Bot Detection

Initial Criteria:

1. Post frequency (daily and yearly quantiles )
2. Time between posts (if user is posting more than 1 post within 60 secs)
3. Post timestamp trend by individual users (if user is always posting at the same time)
4. Near duplicates Posts, Comments (Reddit only) (>90% sim) (not retweets or reshares)
5. Frequency of Engagement (Define Threshold for each platform.)

Model 1: (Multimodal Semi-Supervised)

# 1. User Filtering

Keep only users with at least 2 posts. Users with fewer posts do not exhibit enough behavioral patterns for reliable detection.

# 2. Feature Extraction (User Behavior Analysis)

Extract features across three dimensions: Content, Temporal, Behavioral and Graph Features.

# Content Features

- Text Length (Characters per post)  
- Word Count (Words per post)  
- Text Entropy (Randomness of characters)  
- Lexical Diversity (Unique words ÷ Total words)  
- Hashtag Count  
- Mention Count (@user)  
- URL Count  
- Average Content Similarity (Cosine Similarity using TF-IDF)  
- Near-Duplicate Indicator (Posts >90% text similarity)

# Temporal Features

- Posting Hours, Days  
- Time Between Posts (Average, Median, Standard Deviation)  
- Coefficient of Variation  
- Posting Time Entropy  
- Work Hours / Sleep Hours / Weekend Flags

# Behavioral and Graph Features

- Engagement Ratios (likes, reposts, replies)  
- Sentiment Analysis (happy, sad, neutral ratios)  
- Posting Regularity – Lower entropy = more regular (bot-like)  
- Benford’s Law Deviation:

| **User** | **Likes on 5 Posts** |
| --- | --- |
| **User A** | **[123, 156, 198, 102, 145]** |
| **User B** | **[800, 850, 890, 810, 870]** |
|  |  |
| **User** | **First Digits** |
| User A | : [1, 1, 1, 1, 1] |
| User B | : [8, 8, 8, 8, 8] |
|  |  |

**User A:**

* All likes start with 1.
* 1 appears 100% of the time.
* 1 is supposed to appear often (~30%),  
  **but** not always.
* Still, it's not super suspicious if it's only 5 posts.

**User B:**

* All likes start with 8.
* 8 appears 100% of the time.
* 8 should only appear **5%**, but here it's 100%.
* **Very suspicious**

| **First Digit** | **Expected Probability (%)** |
| --- | --- |
| 1 | 30.1% |
| 2 | 17.6% |
| 3 | 12.5% |
| 4 | 9.7% |
| 5 | 7.9% |
| 6 | 6.7% |
| 7 | 5.8% |
| 8 | 5.1% |
| 9 | 4.6% |

Computes a chi-square deviation score between observed and expected distributions

- Posting Bursts – Rapid posts in short duration of time.  
- Graph-Based Metrics (In-degree, Out-degree, Clustering Coefficient, Community Size)

In-degree: Number of **incoming edges** to a node (user) in the graph. How many users are mentioning, replying to, reposting from, or interacting with this user.

| **Observation** | **Interpretation** |
| --- | --- |
| Very high in-degree | Could be a celebrity or a central figure. |
| Very low in-degree | Normal user or isolated bot. |

# 3. Content Similarity Detection

- Use TF-IDF vectorizer on post texts.  
- Measure pairwise Cosine Similarity (> 0.9 indicates near-duplicate).  
- Features extracted are used as input for ML, NOT for direct rule-based bot detection.

# 4. Unsupervised Pseudo-Labeling (Anomaly Detection Stage)

Use these methods to generate pseudo-labels:  
- Isolation Forest (n\_estimators=100, contamination=0.1)  
- DBSCAN (eps=0.5, min\_samples=5)  
- KMeans (k=2–8 depending on data)  
- PCA + Mahalanobis Distance Outliers

Using multiple unsupervised models = Catches more types of bots = More accurate pseudo-labels.  
Pseudo-label = 1 (likely bot) if flagged by any method.

# 5. Supervised Learning (Validation of Pseudo-Labels)

Train ExtraTreesClassifier with:  
- n\_estimators=100  
- max\_depth=10  
- min\_samples\_split=5  
- class\_weight='balanced'  
Train on engineered features, predict bot probability per post.

# 6. Final Bot Classification Decision

Final bot decision is based purely on supervised ML output:  
- Bot if bot\_probability > optimal threshold (from Precision-Recall curve based on F1 score)  
- No hardcoded threshold decisions.

