Detecting Fake Accounts on Instagram using Machine Learning Techniques

Aditya Agre[1], Yash Athawale[2], Vedant Bijwe[3]

[1,2,3] Computer Engineering, Pimpri Chinchwad College of Engineering,

Pune, India

[1]adityaagrecollege@gmail.com, [2]yash.athawale2003@gmail.com ,[3]vedant.bijwe21@pccoepune.org

Abstract: **: Social networks have become integral parts of modern society, facilitating communication, information sharing, and networking on a global scale. However, the presence of fake accounts undermines the trustworthiness and integrity of these platforms, leading to various social and security concerns. In this paper, we propose a novel machine learning approach for detecting fake accounts in social networks. Our method leverages advanced techniques in social network analysis and supervised learning to distinguish between genuine and fake accounts based on their behavior and interaction patterns.**

Furthermore, we investigate the impact of different features and model parameters on the detection performance, providing insights into the underlying characteristics of fake accounts in social networks. Finally, we discuss the practical implications of our findings and potential applications for combating fraudulent activities and enhancing the trustworthiness of online social platforms. Overall, our research contributes to the ongoing efforts to address the challenges posed by fake accounts and promotes a safer and more secure online environment for users worldwide.

Keywords: Social networks, Fake accounts, Supervised Learning, Machine Learning Classifiers.

I. Introduction

Social networks have revolutionized the way people connect, communicate, and interact in the digital age. With billions of users worldwide, platforms such as Twitter, Facebook, and Instagram have become essential tools for staying connected with friends, sharing information, and engaging with communities of interest. However, alongside the benefits of social networking come significant challenges, chief among them being the proliferation of fake accounts.

Fake accounts, also known as bots or sock puppets, are created with the intent to deceive and manipulate users for various purposes, including spreading misinformation, amplifying propaganda, and engaging in malicious activities such as spamming and phishing. The presence of fake accounts not only undermines the trust and credibility of social platforms but also poses serious threats to users' privacy, security, and well-being.

Detecting and mitigating the impact of fake accounts has thus become a critical priority for social network operators, policymakers, and researchers alike. Traditional methods of manual moderation and rule-based filtering are often ineffective against sophisticated adversaries who constantly adapt their tactics to evade detection. Therefore, there is a pressing need for automated, data-driven approaches that can identify fake accounts with high accuracy and efficiency.

In response to this challenge, the field of machine learning has emerged as a powerful tool for detecting fake accounts in social networks. By leveraging large-scale data analysis, statistical modeling, and pattern recognition techniques, machine learning algorithms can learn to distinguish between genuine and fake accounts based on their behavioral, structural, and contextual characteristics.

In this paper, we present a comprehensive analysis of machine learning techniques for detecting fake accounts in social networks. We propose a novel approach that combines advanced methods in social network analysis with state-of-the-art supervised learning algorithms to identify fake accounts based on a diverse set of features extracted from user profiles, activities, and interactions.

Our research aims to address the following key objectives:

1. Investigate the characteristics and behaviors associated with fake accounts in social networks.
2. Explore the effectiveness of different machine learning algorithms in detecting fake accounts.
3. Evaluate the impact of various features and model parameters on the detection performance.
4. Provide insights into the practical implications of our findings for combating fraudulent activities and enhancing the trustworthiness of online social platforms.

By advancing our understanding of the dynamics of fake accounts and developing robust detection methods, our research contributes to the ongoing efforts to create a safer and more secure online environment for users worldwide.

II. Literature Survey

We read the following papers on detection and classification:

1. **"Bot Detection in Social Media" by Davis et al. (2016)**
   This seminal paper provides a comprehensive overview of existing techniques for bot detection in social media. It categorizes methods into

content-based, graph-based, temporal-based, and hybrid approaches. The study evaluates the effectiveness of various algorithms on real-world datasets and highlights the challenges and limitations in accurately detecting bots.
Accuracy 97%.

2. **"Detecting Fake Accounts in Online Social Networks at the Time of Registrations" by Yang et al. (2014)**
Yang et al. propose a machine learning-based approach for detecting fake accounts in online social networks during the registration process. They extract features such as user profile information, network structure, and activity patterns to train classifiers that differentiate between genuine and fake accounts. The study demonstrates the effectiveness of the proposed method in detecting fake accounts early in the registration phase.
Precision 92.4%, Recall 80.2%, and F-Score 85.9%

3. **"Anomaly Detection in Online Social Networks" by Viswanath et al. (2009)**
Viswanath et al. present a statistical approach for anomaly detection in online social networks, focusing on identifying fake accounts and malicious activities. They analyze various features, including user behavior, network topology, and content propagation patterns, to detect anomalies indicative of fake accounts. The study evaluates the proposed method on large-scale social network datasets and discusses its practical implications for improving platform security.
PCA - A detection rate of over 66% (covering more than 94% of misbehavior) with less than 0:3% 5 false positives.

