MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Big Data Analytics Lab [CSE 3145]

Detection of Anomalous Navigation Patterns in Wikipedia Clickstream

SUBMITTED
BY

Vikasan S Nayak	**230962012**
Dube Prajwal Manoj	**230962024**
Dhruva Goyal	**230962071**

Under the Guidance of

Dr. Anup Bhat B
Assistant Professor
School of Computer Science and Engineering
Manipal Institute of Technology
Manipal, India

July-November 2025

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Manipal
30/10/2025

CERTIFICATE

This is to certify that the project titled **Detection of Anomalous Navigation Patterns in Wikipedia Clickstream** is a record of the bonafide work done by **Vikasan S Nayak (230962012), Dube Prajwal Manoj (230962024), and Dhruva Goyal (230962071)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING) of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2025-2026.

Name and Signature of Examiners:

1. **Dr. Anup Bhat B, Assistant Professor, SCE**
2. **Dr. Ancilla Pinto, Assistant Professor, SCE**

Contents

CHAPTER 1: INTRODUCTION

This project applies Big Data Analytics (BDA) techniques using the PySpark framework to perform anomaly detection on large-scale Wikipedia clickstream dataset. Anomaly detection in web traffic is crucial for identifying unusual user behavior, potential security threats (e.g., bot activity). The analysis uses click-to-click navigation data to model normal traffic patterns and flag statistically and behaviorally abnormal events.

1.1 Objectives

- To preprocess and clean the raw Wikipedia clickstream data (handling missing values, irrelevant records, and data type issues).
- To build a data analytics pipeline in PySpark for feature engineering on the clickstream data (e.g., creating click counts, link-type encoding, and normalized click rates).
- To compare and implement two different anomaly detection models: a Statistical Outlier Detection model (using Z-score on click counts), a Topological model (identifying new navigation paths), and a K-Means Clustering model (for behavioral anomaly detection).
- To generate and present the lists of statistically, topologically, and behaviorally anomalous article pairs/articles from a test dataset.

1.2 Dataset Characteristics

The project utilizes the Wikipedia Clickstream Dataset for English Wikipedia (enwiki) for the months of July and August 2025.

- Dataset Source: Wikipedia Clickstream Data (e.g., clickstream-enwiki-2025-07.tsv, clickstream-enwiki-2025-08.tsv).
- Aim of the Current Project: To detect anomalous navigation patterns or article clicks using big data analytics tools (PySpark and its libraries). This involves identifying statistical outliers in click counts and behavioral anomalies based on article traffic features (in-degree, out-degree, click ratios).

Column Name	Domain Meaning	Type (Categorical/Numeric)	Feature/Target	Info from EDA (Month 1/July)
prev	Source of the click (e.g., previous article, other-search, other-empty)	Categorical (String)	Feature	Contains many unique values; cleaned by dropping 'other-other'
curr	Destination article of the click	Categorical (String)	Feature	The subject of all feature aggregation for clustering
type	Nature of the link (link, external, other)	Categorical (String)	Feature	Predominantly 'link' (approx. 22.9M), followed by 'external' (approx. 12.4M) and 'other' (approx. 790K)
n	Number of clicks in the month	Numeric (Integer)	Feature	Ranges from **10** to **167,194,874**

Table 1.1: Wikipedia Clickstream Dataset Characteristics

CHAPTER 2: METHODOLOGY

This chapter details the data analytics pipeline, from initial data ingestion to the development of the anomaly detection models. The entire process was executed using PySpark to manage the large dataset efficiently.

Fig. 2.1: Flow of data through various stages of the analytics pipeline

2.1 Preprocessing

The need for preprocessing stemmed from the raw TSV format, which lacked a schema and contained records irrelevant to meaningful inter-article traffic analysis.

