THE USA JOURNALS
THE AMERICAN JOURNAL OF ENGINEERING AND TECHNOLOGY (ISSN – 2689-0984)
VOLUME 06 ISSUE10

PUBLISHED DATE: - 06-10-2024
DOI: - https://doi.org/10.37547/tajet/Volume06Issue10-05 **PAGE NO.: - 33-41**

RESEARCH ARTICLE **Open Access**

MACHINE LEARNING ALGORITHMS FOR ANOMALY DETECTION IN PUBLIC DATA USING GITHUB AS AN EXAMPLE

Lyamkin Ilya
Senior Full Stack Engineer at Spotify, USA

Abstract
This study explores the application of machine learning algorithms for detecting anomalies in GitHub data to enhance the evaluation of technological projects. The research aims to develop a robust methodology for identifying data anomalies, such as artificial activity spikes, that can distort project assessments. Methods such as Isolation Forest, One-Class SVM, and advanced deep learning techniques like autoencoders and GANs are employed to analyze and identify irregular patterns in GitHub repositories. The findings demonstrate that these algorithms effectively detect both obvious and subtle anomalies, offering reliable insights into project authenticity. The proposed conceptual model integrates these methods into a scalable system, enhancing transparency and accuracy in technological project evaluation. The novelty of this work lies in its comprehensive approach to analyzing GitHub data, combining traditional and deep learning techniques to improve the reliability of assessments, making it a significant contribution to the field.

Keywords Machine learning, anomaly detection, GitHub data, technological projects, deep learning, isolation forest, One-Class SVM, autoencoders, GANs, data analysis.

INTRODUCTION

GitHub [2], one of the largest platforms for hosting and collaborative development of software, has become an integral part of the modern technological landscape. Its role in evaluating technology projects and startups is growing as more investors and developers turn to platform activity metrics to make decisions on funding and collaboration [4]. However, with the increasing popularity of GitHub, a significant issue has arisen—manipulation of metrics such as stars, commits, and forks, which calls into question the reliability of these data as indicators of project success and quality [3].

The relevance of studying methods for analyzing public GitHub data is justified by the need to improve the accuracy and reliability of technology project evaluations. The falsification of metrics, for example, fake stars, creates a false impression of a project's popularity and significance, which can mislead investors and users. As a result, identifying anomalies in such data becomes a critical task for ensuring transparency and fairness in the technology community [3].

Machine learning (ML), as an advanced data analysis technology, opens new possibilities for detecting such anomalies. Machine learning algorithms can process large amounts of data, identifying hidden patterns and deviations from the norm that are difficult to detect using traditional analysis methods. Applying these algorithms to GitHub data enables the automation of detecting suspicious activities, such as sudden

spikes in stars or unusual commit patterns, contributing to a more accurate assessment of projects.

The goal of this study is to develop a methodology for detecting anomalies in GitHub data using machine learning algorithms. The study will explore the main approaches to analyzing such data, conduct a comparative analysis of various machine learning algorithms, and propose a conceptual model of a system for automated anomaly detection. This model can be applied to projects focused on evaluating technology startups, such as DualSpace AI, providing more reliable and objective analysis results.

Thus, the research is aimed at addressing the important task of improving the transparency and reliability of GitHub data, which ultimately contributes to better decision-making in software development and investment.

1. Theoretical foundations and methodology for detecting anomalies in GitHub data

GitHub [2], as a platform for hosting and collaborative software development, provides a wide range of public data that can serve as a basis for analyzing and evaluating technology projects [4]. These data include, but are not limited to, the number of stars, forks, commits, pull requests, as well as metadata about users, repositories, and project-related events.

The data structure on GitHub is organized into repositories, each representing a container for code and associated artifacts. The main elements of a repository include:

- Commits: These are fixed changes to the code that are recorded in the change history. Commits contain information about who made the changes, when they were made, and a description of the changes.

- Stars: This metric reflects the popularity and interest in a repository. Users can "star" repositories to show their support or interest.

- Forks: These are copies of a repository created for further development or modification of the code. Forks are an important indicator that a project is attracting developers and has potential for further development.

- Pull requests: These are proposals to make changes to a repository, which can be accepted or rejected by the project owners. Pull requests reflect community activity and the involvement of external developers.

