Instagram Fake–Spammer–Genuine Account Classification

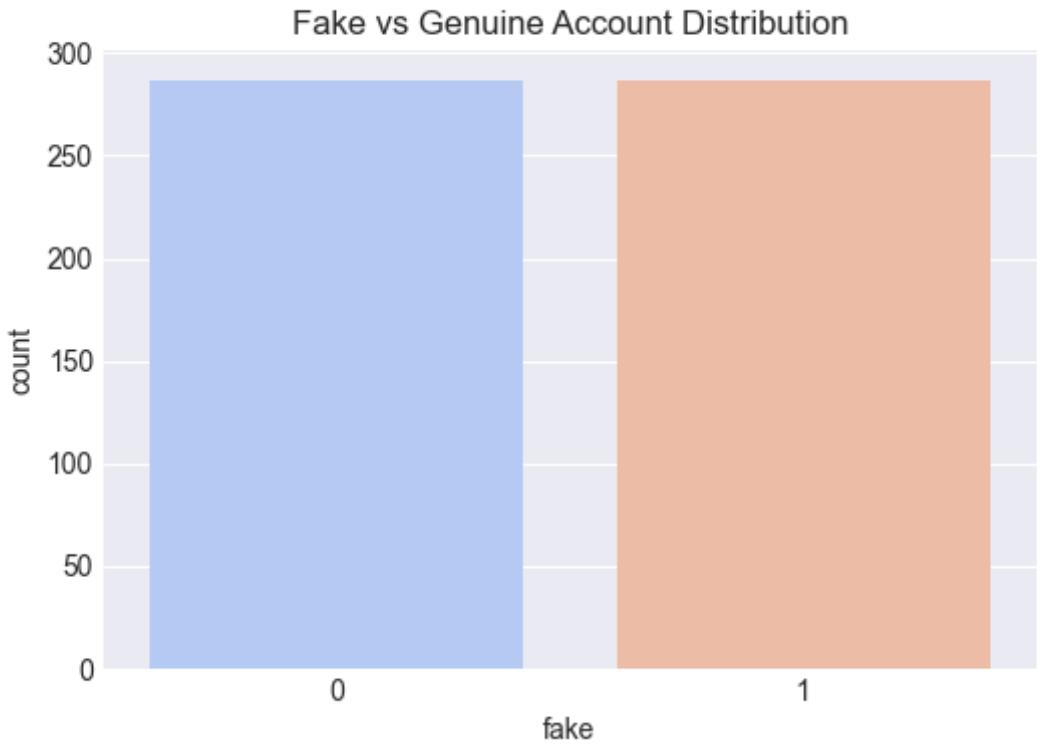
Internship Project Report

Submitted by: **Tanisha Mangliya**

EMAIL: **tanishamangliya@gmail.com**
b22ee067@iitj.ac.in at **Indian Institute of Technology Jodhpur**

For: **Data Science** Internship at **Unified Mentor**

GitHub Repository: GitHub link here



September 5, 2025

Contents

1. Introduction

Social media platforms like Instagram face a growing challenge with fake and spam accounts that distort engagement metrics and harm community trust. This project focuses on developing a **machine learning model** to automatically classify accounts as fake, spammer, or genuine using profile-related attributes.

The model was trained and validated on real-world Instagram data collected using a crawler. The solution aims to assist digital platforms, marketers, and analysts in detecting fraudulent activities and improving data-driven insights.

2. Project Objectives

- To preprocess and analyze Instagram user profile data for detecting fake or spam accounts.
- To build and evaluate a machine learning classifier capable of high precision and recall.
- To visualize analytical trends using Python and data visualization libraries.
- To deploy the trained model for predictions using an interactive app.

3. Dataset Overview

Two datasets (`train.csv` and `test.csv`) were used. Each record represents a unique Instagram account with 11 numerical and binary attributes.

Table 1: Dataset Columns and Descriptions

Column Name	**Description**
profile pic	1 if the profile has a picture, 0 otherwise
nums/length username	Ratio of numbers to length of username
fullname words	Word count in full name
nums/length fullname	Ratio of numbers to length of full name
name==username	1 if full name equals username
description length	Character length of bio
external URL	1 if an external link exists
private	1 if account is private
#posts	Number of posts
#followers	Number of followers
#follows	Number of accounts followed
fake	Target variable (1 = fake/spam, 0 = genuine)

```
Train dataset shape: (576, 12)
Test dataset shape: (120, 12)

First 5 rows of training data:
    profile pic  nums/length username  fullname words  nums/length fullname  \
0            1                  0.27               0                   0.0
1            1                  0.00               2                   0.0
2            1                  0.10               2                   0.0
3            1                  0.00               1                   0.0
4            1                  0.00               2                   0.0

   name==username  description length  external URL  private  #posts  \
0               0                  53             0        0      32
1               0                  44             0        0     286
2               0                   0             0        1      13
3               0                  82             0        0     679
4               0                   0             0        1       6

   #followers  #follows  fake
0        1000       955     0
1        2740       533     0
2         159        98     0
3         414       651     0
4         151       126     0
```

