization of key parameters such as *max depth*, *learning rate*, and *tree estimators*, enhancing the model's robustness and precision. This improvement is particularly crucial for handling imbalanced datasets where subtle patterns in the minority class (vandalism) need to be effectively captured without overfitting or sacrificing generalization.

In the final configuration, XGBoost typically balances a moderate tree depth with a suitable degree of regularization and a learning rate that does not overshoot the loss gradient. This tuned ensemble then forms the backbone of vandalism classification, effectively harnessing the feature-rich representation built in earlier stages of the pipeline.

3.4.3 Training Progress and Insights

The training process of the XGBoost model was monitored through multiple metrics to evaluate convergence and generalization. This section provides insights into the model's performance over iterations, as depicted in Figures 3.3, 3.4, and 3.5.

Accuracy Progress

Figure 3.3 illustrates the progression of training and validation accuracy across 100 iterations. The training accuracy improves rapidly in the initial iterations, stabilizing close to 99.9%. The validation accuracy closely follows, stabilizing near 99.6%. The minimal gap between training and validation accuracy indicates excellent generalization and minimal overfitting.

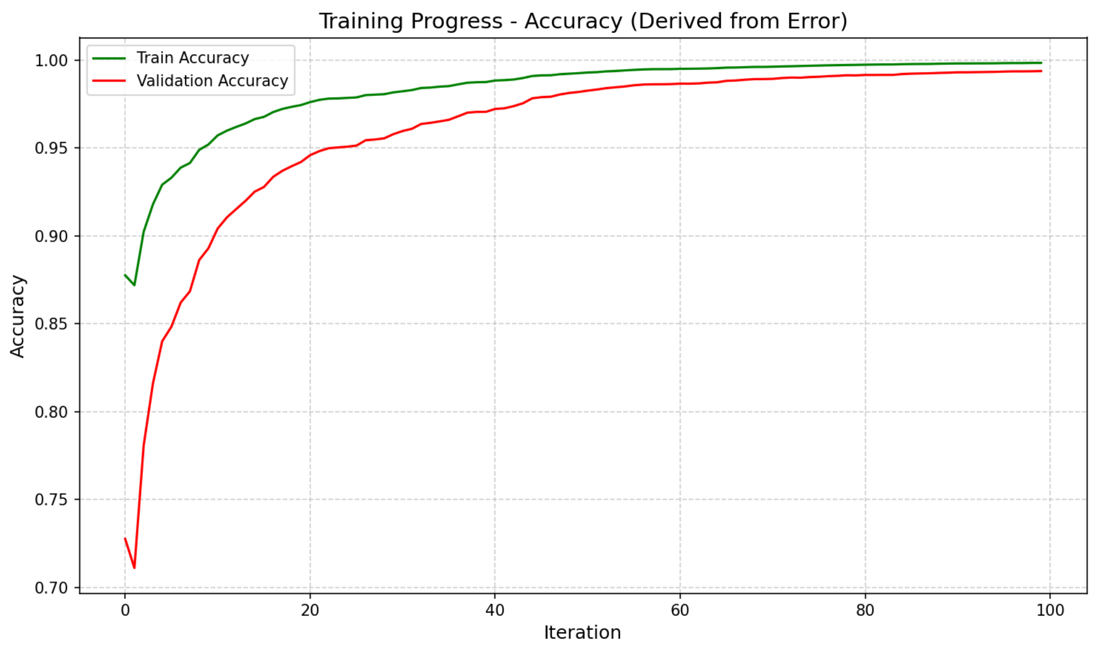


Figure 3.3: Training Progress: Accuracy (Train vs Validation).

Error Reduction

Figure 3.4 shows the error rate progression for both training and validation sets. The training error quickly reduces to near zero, while the validation error also decreases significantly, stabilizing at very low levels. This trend confirms the model's robustness and capacity for generalization.

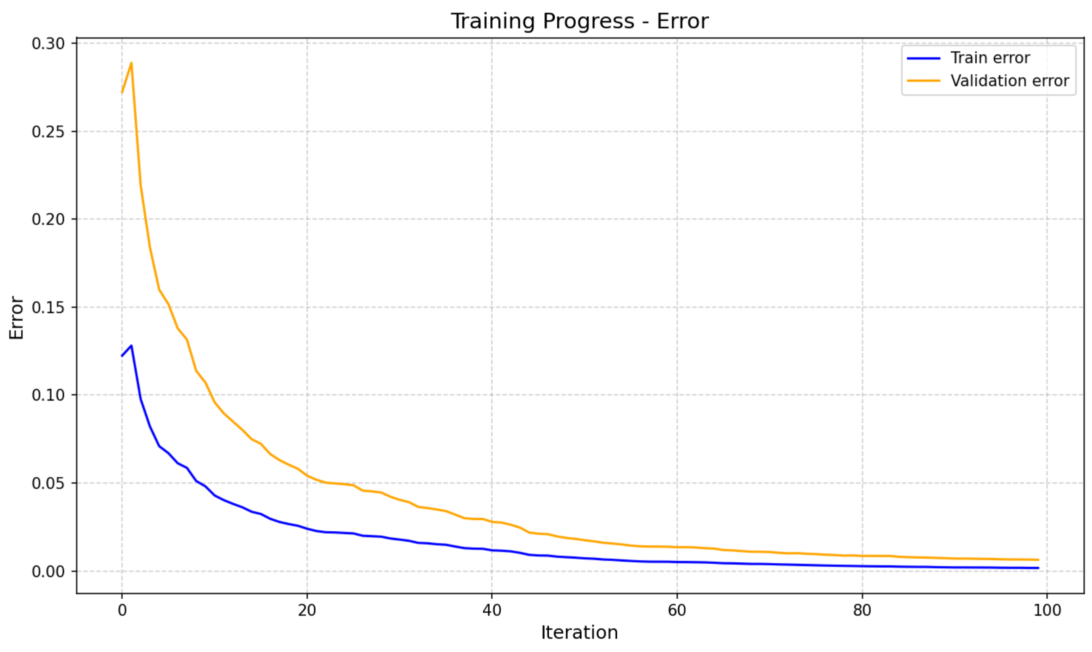


Figure 3.4: Training Progress: Error Rate (Train vs Validation).

Log-Loss Convergence

Log-loss, a measure of prediction confidence, steadily decreases for both training and validation datasets, as depicted in Figure 3.5. Validation log-loss stabilizes around 0.023, signifying high confidence in predictions and consistent model performance.

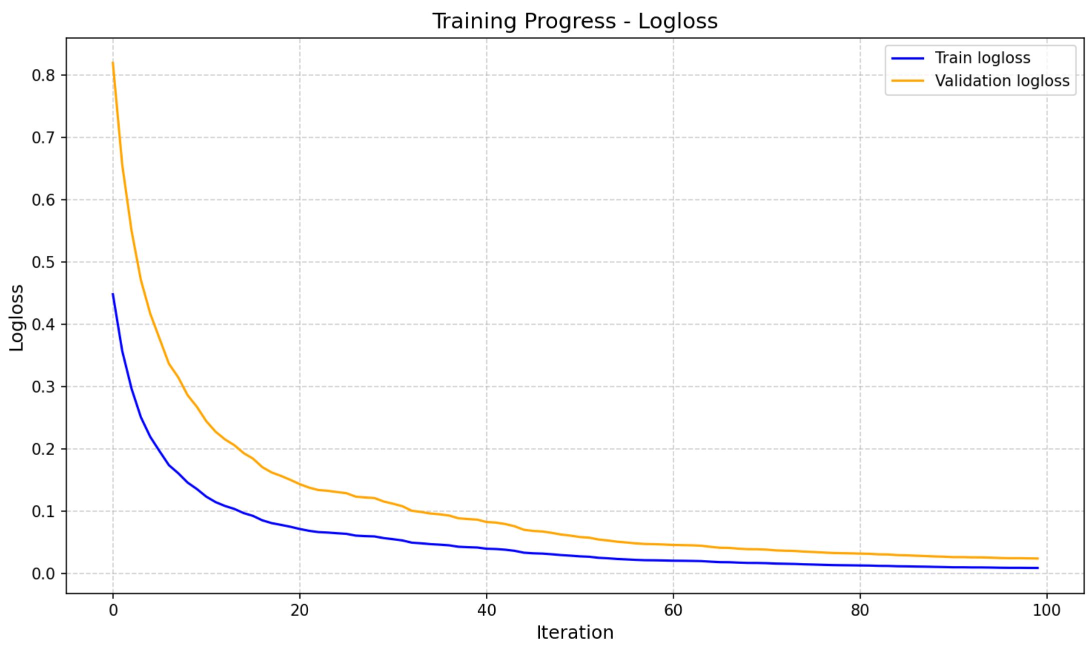


Figure 3.5: Training Progress: Log-Loss (Train vs Validation).

Across these figures, the minimal gap between training and validation metrics reflects strong generalization—critical for real-world OSM data that can vary significantly by region or time. These insights confirm that the final XGBoost model converges effectively, balancing low false positives with a high capture rate for vandalism edits. When combined with the chunk-based data ingestion pipeline and carefully engineered features (§3.2.2), this trained classifier forms the linchpin of our vandalism detection system, ready to perform inference on live or monthly OSM updates.

3.5 Enhancing Changeset-Level Detection

In OpenStreetMap (OSM), a *changeset* encapsulates a group of edits uploaded in one session, potentially spanning extensive geographic areas or multiple features. The same XGBoost pipeline discussed in previous sections can be trained on changeset-level data to detect vandalism for each changeset. However, additional techniques—the *hyper-classifier* and *meta-classifier*—provide further refinement by leveraging per-contribution predictions and fusing them with changeset-level insights.

3.5.1 Baseline Changeset Pipeline

When operating on changeset data, the pipeline treats each *changeset* as a single data instance, extracting session-level attributes as features. Although these features—discussed in Section 3.2.2—can capture large-scale vandalism (e.g., mass deletions), they may overlook subtle patterns of dispersed suspicious edits. An XGBoost classifier trained on these changeset-level attributes forms the baseline changeset approach.

High-Level Approach.

1. **Data Ingestion**: Load changeset-level records, each labeled as vandalism or non-vandalism.
2. **Feature Extraction**: Incorporate bounding-box coordinates, timestamps, user statistics, and edit counts as input to the pipeline.
3. **Training**: Fit an XGBoost model to classify each changeset, producing a vandalism probability per changeset.

This baseline method effectively detects obvious session-wide disruptions (e.g., hundreds of deleted roads) but may miss more nuanced, distributed forms of vandalism if they lack a clear bounding-box or user-based signature.

3.5.2 Hyper-Classifier: Aggregating Per-Contribution Predictions

The term *hyper-classifier* reflects its role as a higher-order model that builds upon contribution-level predictions rather than raw features alone. Instead of relying solely on changeset metadata, it aggregates per-edit suspicion scores to form a more holistic assessment of vandalism within a changeset.