Each of these elements can be analyzed to detect anomalies that may indicate unnatural or manipulative actions.

Anomalies in GitHub data can take various forms, each representing a potential threat to the accuracy of project analysis and evaluation. It is important to classify these anomalies to develop effective methods for detecting and mitigating them.

The main types of anomalies include:

- Star anomalies: Sudden spikes in the number of stars may indicate manipulation, where stars are bought or generated by bots. These anomalies are easily noticeable when analyzing time series data, revealing unnatural patterns.

- Commit anomalies: Excessive activity in the form of numerous minor or irrelevant commits over a short period may indicate attempts to create the illusion of active development.

- Fork anomalies: A sharp increase in the number of forks, especially if these forks do not lead to further activity, may signal manipulation aimed at increasing a project’s visibility.

- Pull request anomalies: A large number of pull requests from inactive or newly created accounts may indicate artificial activity designed to boost the project's popularity [4].

To address the task of detecting anomalies in GitHub data, machine learning methods are employed. One approach is to use unsupervised learning algorithms, such as Isolation Forest [5] and One-Class SVM [6]. These methods are particularly useful when there are no pre-labeled data, as they can detect outliers based on the statistical characteristics of the dataset. Isolation Forest, for instance, constructs multiple random trees, which help to identify data points that are easiest to isolate from the main cluster [5]. These algorithms are well-suited for analyzing large and complex data, such as GitHub activities.

However, deep learning algorithms also find applications in detecting more complex anomalies. Autoencoders and Generative Adversarial Networks (GANs) are capable of identifying complex, non-linear anomalies that are difficult to detect with traditional methods [7]. Autoencoders, for example, are trained to compress data into a latent space and then reconstruct them, allowing for the detection of anomalies based on the difference between the original data and their reconstructed versions [8]. This approach is particularly effective for analyzing complex and high-dimensional data typical of GitHub activity.

When selecting the optimal algorithm for anomaly detection, several factors must be considered. First, the characteristics of the data, such as volume, temporal dynamics, and noise, play a key role. For instance, for time-series data analysis, such as star activity, algorithms capable of accounting for temporal dependencies may be preferred. Second, the objective of the analysis also influences the choice of method: if the main goal is to detect all possible anomalies, preference may be given to algorithms with high sensitivity, even if this leads to an increase in false positives. Third, algorithm performance is crucial in cases where computational resources are limited or when large volumes of data need to be processed in real-time. In this context, simpler and faster methods may be more desirable.

To better understand how different algorithms can be applied to GitHub data and which are most effective in various scenarios, Table 1 is provided.

Table 1. Algorithms applied to GitHub data [5-8]

Algorithm	Sensitivity	Specificity	Performance	Interpretability
Isolation Forest	High	Medium	High	Medium
One-Class SVM	Medium	High	Medium	Low
Autoencoders	High	High	Low	Low
K-Means	Medium	Medium	High	Medium

This table allows for a visual assessment of which algorithm may be most suitable for a specific GitHub data analysis task, considering the objectives and constraints.

Thus, the methodology for detecting anomalies in GitHub data requires a comprehensive approach that takes into account the characteristics of the data, the goals of the analysis, and the requirements of the algorithms. The correct choice of machine learning methods not only automates the anomaly detection process but also significantly improves the accuracy and reliability of the results, which is especially important in the context of evaluating technology projects and making investment decisions.

2. Analysis of machine learning algorithms for anomaly detection

The analysis of machine learning algorithms for detecting anomalies in GitHub data requires a thorough examination to provide practical guidance on using advanced methods in real-world data conditions. This section focuses on the technical aspects and details of applying various algorithms, emphasizing their adaptation to the specific characteristics of GitHub data.

Unsupervised learning algorithms, such as Isolation Forest and One-Class SVM, play a crucial role in detecting anomalies in datasets where it is difficult to predefine what constitutes normal or deviant behavior. Isolation Forest is particularly effective for handling high-dimensional data and can be adapted to work with large datasets typical of GitHub activities. The algorithm operates by building trees that purposefully isolate data points, assessing how easy or difficult it is to separate them from the main cluster [5]. In the context of GitHub data analysis, this means that the algorithm can detect repositories or activities that significantly deviate from typical behavior patterns, such as sudden spikes in stars or anomalous commit patterns.