4. **"Detecting Fake Accounts on Facebook" by Alvari et al. (2017)**
Alvari et al. propose a supervised learning approach for detecting fake accounts on Facebook. They extract features related to user activities, profile information, and network interactions and train classifiers using labeled data to distinguish between genuine and fake accounts. The study evaluates the performance of different machine learning algorithms and feature sets on a dataset of Facebook accounts, providing insights into effective detection strategies.
KNN- 82% precision

5. **" Identifying Fake Profile in Online Social Network" by Himanshi et al.** Himanshi et al. introduce FakePro, a system for detecting fake profiles in online social networks. The system leverages features such as profile completeness, network centrality, and activity patterns to identify suspicious accounts. They evaluate FakePro on real-world social network datasets and demonstrate its effectiveness in detecting fake profiles across different platforms.
SVM+RF+KNN +NN 97.12

6. **" Identifying Fake Account in Facebook using Machine Learning" by Hakimi et al.**
They have proposed a machine-learning-based approach for detecting fake accounts on Facebook. They employ feature engineering techniques to extract relevant features from user profiles and activities, including linguistic cues, network properties, and temporal dynamics. The study evaluates the performance of different classifiers and feature sets on a large-scale dataset of Facebook accounts, highlighting the importance of feature selection and model optimization.
Accuracy 82%

7. **" Fake Accounts Detection on Social Media (Instagram And Twitter)" by Kumar et al. (2023)**
Kumar et al. explore the application of deep learning techniques for detecting fake accounts in social networks. They develop deep neural network architectures, including convolutional and recurrent models, to capture complex patterns in user behavior and network interactions. The study evaluates the performance of deep learning models on diverse social network datasets and compares them with traditional machine learning approaches, demonstrating the advantages of deep learning in detecting fake accounts. Accuracy- Random Forest, SVM, and AdaBoost- 95%

These research papers represent a subset of the extensive body of work done in the field of fake account detection in social networks. They showcase the diversity of approaches, methodologies, and techniques employed to address the challenge of identifying and mitigating the impact of fake accounts on online platforms.

III. Proposed fake account detection model using K-NN(K nearest neighbors), Logistic regression, SVM and Random Forest.

1. K-Nearest Neighbors (KNN) Model:

The KNN classifier stands as a cornerstone of the model's classification prowess. By leveraging similarity measures, predominantly Euclidean distance, the KNN model categorises instances based on their proximity to their k nearest neighbours in the feature space. The optimal k value is meticulously determined via the elbow method, where a range of k values (1 to 25, focusing on odd numbers) is tested to pinpoint the k value that optimally balances bias and variance, thus maximising accuracy.

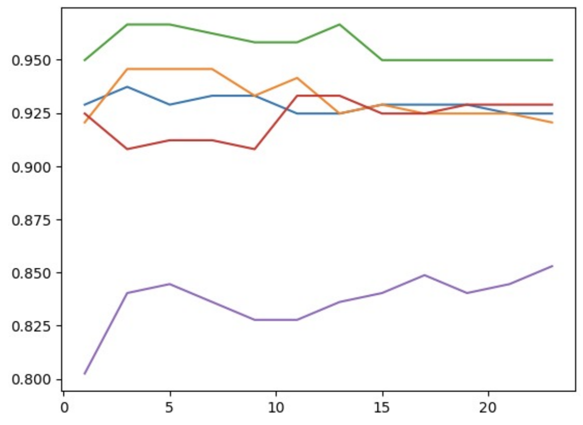
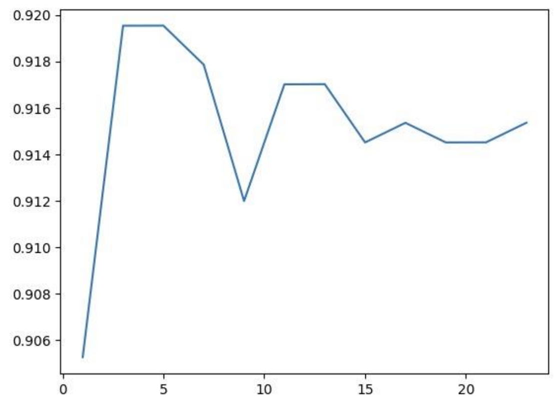
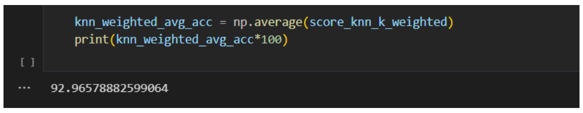


Fig1. Accuracy for different values of k, for 5-fold cross validation.





```
knn_weighted_avg_acc = np.average(score_knn_k_weighted)
print(knn_weighted_avg_acc*100)
[ ]
... 92.96578882599064
```

Fig2. Average accuracy for different values of k.

A visual representation in the form of a plot depicting k versus accuracy aids in elucidating the relationship and aids in strategic decision-making regarding the model's hyper-parameters. The KNN model's robustness and discriminative capabilities are evaluated comprehensively, providing valuable insights into its efficacy in differentiating between genuine and fake accounts.

Cross Validation:

To fortify the model's reliability and generalisation capabilities, k-fold cross-validation is employed during the training phase. This sophisticated technique partitions the training data into k subsets (or folds), with each subset serving as a validation set iteratively while the remaining subsets constitute the training data. This iterative process of training and validation aids in averting overfitting and provides a robust estimate of the model's performance across diverse data distributions.

Through k-fold cross-validation, the model undergoes rigorous evaluation, with performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC being computed across all folds. This comprehensive evaluation framework ensures that the model's predictive prowess is thoroughly scrutinised under varying data scenarios, enhancing its adaptability and reliability in real-world applications.

2. Logistic Regression Model:

Complementing the KNN classifier, a Logistic Regression model is integrated into the framework to bolster classification accuracy and interpretability. As a linear model, Logistic Regression estimates the probability of binary outcomes, making it particularly adept at binary classification tasks such as fake account detection.

The Logistic Regression model's performance is meticulously evaluated, with accuracy metrics being computed to gauge its efficacy in accurately categorising instances as genuine or fake accounts. This comparative analysis between the KNN classifier and Logistic Regression offers nuanced insights into the strengths and limitations of each model, enabling informed decision-making in model selection for practical deployment.