- *Data Ingestion and Schema:* Raw TSV files for July (Month 1) and August (Month 2) 2025 were loaded into Spark DataFrames, with the schema inferred, resulting in 36,131,419 rows for Month 1. Columns were manually assigned: prev, curr, type, and n.
- *Data Cleaning:* A check confirmed the absence of null values in all columns. The rows where the prev column was 'other-other' were filtered out because they offered no meaningful context about the user's source, reducing the July dataset by 201,202 rows.
- *Storage Optimization:* The cleaned data was persisted to Parquet format. This was done to leverage columnar storage, ensuring faster read speeds for subsequent analytical queries, preserving the schema, and achieving higher data compression compared to the original TSV format. Additionally, Parquet is more efficient than TSV for big data analytics because it supports predicate pushdown (filtering data at the storage level before reading it into memory) and selective column reading, significantly reducing I/O and improving query performance on large datasets.

2.2 Feature Engineering

To prepare the data for modeling, the following feature transformations were applied:

- *Target Encoding:* The categorical *type* column was numerically encoded using *pyspark.sql.functions.when* to create *type_encoded* where-
    - link = 0

    - external = 1
    - other = 2

- *Click Normalization:* Due to the severe positive skew and large range of the raw click count n was highly skewed (minimum 10 and maximum 167M). To mitigate the effect of extreme outliers and achieve a more normal distribution, the log(1+x) transformation (*log1p* function) was applied to create the *normalized_clicks* column. This transformation mitigated the influence of extreme outliers and produced a more Gaussian-like distribution, suitable for statistical and distance-based models.

2.3 Anomaly Detection Methodology

The dataset was split, with the cleaned and engineered July data (Month 1) serving as the training/baseline data and the August data (Month 2) serving as the test data for anomaly detection.

2.3.1 Statistical Outlier Detection (Z-Score)

Statistical Outlier Detection using Z-score is a method to identify data points that are significantly different from the average of the dataset. The Z-score measures exactly how many standard deviations a specific data point is away from the dataset's mean.

In this project, this method was not used to analyze an article's overall behavior. Instead, it was applied row-by-row to find individual navigation paths (prev -> curr) where the traffic volume was a statistical anomaly.

A high Z-score on a path signifies that the number of clicks from *a* specific previous page to the specific current page is abnormally high (or low) compared to the normal click volume we learned from all paths in Month 1.

The process was split into a training phase (learning the "norm" from Month 1) and a testing phase (scoring Month 2 against that norm):

- *Training:* The mean and standard deviation of normalized_clicks were calculated from the Month 1 training set (mean=3.656, standard deviation =1.198). This established the normal traffic volume for any given path, giving a baseline for comparison.
- *Anomaly Calculation:* These baseline statistics (mean and stddev from Month 1) were then used to calculate a Z-score for every record in the Month 2 (August) test set. The z-score formula was applied using a *withColumn* transformation.

- *Threshold:* An event was classified as a statistical outlier if its absolute Z-score was greater than 3. This is a standard statistical threshold that identifies points outside of 99.7% of a normal distribution. This filtering operation identified **119,506** individual navigation paths in Month 2 as statistical outliers. The paths with the highest scores, like *other-empty->Main_Page*, were the most extreme volume anomalies.

2.3.2 Topological Anomaly Detection (New paths)

This is a graph-based method that identifies structural changes in the navigation network. Unlike the Z-score method, it is not concerned with traffic volume (how many clicks) but with the existence of a navigation path.

In this project, this method was used to find all brand-new navigation paths (*prev -> curr*) that appeared in Month 2 (August) but did not exist at all in Month 1 (July).

A topological anomaly is a path that has never been seen before. It represents a new connection in the Wikipedia graph, which could be caused by a new link being added to a page, a new search query, or a new pattern of user behavior.

In the context of this project, a path, or conceptually an edge, is the direct, one-way navigation link between two Wikipedia articles. The *prev* column is the source of the path (the page the user came from). The *curr* column is the destination of the path (the page the user landed on).