To overcome the limitations of purely changeset-level attributes, the pipeline incorporates the *hyper-classifier*, which aggregates the output of the contribution-level XGBoost model. Rather than solely relying on bounding-box or changeset metadata, the hyper-classifier builds new features summarizing how suspicious each edit is within a changeset.

Aggregated Signals.

1. **Per-Contribution Probabilities**: For each edit, the contribution-level model outputs a vandalism probability (`pred_prob`).

2. **Group by Changeset**: All edits sharing a `changeset_id` are collected, and their probabilities undergo statistical summarization (e.g., mean, median, quantiles, standard deviation, proportion of edits above a threshold).

3. **Feature Matrix**: A new dataset emerges, where each row is a changeset, augmented with these aggregated per-edit probabilities.

4. **Hyper-Classifier Training**: An additional XGBoost model is trained on this feature matrix to classify entire changesets, potentially capturing session-wide malicious intent that the baseline changeset pipeline might miss.

By consolidating the edit-level suspicion scores, the hyper-classifier aims to exceed the performance of the baseline changeset-level approach. The underlying assumption is that if many edits within a single changeset are moderately suspicious, the collective signal is significantly stronger than each edit's individual suspicion.

3.5.3 Meta-Classifier: Combining Baseline and Hyper-Classifier Outputs

While the baseline changeset pipeline capitalizes on changeset metadata, and the hyper-classifier exploits aggregated per-edit probabilities, each perspective covers different facets of vandalism. The *meta-classifier* merges these two probability streams into a single final prediction. The term *meta-classifier* [?] refers to its role in integrating predictions from multiple models at a higher level. By integrating these distinct probability streams, the meta-classifier leverages complementary signals from both sources, aiming to enhance detection performance beyond what either model achieves independently.

1. **Obtain Baseline Changeset Probability**: Run the XGBoost model on changeset data, generating a vandalism probability.

2. **Obtain Hyper-Classifier Probability**: Compute the probability that the same changeset is vandalism based on aggregated per-edit signals.

3. **Construct a New Dataset**: Each changeset now has two predicted probabilities ($\hat{p}_{\text{baseline}}$ and $\hat{p}_{\text{hyper}}$), along with a true label indicating whether it is vandalism.

4. **Train Meta-Classifier**: A separate XGBoost or logistic regression model maps these two probabilities (and optionally a few additional features) to a final vandalism classification.

During inference, the pipeline once again obtains predictions from both the baseline model and the hyper-classifier, feeding them into the meta-classifier to arrive at a single decision.

To facilitate the hyper-classifier and meta-classifier, changeset IDs must be consistent between datasets. If a changeset or its constituent edits are missing in the contribution-level data, they cannot be aggregated. Hence, both the hyper-classifier and meta-classifier pipelines require a matched set of edits and changesets.

Conclusion. The multi-stage approach to changeset-level detection (baseline pipeline, hyper-classifier, and meta-classifier) highlights the system's flexibility. The baseline pipeline uses a straightforward model, the hyper-classifier exploits fine-grained edit suspicions, and the meta-classifier fuses both perspectives. Although additional complexity is introduced, each stage addresses a different source of vandalism signals.

All three methods are independently trained using XGBoost. The pipeline organizes each stage's training and evaluation data in alignment with the changeset splits outlined in Section 3.3, ensuring no data leakage between these overlapping models. Chapter 4 provides empirical results, showing how these layered models can substantially improve detection accuracy across diverse OSM vandalism scenarios.

3.6 Evaluation Framework and Methodology

A rigorous evaluation strategy is essential to ensure that the vandalism detection models generalize effectively across diverse OpenStreetMap (OSM) editing behaviors. This section outlines the methodological approach for assessing model performance, including the choice of evaluation metrics, statistical validation techniques, and geographical assessments.

3.6.1 Evaluation Metrics

Measuring vandalism detection performance involves navigating a trade-off between false alarms and undetected malicious edits. The thesis employs a suite of metrics that highlight different facets of classification quality.

Precision, Recall, and F1-score. These metrics encapsulate distinct but complementary concerns:

- **Precision**: Among the edits flagged as vandalism, how many are truly malicious?
- **Recall**: Of all vandalism edits in the dataset, how many did the model catch?
- **F1-score**: The harmonic mean of precision and recall, penalizing weaknesses in either dimension.

Because vandalism typically makes up a minority of OSM edits, achieving strong recall (to catch most malicious contributions) without overwhelming moderators with false positives is paramount.

AUC-PR vs. AUC-ROC. When dealing with class imbalance, the area under the precision-recall curve (AUC-PR) can provide a clearer snapshot of performance than AUC-ROC. AUC-PR emphasizes how effectively the model maintains high recall while keeping false positives in check—a crucial balance in real-time vandalism detection. Nevertheless, the area under the receiver operating characteristic curve (AUC-ROC) remains valuable for a broader sense of discriminative power. Later sections present confusion matrices and curves (Figure 4.1, Figure 4.2a, Figure 4.2b) that illustrate how these metrics play out in practice, demonstrating the classifier's capacity to distinguish vandalism from normal edits.

3.6.2 Bootstrapping for Confidence Intervals

While a single test set offers a snapshot of model performance, **bootstrapping** [?] yields deeper insight into metric stability and variance. The procedure involves:

1. *Resampling* the test set with replacement multiple times (1,000 iterations).
2. *Evaluating* metrics (precision, recall, F1-score, AUC-PR, etc.) for each resampled dataset.

3. *Aggregating* these results into distributions, computing mean, median, and confidence intervals.

This Monte Carlo–style approach clarifies how sensitive the classifier is to the particular composition of the test set. Narrow intervals suggest a robust, reliable model, while wider bounds may indicate volatility in classification performance. Refer to Table 4.2 for an example of how these confidence intervals provide a nuanced perspective on vandalism detection reliability.

3.6.3 Geographical Evaluations

OpenStreetMap (OSM) is a global resource, edited by contributors across diverse continents and countries. Because mapping practices, data availability, and local conventions can differ substantially from one region to another, a vandalism-detection model that excels in one geographical area may falter in another. To ensure broad applicability, we incorporate *geographical evaluations*, systematically breaking down performance by region or continent.

In practice, this entails tagging each data instance—whether a single edit or an entire changeset—with the relevant continent or country. During post-processing, the model's predictions are grouped by these spatial labels, and metrics such as precision, recall, and F1-score are recalculated within each group. If certain areas experience high error rates (e.g., a lower recall in Africa or an inflated false-positive rate in Asia), it can signal that the model's learned patterns do not generalize effectively to those contexts. Explanations might include insufficient training samples from that region, differences in local tagging conventions, or a higher prevalence of certain map features that the classifier has not learned to handle.

Such an analysis also paves the way for region-specific remediation. If the classifier repeatedly underperforms in an underrepresented continent, it may be beneficial to supplement the training set with additional data from that area, refine certain features (e.g., bounding-box logic for roads in mountainous terrain), or even implement custom logic that triggers extra checks in regions with historically higher vandalism rates. Further, by mapping the model's false positives and false negatives spatially, domain experts or OSM community volunteers can better prioritize manual validation and adopt targeted improvements. Although the final numeric results of these geographical evaluations appear later in the thesis, the methodological framework itself—spatial tagging and region-specific performance scoring—embeds seamlessly within the broader pipeline and provides crucial diagnostic feedback on regional biases or gaps in generalization.

Summary of the Evaluation Framework and Methodology. This section outlines the structured approach used to assess the model's performance, ensuring a thorough evaluation of its generalization capabilities. By integrating multiple evaluation metrics, bootstrap-based statistical validation, and geographical assessments, the framework provides a robust methodology for measuring classifier effectiveness. These methods allow for a nuanced analysis of precision-recall trade-offs, model stability, and regional performance variations. The results of these evaluations are discussed in detail in Chapter 4, where the impact of these assessment strategies on the model's real-world applicability is examined.

3.7 Summary

In sum, the methodological framework presented across this chapter unites a range of strategies to detect vandalism at multiple levels of granularity and to validate performance under realistic conditions. The pipeline begins by loading and cleaning raw OSM data (§3.2.1 and §3.2.2), ensuring that each edit or changeset has consistent, well-defined features capturing geometry, user history, tags, and other relevant attributes. From there, the main classification approach utilizes a finely tuned XGBoost model (§3.4) to learn which patterns correlate with malicious edits—a process guided by appropriate split strategies, robust evaluation metrics, and bootstrapping for confidence intervals.

Because vandalism can appear at different scales, the pipeline extends beyond per-contribution classification to changeset-level modeling, leveraging both a baseline changeset model (built on changeset metadata) and a hyper-classifier (aggregating probabilities derived from individual edits). Finally, a meta-classifier can fuse the outputs of these changeset-level classifiers, capitalizing on the complementary nature of changeset metadata and aggregated per-contribution signals (§3.5).

The evaluation framework (§3.6) ensures a rigorous assessment of the vandalism detection pipeline through various evaluation metrics, bootstrapping for confidence intervals, and geographical evaluations. Together, these strategies provide a comprehensive validation framework, ensuring both predictive power and adaptability across diverse OSM data distributions.

Taken together, these components illustrate a method that is both *comprehensive*—covering local anomalies, aggregated session patterns, and large-scale evaluation—and *flexible* in addressing multiple forms of vandalism. The next chapters will present empirical results demonstrating the effectiveness of this approach, accompanied by deeper discussions of how these methodological choices impact performance in real-world scenarios.

4 Results

This chapter presents a comprehensive evaluation of the machine learning models developed for vandalism detection in OpenStreetMap (OSM). The analysis examines the model's generalization across diverse data splits, prediction robustness, and the impact of User and OSM element features (U, OE features) on detection performance. Additionally, the results highlight key performance trends, including differences in model effectiveness based on random, geographical, and temporal splits, as well as insights gained from confidence interval estimations and real-world application scenarios.

4.1 Model Performance on Contribution Data

The contribution-level model operates at the granularity of individual edits, identifying malicious modifications before they aggregate into broader upload sessions. To comprehensively assess this model, we tested three distinct data split strategies:

- **Random Split**: Random partitioning into training, validation, and test sets, using 42 as the random seed. The sizes were configured as `TEST_SIZE` = 150,000, `VAL_SIZE` = 50,000, and a remaining large training set.
- **Geographic Split (by Continent)**: Training on specified regions (Oceania, Europe, South America), validating on others (Africa, Antarctica, Other), and testing on North America/Asia. The split key was `'continent'`, reflecting real-world geographic variability.

- **Temporal Split**: Training on 2018–2019 data, validating on 2015, and testing on 2017 (`TRAIN_YEARS`, `VAL_YEARS`, `TEST_YEARS`), thus evaluating how well the model generalizes to later edits.