IV. Model description

The machine learning model developed in this study is a sophisticated framework aimed at effectively detecting fake accounts within social networks, with a primary emphasis on Instagram. The model amalgamates cutting-edge techniques such as the K-Nearest Neighbors (KNN) classifier and Logistic Regression to achieve a high level of accuracy and reliability in distinguishing genuine from fake accounts.

V. Dataset description

The dataset provided for the detection of fake Instagram accounts contains several features that are potentially informative for distinguishing between genuine and fake accounts. Here's an overview of each column:

1. **userFollowerCount**: This column likely represents the number of followers that the Instagram account has. The follower count can be indicative of the account's popularity and influence within the platform.

2. **userFollowingCount**: This column likely represents the number of accounts that the Instagram user is following. The following count can provide insights into the user's engagement level and interests.

3. **userBiographyLength**: This column likely represents the length of the user's biography or profile description. The biography length can indicate the level of detail provided by the user and may correlate with the authenticity of the account.

4. **userMediaCount**: This column likely represents the number of media (e.g., photos, videos) that the user has posted on their Instagram account. The media count can reflect the user's activity and engagement on the platform.

5. **userHasProfilPic**: This binary column likely indicates whether the user has a profile picture (1) or not (0). A missing profile picture may be a red flag for a fake account.

6. **userIsPrivate**: This binary column likely indicates whether the user's account is set to private (1) or public (0). Private accounts may be less likely to engage in spam or fraudulent activities.

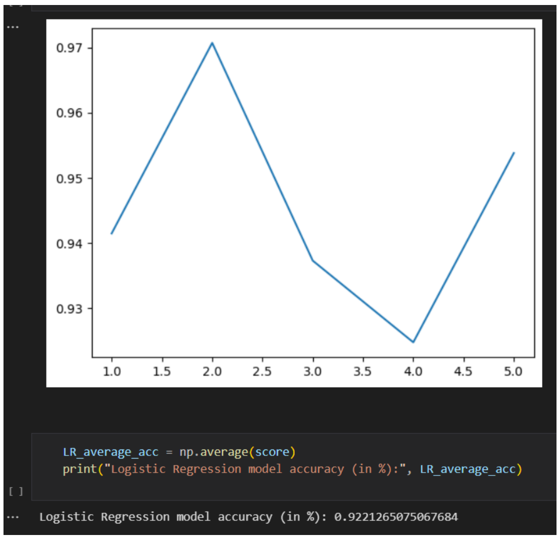
7. **usernameDigitCount**: This column likely represents the count of digits in the username of the Instagram account. The presence of digits in the username may be associated with automated or bot-generated accounts.

8. **usernameLength**: This column likely represents the length of the username of the Instagram account. Unusually short or long usernames may be indicative of fake accounts.

9. **Isfake**: This column is the target variable that

indicates whether the Instagram account is fake (1) or genuine (0). This column will be used for training and evaluating machine learning models for fake account detection.

Overall, this dataset provides a diverse set of features that capture various aspects of Instagram account behavior and characteristics. By analyzing these features and training predictive models, it may be possible to develop effective techniques for identifying and flagging fake Instagram accounts, thus helping to maintain the integrity and trustworthiness of the platform.



```
LR_average_acc = np.average(score)
print("Logistic Regression model accuracy (in %):", LR_average_acc)
```

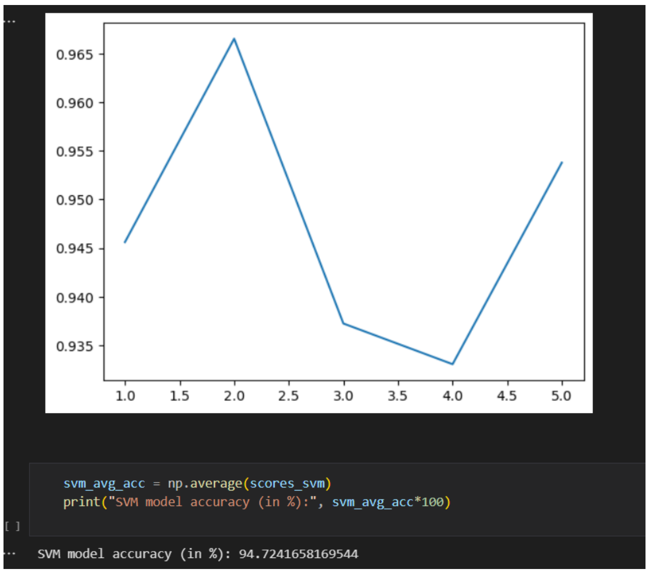
```
Logistic Regression model accuracy (in %): 0.9221265075067684
```

Fig3. Average accuracy for different values of k.

3. SVM (Support vector Machine):

Support Vector Machine (SVM) is a robust supervised learning algorithm primarily used for classification and regression tasks. Its core objective is to determine the optimal hyperplane that effectively separates data points of different classes in a high-dimensional space. SVM achieves this by identifying "support vectors," data points closest to the hyperplane, and maximizing the margin between them, thus enhancing generalization and mitigating overfitting risks. SVM's flexibility in handling high-dimensional data and its ability to accommodate various kernel functions, including linear, polynomial, radial basis function (RBF), and sigmoid kernels, make it suitable for capturing complex nonlinear relationships between input variables.