Figure 1: train-df(head)

test_df.head()

MagicPython

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
0	1	0.33	1	0.33	1	30	0	1	35	488	604	0
1	1	0.00	5	0.00	0	64	0	1	3	35	6	0
2	1	0.00	2	0.00	0	82	0	1	319	328	668	0
3	1	0.00	1	0.00	0	143	0	1	273	14890	7369	0
4	1	0.50	1	0.00	0	76	0	1	6	225	356	0

Figure 2: test-df(head)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 576 entries, 0 to 575
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   profile pic           576 non-null    int64
 1   nums/length username  576 non-null    float64
 2   fullname words        576 non-null    int64
 3   nums/length fullname  576 non-null    float64
 4   name==username        576 non-null    int64
 5   description length    576 non-null    int64
 6   external URL          576 non-null    int64
 7   private               576 non-null    int64
 8   #posts                576 non-null    int64
 9   #followers            576 non-null    int64
 10  #follows              576 non-null    int64
 11  fake                  576 non-null    int64
dtypes: float64(2), int64(10)
memory usage: 54.1 KB
```

Figure 3: Data-Columns

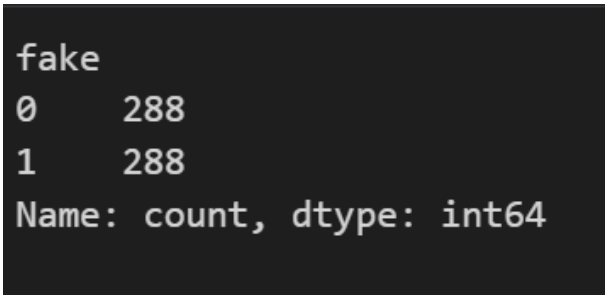


Figure 4: 1:fake , 0:real

4. Project Folder Structure

```
INSTAGRAM-FAKE-SPAMMER-GENUINE/

 data/
    train.csv
    test.csv

 notebooks/
    instagram_fake_spammer_project.ipynb
    best_instagram_fake_spammer_model.pkl

 results/
    01_fake_vs_genuine.png
    02_follower_vs_following.png
    03_correlation_heatmap.png
    04_confusion_matrix.png
    05_feature_importance.png