The following sections compare the model's performance under these scenarios, with and without U, OE features.

4.1.1 Model Performance Across Split Types

Table 4.1 summarizes the performance metrics across random, geographic, and temporal splits. These metrics include *Accuracy*, *Precision*, *Recall*, *F1 score*, *AUC-PR*, and *ROC-AUC*. Each row indicates whether user and OSM element features are enabled.

Table 4.1: Vandalism detection performance for different data split types regarding accuracy, precision, recall, F1 score, AUC-PR, and ROC-AUC. *Note: U, OE features indicate User and OSM Element features.*

Data Split	Model Type	Accuracy	Precision	Recall	F1 Score	AUC-PR	ROC-AUC
Random Split	With U, OE	**0.9931**	**0.9673**	**0.9994**	**0.9831**	**0.99969**	**0.99991**
	Without U, OE	0.9742	0.8868	0.9986	0.9394	0.99742	0.99936
Geographic Split	With U, OE	**0.8063**	**0.7818**	0.7706	**0.7762**	**0.87210**	**0.91040**
	Without U, OE	0.5702	0.5155	**0.9884**	0.6776	0.61682	0.63373
Temporal Split	With U, OE	**0.7213**	**0.7460**	**0.8379**	**0.7893**	**0.85336**	0.79453
	Without U, OE	0.7093	0.7106	0.7968	0.7512	0.83770	**0.80316**

Random Split: This configuration treats the entire dataset uniformly, distributing samples randomly into train, validation, and test partitions. With U, OE features, the model achieves high accuracy (99.31%) and AUC-PR (99.969%), reflecting a robust ability to identify vandalism. Recall is notably strong at 99.94%, ensuring minimal missed vandalism. When these features are disabled, accuracy decreases to 97.42%, indicating that user and OSM element attributes are critical for capturing nuanced vandalism behaviors.

Geographical Split: In this setup, training occurs on some continents while validation and testing occur on distinct regions. While the with-U, OE model attains a decent accuracy of 80.63%, it underscores the challenges of regional variability, especially compared to the random split. The performance drop when excluding U, OE features (accuracy plunging to 57.02%) suggests that user behavior and local OSM element patterns are pivotal for handling cross-region data differences.

Temporal Split: Here, older data (e.g., 2018–2019) trains the model, while newer edits (e.g., 2017 or other designated test year) form the test set. Accuracy (72.13%) and recall (83.79%) remain decent with U, OE features, highlighting the model's ability to adapt over time, though not as strongly as under random conditions. Removing U, OE features diminishes performance further, revealing the difficulty of extrapolating older user behavior into newer vandalism patterns without comprehensive historical signals.

Observations and Significance: Across all split types, user and OSM element features consistently bolster metrics—most strikingly in geographical splits. Varying data distributions (regional or temporal) amplify the model's need for contextual signals, underscoring user-level editing traits and OSM geometry changes as essential components in detecting and understanding vandalism. These findings guide subsequent steps in evaluating our best model configurations in greater detail.

4.1.2 Visualizing Metrics for the Model with Optimal Configuration

This subsection focuses on the best-performing configuration—namely, the *random split* combined with user and OSM element (U, OE) features. By presenting and interpreting key visualizations, we aim to provide deeper insights into the model's classification strengths and limitations.

Confusion Matrix. Figure 4.1 shows the distribution of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for vandalism detection. Numerically, the model correctly identifies the majority of non-vandalism contributions (TN) and vandalism contributions (TP), while maintaining a minimal fraction of misclassifications (FP and FN). This balance is critical for OSM workflows, as excessive FP can overwhelm human reviewers, while excessive FN risks undetected malicious edits.

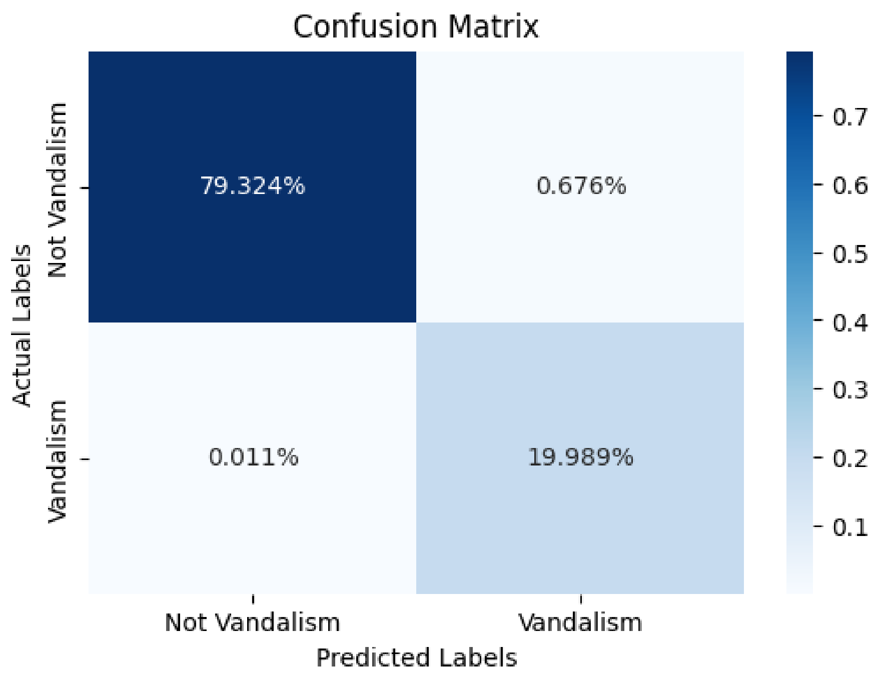
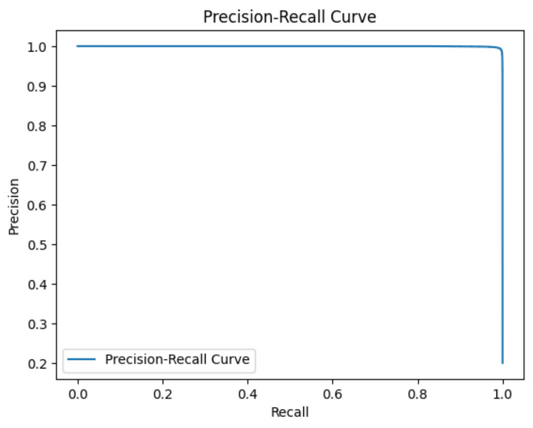


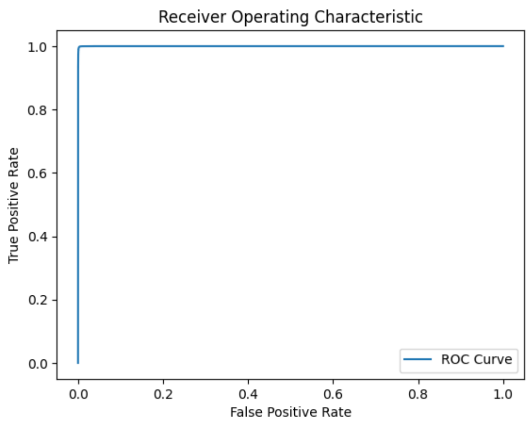
Figure 4.1: Confusion matrix for the model with best configuration (random split with U, OE features).

Precision-Recall Curve. In vandalism detection, vandalism contributions typically form the minority class. Figure 4.2a plots precision versus recall across varying classification thresholds, highlighting how effectively the model minimizes false positives (maintains high precision) while capturing the majority of true vandalism edits (high recall). The near-ideal curve indicates that precision remains high across a wide range of recall levels—a crucial property when balancing the cost of manual review versus missed vandalism incidents.

ROC Curve. The receiver operating characteristic (ROC) curve in Figure 4.2b illustrates the relationship between the true positive rate (TPR) and false positive rate (FPR). This best model closely hugs the top-left corner, yielding an AUC near 1.0—a strong indicator that the model consistently distinguishes vandalism from legitimate contributions without inflating FPR. Such performance is pivotal for OSM's large-scale environment, where even a small misclassification rate can translate into numerous erroneous flags or undetected vandal edits.



(a) Precision-recall curve for the best-performing contribution-level model.



(b) ROC curve for the best-performing contribution-level model.

Figure 4.2: Precision-recall curve and ROC curve for the contribution-level model with best configuration.

Overall Observations. These visualizations reinforce the conclusions drawn from the quantitative metrics:

- **Robust Classification**: The confusion matrix reveals very few misclassifications, implying minimal risk of undetected vandalism or unnecessary alerts.
- **Minority Class Excellence**: High precision-recall performance confirms that the model excels at flagging vandalism without overwhelming moderators with false positives.
- **Stable Threshold Behavior**: Both PR and ROC curves remain favorable

across a wide range of thresholds, indicating stability and flexibility in real-world scenarios where decision boundaries may shift.

These findings also underscore the importance of incorporating user and OSM element features into the detection pipeline, as they substantially enhance the model's ability to capture subtle vandalism signals. The next section explores the robustness of these performance metrics through bootstrapping, providing confidence intervals to gauge variance in model predictions.

4.1.3 Bootstrap Results

While the evaluation metrics in the Section 4.1.2 highlight the strong performance of the model on a fixed test set, it is equally important to assess the *stability* of these metrics under repeated sampling. Bootstrapping provides a robust mechanism for estimating variance and constructing confidence intervals, thereby revealing how sensitive the model is to variations in the test data.

Experimental Setup. The bootstrapping procedure involved drawing 1,000 random samples (with replacement) from the test set. Each sample maintained the same number of observations as the original test set, and the model's metrics (Accuracy, Precision, Recall, F1-score, AUC-ROC, and AUC-PR) were computed for each iteration. Table 4.2 presents the aggregated statistics over the 1,000 bootstrap iterations, providing insight into the mean, standard deviation, and 95% confidence intervals for each metric.

Table 4.2: Bootstrap Metrics (1,000 Iterations) for the Contribution-Level Model with Best Configuration (Random Split with U/O Features).

Metric	**Mean**	**Std Dev**	**95% CI Lower**	**95% CI Upper**
Accuracy	0.99313	0.00022	0.99270	0.99357
Precision	0.96733	0.00105	0.96527	0.96941
Recall	0.99943	0.00014	0.99914	0.99967
F1-Score	0.98312	0.00054	0.98203	0.98418
AUC-ROC	0.99992	0.00001	0.99989	0.99994
AUC-PR	0.99970	0.00003	0.99963	0.99976

Interpretation of Bootstrap Statistics. These bootstrap-derived statistics offer several key insights into model reliability:

- **Consistency of Accuracy**: The mean accuracy of 0.99313, coupled with a low standard deviation (0.00022), suggests the model consistently maintains correct classification above 99% of the time.
- **Precision Stability**: A mean precision of 0.96733 with tight confidence bounds (0.96527–0.96941) implies minimal fluctuation in false positives. This stability is crucial for real-world usage, where false alarms can burden moderators.
- **High Recall across Samples**: Recall remains exceptionally high at 0.99943, indicating that the model almost never misses vandalism, regardless of which subset of the test set is sampled.
- **Robust F1-Score**: The F1-score hovers around 0.983, reflecting a strong balance between precision and recall.