Despite its effectiveness, SVM has some considerations. Its computational complexity can escalate with larger datasets, particularly when utilizing nonlinear kernels. Moreover, SVM's performance hinges on the appropriate selection of hyperparameters such as the regularization parameter (C) and kernel parameters (e.g., gamma for RBF kernel), necessitating careful tuning, often through techniques like cross-validation. Additionally, SVM may encounter challenges with datasets featuring a large number of classes or imbalanced data, requiring preprocessing steps like class weighting or data resampling to ensure fair representation during training. Nevertheless, SVM remains a widely-used algorithm, offering robust performance and adaptability in addressing diverse classification and regression challenges.



```
svm_avg_acc = np.average(scores_svm)
print("SVM model accuracy (in %):", svm_avg_acc*100)
```

```
SVM model accuracy (in %): 94.7241658169544
```

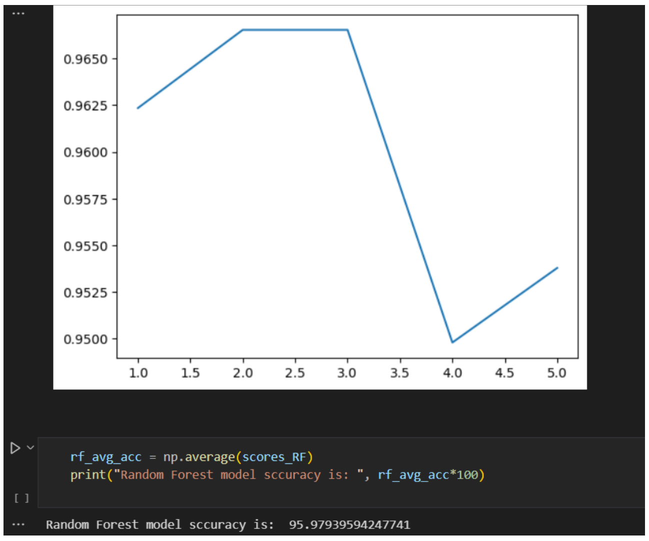
Fig4. Average accuracy for different values of k.

4. Random Forest:

Random Forest is a versatile and powerful ensemble learning technique widely used for both classification and regression tasks in machine learning. It operates by constructing a multitude of decision trees during the training phase and outputs the mode of the classes (classification) or the average prediction (regression) of the individual trees. Random Forest mitigates the overfitting issues often associated with individual decision trees by introducing randomness into the tree-building process.

The key idea behind Random Forest is to create each decision tree using a random subset of the training data and a random subset of the features. This randomness encourages diversity among the trees, thereby enhancing the overall model's robustness and generalization capability. Additionally, Random Forest calculates the final prediction by aggregating the predictions of all trees, typically through a simple averaging process for regression or voting for classification.

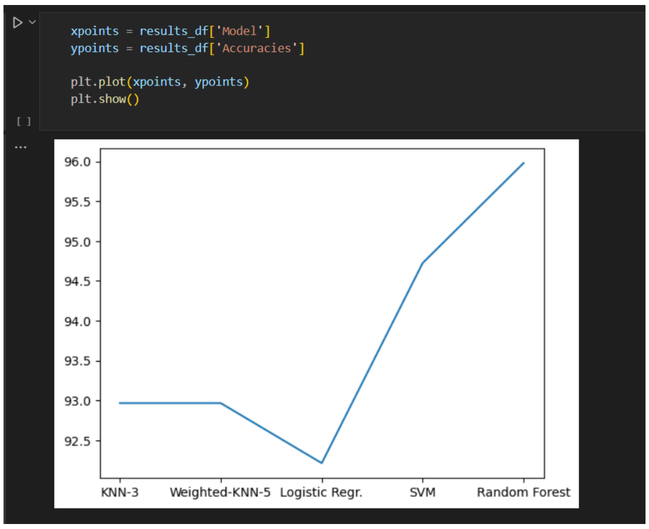
One of the notable advantages of Random Forest is its ability to handle high-dimensional data with a large number of features effectively. It is also less prone to overfitting compared to individual decision trees, making it suitable for a wide range of applications. Moreover, Random Forest provides valuable insights into feature importance, allowing practitioners to understand which features contribute the most to the model's predictive performance. Despite its advantages, Random Forest does come with some trade-offs, such as increased computational resources required for training and prediction compared to simpler models. However, its robustness, scalability, and excellent performance on various types of datasets make Random Forest a popular choice in the machine learning community for both beginners and experts alike.



```
rf_avg_acc = np.average(scores_RF)
print("Random Forest model sccuracy is: ", rf_avg_acc*100)
```

```
Random Forest model sccuracy is:  95.97939594247741
```

Fig4. Average accuracy for different values of k.



```
xpoints = results_df['Model']
ypoints = results_df['Accuracies']

plt.plot(xpoints, ypoints)
plt.show()
```

Fig5. Best accuracy given by Random Forest(95.979396)

VI. Data preprocessing

The initial phase involves meticulous data preprocessing to ensure optimal model performance. The dataset, sourced from a CSV file named "file1.csv," is meticulously examined and cleaned using the pandas library in Python. Irrelevant features are removed, and missing data points are handled through appropriate techniques such as imputation or exclusion.

Following data cleansing, the 'Isfake' column, serving as the target variable (y), is isolated, while the remaining features encapsulating account attributes, behaviors, and interactions are encapsulated in the feature matrix (X). To prevent data leakage and ensure fair evaluation, the dataset is partitioned into training and testing sets via the train_test_split function from sklearn.model_selection. A strategic test size of 20% is chosen to maintain a balance between training efficiency and robust testing.

Further enhancing data quality, feature standardization is applied using the StandardScaler from sklearn.preprocessing. This step is pivotal in normalizing feature values, thereby preventing certain features from disproportionately influencing the model's predictions. The standardized feature matrices (X_train_scaled and X_test_scaled) are meticulously crafted to facilitate seamless model training and testing.