This method uses the Month 1 data to build a known map of all valid navigation paths and then compares the Month 2 map against it to find the differences.:

- *Training:* A DataFrame named *baseline_edges* was created by selecting the *prev* and *curr* columns from the Month 1 (July) training data and applying a *distinct()* operation.This resulted in a set of all unique, normal navigation paths that occurred in the baseline month, representing a graph.
- *Data Preparation:* The same process was applied to the Month 2 (August) test data to create *test_edges*, a distinct set of all unique paths that occurred in the test month making up the *current* edges.
- *Anomaly Calculation:* The core of this method is a *left_anti join*. The test_edges DataFrame (Month 2) was joined with *baseline_edges* (Month 1) using *how="left_anti"*.A left_anti join returns only the rows from the *test_edges* (left DataFrame) that do not have a

match in the *baseline_edges* (right DataFrame). The result, *topological_anomalies*, is a DataFrame containing every *prev-curr* pair that appeared in August but was not present in July.

- *Threshold:* By definition, every row returned by the *left_anti join* is a topological anomaly. No further thresholding is required. This process identified 1,141,450 new, previously unseen navigation paths in Month 2.

2.3.3 Behavioral Anomaly Detection (K-Means Clustering)

Behavioral Anomaly Detection is an unsupervised learning that identifies data points with unusual patterns of behavior.

In this project, *kmeans* method was applied to the articles, or conceptually the nodes in the graph, not individual paths. It works by grouping articles into clusters based on their overall traffic profiles. A traffic profile is a feature vector that describes an article's complete navigation behavior, its role as both a destination (how people arrive) and a source (where people go next).

An article is considered a behavioral anomaly if its traffic profile is very different from all normal profiles, which is measured as a large distance from its nearest cluster center.

- *Feature Aggregation:* First, the data was transformed from a list of paths to a feature vector for each unique article. This was done on Month 1 training data.
    - *Inbound features* (*in_agg*): The data was grouped by the *curr* column (the article) to calculate:
        - *in_degree*- The count of unique previous pages, using *countDistinct("prev")*.
        - *in_clicks*- The sum of *normalized_clicks* arriving at this article.
        - *in_events*- The count of inbound path records.
    - *Outbound features (out_agg*): The data was grouped by the *prev* column to calculate:

        - *out_degree*- The count of unique destination pages, using countDistinct("curr").
        - *out_clicks*- The sum of *normalized_clicks* leaving this article.
        - *out_events*- The count of outbound path records.
    - *Full Feature Vector:* A *full_outer join* combined *in_agg* and *out_agg* on the article name (*curr column*). This was crucial to keep all articles that were only sources or only destinations. A *fillna(0)* was applied to handle the nulls created by this join.
    - *Derived Ratio:* Using *withColumn*, three relational features were created to capture the behavior of the traffic:
        - *ratio_out_in- out_clicks* / (*in_clicks* + 1.0). A ratio that compares the total normalized click volume leaving an article to the total normalized click volume entering it. It defines whether an article is a source (value>1.0, more traffic flows out) or a destination (value<1.0) for traffic.
        - bounce_rate- 1-(out_clicks / (in_clicks + 1.0)). This measures what fraction of inbound traffic stops at this article, i.e., does not navigate to another Wikipedia article. High rate (near 1.0) is a *dead-end* page. Almost all traffic that arrives stops here. It has high *in_clicks* but very low *out_clicks*. Low (near 0.0) is a pass-through page and nearly all traffic that arrives also leaves. Negative Value is a strong source. A negative bounce rate simply means the *ratio_out_in* was greater than 1, so it is a net exporter of clicks.
        - in_out_event_ratio- in_events / (out_events + 1.0). This ratio compares the total number of inbound navigation paths to the total number of outbound navigation paths. High ratio (>> 1.0) is a *gathering point.* The article is a popular destination from many different sources (*in_events* is high), but users don't navigate from it very much (*out_events* is low). Low ratio (<< 1.0) a *distributor*. The article is a major starting point for navigation to many different pages (*out_events* is high), but it doesn't have many pages linking to it (*in_events* is low).