- **Near-Perfect AUC Metrics**: Both AUC-ROC (0.99992) and AUC-PR (0.99970) underscore the model's powerful discriminative capabilities, with confidence intervals confined to a very narrow range.

The bootstrap analysis confirms that the high performance reported in Section 4.1.2 is not an artifact of a particular test split. Rather, the model maintains robust accuracy and class-specific metrics across repeated subsampling. This consistency highlights the effectiveness of incorporating user and OSM element features, reinforcing the conclusion that these feature sets are integral to robust vandalism detection. It also suggests that the model can be trusted to deliver stable results in production-like scenarios, where incoming edits may slightly differ from those observed in the test set.

4.1.4 Geographical Evaluation Results

This section presents a continent-wise evaluation of our model ran on contribution data with the best configuration, illustrating how well it adapts to regional variations in OpenStreetMap (OSM) vandalism. Table **??** summarizes key metrics (accuracy, precision, recall, F1-score, AUC-ROC, AUC-PR) across eight continent categories in the dataset, revealing consistently high performance with minor discrepancies attributed to data distribution and sample sizes.
Observations and Insights.

- **Overall High Accuracy Across Continents:** Most regions display above 98% accuracy, underscoring the model's robust generalization to different geographic contexts. Even South America, with the lowest accuracy of 98.91%, remains close to the top end.

- **Near-Perfect Results in Antarctica:** Antarctica has only 23 samples, all classified correctly, yielding 100% metrics. This perfect performance partly reflects the small sample size, making it easier to achieve flawless classification.

Table 4.3: Continent-Wise Metrics for the Model with Best Configuration

Continent	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**
Africa	0.9973	0.9861	0.9998	0.9929	0.99999
Antarctica	1.0000	1.0000	1.0000	1.0000	1.00000
Asia	0.9927	0.9474	0.9990	0.9725	0.99993
Europe	0.9918	0.9601	0.9989	0.9791	0.99987
North America	0.9923	0.9623	1.0000	0.9808	0.99987
Oceania	0.9939	0.9835	0.9992	0.9913	0.99987
Other	0.9968	0.9904	1.0000	0.9952	0.99999
South America	0.9891	0.9672	0.9996	0.9831	0.99994

- **Minor Variation in Precision and Recall:** Precision hovers mostly above 96%—indicating few false positives—and recall consistently nears 100% in most continents, signifying that vandalism edits are rarely missed. Africa, for instance, shows a near-perfect recall of 0.9998, illustrating minimal undetected vandalism in that region's dataset.

- **High AUC-ROC and AUC-PR Values:** All regions achieve AUC-ROC scores above 0.9998, revealing superior discrimination between vandalism and non-vandalism edits. In practice, this means the model selects a threshold that balances precision and recall effectively across diverse geographies.

- **Potential Influences of Data Distribution:** Africa, Asia, and Europe comprise large numbers of edits, offering the model abundant training examples. Regions like Antarctica or Oceania have fewer edits, which can lead to either perfect performance (e.g., Antarctica) or minor classification errors if the training distribution does not fully capture these areas' editing patterns.

These results validate the model's capacity to handle global OSM edits, maintaining consistently high performance despite potential continental differences in editing style, user demographics, or map features. Although local factors (e.g., region-specific vandalism forms) could still pose challenges, the results suggest that the best model addresses the majority of vandalism scenarios around the world.

The next subsection (§**??**) delves deeper into why user and OSM element features bolster performance, comparing metrics between models with and without these features to illustrate their impact in a more nuanced manner.

4.1.5 Performance Gains from User and OSM Element Features

User and OSM element (**U/O**) features have a pronounced effect on the model's ability to distinguish malicious edits from legitimate contributions. This subsection provides both quantitative and qualitative insights into why these features matter, referencing comparative performance metrics, feature importance rankings, and domain-specific rationales.

Comparative Performance. Figure **??** highlights how metrics such as precision, recall, F1-score, AUC-PR, and AUC-ROC increase significantly when U/O features are enabled compared to a baseline model that omits them. For instance, precision rises sharply due to the model's enhanced capacity to identify truly vandalistic edits, minimizing spurious alerts. Similarly, recall improves, capturing more actual vandalism without sacrificing precision.

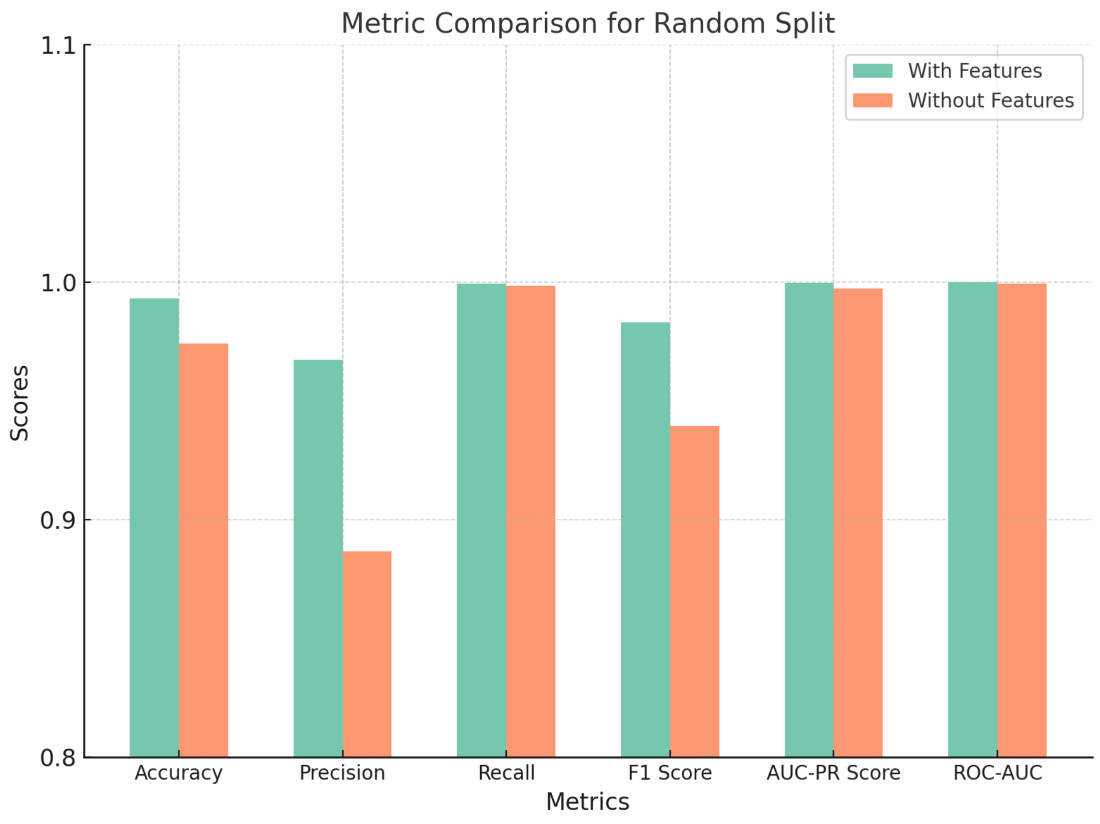


Figure 4.3: Metric comparison for models with and without user and OSM element (U/O) features, demonstrating clear performance gains when U/O features are included.

Feature Importance. Beyond raw metrics, the XGBoost framework enables interpretability via feature importance scores. Figure **??** shows the top 20 features ranked by total gain, revealing a dominant presence of U/O features. High-ranking user-centric attributes, such as `user_n_changesets_cum` or `user_previous_edit_timestamp`, highlight how user histories and editing patterns offer critical clues. OSM element attributes, like `n_edits` (cumulative edits on an object) or `element_previous_edit_timestamp`, capture the recency and frequency of map-object changes, effectively flagging suspicious edit bursts or back-and-forth modifications.

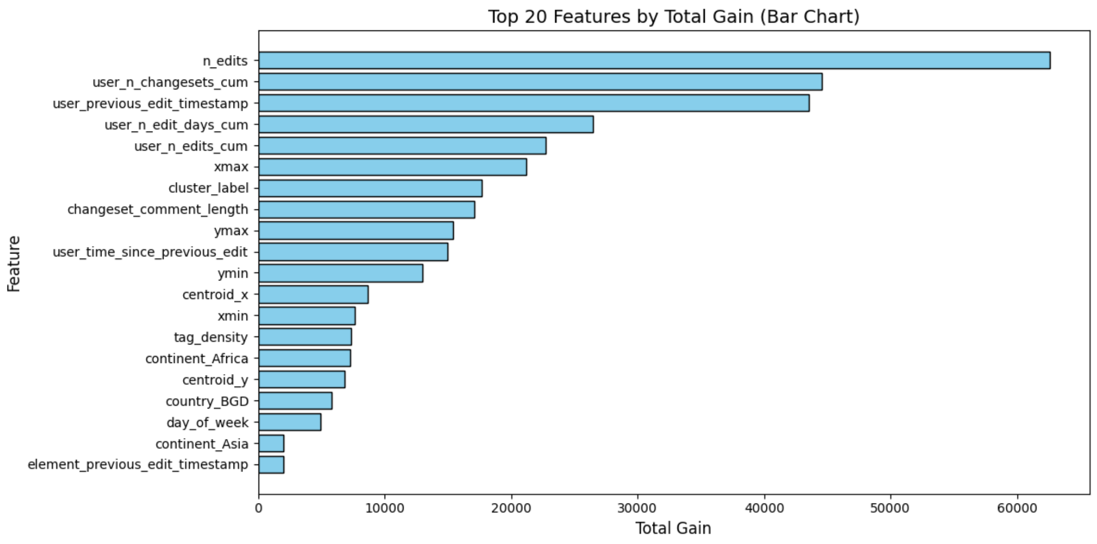


Figure 4.4: Top 20 features by total gain in the XGBoost model. U/O features dominate the upper ranks, underscoring their value in vandalism detection.

Comparative Analysis of Geo-Split Models

To evaluate the importance of user and OSM element features, an experiment was conducted comparing two models: one with user and OSM element features and one without. The analysis leveraged geo-splits by training, validating, and testing the models on geographically distinct regions such as Asia, Africa, Europe, Oceania, North America, South America, Antarctica, and Other. The bootstrap metrics were collected across multiple valid combinations of train, test, and validation regions, and the results are presented in Figure **??**.