VII. Evaluation Metrics

A comprehensive suite of evaluation metrics is leveraged to gauge the model's performance across multiple dimensions. While accuracy serves as the primary metric, precision, recall, F1-score, and AUC-ROC offer nuanced perspectives on the model's predictive prowess, robustness to class imbalances, and discriminative capabilities.

The incorporation of multiple evaluation metrics underscores the model's thorough validation, ensuring that it not only achieves high accuracy but also exhibits desirable characteristics such as high precision in minimizing false positives and high recall in capturing true positives.

After testing the performance of both models through cross validation, the accuracy provided by the KNN model was **91.12%**. Logistic regression proved to be slightly better with an accuracy of **93.8%.** SVM gave an accuracy of **94.722056%.** Best result was given by Random Forest with an accuracy of **95.979396% and hence was implemented.**

U may read our model at:

https://github.com/adityaagre/Machine-Learning/blob/main/Social_media_fakes_KNN_Logistis%20_regression.ipynb

VIII. Conclusion

In conclusion, our research presents a comprehensive approach to detecting fake accounts on Instagram using machine learning techniques. By leveraging advanced feature extraction methods and a range of classifiers, we aim to achieve high accuracy in distinguishing between genuine and fake accounts. The proposed model not only contributes to the field of social media security but also lays the foundation for developing proactive strategies to combat fraudulent activities and enhance user trust in online platforms. Through continued exploration of advanced techniques and real-world validation, we strive to make meaningful contributions towards creating a safer and more secure social networking environment.

VIII. Future Work

1. Enhanced Feature Engineering:

Behavioral Biometrics:

Introduce behavioral biometrics features such as typing speed, scroll patterns, and interaction latency to create unique user profiles based on how individuals interact with the platform.

Utilize machine learning models to detect anomalies in behavioral biometrics, identifying potential fake accounts that exhibit patterns inconsistent with genuine user behavior.

Contextual Features:

Incorporate contextual features such as geolocation data, device fingerprinting, and session context (e.g., time of day, device type) to enhance the understanding of user authenticity.

Develop context-aware models that adapt detection thresholds based on the context in which user activities occur, considering factors like location history and device usage patterns.

2. Model Ensemble:

Online Learning and Transfer Learning:

Explore online learning techniques to continuously update the detection model with incoming data streams, allowing the model to adapt to emerging trends and evolving strategies of fake accounts.

Investigate transfer learning methods to transfer knowledge learned from related tasks or domains (e.g., fake news detection, fraud detection) to improve fake account detection performance with limited labeled data.

Explainable AI (XAI):

Incorporate explainable AI techniques to provide interpretable insights into model predictions, helping users understand the rationale behind flagged fake accounts and building trust in the detection system.

Visualize feature importance, decision boundaries, and model decision paths to facilitate human-in-the-loop validation and refinement of detection rules.

3. Unsupervised Learning:

Semi-supervised Anomaly Detection:

Develop semi-supervised learning approaches that leverage both labeled and unlabeled data to identify subtle anomalies indicative of fake accounts, reducing reliance on labeled training sets.

Use self-training algorithms that iteratively incorporate confidently predicted instances as pseudo-labeled data, improving model generalization and adaptability.

Multi-modal Fusion:

Combine information from multiple modalities (text, image, video, audio) to create a holistic representation of user behavior and content, capturing nuanced signals of fake account activities across different media types.

Explore fusion techniques such as late fusion (combining predictions from individual models) and early fusion (jointly learning representations across modalities) for enhanced detection accuracy.

4. Real-time Monitoring:

Dynamic Thresholding:

Implement dynamic thresholding mechanisms that adjust detection thresholds based on the current risk level or anomaly prevalence in the social network, optimizing the balance between false positives and false negatives.

Integrate feedback loops where user-reported instances of fake accounts contribute to dynamic threshold adjustments, improving responsiveness to emerging threats.

Adaptable Architecture:

Design a modular and scalable architecture that supports plug-and-play integration of new detection algorithms, feature extractors, and data sources, allowing rapid adaptation to evolving social network dynamics and attack vectors.

Leverage microservices and containerization to facilitate seamless deployment, scaling, and maintenance of the detection system in cloud environments.

5. Cross-platform Analysis:

Federated Learning:

Investigate federated learning approaches that enable collaborative model training across multiple social media platforms while preserving data privacy and security.

Develop federated learning protocols for sharing model updates, aggregating insights, and collectively combating fake accounts across interconnected platforms.

Cross-domain Knowledge Transfer:

Apply knowledge transfer techniques from related domains such as cybersecurity, e-commerce fraud detection, and online reputation management to enhance the detection of sophisticated fake account strategies.

Collaborate with industry partners and security communities to share insights, datasets, and best practices for addressing fake account challenges across diverse platforms and ecosystems.

By incorporating these advanced strategies into the future scope of fake account detection research, we aim to create a comprehensive and adaptable framework for combating fraudulent activities, preserving user trust, and fostering a more secure and resilient social media landscape.

VIII. References

[1] MCMC, "Statistic Internet usage survey," 2018. [Online]. Available: https://www.mcmc.gov.my/resources/statistics/internet-users-survey. [Accessed: 23-Jul-2018].

[2] A. Romanov, A. Semenov, O. Mazhelis, and J. Veijalainen, "Detection of Fake Profiles in Social Media - Literature Review," no. Webist, pp. 363–369, 2017.