- *Feature Standardization:* K-Means is a distance-based algorithm, so features with large scales (such as *in_clicks*) can dominate features with small scales (like *bounce_rate*). Therefore, standardization was essential to bring all features into a comparable range.
    - *Vector Assembler*- A *VectorAssembler* was used to combine all 7 features (*in_degree, out_degree, in_clicks, out_clicks, ratio_out_in, bounce_rate, in_out_event_ratio*) into a single vector column named *features_raw*.
    - *Scaling*- A *StandardScaler* was fit on the *features_raw* column of the Month 1 data. This model learned the mean and standard deviation of each feature and was configured with *withMean*=True and *withStd*=True to transform the data, ensuring all features had a mean of 0 and a standard deviation of 1.
- *Model Training:* Learning the trends from Month 1.
    - *K-Means Training*- A *KMeans* model was trained on the scaled, vectorized Month 1 data. The model was then configured with k=10 clusters and a seed=42 for reproducibility.
    - *Saving the models*- The trained *StandardScalerModel* and *KMeansModel* were both saved to disk. This was the most critical step, as it froze the definition of normal behavior learned from Month 1, providing a model to compare with adjacent months to analyze behaviors.
- *Anomaly Scoring:* This is where the detection for Month 2 took place.
    - *Feature Aggregation*- The exact same 7-feature aggregation process for Month 1 was applied to the Month 2 test data.
    - *Loading Baseline Models*- The *StandardScalerModel* and *KMeansModel* that were trained on Month 1 were loaded from disk.
    - *Transform Test Data*- The Month 2 feature vectors were transformed using the loaded *StandardScalerModel*. This scaled the Month 2 data relative to Month 1.
    - *Predict Clusters*- The loaded *KMeansModel* was used to *transform()* the scaled Month 2 data. This assigned each Month 2 article to the closest cluster from Month 1.

    - *Calculate Anomaly Score*- A User-Defined Function (UDF) was created to calculate the squared Euclidean distance between an article's feature vector and the center of the cluster it was assigned to. This distance was saved as the final anomaly score, *distance_to_center*.

- *Thresholding and Filtering:* A simple top-10 list is not statistically robust to find anomalies. Therefore, a threshold was calculated from the Month 2 anomaly scores. The *mean* and *stddev* of the *distance_to_center* column were calculated. An article was classified as a formal behavioral anomaly if its score was greater than the- *mean* + (3 * *stddev*). This formula comes from the normally distribution of data, where 99.7% of the daat lies within 3σ, and hence anythiing beyond that may be classifed as an anolmaly to the dataset. This rigorous process filtered the entire set of articles down to the top 10 most significant behavioral anomalies.

CHAPTER 3: RESULTS AND ANALYSIS

The anomaly detection methodologies were executed on the Month 2 (August 2025) test data using the models and baselines obtained from the Month 1 (July 2025) data. Each method successfully identified a distinct set of anomalies, which were then synthesized to produce robust, high-confidence outcomes.

3.1 Statistical Outlier Analysis (Z-Score)

This method identified 119,506 individual navigation paths as statistical outliers, meaning their *normalized_clicks* volume was more than 3 standard deviations from the norm established in Month 1. These represent paths with extreme and unusual traffic spikes.

- Training Statistics (Month 1/July)-
    - Mean: 3.656
    - Standard Deviation: 1.198
- Outlier Count (Month 2/August)-
    - Total Statistical Outliers (|Z| > 3): 119,506

The top 10 most anomalous paths (by highest Z-score) were:

prev	*curr*	*Z-Score*
other-empty	Main_Page	12.745
other-search	Coolie_(2025_film)	9.783
other-empty	Deaths_in_2025	8.950
other-search	Saiyaara	8.893
other-search	Main_page	8.781
other-search	Amanda_Knox	8.698
other-external	Utrecht	8.669
other-search	Frank_Caprio	8.426
other-search	My_Oxford_Year	8.424
other-search	Terence_Stamp	8.396

Table 3.1.1: Top 10 Statistical Anomalies

3.2 Topological Anomalies (New Paths)

This method identified 1,141,450 new navigation paths. These are paths that existed in the Month 2 data but had no precedent in the entire Month 1 dataset, indicating that there is a structural change in the navigation graph (which can be thought of as a tree with articles).