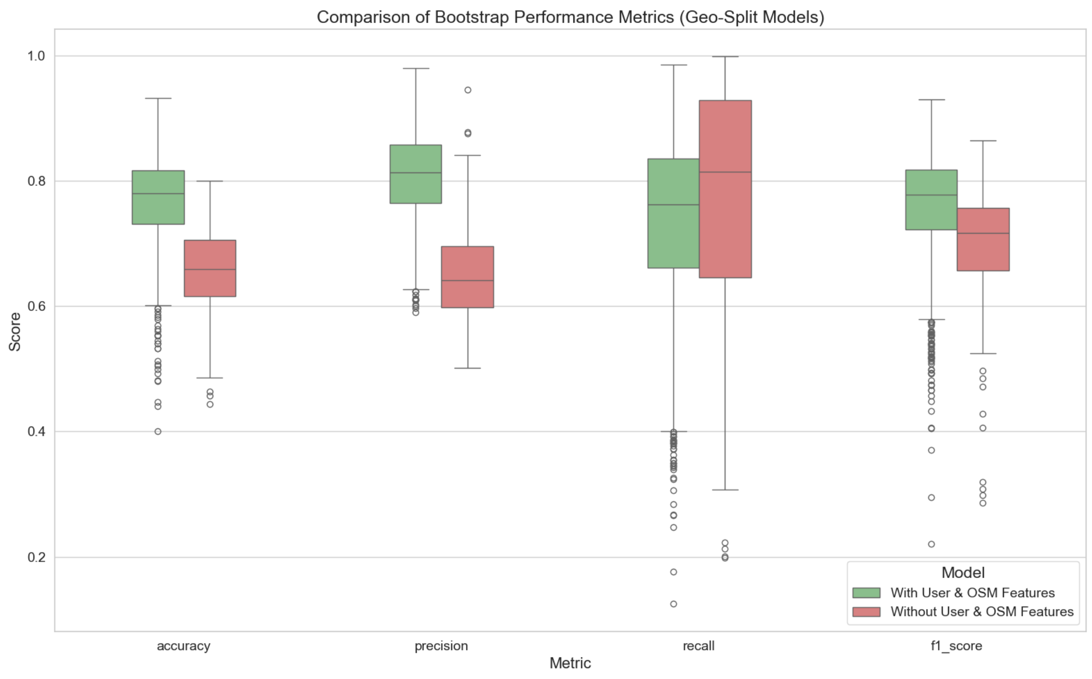


Figure 4.5: Comparison of Bootstrap Performance Metrics (Geo-Split Models). The box plots compare the performance of models trained with and without user and OSM features across accuracy, precision, recall, and F1-score.

From the box plot shown in Figure **??**, the following observations can be made regarding the performance of the two models:

1. Accuracy: The model with user and OSM features demonstrates a higher median accuracy, with most values clustering around 0.8. In contrast, the model without these features has a lower median accuracy, centered around 0.65. While the interquartile range is narrower for the model without features, the overall performance is less consistent and inferior.

2. Precision: The feature-rich model exhibits a higher median precision, indicating better performance in identifying relevant results. The precision for the model without user and OSM features is significantly lower, with a narrower spread, suggesting limited relevance detection.

3. Recall: The model with user and OSM features has a slightly lower median recall compared to its other metrics, but it still outperforms the feature-less model. The feature-less model shows a slightly higher recall but with significantly larger variability, implying inconsistent performance across different geo-splits.

4. F1 Score: The model with user and OSM features maintains a higher median F1 score, reflecting a better balance between precision and recall. The feature-less model has a noticeably lower F1 score, indicating a suboptimal trade-off between these metrics.

Overall Insights: The results highlight the clear advantage of incorporating user and OSM element features into the model. These features improve performance across all metrics, with the most significant gains observed in precision and accuracy. While there is some overlap in recall performance, the feature-rich model remains more consistent. This analysis underscores the importance of user behavior and OSM metadata in capturing spatial and contextual nuances, ultimately enhancing the model's ability to detect vandalism effectively.

Discussion and Domain Rationale. The synergy between user and element features is particularly evident in complex vandalism scenarios. For example:

1. **User Histories and Behavior Patterns**: Attributes such as `user_n_changesets_cum` and `user_n_edit_days_cum` reveal long-term behavioral trends. A contributor who shows abnormally high activity in a short span or a history of contentious edits is more likely to commit vandalism. Thus, user-based metrics add a layer of personalized detection that purely content-based features might miss.

2. **OSM Element Details**: Indicators like `n_edits` for a specific road or building denote how frequently the object has been changed. A series of rapid, repetitive edits on the same element may signal disputes, edit wars, or

back-and-forth vandalism. By tying suspicious activity to the element level, the model pinpoints hot spots in the map.

3. **Spatial/Temporal Contexts**: Combining user and element signals with geometry and timestamps further refines predictions. For instance, a user who rarely edits outside a specific region but suddenly modifies data in a distant locale may trigger a higher vandalism likelihood. Similarly, temporal features (`user_time_since_previous_edit`) help detect bursts of edits that deviate from normal user patterns.

4. **Robustness Across Splits**: As evidenced in Table 4.1, models with U/O features exhibit consistent gains under random, geographic, and temporal splits. By capturing both who is editing and what is being edited, the model generalizes better to unseen regions or future data.

These observations explain the metrics gap between models with and without U/O features. In essence, user features highlight contributor credibility and patterns, while element features enable a fine-grained view of object-based anomalies. Their combined effect sustains high precision and recall across various data splits, rendering the detection pipeline significantly more robust against the diverse forms of vandalism encountered in OSM.

4.2 Model Performance on Changeset Data

This section evaluates three approaches for vandalism detection at the *changeset* level:

A **baseline XGBoost model** (relying on changeset-level features),

The **hyper-classifier** (aggregating per-contribution probabilities), and

The **meta-classifier** (fusing both model outputs).

As introduced in Section 3.5, each approach targets session-wide vandalism signals from a different angle. The baseline model considers direct changeset attributes (e.g., bounding box, user metadata), while the hyper-classifier leverages fine-grained edit-level suspicion scores; the meta-classifier unifies these perspectives, aiming for further gains.

4.2.1 Model Comparisons and Results

Table **??** details the key metrics—Accuracy, Precision, Recall, F1-score, and AUC-ROC—on the test set for all three models. Notably, the hyper-classifier achieves a substantial improvement over the baseline, while the meta-classifier slightly outperforms the hyper-classifier in most metrics except precision.

Table 4.4: Comparison of Changeset-Level Models

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**
Baseline XGBoost	0.8712	0.8709	0.8773	0.8741	0.9444
Hyper-Classifier	0.9754	**0.9685**	0.9837	0.9761	**0.9959**
Meta-Classifier	**0.9759**	0.9654	**0.9882**	**0.9766**	0.9954

Baseline XGBoost (Changeset). At around 87% accuracy and a 0.9444 AUC-ROC, the baseline approach demonstrates moderate efficacy in detecting highly disruptive changesets. Its recall of 0.8773 ensures that a large portion of malicious sessions are detected, but it falls short in separating more subtle vandalism patterns.

Hyper-Classifier. By summarizing suspicious-edit probabilities into changeset-level statistics, the hyper-classifier significantly boosts both recall (0.9837) and precision (0.9685), pushing accuracy to 97.54%. This jump underscores the value of aggregated per-edit signals in exposing large-scale or distributed vandalism that might appear benign when viewed solely through baseline changeset attributes.

Meta-Classifier. Combining the baseline model's probability with the hyper-classifier's output yields a slight performance gain in overall accuracy (0.9759) and F1-score (0.9766), although precision dips marginally compared to the hyper-classifier. This mixed result suggests that while integrating changeset features with aggregated edit-level information can improve certain metrics (notably recall), additional tuning of meta-classifier hyperparameters may be required to fully exploit the complementary signals.

4.2.2 Evaluation via ROC and PR Curves

The ROC and PR curves in Figure **??** provide additional insights into model performance.

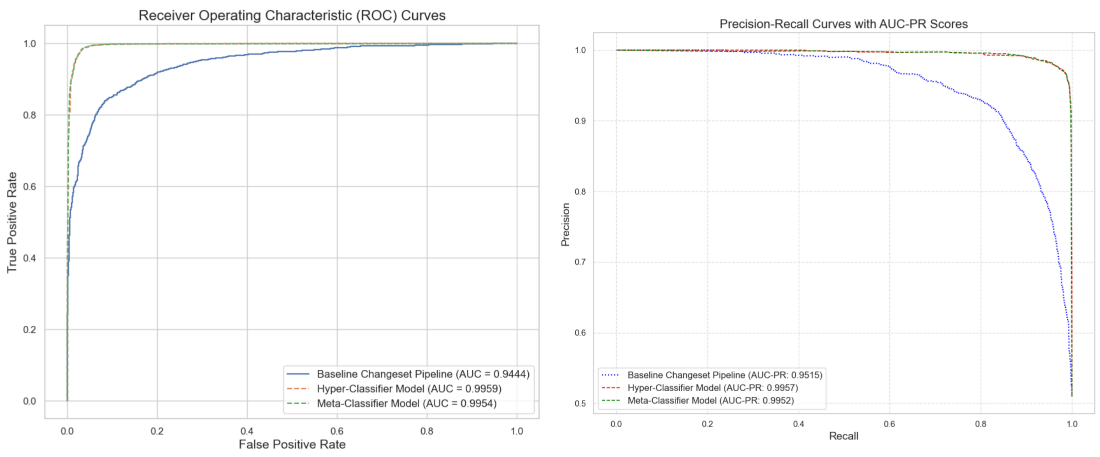


Figure 4.6: Comparison of ROC and PR Curves for Baseline Changeset Pipeline, Hyper-Classifier, and Meta-Classifier Models. AUC and AUC-PR scores are annotated in the legend.

Insights from ROC Curve. The ROC curve highlights the trade-off between true positive rate and false positive rate. Both the Hyper-Classifier and Meta-Classifier achieved AUC-ROC scores above 0.995, with the Hyper-Classifier slightly outperforming the Meta-Classifier. In contrast, the Baseline Changeset Pipeline achieved a lower AUC-ROC of 0.944, underscoring its limited ability to discern between positive and negative samples.

Insights from PR Curve. The PR curve illustrates precision-recall trade-offs, particularly useful for imbalanced datasets. The Hyper-Classifier achieved the highest AUC-PR score (0.9957), followed closely by the Meta-Classifier (0.9952). The Baseline Changeset Pipeline, however, showed a significantly lower AUC-PR score (0.9515), reflecting its weaker recall and precision balance compared to the other models.

4.2.3 Discussion and Insights

- **Strength of Aggregated Edit Signals**: The hyper-classifier's substantial improvement over the baseline emphasizes how distributed edit suspicions can reveal otherwise hidden session-wide vandalism patterns.

