Combating Social Bots with Machine Learning

Willy Chan
CS 109 Challenge Submission
Stanford University
Stanford, CA
`willyc@stanford.edu`

1 Introduction

This project addresses the pressing concern of identifying social bots on digital platforms, which is paramount in an age rife with digital misinformation. A logistic regression model was crafted, utilizing a dataset that includes user age, post count, average daily activity, and bot likelihood scores. This model aims to identify patterns suggestive of bot-like behavior. A standard logistic regression approach was adopted, focusing on the direct relationship between the provided user metrics and their likelihood of being bots. The model's insights underline the complex interplay between user engagement metrics and bot likelihood, contributing to the broader challenge on the influence of social bots in shaping the social media ecosystem.

2 Background

The dataset, originating from Indiana University researchers (Shao et al.), examines the influence of social bots in disseminating low-credibility content on social media. Grounded in academic rigor and detailed methodology, it aims to unravel the dynamics of bot activities and their effect on public opinion and online conversations. The dataset includes crucial metrics such as user age, posting frequency, and a bot likelihood score, providing a detailed view of user behavior patterns and the tactics employed by such accounts to spread potentially harmful content.

The data focuses on user interactions across social media platforms, distinguishing between human users and social bots. Key attributes include:

- Age (days): Reflecting the account's longevity in days.
- Count: Total number of user posts.
- Activity: Daily average posting rate.
- Bot Score (English): A scale from 0 to 1, predicting the likelihood of an account being a bot, with higher scores indicating a greater bot probability.

3 Model Development and Data Preparation

The dataset was divided into training and test subsets, following the standard 80:20 split. The training set encompassed 8,952 samples, while the test set included 2,238 samples, each characterized by three features: age, count, and activity. Given that Bot Score represents a likelihood, I converted it into a binary classification problem by selecting the common threshold of 0.5.

Preprint. Under review.

4 Logistic Regression Model

4.1 Approach and Probability Concepts

The logistic regression model is a fundamental tool in statistical classification, especially suitable for binary classification tasks like our bot detection problem. The core idea of logistic regression is to model the probability that a given input belongs to a certain class (in this case, a bot or not a bot).

4.2 Model Formulation

The logistic regression model calculates probabilities using the logistic function, which is an S-shaped curve that maps any real-valued number into a value between 0 and 1. For a set of features $x = (x_1, x_2, ..., x_n)$ and corresponding weights $w = (w_1, w_2, ..., w_n)$, the probability that x belongs to the positive (bot) class is given by $p(y = 1|x) = \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+...+w_nx_n)}}$, where e is the base of the natural log and w_0 is the bias term.

4.3 Parameter Estimation

To estimate the parameters w from the data, we use the method of maximum likelihood estimation (MLE). MLE seeks to find the parameter values that maximize the likelihood of observing the given data. In logistic regression, this translates to maximizing the product of probabilities assigned to the actual outcomes across all data points.

Given a dataset with m samples $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$ where $y^{(i)}$ is the actual class of the i-th sample, the likelihood L is $L(w) = \prod_{i=1}^{m} P(y^{(i)}|x^{(i)})$. Finding the argmax of $L(w)$ is equivalent to maximizing its log-likelihood, which is more computationally convienient:
$LL(w) = \sum_{i=1}^{m} [y^{(i)} \log(P(y = 1|x^{(i)}) + (1 - y^{(i)}) \log(1 - P(y = 1|x^{(i)}))]$

4.4 Gradient Descent for Optimization

To find the weights that maximize the log-likelihood, we can use gradient descent, an iterative optimization algorithm. Starting with an initial guess for w, we iteratively adjust this guess in the direction that increases the log-likelihood. The update rule at each iteration k is $w^{k+1} = w^k + \alpha \Delta LL(w^k)$, where α is the learning rate and $\Delta LL(w^k)$ is the gradient of the log-likelihood with respect to w at iteration k.

5 Classification

5.1 Coding the Model

The logistic regression model was implemented in Python without relying on predefined libraries like scikit-learn. The main steps in the implementation included:

Initialization: The model initializes weights and bias to zero. These parameters are crucial for predicting the probability of a user being a bot.

```
class LogisticRegressionClass:
    def __init__(self, learning_rate=0.01, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None
```

Training: the model was trained using the training subset of the data. In each iteration, the weights were updated in the direction that increases the log-likelihood. The sigmoid function is applied to transform the linear combination of inputs and weights into a probability between 0 and 1. The end of the training method also employs gradient descent for optimizing the model's weights. This iterative algorithm adjusts weights to maximize the log-likelihood of observing the training data under the model. For classification, the model calculates the probability of being a bot for each user. If this probability is above 0.5, the user is classified as a bot; otherwise, as a non-bot.

```
def train(self, X, y):
    # Initialize parameters
    num_samples, num_features = X.shape
    self.weights = np.zeros(num_features)
    self.bias = 0
    # Gradient descent
    for _ in range(self.num_iterations):
        model = np.dot(X, self.weights) + self.bias
        predictions = self.sigmoid(model)

        # Compute gradients
        dw = (1 / num_samples) * np.dot(X.T, (predictions - y))
        db = (1 / num_samples) * np.sum(predictions - y)
        # Update weights
        self.weights -= self.learning_rate * dw
        self.bias -= self.learning_rate * db
```

5.2 Results and Discussion

5.3 Conclusion

In conclusion, the logistic regression model developed for identifying social bots achieved a notable accuracy of approximately 88.89% on the testing dataset, primarily excelling in recognizing non-bot users. However, its performance in accurately detecting bots was limited, underscoring the need for further model refinement.

The final weights of the model provide key insights:

- **Age** Weight: -13.8906107. Indicates a lower likelihood of being a bot for older accounts, suggesting account age is a significant factor.
- **Number of Posts** Weight: -315.805460. Suggests a strong inverse relationship between post count and bot likelihood, with higher posting activity being less characteristic of bots (a counter-intuitive notion for this particular dataset).
- **Activity** Weight: 0.00779994128. Shows a minimal positive correlation with bot likelihood, implying that daily activity rate has a slight but not substantial impact.

These weights reveal that account longevity and posting frequency are pivotal in distinguishing bots, offering a counterintuitive perspective on typical bot behavior.

This project underscores the complexities involved in bot detection on social media platforms. While the model provides a solid foundation, the intricacies of bot behaviors necessitate a deeper exploration of features and possibly more sophisticated modeling techniques. Future research could focus on enhancing feature engineering, addressing dataset imbalances, and exploring advanced machine learning algorithms to achieve a more nuanced understanding and effective detection of social bots. This work contributes to the ongoing efforts in ensuring the integrity and trustworthiness of digital social interactions.

References

[1] Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Kaicheng Yang, Alessandro Flammini, Filippo Menczer: “The spread of low-credibility content by social bots”, 2017, Nature Communications, 9: 4787, 2018; [http://arxiv.org/abs/1707.07592 arXiv:1707.07592]. DOI: [https://dx.doi.org/10.1038/s41467-018-06930-7 10.1038/s41467-018-06930-7].