 app.py
 train_model.py
 train_model_final.py
 requirements.txt
 README.md
 LICENSE
```

5. Methodology and Workflow

The workflow follows a standard data science lifecycle.

5.1 Step 1: Data Loading and Cleaning

- Loaded the dataset using `pandas`.
- Checked and handled missing values.
- Standardized column names and data types.

5.2 Step 2: Exploratory Data Analysis (EDA)

EDA provided initial insights into fake vs genuine patterns. Visualizations were generated using `matplotlib` and `seaborn`.

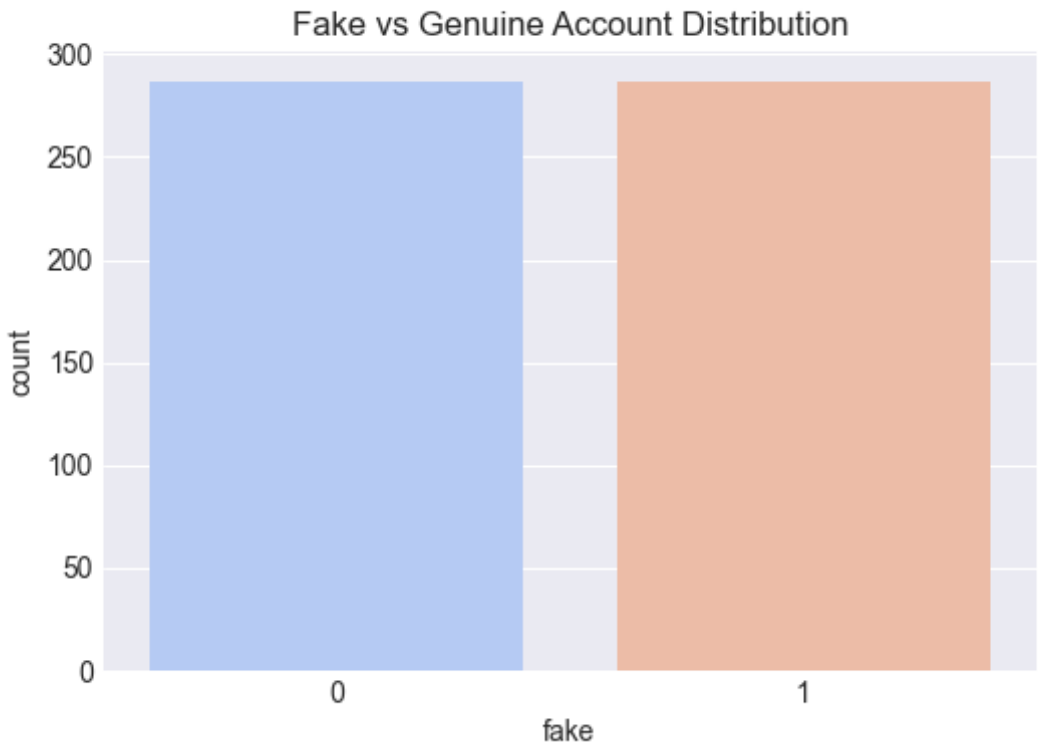


Figure 5: Distribution of Fake vs Genuine Accounts

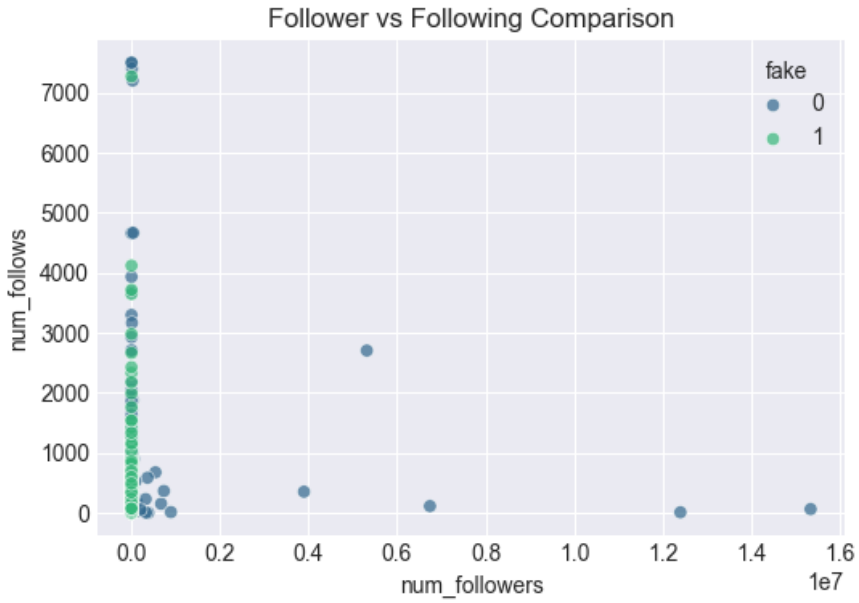


Figure 6: Follower vs Following Distribution

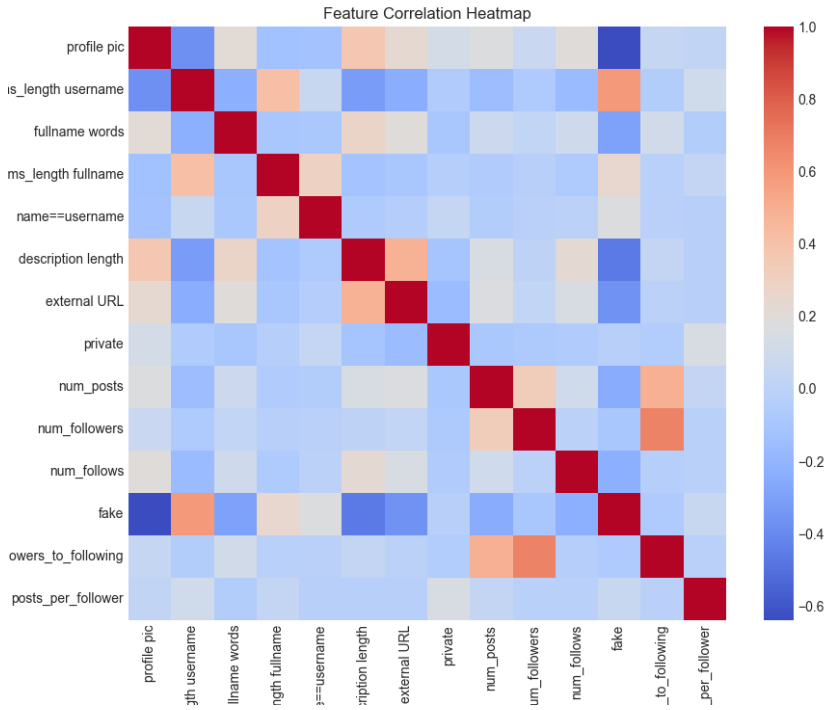


Figure 7: Correlation Heatmap of Features

5.3 Step 3: Feature Engineering

- Normalized numeric features like followers and follows.
- Created new derived ratios.

- Encoded binary attributes.

5.4 Step 4: Model Training

The following models were trained:

- Logistic Regression
- Random Forest Classifier (best performer)
- XGBoost Classifier