- **Slight Edge for Meta-Classifier**: Although the meta-classifier does not universally surpass the hyper-classifier, it achieves higher accuracy, recall, and F1-score, indicating that some synergy exists between baseline changeset features and aggregated probabilities. Precision is slightly lower (0.9654 vs. 0.9685), illustrating a trade-off where gains in recall occasionally come at the cost of more false positives.

- **Future Tuning Potential**: Further optimization of meta-classifier hyper-parameters—such as tree depth, regularization, or ensemble weighting—may yield additional benefits. If the baseline model introduces new features or the hyper-classifier refines its per-edit scoring, the meta-classifier's fusion could become more impactful.

Overall, while the hyper-classifier establishes a strong new baseline (particularly in precision), the meta-classifier surpasses it in certain metrics (e.g., accuracy, recall, and F1-score). These findings confirm the potential of merging high-level changeset attributes with aggregated per-contribution signals, but also suggest ongoing refinement may be required to achieve consistent gains across all metrics.

5 Application: Vandalism in OpenStreetMap

This chapter presents an analysis of the application of the proposed vandalism detection model on over two years of real-time OpenStreetMap (OSM) contributions (from January 2022 to July 2024). Using the prediction results, we evaluate temporal trends, highlight significant deviations, and discuss the implications for identifying and mitigating vandalism in OSM contributions.

5.1 Overview of Vandalism Predictions

The analysis covers monthly contributions and corresponding vandalism predictions over the study period. Predictions are based on the model's application to real-time datasets, enabling continuous monitoring of contributions for potential vandalism. Figures **??** and **??** illustrate these trends using bar and line graphs, respectively.

5.2 Key Observations and Analysis

5.2.1 General Trends

Over the study period, OSM saw significant variation in monthly contributions, ranging from approximately **19 million** (July 2024) to **39 million** (August 2023). Vandalism counts exhibited similar variability, with predictions ranging from around **46,000** (December 2023) to a peak of over **1.3 million** (August 2023). These figures highlight a mix of stable months and periods of sudden spikes in vandalism, as seen in Figures **??** and **??**.

5.2.2 Deviations and Notable Outliers

August 2023 Spike: The model flagged a dramatic increase in vandalism predictions, reaching over **1.3 million** entries, far exceeding the monthly average (Figure **??**). This spike may be attributed to coordinated malicious activities or errors in contribution uploads. Notably, contributions also peaked at 39 million during this month, suggesting heightened activity.

July 2023 Increase: Another substantial increase in vandalism predictions (over **207,000** entries) aligns with a high number of contributions for that month (**24.7 million**). While less extreme than August, this warrants further investigation to determine whether user behavior changes or systemic factors were responsible.

Temporal Stability in Early 2024: From January to July 2024, vandalism counts appeared relatively stable, ranging between **49,000 and 72,000** predictions per month. This stability may reflect improved moderation or reduced malicious activity during this period.

5.2.3 Seasonal Variability

Seasonal patterns suggest that vandalism may fluctuate based on user activity trends. For instance, higher vandalism predictions often coincided with months of elevated contributions (Figure **??**), while periods like late 2022 and early 2023 exhibited comparatively lower vandalism rates.

5.3 Implications for Real-Time Monitoring

The insights gained from these predictions underscore the importance of implementing robust, scalable vandalism detection mechanisms in OSM. Key takeaways include:

- **Spike Detection:** The ability to detect significant deviations (e.g., August 2023) provides an opportunity to investigate and address potential systemic issues or targeted vandalism campaigns.
- **Trend Analysis:** Regular monitoring of vandalism trends can help prioritize regions or periods for manual review by moderators, improving the overall quality of OSM contributions.
- **Scalability of the Model:** Despite the variability in contributions and vandalism counts, the model effectively scaled to handle datasets of varying sizes, demonstrating its practicality for large-scale applications.

5.4 Visualizing Results

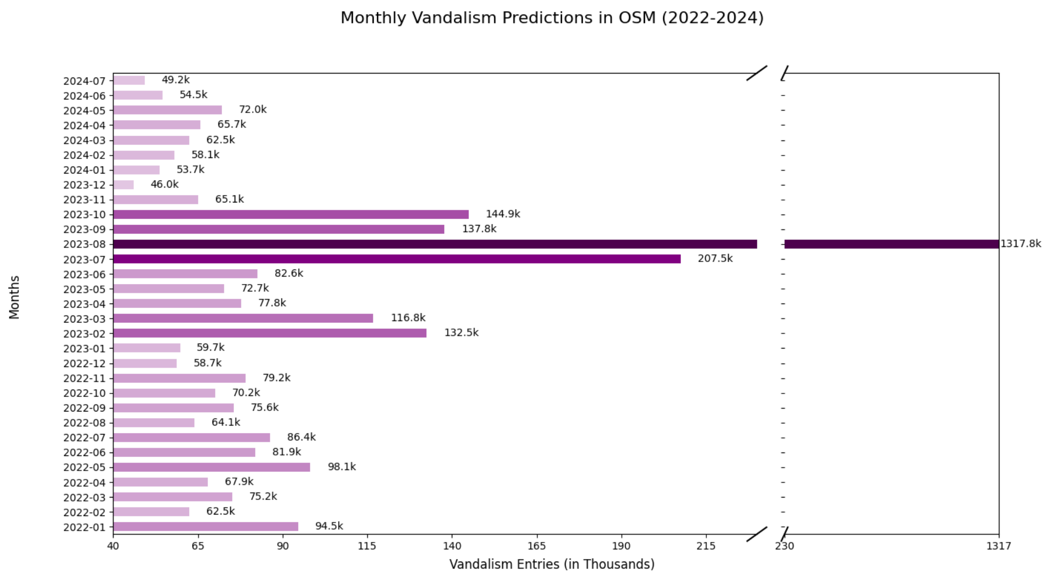


Figure 5.1: Monthly vandalism predictions in OSM (2022–2024) – Bar Graph. The spike in August 2023 is evident, indicating a significant deviation from regular vandalism patterns.

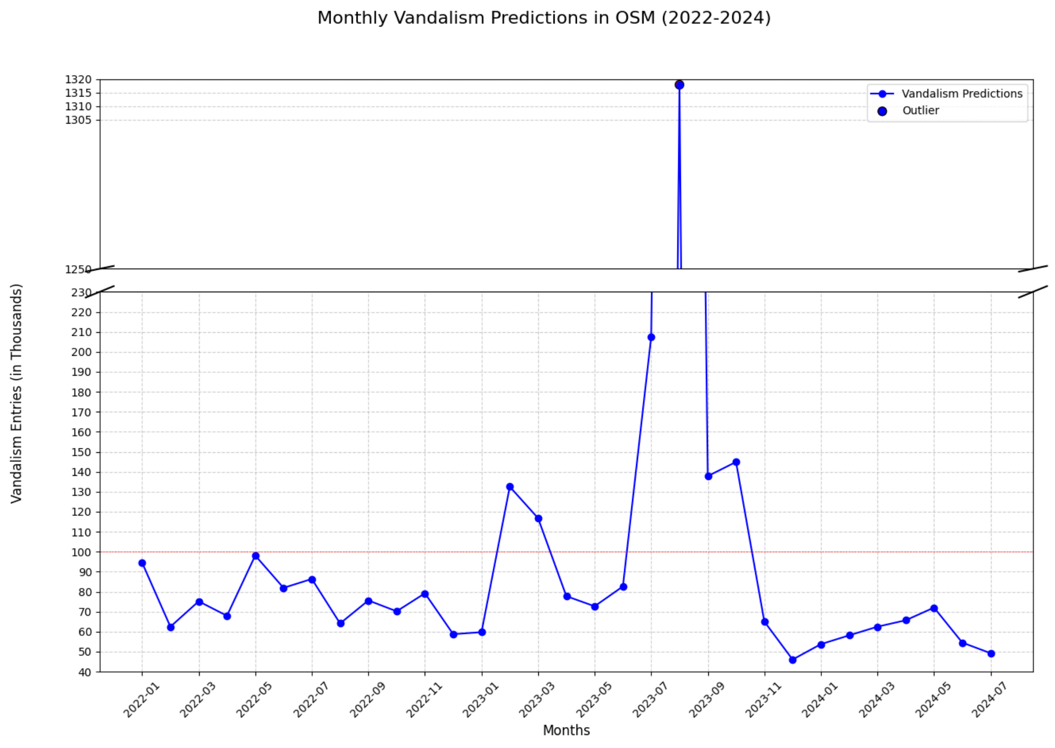


Figure 5.2: Monthly vandalism predictions in OSM (2022–2024) – Line Graph. Seasonal patterns and notable spikes, such as in August 2023, are highlighted.

Interactive Heatmap Visualization

To further understand the spatial distribution of vandalism, a heatmap was generated showing vandalism predictions across the world. The heatmap is available for interactive exploration, allowing users to visualize patterns for each month from January 2022 to July 2024. The static image in Figure **??** shows the distribution of vandalism for August 2023, where the spike in predictions aligns with regions experiencing heightened activity.

The full heatmap is available in our GitHub repository at Link. *To view it in a browser, download the raw file and rename its extension from* `.txt` *to* `.html`*.*

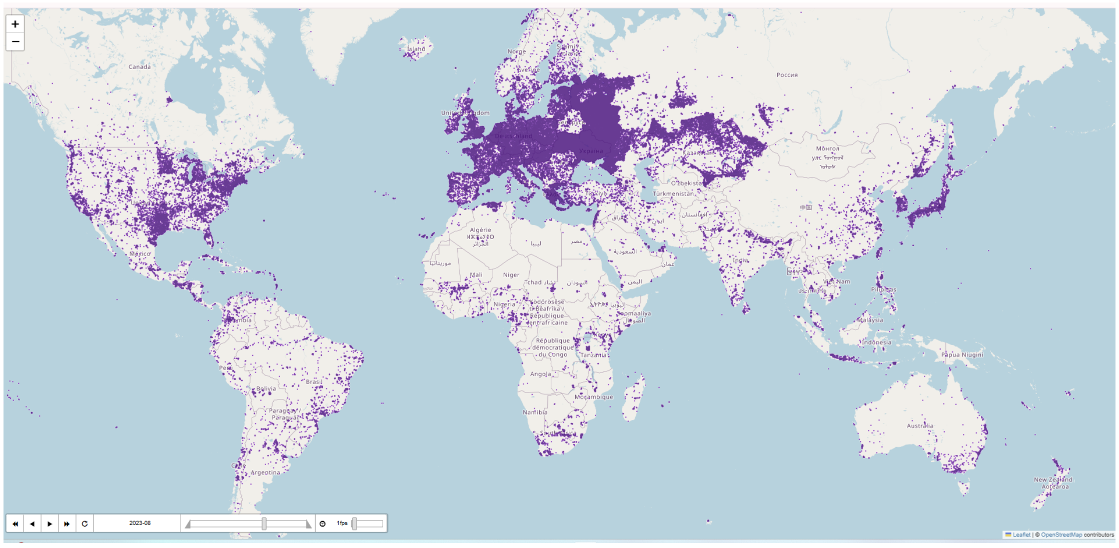


Figure 5.3: Heatmap of vandalism predictions for August 2023. The spike in vandalism activity is prominently visible across highly active regions.