While this number is large due to the changing trends and interests of users in the subsequent month, analyzing the destinations of these new paths is highly revealing. The *Main_Page* was, by far, the most common destination for new, previously unseen navigation paths.

curr	*New Path Count*
Main_Page	7,105
United_States	252
YouTube	230
A	203
World_War_II	160
13	151
New_York_City	146
Weapons_(2025_film	134
United_Kingdom	134
Soviet_Union	133

Table 3.2.1: Top 10 Statistical Anomalies

3.3 Behavioral Anomaly Analysis (K-Means)

This method identified 10 articles as significant behavioral anomalies. These are articles whose traffic profiles (the 7-feature vector) were more than 3 standard deviations from their assigned cluster's mean distance.

These 10 articles represent the most fundamentally unusual *actors* in the Wikipedia network for Month 2, hinting at a bot behavior.

- Distance Statistics (Month 2):
    - Mean Distance : 0.531
    - Standard Deviation : 299.034
    - Anomaly Threshold Mean Distance + 3 x Standard Deviation= 897.633
- Anomaly Count (Month 2):
    - Total Formal Behavioral Anomalies (Distance > Threshold): 10

curr	*Anomaly Score (Distance)*
Main_Page	433,915.1
other-search	248,994.4
other-empty	149,393.2
A	79,571.3
other-internal	34,633.2
Issue_(genealogy)	13,659.2
other-external	8,568.0
Birth_name	1,379.8
Incumbent	1,001.1
Killed_in_action	910.9

Table 3.3.1: Top 10 Behavioral Anomalies by Anomaly Score

3.4 Synthesis: High-Confidence Anomaly Detection

The true analytical power of this project comes from synthesizing the results of the three independent methods. While the Statistical and Topological methods flagged over a million paths, the Behavioral method found only 10 articles. The overlap between these sets provides a high-confidence list of true anomalies.

3.4.1 Overlap Analysis

- Flagged by All 3 Methods- 5 articles
- Flagged by Statistical and Behavioral- 5 articles
- Flagged by Topological AND Behavioral: 6 articles
- Flagged by Statistical AND Topological: 56,913 articles

The key finding from this synthesis is that the K-Means model, used to study behavior, was extremely effective. All 10 articles it flagged were also flagged by at least one other method, and 5 of them were flagged by all three, confirming the anomalous nature of the 3.

By joining the results from the Statistical and Behavioral models, we can see why these 5 articles are so anomalous. The K-Means (Behavioral) model tells us the "WHAT"- "This article's overall traffic profile is weird." The Z-Score (Statistical) model tells us the "WHERE"- "This specific path leading to this article had a massive volume spike."

article	*Behavioral Anomaly Score (Distance)*	*Statistical Anomaly (Avg Z-Score)*	*Statistical Anomaly (Hot Path Count)*
Main_Page	433,915.1	4.12	20
A	79,571.3	7.18	1
Issue_(genealogy)	13,659.2	3.46	1
Incumbent	1,001.1	3.24	1
Killed_in_action	910.9	3.52	2

Table 3.4.1: High-Confidence Anomalies (Flagged by Statistical & Behavioral Methods