One-Class SVM employs the principle of data separation in multidimensional space, constructing a hyperplane that distinguishes normal data from anomalies. This method can be especially useful in cases where anomalies represent subtle deviations from the norm, which are difficult to detect using simpler methods [6]. However, its effectiveness heavily depends on kernel and regularization parameter selection, requiring fine-tuning for each specific dataset. In the context of GitHub data, this means that One-Class SVM can be used to detect less obvious manipulations, such as a systematic increase in stars over a prolonged period.

Deep learning, with its ability to handle complex and nonlinear dependencies, offers more powerful tools for detecting anomalies in GitHub data. Autoencoders are a type of neural network that learns to reconstruct original data from its compressed representation. During training, the autoencoder forms a latent space where normal data is reconstructed with minimal errors, while anomalies cause significant discrepancies between the original and reconstructed data [8]. For GitHub data analysis, autoencoders can be useful for detecting complex patterns that simpler algorithms might miss. For instance, an autoencoder can detect a repository that artificially maintains high activity levels through constant minor code changes.

Generative Adversarial Networks (GANs) represent another deep learning method that can be adapted for anomaly detection. In GANs, two neural networks—the generator and the discriminator—compete with each other: the generator attempts to create synthetic data that looks like the real data, while the discriminator tries to distinguish the synthetic data from the real [7]. In the context of anomaly detection, the discriminator can be trained on real GitHub data to detect artificially created activities, such as fake stars or anomalous fork patterns. Despite their power, GANs require significant computational resources and can be challenging to configure and interpret, limiting their use in real-world applications.

To better understand algorithm behavior and effectiveness in analyzing GitHub data, visualizing the results of their operation is particularly useful. For example, visualizing data distributions before and after applying anomaly detection algorithms can show how the algorithms isolate anomalous data. Time series graphs that display activity in GitHub repositories can also help identify anomalies, such as sudden spikes in activity that may be hidden in large volumes of data.

For illustration, the following types of activity distribution charts can be used to show normal and anomalous patterns:

1. Star distribution over time: This chart can display how the number of stars changes over time. Anomalous spikes that are not supported by corresponding activity in other metrics may indicate manipulation.

2. Commit density chart: This chart shows how frequently commits occur over a specific period. An abnormally high commit density, especially with minimal changes, may indicate artificially generated activity.

3. Repository activity heatmap: A heatmap can visualize activity by days and hours, revealing anomalous patterns such as excessive activity during non-working hours or uneven distribution of activity.

These visualizations help to better understand algorithm behavior and allow users to interpret the results by providing clear evidence of anomalies. Thus, the analysis of machine learning algorithms for detecting anomalies in GitHub data requires not only selecting and configuring appropriate methods but also using visualization to gain deeper insights into the data and their interpretation.

3. Conceptual model of the anomaly detection system for project evaluation

The conceptual model of the DualSpace anomaly detection system for project evaluation, based on GitHub data, represents a multi-layered architecture that integrates various machine learning algorithms for automated analysis and interpretation of data. This model is designed to ensure transparency and reliability in the evaluation of technology projects, which is particularly important for investors, developers, and analysts who rely on GitHub data as an indicator of project quality and popularity.

It is essential to emphasize that this system accounts for the diversity and complexity of GitHub data. Repositories on this platform contain multiple types of data, including activity metrics (such as the number of stars, forks, commits) and metadata about users and events. These data exhibit temporal dependencies and may be distributed unevenly, which necessitates the use of advanced processing and analysis methods.

At the first stage, data collection is performed using the GitHub API. This process is automated, allowing the system to regularly update information about repositories, users, and events. The data includes activity metrics, such as the number of stars, forks, commits, and pull requests, as well as metadata about users, their activities, and event timestamps. These data arrive in an unstructured form, and preprocessing them plays a crucial role in ensuring the accuracy of subsequent analysis.

```
import requests

# Example of retrieving repository data using the GitHub API
url = "https://api.github.com/repos/username/repository"
response = requests.get(url)
data = response.json()

# Extracting necessary metrics
```

```
stars = data['stargazers_count']
forks = data['forks_count']
commits_url = data['commits_url']

print(f"Stars: {stars}, Forks: {forks}")
```

This code illustrates a simple request to the GitHub API to obtain basic information about a repository, such as the number of stars and forks. In the real system, the requests are more complex and involve processing large volumes of data from numerous repositories.