5.5 Summary

The application of the vandalism detection model to over two years of real-time data illustrates its effectiveness in identifying malicious contributions and its adaptability to dynamic OSM editing trends. While general predictions align with expected patterns, significant deviations (e.g., August 2023) highlight the need for continued refinement of detection systems and collaborative efforts to mitigate vandalism in OSM.

6 Discussions and Future Work

This chapter reflects on the key insights gained from implementing and evaluating the proposed vandalism detection framework, addressing the objectives and research questions put forth in the introduction chapter (§1.4). In particular, it discusses the model's practical integration into real-world OSM workflows, the importance of user and OSM element features, and the feasibility of scaling the approach to high-volume data streams. The chapter concludes with an outline of potential directions for future work to bolster the robustness and adaptability of this system.

6.1 Discussion

6.1.1 Achieving Robust Detection in Real-World Settings

A primary ambition of this thesis was to train and validate the model on a moderately sized dataset (approximately 0.6 million labeled contributions) while ensuring that it generalizes effectively to real-time OSM edits. Preliminary real-world tests indicate that around 30 million new contributions may arrive monthly, underscoring the necessity of a pipeline that not only excels in predictive accuracy but also maintains rapid inference times.

- **Prediction as a New Labeled Dataset:** The high precision and recall observed in Chapter 4 suggest that model outputs can serve as a high-confidence labeled set, marking suspicious edits for manual review. This iterative process can continually refine the training corpus, thus improving the model over time.

- **Integration at HeiGIT:** Given the pipeline's chunk-based architecture and proven scalability, it can be seamlessly linked to HeiGIT's existing OSM data stream. The inference process (roughly 1.5 hours for monthly batch files) fits within the operational window of many data ingestion workflows, enabling near real-time identification of vandalism in 5-minute snapshots.

- **Inference Efficiency:** Despite the comprehensive set of features, including user histories and OSM element metadata, the final XGBoost classifier delivers swift predictions. This efficiency is pivotal in continuously monitoring OSM's stream of edits, ensuring that flagged cases reach reviewers swiftly.

6.1.2 Importance of User and OSM Element Features

As discussed in Chapter 4 and elaborated in Section **??**, user-focused attributes (e.g., cumulative changesets, edit frequencies) and OSM element features (e.g., edit histories of specific objects) substantially enhance the model's ability to identify vandalism. Empirical evidence underscores their importance in capturing subtle behavioral and spatial patterns that improve detection accuracy:

- **Better Geographical and Temporal Generalization:** Models enriched with user and element features performed more consistently, even under geographic (Figure **??**) and temporal splits, indicating that these signals capture enduring behavioral and object-based patterns less dependent on region or time.

- **Heightened Precision and Recall:** The metrics in Figure **??** confirm large gains in precision and recall with these features enabled, particularly in high-variance settings. The boosted precision lowers false alarms, while higher recall ensures fewer malicious edits evade detection.

By integrating insights about who is editing and what is being edited, the model robustly identifies anomalous behaviors, even when novel forms of vandalism arise.

6.1.3 Outcome Relative to Thesis Objectives and Research Questions

The objectives introduced in Section 1.4 aimed to build a feature-rich, scalable ML pipeline capable of detecting vandalism with high accuracy while remaining computationally efficient. Key achievements include:

- **Comprehensive Feature Engineering:** Spatial, temporal, user, and textual attributes were combined to capture the diversity of OSM data. Chapter 4 detailed how these features bolster accuracy in different split configurations.
- **Scalable ML Pipeline:** Chunk-based data loading and parallelized feature extraction allow the system to process millions of edits per month with minimal overhead. Inference times of roughly 1.5 hours for monthly data or real-time detection within a 5-minute window confirm operational viability.
- **High Predictive Performance:** The best XGBoost-based model attained precision and recall above 95% in random splits, with robust performance in geographic and temporal splits as well. This aligns with the goal of minimizing false positives while maintaining strong detection of genuine vandalism.
- **User Behavior and Contextual Cues:** User-history features proved crucial for capturing recidivist or abrupt editing behaviors, while OSM element metadata flagged localized edit wars or repeated tampering. These findings address the research question of which signals most enhance vandalism detection.

Nonetheless, certain research questions persist around sustaining performance under evolving vandalism strategies and guaranteeing adaptation to newly emerging edit patterns.

6.2 Future Work

Despite the demonstrated success, several avenues exist for further enhancement of the proposed system:

1. **Real-Time Streaming and Continuous Learning:** While the current pipeline is chunk-based and efficient enough for monthly or near real-time data, an incremental or online learning framework could further reduce latency. This would allow immediate model updates in response to new edit behaviors or flagged vandalism cases.

2. **Refined Data Labeling at Scale:** The thesis leverages a labeled dataset of around 0.6 million contributions. As predictions are integrated into OSM's data stream at HeiGIT, an iterative feedback loop could generate more extensive, high-quality labels. Active learning or crowdsourced verification could refine class boundaries, especially for ambiguous edits.

3. **Deep Learning Architectures and Transfer Learning:** While XGBoost proved optimal for tabular data, recent developments in transformer-based models (e.g., FT-Transformers) may yield benefits for extremely high-dimensional or sparse features. Future experiments can compare these architectures if data volumes justify the additional computational overhead.

4. **Localized Models and Domain Adaptation:** Regional or city-level models might outperform a single global model, especially in highly specialized editing contexts. Domain adaptation techniques could further enhance cross-region or cross-time performance by reducing calibration overhead when shifting between geographies or time periods.

5. **In-Depth Interpretability and Explanation:** While feature importances help unravel model decisions, advanced interpretability tools—such as SHAP (SHapley Additive exPlanations)—could clarify how each user-centric or OSM element attribute influences individual predictions. This granularity aids community trust and fosters collaborative improvement of detection rules.

By pursuing these directions, the vandalism detection pipeline can stay aligned with OSM's ever-evolving landscape. Enhanced real-time responsiveness, larger labeled datasets, and advanced interpretability or domain adaptation would expand both the technical depth and practical utility of the framework.

6.3 Conclusion

In summary, the research has demonstrated the feasibility and effectiveness of a feature-rich, XGBoost-based pipeline to detect vandalism across diverse OSM editing patterns. The system's strong performance—further underscored by user and OSM element features—indicates a viable path toward large-scale, near real-time deployment. As OSM continues to grow and editing behaviors change, ongoing improvements to data labeling, model updating, and region-specific adaptation will be essential to sustain the high detection rates exhibited in this thesis.

A Project Setup and Usage

This appendix provides an overview of the project layout, installation guidelines, and quick-start steps for reproducing the vandalism detection pipeline described in this thesis. While not exhaustive, the following structure offers a snapshot of key scripts, directories, and configuration files.

A.1 Repository Structure

A typical arrangement of the project files might appear as follows:

```
./
    ml_training_and_eval_pipeline.py
    prediction_pipeline_for_monthly_files.py
    prediction_pipeline_for_smaller_daily_files.py
    README.md
    requirements.txt

    data/
        changeset_data/
            output/
            processed/
            raw/
            Visualization/
        contribution_data/
            output/
            processed/
        raw/
        Visualization/

docs/
    discussions.md
    feature.md
    thesis.tex
    (other .md files with research notes)

logs/
    changeset_pipeline.log
    contribution_pipeline.log
    pipeline.log

models/
    changeset_model/
        random/
        geographic/
        temporal/
        (subdirectories for hyper_classifier, meta_classifier)
    contribution_model/
        random/
        geographic/
        temporal/
        (subdirectories for hyper_classifier, meta_classifier)

notebooks/
    visualization_and_analysis_changeset_data.ipynb
    visualization_and_analysis_contribution_data.ipynb
    (other exploratory notebooks)

src/
```

```
        data_loading.py
        feature_engineering.py
        data_splitting.py
        clustering.py
        training.py
        evaluation.py
        preprocessing.py
        hyper_parameter_search.py
        meta_classifier.py
        config.py
        ...
        hyper_classifier/
            hyper_classifier_main.py
            hyper_classifier_training.py
            hyper_classifier_data_splitting.py
        data_gathering/
            contributions/
            changesets/
            utility_codes/
        plots/
            plots.py
            box_plot_for_bootstraps.py
        (other utility codes and scripts)
```

Key folders and scripts:

- `ml_training_and_eval_pipeline.py`: End-to-end pipeline orchestrating data preprocessing, feature extraction, model training, and evaluation for OSM vandalism detection.
- `data/`: Contains raw OSM data, processed datasets, and subfolders for storing output artifacts (e.g., Parquet files, visualizations).

- `docs/`: Collection of research notes, design documents, and auxiliary Markdown files. May include the main Latex thesis file.
- `logs/`: Houses log files for different pipeline executions (changeset-level, contribution-level, or temporal splits).
- `models/`: Stores trained model artifacts (`pkl` files) for both changeset and contribution models, along with hyper- and meta-classifiers.
- `notebooks/`: Jupyter notebooks for exploratory data analysis and visualization.
- `src/`: Primary source code implementing data loading, feature engineering, clustering, training, and evaluation logic. Subdirectories like `hyper_classifier/` hold specialized scripts for aggregated detection.

A.2 Installation and Dependencies

1. **Clone the Repository:**
   https://github.com/rspseshasai/vandalism_detection_osm.git
2. **Set Up a Python Virtual Environment (Optional but Recommended):**
   ```
   python -m venv venv
   source venv/bin/activate        # On Linux or macOS
   .\venv\Scripts\activate         # On Windows
   ```
3. **Install Required Packages:**
   ```
   pip install --upgrade pip
   pip install -r requirements.txt
   ```

This installs necessary dependencies such as `xgboost`, `pandas`, `numpy`, and `scikit-learn`.

A.3 Getting Started

1. **Prepare Data:** Place the relevant OSM data files (either raw monthly extracts or previously processed contributions/changesets) into the `data/` directory. Adjust directory paths in scripts or configuration files if necessary.

2. **Configure Settings:** Edit `config.py` (or equivalent) to specify hyperparameters, file paths, and logging preferences. This might include chunk sizes for data loading or hyperparameter ranges for model optimization.