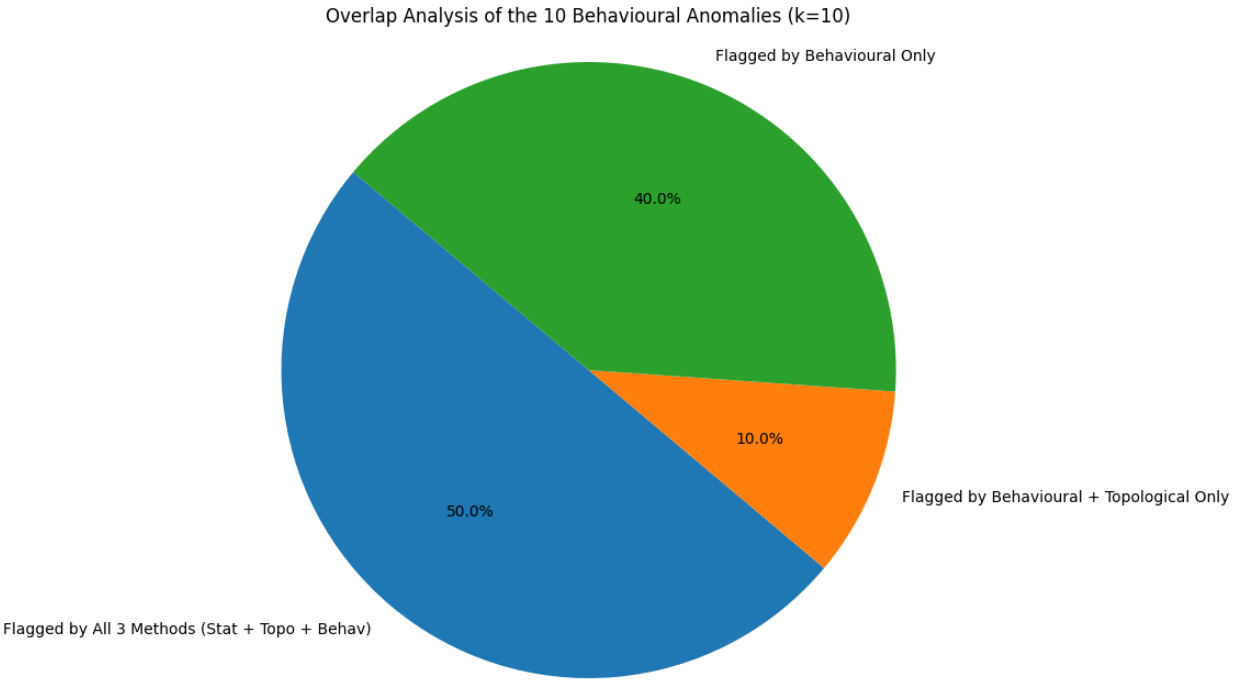

Figure 3.4.1: overlap breakdown for 10 behavioral anomalies

3.4.2 Interpretation of Results

Main_Page: This is unambiguously the most anomalous article. It has the highest possible behavioral anomaly score, meaning its traffic profile is completely unique and far from any normal cluster. It was also the destination for over 7,000 new paths and was involved in 20 different paths with statistically extreme traffic spikes (average Z-score of 4.12). The model correctly identified it as a statistical outlier, not a malicious actor. Its anomaly score confirms that it is a unique hub. This indicates a general, sitewide increase in users returning to the *Main_Page* from various

sources in August compared to July. This could be due to a UI change, a popular event, or a new search engine behavior.

A: This is the second most anomalous article. It has an extremely high behavioral anomaly score (79,571) and was also flagged by the topological method. Most importantly, the statistical method found that its anomalous path had an average Z-score of 7.18. This article didn't just have high traffic; it had a singular, massive spike on one path.

Issue_(genealogy), Incumbent), Killed_in_action: These three articles represent a clear signal of anomaly. All three were flagged by all methods, showing they have a unique behavioral profile, were the destination for new paths, and experienced at least one high-volume (Z-score > 3) traffic spike. These are proper anomalies that are very different from normal user behavior. This strongly suggests these articles were targeted by automated activity (bots) or experienced a sudden, non-human surge in traffic in August.