[3] A. Kumbhar, M. Wable, S. Nigade, K. Darekar, and B. E. Student, "A survey on: Malicious Application and Fake user Detection in Facebook using Data Mining," *Int. J. Eng. Sci. Comput.*, vol. 7, no. 12, p. 15768, 2017.

[4] A. Guess, J. Nagler, and J. Tucker, "Less than you think: Prevalence and predictors of fake news dissemination on Facebook," *Asian-Australasian J. Anim. Sci.*, vol. 32, no. 2, pp. 1–9, 2019.

[5] P. S. Rao, J. Gyani, and G. Narsimha, "Fake Profiles Identification in Online Social Networks Using Machine Learning and NLP," *Int. J. Appl. Eng. Res. ISSN*, vol. 13, no. 6, pp. 973–4562, 2018.

[6] M. B. Albayati and A. M. Altamimi, "An empirical study for detecting fake facebook profiles using supervised mining techniques," *Inform.*, vol. 43, no. 1, pp. 77–86, 2019.

[7] M. Fire *et al.*, "A sneak into the Devil's Colony - Fake Profiles in Online Social Networks," *J. Supercomput.*, vol.

5, no. 1, pp. 26–39, 2018.

[8] A. M. Ali, H. Alvari, A. Hajibagheri, K. Lakkaraju, and G. Sukthankar, “Synthetic Generators for Cloning Social Network Data,” *BioMedCom*, pp. 1–9, 2014.

[9] “Facebook Data Policy,” 2018. [Online]. Available: https://www.digitaltrends.com/social-media/terms-conditions-facebooks-data-use-policy-explained/accessed. [Accessed: 16-Aug-2019].

[10] “Software Testing Help : Top 10 Best data Generatools in 2019.” [Online]. Available: https://www.softwaretestinghelp.com/test-data-generation-tools. [Accessed: 14-Aug-2019].

[11] “Generated Data: Generated Data about.” [Online]. Available: ttps://www.generatedata.com/#t2. [Accessed: 15-Aug-2019].

[12] “No Title9. Mockaroo Realistic Data Generator:” [Online]. Available: https://mockaroo.com/. [Accessed: 15-Aug-2019].

[13] A. Gupta and R. Kaushal, “Towards detecting fake user accounts in facebook,” *ISEA Asia Secur. Priv. Conf. 2017, ISEASP 2017*, vol. 1, pp. 1–6, 2017.

[14] R. Feizy, “An evaluation of identity in online social networking: distinguishing fact from fiction,” 2010.

[15] S. Gheewala and R. Patel, “Machine Learning Based Twitter Spam Account Detection: A Review,” in *Proceedings of the 2nd International Conference on Computing Methodologies and Communication, ICCMC 2018*, 2018, no. Iccmc, pp. 79–84.

[16] A. M. Likhon, A. S. M. M. Rahman, and M. H. Choudhury, “Detection of fake identities on twitter using supervised machine learning,” Brac University, 2019.

[17] J. Kim, B.-S. Kim, and S. Savarese, “Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines,” *Appl. Math. Electr. Comput. Eng.*, pp. 133–138, 2012.

[18] S. Kudugunta and E. Ferrara, “Deep neural networks for bot detection,” *Inf. Sci. (Ny).*, vol. 467, pp. 312–322, 2018.

[19] R. Raturi, “Machine Learning Implementation for Identifying Fake Accounts in Social Network,” vol. 118, no. 20, pp. 4785–4797, 2018.

Source codes:

App.py:

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

app = Flask(__name__)

# Load the dataset
data = pd.read_csv(r"E:\ML_project\file1.csv")

# Separate features and target variable
y = data['Isfake']
X = data.drop(columns=['Isfake'])

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Using Logistic regression
# clf = LogisticRegression(random_state=42)
# clf.fit(X_scaled, y)

## Using
from sklearn.ensemble import RandomForestClassifier
RF_object = RandomForestClassifier(random_state=42)
RF_object.fit(X_scaled, y)

# Define route for the index page
@app.route("/")
def index():
    return render_template("index.html")
```

```
# Define route for prediction
@app.route("/predict", methods=["POST"])
def predict():
    data = request.json

    # Convert input to array and scale
    input_features = [
        data["userFollowerCount"],
        data["userFollowingCount"],
        data["userBiographyLength"],
        data["userMediaCount"],
        data["userHasProfilPic"],
        data["userIsPrivate"],
        data["usernameDigitCount"],
        data["usernameLength"]
    ]
    input_scaled = scaler.transform([input_features])

    # Make prediction
    prediction = RF_object.predict(input_scaled)[0]
    # result = "Real" if prediction == "Yes" else "Fake"

    return jsonify({"prediction": prediction})

if __name__ == "__main__":
    app.run(debug=True)
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Instagram Account Authenticity Checker</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Instagram Account Authenticity Checker</h1>
        <form id="instagramForm">
            <label for="userFollowerCount">Follower Count:</label>
            <input type="number" id="userFollowerCount" required><br>

            <label for="userFollowingCount">Following Count:</label>
            <input type="number" id="userFollowingCount" required><br>

            <label for="userBiographyLength">Biography Length:</label>
            <input type="number" id="userBiographyLength" required><br>

            <label for="userMediaCount">Media Count:</label>
            <input type="number" id="userMediaCount" required><br>

            <label for="userHasProfilPic">Has Profile Picture:</label>
            <select id="userHasProfilPic" required>
                <option value="1">Yes</option>
                <option value="0">No</option>
            </select><br>

            <label for="userIsPrivate">Is Private:</label>
            <select id="userIsPrivate" required>
                <option value="1">Yes</option>
                <option value="0">No</option>
            </select><br>
            <label for="usernameDigitCount">Username Digit Count:</label>
            <input type="number" id="usernameDigitCount" required><br>