3. **Run the Pipeline:** Execute the main pipeline script:

   ```
   python ml_training_and_eval_pipeline.py
   ```

   This orchestrates data loading, preprocessing, training, and evaluation. Logs are saved in `logs/`, while trained models and evaluation artifacts appear in `models/` and `data/output/` or `visualization` directories.

4. **Review Results:** Examine logs for run summaries, evaluate performance metrics (e.g., confusion matrices, ROC curves), and verify output data in `data/output/`. For large datasets, consider using chunk-based processing or streaming logic embedded in `src/data_loading.py`.

A.4 Additional Notes and Resources

- **Inference Pipelines**: - The scripts `prediction_pipeline_for_monthly_files.py`, and `prediction_pipeline_for_smaller_daily_files.py` facilitate inference on different data ingestion schedules.The former is optimized for large-scale monthly datasets, ensuring efficient batch processing, while the latter enables real-time or near-real-time inference on smaller daily updates. This distinction allows the system to adapt seamlessly to varying data processing needs while maintaining accuracy and efficiency.

- **Notebooks for Exploration**: In `notebooks/`, several `.ipynb` files allow interactive data analysis, feature visualization, or ablation experiments.
- **Model Checkpoints**: The `models/` folder maintains the final trained models (including hyper- and meta-classifiers). Examine timestamps or naming conventions to track model versions.
- **Documentation and Discussion**: The `docs/` directory includes supporting Markdown documents detailing design decisions, alternative strategies, and future directions.
- **Extensibility**: To extend functionalities (e.g., new features or classifiers), consider placing additional scripts in `src/` and referencing them from the pipeline. The chunk-based approach ensures minimal memory overhead, facilitating seamless scaling for large monthly or daily files.

A.5 Summary

This appendix outlines the core repository structure and key steps for deploying the vandalism detection pipeline. By adhering to this organizational model and following the installation/configuration guidelines, researchers and practitioners can replicate the results described in this thesis, customize the framework for different data sources, or integrate new classification methodologies. The modular nature of the design encourages iterative improvements while safeguarding consistency, scalability, and reproducibility throughout the evolution of OpenStreetMap data.

Bibliography

[1] *OpenStreetMap*: https://www.openstreetmap.org/

[2] *Vandalism in OSM*: https://wiki.openstreetmap.org/wiki/Vandalism

[3] *XGBoost Documentation*: https://xgboost.readthedocs.io/en/stable/

[4] *Heidelberg Institute for Geoinformation Technology (HeiGIT)*: https://heigit.org/

[5] *Changeset in OSM*: https://wiki.openstreetmap.org/wiki/Changeset

[6] Jamal Jokar Arsanjani, Alexander Zipf, Peter Mooney, and Marco Helbich. 2015. *An Introduction to OpenStreetMap in Geographic Information Science: Experiences, Research, and Applications.* Available at: https://link.springer.com/book/10.1007/978-3-319-14280-7

[7] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. 2021. *WorldKG: A World-Scale Geographic Knowledge Graph.* Available at: https://arxiv.org/abs/2109.10036

[8] Y. Li, J. Anderson, and Y. Niu. *Vandalism Detection in OpenStreetMap via User Embeddings.* Proceedings of the 30th ACM International Conference on Information and Knowledge Management, 2021. Available at: https://dl.acm.org/doi/10.1145/3459637.3482213

[9] N. Tempelmeier and E. Demidova, *Attention-Based Vandalism Detection in OpenStreetMap*, Proceedings of The Web Conference 2022, 2022. Available at: https://arxiv.org/abs/2201.10406.

[10] W. Almeida. *OSMCha: OpenStreetMap Changeset Analyzer.* 2016. Available at: https://osmcha.org/

[11] B. Adler, L. De Alfaro, and I. Pye. *Detecting Wikipedia vandalism using WikiTrust.* CLEF 2010. Available at: https://luca.dealfaro.com/papers/10/c62-pan2010.pdf

[12] S. Heindorf, M. Potthast, G. Engels, and B. Stein. *Vandalism detection in Wikidata.* ACM International Conference on Information and Knowledge Management, 2016. Available at: https://dl.acm.org/doi/10.1145/2983323.2983740

[13] S. Heindorf, M. Potthast, G. Engels, and B. Stein. *Towards vandalism detection in knowledge bases: Corpus construction and analysis.* Available at: https://dl.acm.org/doi/10.1145/2766462.2767804

[14] Juan R. Martinez-Rico, Juan Martinez-Romo, and Lourdes Araujo. *Can deep learning techniques improve classification performance of vandalism detection in Wikipedia?* Engineering Applications of Artificial Intelligence 78, 2019. Available at: https://www.researchgate.net/publication/330801258_Can_deep_learning_techniques_improve_classification_performance_of_vandalism_detection_in_Wikipedia

[15] P. Neis, M. Goetz, and A. Zipf. *Towards automatic vandalism detection in OpenStreetMap.* ISPRS International Journal of Geo-Information, 2012. Available at: https://www.mdpi.com/2220-9964/1/3/315

[16] Q. T. Truong, G. Touya, and C. de Runz. *Towards Vandalism Detection in OpenStreetMap Through a Data Driven Approach.* GIScience

2018. Available at: https://drops.dagstuhl.de/storage/00lipics/lipics-vol114-giscience2018/LIPIcs.GISCIENCE.2018.61/LIPIcs.GISCIENCE.2018.61.pdf

[17] Q. T. Truong, G. Touya, and C. de Runz. *OSMWatchman: Learning How to Detect Vandalized Contributions in OSM Using a Random Forest Classifier.* ISPRS International Journal of Geo-Information, 2020. Available at: http://mdpi.com/2220-9964/9/9/504

[18] Tutorialspoint. *XGBoost Architecture.* Available at: https://www.tutorialspoint.com/xgboost/xgboost-architecture.htm

[19] S. O. Arik and T. Pfister. *TabNet: Attentive interpretable tabular learning.* Proceedings of the AAAI Conference on Artificial Intelligence, 2021. Available at: https://arxiv.org/abs/1908.07442

[20] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. *Revisiting deep learning models for tabular data.* Advances in Neural Information Processing Systems, 2021. Available at: https://arxiv.org/abs/2106.11959

[21] T. Chen and C. Guestrin. *XGBoost: A Scalable Tree Boosting System.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. Available at: https://dl.acm.org/doi/10.1145/2939672.2939785

[22] J. H. Friedman. *Greedy Function Approximation: A Gradient Boosting Machine.* Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001. Available at: https://projecteuclid.org/euclid.aos/1013203451

[23] J. Bergstra and Y. Bengio, *Random Search for Hyper-Parameter Optimization*, Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012. Available at: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

[24] B. T. Adler, L. De Alfaro, S. M. Mola-Velasco, P. Rosso, and A. G. West, *Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features*, International Conference on Intelligent Text Processing and Computational Linguistics, 2011. Available at: https://www.researchgate.net/publication/51527727_Wikipedia_Vandalism_Detection_Combining_Natural_Language_Metadata_and_Reputation_Features.

[25] K. Y. Itakura and C. L. A. Clarke, *Using Dynamic Markov Compression to Detect Vandalism in Wikipedia*, Proceedings of the 32nd ACM SIGIR Conference on Research and Development in Information Retrieval, 2009. Available at: https://dl.acm.org/doi/10.1145/1571941.1572146.

[26] S. Kumar, F. Spezzano, and V. S. Subrahmanian, *VEWS: A Wikipedia Vandal Early Warning System*, Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015. Available at: https://arxiv.org/abs/1507.01272.

[27] K. McKeown and W. Wang, *"Got You!": Automatic Vandalism Detection in Wikipedia with Web-Based Shallow Syntactic-Semantic Modeling*, Proceedings of the 23rd International Conference on Computational Linguistics, 2010. Available at: https://aclanthology.org/C10-1129.pdf.

[28] K. Smets, B. Goethals, and B. Verdonk, *Automatic Vandalism Detection in Wikipedia: Towards a Machine Learning Approach*, AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy, 2008. Available at: https://www.researchgate.net/publication/254460840_Automatic_Vandalism_Detection_in_Wikipedia_Towards_a_Machine_Learning_Approach.

[29] K.-N. Tran and P. Christen, *Cross-Language Learning from Bots and Users to Detect Vandalism on Wikipedia*, IEEE Transactions on Knowledge and Data

Engineering, vol. 27, no. 3, pp. 673–685, 2014. Available at: https://dl.acm.org/doi/10.1109/TKDE.2014.2339844.

[30] S. Yuan, P. Zheng, X. Wu, and Y. Xiang, *Wikipedia Vandal Early Detection: From User Behavior to User Embedding*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2017. Available at: https://link.springer.com/chapter/10.1007/978-3-319-71249-9_50.

[31] R. Crescenzi, M. Fernandez, F. A. Garcia Calabria, P. Albani, D. Tauziet, A. Baravalle, and A. S. D'Ambrosio, *A Production-Oriented Approach for Vandalism Detection in Wikidata—The Buffaloberry Vandalism Detector at WSDM Cup 2017*, arXiv preprint, 2017. Available at: https://arxiv.org/abs/1712.06919.

[32] A. Grigorev, *Large-Scale Vandalism Detection with Linear Classifiers—The Conkerberry Vandalism Detector at WSDM Cup 2017*, arXiv preprint, 2017. Available at: https://arxiv.org/abs/1712.06920.

[33] S. Heindorf, M. Potthast, G. Engels, and B. Stein, *Overview of the Wikidata Vandalism Detection Task at WSDM Cup 2017*, arXiv preprint, 2017. Available at: https://arxiv.org/abs/1712.05956.

[34] A. Sarabadani, A. Halfaker, and D. Taraborelli, *Building Automated Vandalism Detection Tools for Wikidata*, Proceedings of the 26th International Conference on World Wide Web Companion, 2017. Available at: https://dl.acm.org/doi/10.1145/3041021.3053366.

[35] *Introduction to Bootstrapping*: https://carpentries-incubator.github.io/machine-learning-novice-python/07-bootstrapping/index.html

[36] Envisioning.io, *Meta-Classifier*, Available at: https://www.envisioning.io/vocab/meta-classifier.

[37] G. Fahad, N. Alshomrani, A. Bahkali, A. S. Khan, M. T. Alotaibi, and A. A. Zafar, *Comprehensive Review of K-Means Clustering Algorithms*, ResearchGate, 2021. Available at: https://www.researchgate.net/publication/354547481_Comprehensive_Review_of_K-Means_Clustering_Algorithms.

[38] Scikit-learn, *sklearn.cluster.KMeans — scikit-learn 1.0.2 documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[39] IBM, *K-Means Clustering*, IBM Think Blog. Available at: https://www.ibm.com/think/topics/k-means-clustering.

[40] Scikit-learn, *Cross-validation: evaluating estimator performance*, Available at: https://scikit-learn.org/stable/modules/cross_validation.html.