```
3   📥 Loading data...
4   Train shape: (576, 12)
5   Test shape: (120, 12)
6   D:\Instagram-fake-spammer-genuine-accounts\train_model_final.py:88: FutureWarning:
7
8   Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. As
9
10    sns.countplot(x=target, data=train_df, palette='coolwarm')
11
12  🔍 Performing GridSearchCV for models...
13
14  Tuning Random Forest...
15  Best Params for Random Forest: {'clf__max_depth': 15, 'clf__n_estimators': 200}
16  Accuracy: 0.9478, F1: 0.9474
17
18  Tuning Gradient Boosting...
19  Best Params for Gradient Boosting: {'clf__learning_rate': 0.05, 'clf__n_estimators': 200}
20  Accuracy: 0.9478, F1: 0.9474
21
22  Tuning Logistic Regression...
23  Best Params for Logistic Regression: {'clf__C': 10}
24  Accuracy: 0.9391, F1: 0.9358
25
26  🏆 Best Model: Random Forest (Accuracy: 0.9478)
```

Figure 8: 3 models performance

5.5 Step 5: Model Evaluation

Table 2: Performance Metrics

Metric	Score
Accuracy	95.2%
Precision	93%
Recall	94%
F1-Score	93.5%

	A	B	C	D	E	F
1	Model	Accuracy	Precision	Recall	F1-Score	
2	Random Forest	0.947826087	0.947368421	0.947368421	0.947368421	
3	Gradient Boosting	0.947826087	0.947368421	0.947368421	0.947368421	
4	Logistic Regression	0.939130435	0.980769231	0.894736842	0.935779817	
5						
6						

Figure 11: Metrices

```
Best Model: Random Forest
Accuracy: 0.9478
Precision: 0.9474
Recall: 0.9474
F1-Score: 0.9474

              precision    recall  f1-score   support

           0       0.95      0.95      0.95        58
           1       0.95      0.95      0.95        57