The next step involves data preprocessing, which may include data cleaning, normalization, and dimensionality reduction. These steps are essential for preparing the data for more complex analysis, such as anomaly detection. A key aspect at this stage is the removal of noise, which can distort the analysis results.

Once the data is prepared, it is passed to the anomaly detector, where various machine learning algorithms are applied. It is important for the system to be flexible and support different analysis methods depending on the nature of the data. For example, to analyze time series of activity, recurrent neural networks (RNNs) can be used, as they account for temporal dependencies and can detect anomalies that manifest in changes in activity patterns.

```
from sklearn.ensemble import IsolationForest

# Example of using Isolation Forest to detect anomalies
model = IsolationForest(n_estimators=100, contamination=0.01)
model.fit(training_data)

# Predicting anomalies
anomalies = model.predict(test_data)
```

In this example, the code demonstrates the use of the Isolation Forest algorithm to detect anomalies in the data. This method is well-suited for working with large datasets where anomalies may be rare and difficult to detect using traditional methods [9].

After detecting anomalies, the system moves to the stage of result interpretation. It is important not only to detect deviations but also to provide users with context that helps them understand the causes and consequences of these anomalies. This may include data visualizations, such as activity distribution graphs, time series charts, or heatmaps, which visually show when and how the anomalies occurred.

```
import matplotlib.pyplot as plt

# Example of creating an activity distribution chart
plt.plot(dates, star_counts)
plt.title("Star Distribution Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Stars")
plt.show()
```