CHAPTER 4: CONCLUSION

4.1 Summary of Findings

This project successfully implemented and evaluated a multi-stage Big Data Analytics pipeline in PySpark to detect anomalous navigation patterns in the Wikipedia clickstream data. By using the July 2025 dataset as a baseline and the August 2025 dataset as a testbed, three distinct types of anomalies were identified:

1. Statistical (Z-Score): Flagged 119,506 individual navigation paths that exhibited extreme traffic volume *(normalized_clicks)* compared to the norm.
2. Topological (New Paths): Flagged 1,141,450 new paths that existed in August but had no precedent in the July data, indicating a structural change in the graph.
3. Behavioral (K-Means): Flagged 10 individual articles whose overall traffic profiles (a 7-feature vector) were statistically distant from all normal behavioral clusters learned from the baseline data.

The project's key result was the convergence of these three models and the analysis of results obtained from behavioral and statistical models. The 10 articles flagged by the behavioral model were all confirmed by at least one other method. Most importantly, 5 articles *(Main_Page, A, Issue_(genealogy), Incumbent, Killed_in_action)* were flagged by all three methods, confirming them as the highest-confidence anomalies in the dataset.

4.2 Interpretation of High-Confidence Anomalies

A core objective was to identify potential bot activity. The results show that our models flagged two different types of anomalies: expected functional outliers and suspicious behavioral outliers.

4.2.1 Expected Anomaly (Functional Outlier): *Main_Page*

The *Main_Page* was correctly identified as the number 1 anomaly, which serves as a successful validation of the K-Means model. *Main_Page* is not a bot but a known anomaly or a unique entity.

Its behavioral profile (highest anomaly score of 433,915.1) is functionally unique and unlike any other article. Its 7,105 new topological paths reflect the site's dynamic nature (new articles, new search referrers) rather than some malicious activity. The model correctly isolated *Main_Page* proving its ability to find unique entities.

4.2.2 Suspicious Anomalies (Potential Bot Activity): A, Issue_(genealogy), etc.

These articles represent the true, suspicious findings of the project. Unlike *Main_Page*, they are not expected to have unique, high-traffic profiles.

Article A: This article had the 2nd highest behavioral anomaly score (79,571.3).It had an average statistical Z-score of 7.18. A Z-score of this magnitude is not indicative of human trend but is a classic example of a click bot, which is an automated script targeting a single path with extreme, repetitive volume.

Issue_(genealogy), Incumbent, Killed_in_action: These articles followed the same pattern as *A*. All were flagged by all three methods and showed a combination of a unique behavioral profile and a high statistical Z-score (>3.0).

This combination of a unique profile and a huge, non-human Z-score makes these articles prime candidates for bot-like activity and demands further investigation by the website developers.

4.3 Limitations and Future work

4.3.1 Limitations

- The analysis was limited to two months of data. This short window cannot model long-term seasonality, meaning some anomalies could be normal annual events.

4.3.2 Future Work

- Expanding the analysis to a full 12+ months to build a more robust model of normal behavior that accounts for seasonality.
- Adding temporal features (such as hour-of-day) to the behavioral profile to better distinguish human from bot activity.

4.4 Insights

By synthesizing the results of statistical, topological, and behavioral models, this project demonstrated an effective method for filtering over a million potential anomalies down to a small, high-confidence list of anomalies. This approach successfully identified non-obvious anomalous articles (like A) and provided strong evidence for interpreting their behavior as programmatic bot activity, fulfilling the project's core objectives.

REFERENCES

1. Wikimedia Foundation, "Clickstream data," 2025. [Online]. Available: https://dumps.wikimedia.org/other/clickstream/
2. Apache Software Foundation, "Apache Spark: A unified analytics engine for large-scale data processing," 2025. [Online]. Available: https://spark.apache.org/
3. Wikimedia Foundation, "Research:Wikipedia clickstream," meta.wikimedia.org, 2024. [Online]. Available: https://meta.wikimedia.org/wiki/Research:Wikipedia_clickstream.
4. A. Gupta, "Handling Outliers using Z-Score | Explained," Kaggle, 2022. [Online]. Available: https://www.kaggle.com/code/everydaycodings/handling-outliers-using-z-score-explained/notebook