# 7. Aggregation at Author Level

Aggregate post-level results to author-level:  
- Average bot probability  
- Confidence based on number of posts  
- Assign final bot label to authors.

# 8. Visualization and Reporting

- Bot Probability Distribution Histogram  
- Feature Importance Chart  
- Posting Activity Heatmaps  
- Bot vs Human Feature Distribution Plots

# Hyperparameters:

- Isolation Forest: n\_estimators=100, contamination=0.1  
- DBSCAN: eps=0.5, min\_samples=5  
- KMeans: n\_clusters=2–8  
- PCA: Mahalanobis threshold = mean + 3\*std  
- ExtraTreesClassifier: n\_estimators=100, max\_depth=10  
- Content Similarity Threshold: >0.9

# Research Papers Referenced

1. **Multimodal Twitter Bot Detection Framework -** <https://link.springer.com/article/10.1007/s13278-025-01435-w>

**2.** **Bot Detection Using a Mixture of Experts -** <https://ojs.aaai.org/index.php/ICWSM/article/view/22179>

**3.** **Unsupervised Detection of Coordinated Fake-Follower Campaigns** - <https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-024-00499-6>

- BotDGT (2024): Temporal behavioral bot patterns  
- ETS-MM (2025): Multi-stage bot detection framework  
- BotShape (2023): Temporal posting shape detection  
- Structural Information Theory (2024): Network-based bot detection  
Eg: DeBot, UnDBot, BotWalk are some semi supervised learning methods for bot detection.

(GMAE2-CGNN) Model-2

# 1.Data preprocessing: At least two posts required.

# 2. Feature Extraction

Features generated at the post level:  
- Textual Features: Post text length, word count, text entropy.  
- Sentiment Features: Ratios of happy, sad, and neutral posts.  
- Engagement Features: Reply count, repost count, like count.  
- Temporal Features: Hour of posting, day of week, posting regularity.  
- Feature Normalization: Standardized (zero mean, unit variance).

# 3. Graph Construction

- Nodes: Each user is represented as a node.  
- Edges:  
 - Created if one user mentions another or if posts are highly textually similar.  
- Graph Matrix: Sparse adjacency matrix.  
- Important Note: Only mentions and text similarity determine edges. Other features are processed separately by the feature encoder.

# 4. Model Architecture

The model has two components:  
  
a. Feature Encoder (Multi-Layer Neural Network)  
- Input Layer: Number of engineered features (~30–40).  
- Hidden Layers: 128 → 64 → 32 units (BatchNorm, LeakyReLU, Dropout 0.2 after each).  
- Output: 32-dimensional user feature embedding.  
  
b. Graph Encoder (Simple Graph Neural Network)  
- First Linear Layer: 32 → 32 units, ReLU activation.  
- Graph Aggregation: Neighborhood information aggregation.  
- Second Linear Layer: 32 → 16 units, ReLU activation.

# 5. Training Methodology (Self-Supervised Learning)

- No manual labels are used.  
- Contrastive learning objective:  
 - Connected users have similar embeddings.  
 - Disconnected users have different embeddings.  
- This method is self-supervised, learning purely from graph structure and feature similarity.

# 6. Bot Detection (Post-Training)

After training:  
- Extract final user embeddings (16-dimensional).  
- Apply unsupervised clustering (KMeans).  
- Identify bot-like users based on cluster separation.  
- Final classification is completely unsupervised.

# 7. Final Output

Each user receives:  
- A low-dimensional embedding vector.  
- A cluster assignment.  
- A bot likelihood score based on clustering position and isolation.

# Hyperparameters Summary

- Feature Encoder Hidden Layers: [128, 64, 32]  
- Dropout Rate: 0.2  
- Graph Encoder Hidden Dim: 32  
- Graph Encoder Output Dim: 16  
- Optimizer: Adam  
- Learning Rate: 0.001  
- Batch Size: 64  
- Epochs: 30–50  
- Max Users Sampled: 10,000  
- Max Posts per User: 50  
- Clustering Algorithm: KMeans

# Research Paper References

- Unsupervised Bot Detection via Graph and Behavior features - <https://arxiv.org/abs/2404.13595>  
- GMAE2

# Important Clarification

- CGNN (Conditional Graph Neural Networks) are originally supervised.  
- In this project, CGNN is modified for self-supervised learning using contrastive loss without ground truth labels.  
- The final bot detection is fully unsupervised after clustering the embeddings.

# Final Summary

This project combines user behavior features and user connection graphs, applies self-supervised contrastive learning to generate embeddings, and detects bots through unsupervised clustering without requiring labeled data.