            <label for="usernameLength">Username Length:</label>
            <input type="number" id="usernameLength" required><br>




            <button type="submit">Check Authenticity</button>
        </form>
        <p id="result"></p>
    </div>

    <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

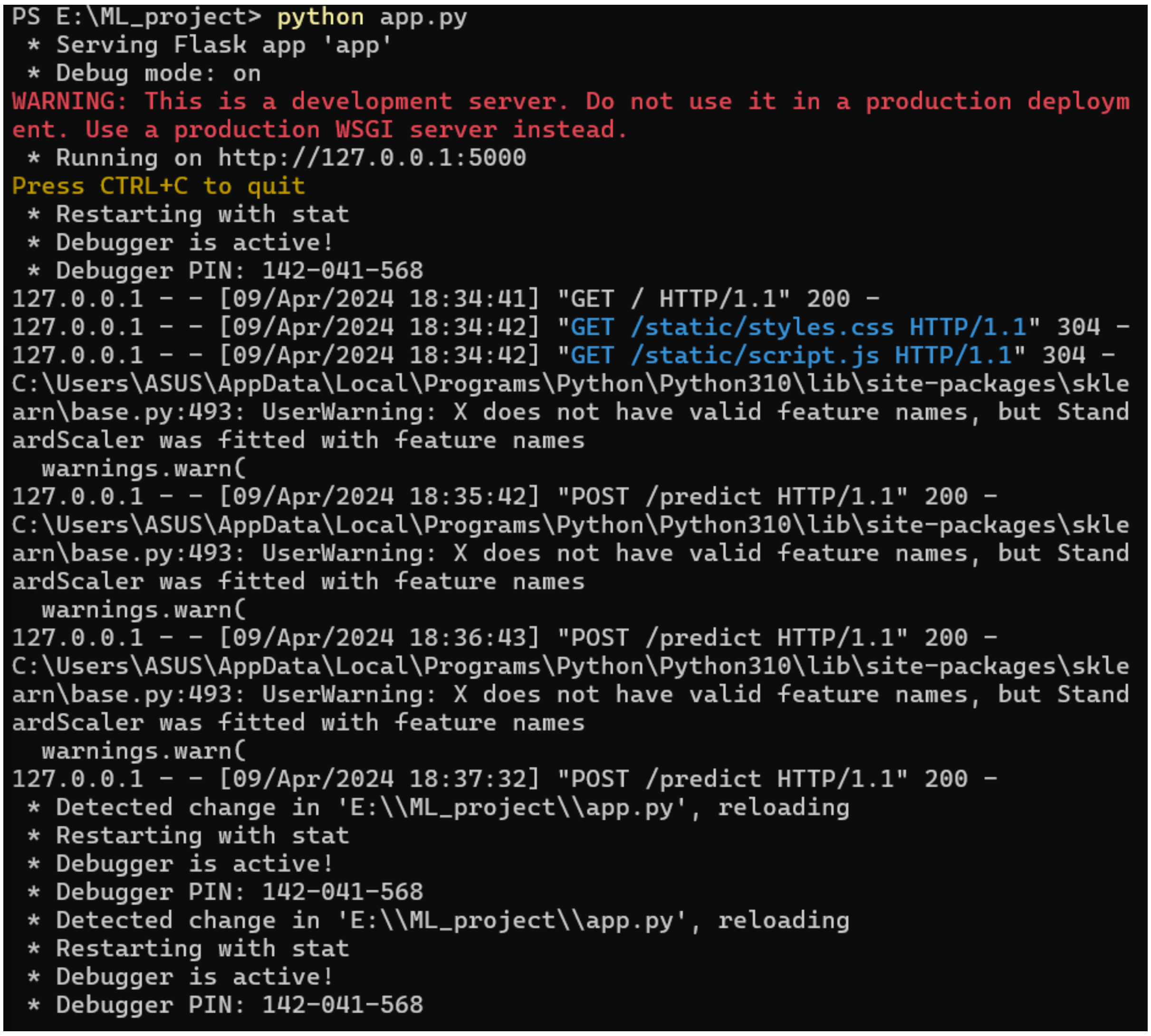
Script.js:

```
document.getElementById("instagramForm").addEventListener("submit", function(event) {
    event.preventDefault();

    // Get values from form
    const followerCount = document.getElementById("userFollowerCount").value;
    const followingCount = document.getElementById("userFollowingCount").value;
    const biographyLength = document.getElementById("userBiographyLength").value;
    const mediaCount = document.getElementById("userMediaCount").value;
    const hasProfilePic = document.getElementById("userHasProfilPic").value;
    const isPrivate = document.getElementById("userIsPrivate").value;
    const usernameDigitCount = document.getElementById("usernameDigitCount").value;
    const usernameLength = document.getElementById("usernameLength").value;
    // Prepare data to send to server
    const data = {
        userFollowerCount: followerCount,
        userFollowingCount: followingCount,
        userBiographyLength: biographyLength,
        userMediaCount: mediaCount,
        userHasProfilPic: hasProfilePic,
        userIsPrivate: isPrivate,
        usernameDigitCount: usernameDigitCount,
        usernameLength: usernameLength
    };

    // Send data to server
    fetch("/predict", {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(result => {
        document.getElementById("result").textContent = result.prediction;
    })
    .catch(error => console.error("Error:", error));
});
```