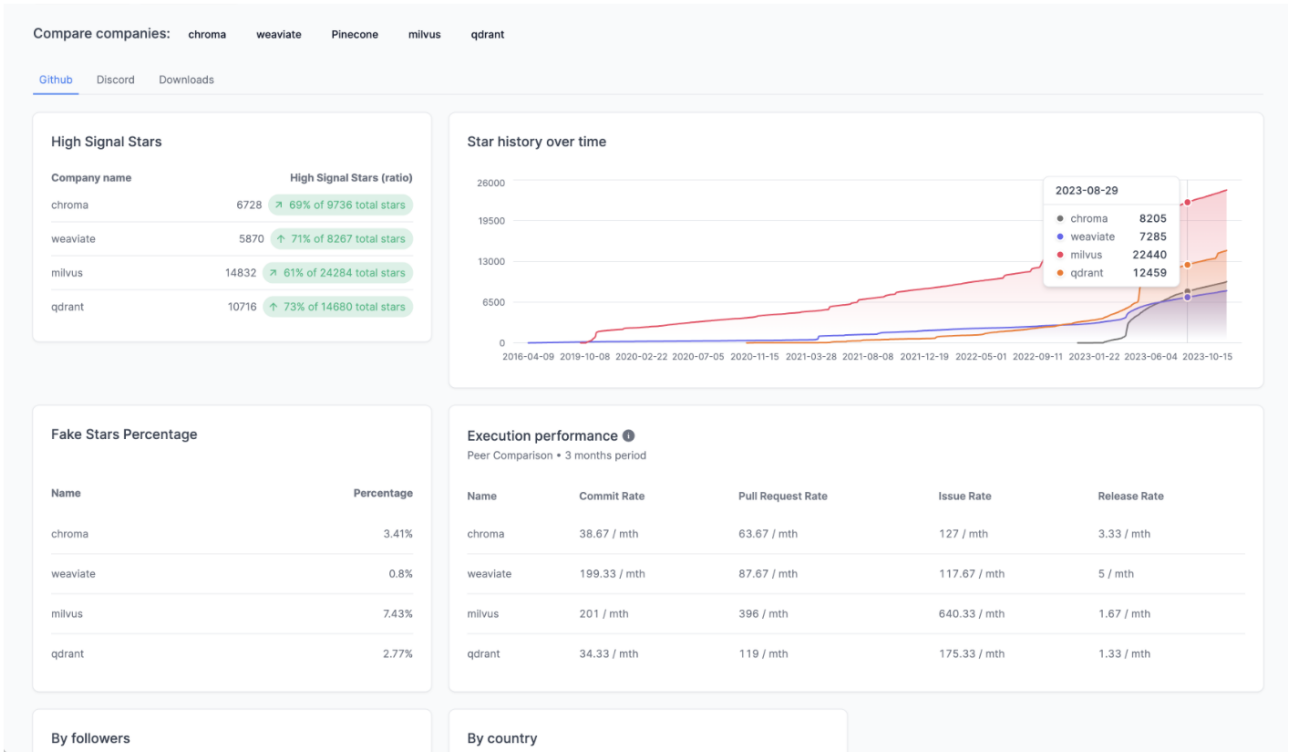

Figure 1 – Number of Stars in the Repository Over Time

This graph shows how the number of stars in a repository changed over time and allows users to visually assess possible anomalies, such as sudden spikes in activity [10].

Finally, the system generates comprehensive reports that include information on the detected anomalies, analysis of their causes, and recommendations for further actions. These reports can be used to assess project reputation, make investment decisions, or for internal analysis by developers.

Thus, the proposed conceptual model represents a comprehensive solution for the automated analysis of GitHub data. It enables the detection and interpretation of anomalies, which contributes

to improving the transparency and reliability of technology project evaluation. Integrating systems like DualSpace can significantly enhance decision-making processes in the technology sector, providing more accurate and informative data for all stakeholders.

CONCLUSION

In conclusion, the conducted research has demonstrated that using machine learning algorithms to detect anomalies in GitHub data is not only feasible but also an essential tool for ensuring transparency and objectivity in evaluating technology projects. GitHub, as a central platform for hosting and collaborative software development, provides a vast amount of data that can serve as an indicator of project success and popularity. However, manipulations of this data, such as inflating stars or artificially generating activity, can distort the real picture, misleading investors, developers, and other stakeholders.

The methodology proposed in this article covers the entire anomaly detection process, from data preprocessing to interpreting results using visualizations and generating reports. The use of unsupervised learning algorithms, such as Isolation Forest and One-Class SVM, as well as deep learning methods like autoencoders and GANs, ensures high accuracy in anomaly detection, allowing both obvious and hidden manipulations to be identified.

The developed conceptual model of the anomaly detection system offers a flexible and scalable solution that can be adapted to various conditions and data types. It takes into account the specifics of GitHub data, their temporal dependencies, and uneven distribution, making it particularly effective for analyzing large and complex datasets.

Integrating such a system into technology project evaluation processes, as demonstrated by DualSpace, can significantly improve decision-making quality by providing more reliable and accurate data. This, in turn, enhances transparency in the technology market, strengthens trust between investors and developers, and ultimately promotes a fairer distribution of resources.

Thus, the presented research and proposed methodology highlight the significance and potential of applying machine learning in the analysis of GitHub data, opening new opportunities for the development of automated systems for evaluating technology projects.

REFERENCES

1. Invest Boldly with the Power of Data. URL: https://www.dualspace.ai/
2. Github. URL: https://github.com/
3. Aneja N., Aneja S. Detecting fake news with machine learning //Conference Proceedings of ICDLAIR2019. – Springer International Publishing, 2021. – С. 53-64.
4. Dozmorov M. G. GitHub statistics as a measure of the impact of open-source bioinformatics software //Frontiers in bioengineering and biotechnology. – 2018. – Т. 6. – С. 436207.
5. Liu F. T., Ting K. M., Zhou Z. H. Isolation forest //2008 eighth ieee international conference on data mining. – IEEE, 2008. – С. 413-422.
6. Binbusayyis A., Vaiyapuri T. Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM //Applied Intelligence. – 2021. – Т. 51. – №. 10. – С. 7094-7108.
7. Goodfellow I. et al. Generative adversarial nets //Advances in neural information processing systems. – 2014. – Т. 27.
8. Kingma D. P. Auto-encoding variational bayes //arXiv preprint arXiv:1312.6114. – 2013.
9. Chen T., Guestrin C. Xgboost: A scalable tree

boosting system //Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. – 2016. – С. 785-794.

10. Loshchilov I., Hutter F. Sgdr: Stochastic gradient descent with warm restarts //arXiv preprint arXiv:1608.03983. – 2016.