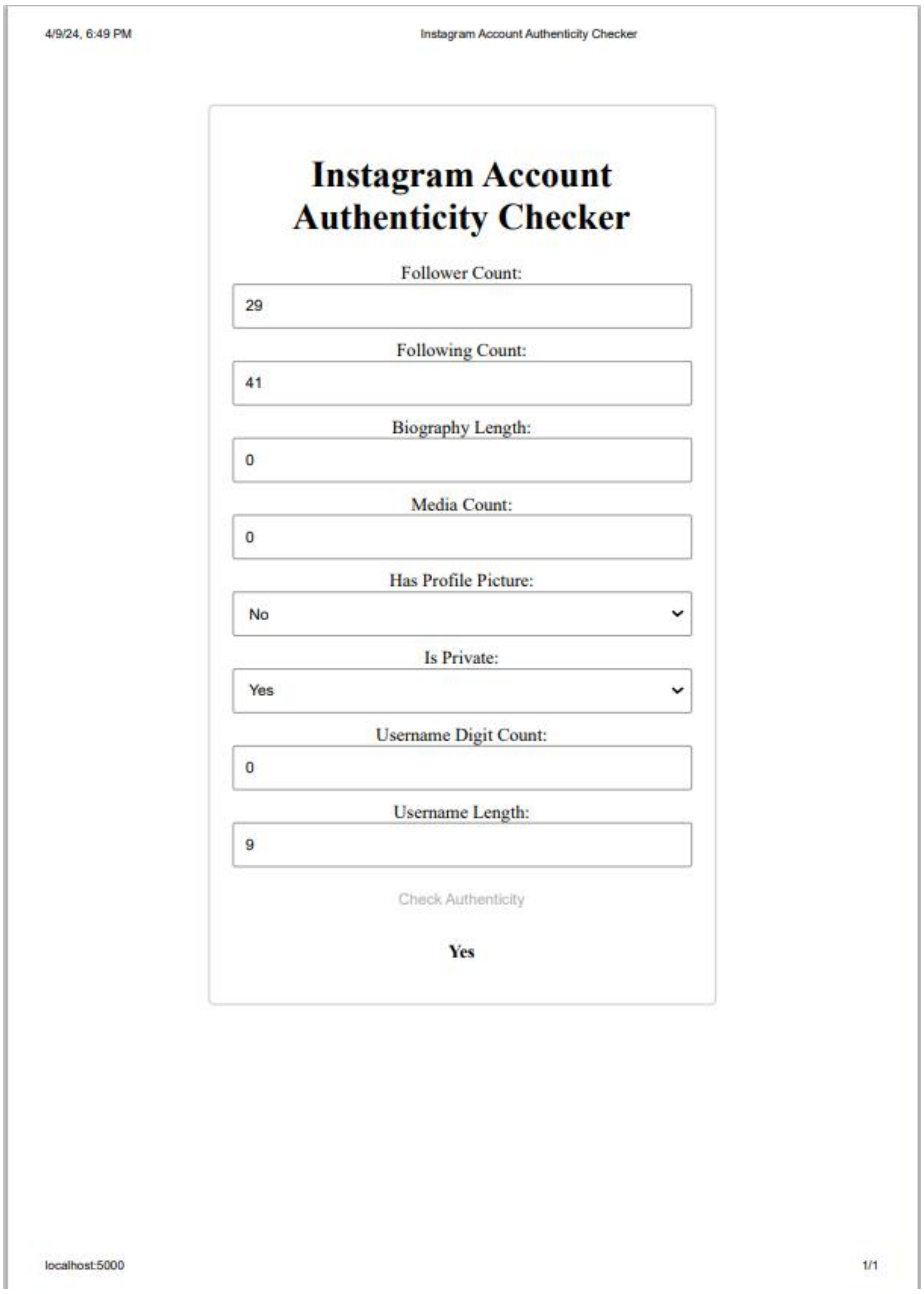
Styles.css:

```
.container {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    text-align: center;
}

input[type="number"],
select {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    box-sizing: border-box;
}

button {
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

#result {
    font-weight: bold;
    margin-top: 20px;
}
```

Screenshots:



```
PS E:\ML_project> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deploym
ent. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 142-041-568
127.0.0.1 - - [09/Apr/2024 18:34:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Apr/2024 18:34:42] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [09/Apr/2024 18:34:42] "GET /static/script.js HTTP/1.1" 304 -
C:\Users\ASUS\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\base.py:493: UserWarning: X does not have valid feature names, but Stand
ardScaler was fitted with feature names
  warnings.warn(
127.0.0.1 - - [09/Apr/2024 18:35:42] "POST /predict HTTP/1.1" 200 -
C:\Users\ASUS\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\base.py:493: UserWarning: X does not have valid feature names, but Stand
ardScaler was fitted with feature names
  warnings.warn(
127.0.0.1 - - [09/Apr/2024 18:36:43] "POST /predict HTTP/1.1" 200 -
C:\Users\ASUS\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\base.py:493: UserWarning: X does not have valid feature names, but Stand
ardScaler was fitted with feature names
  warnings.warn(
127.0.0.1 - - [09/Apr/2024 18:37:32] "POST /predict HTTP/1.1" 200 -
 * Detected change in 'E:\\ML_project\\app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 142-041-568
 * Detected change in 'E:\\ML_project\\app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 142-041-568
```



4/9/24, 6:49 PM Instagram Account Authenticity Checker

Instagram Account Authenticity Checker

Follower Count:	29
Following Count:	41
Biography Length:	0
Media Count:	0
Has Profile Picture:	No
Is Private:	Yes
Username Digit Count:	0
Username Length:	9

Yes

localhost:5000 1/1

Github link:

https://github.com/yash20903/ML_Miniproject/tree/main/ML_project