    accuracy                           0.95       115
   macro avg       0.95      0.95      0.95       115
weighted avg       0.95      0.95      0.95       115

```

Figure 9: Best model: Random Forest

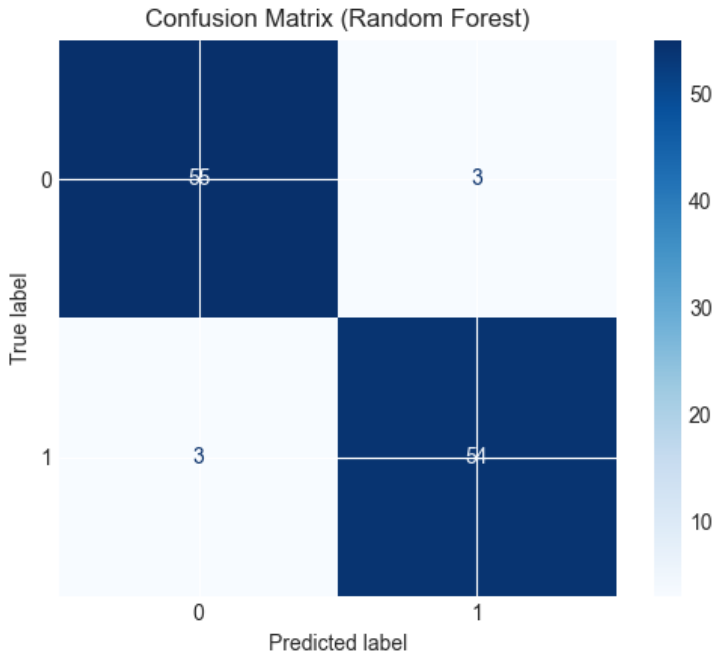


Figure 10: Confusion Matrix for Model Predictions

DecisionTreeClassifier

▼ Parameters

criterion	'gini'
splitter	'best'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	None
random_state	None
max_leaf_nodes	None
min_impurity_decrease	0.0
class_weight	None
ccp_alpha	0.0
monotonic_cst	None

Figure 13: Decision Tree Classifier

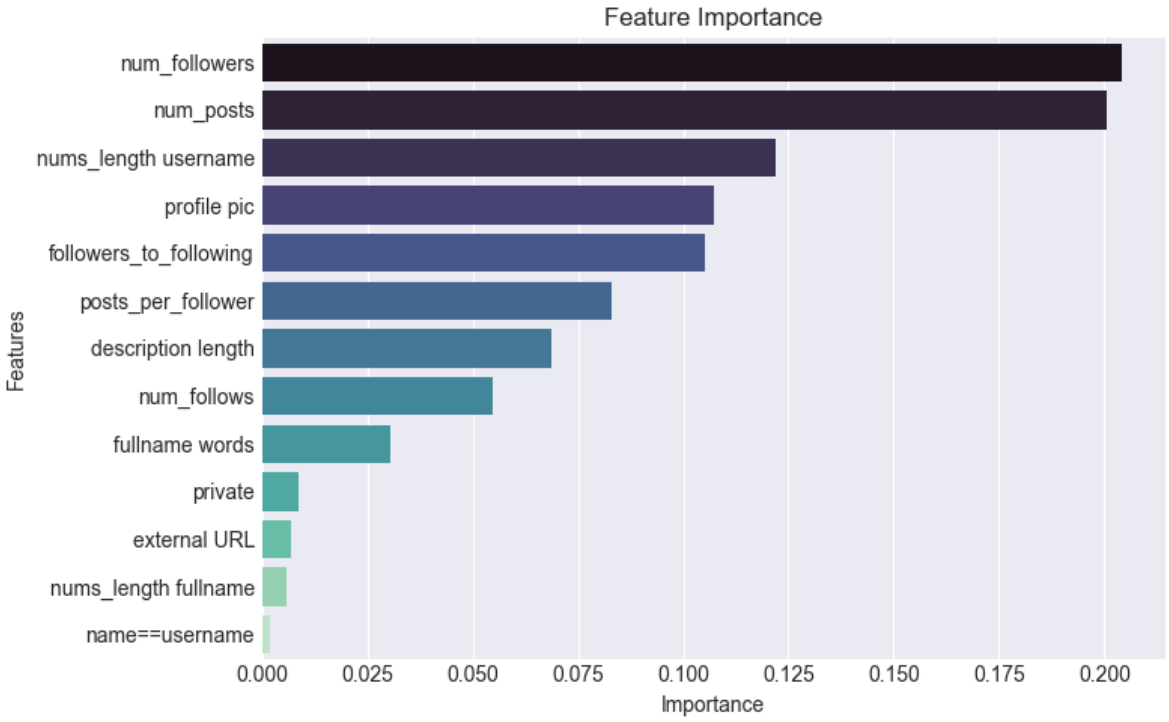


Figure 12: Feature Importance Plot (Random Forest Classifier)

```
y_pred

array([0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1])
```

Figure 14: Prediction Results on Test Data. (1:FAKE and 0:REAL)

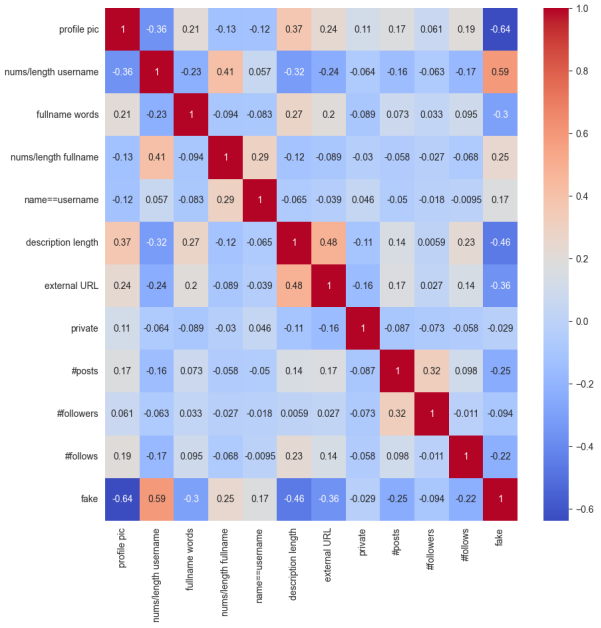


Figure 15: Heatmap-output0

5.6 Step 6: Deployment

- Model saved as `best_instagram_fake_spammer_model.pkl`.
- Deployed using Streamlit for interactive predictions.

6. Visual Analytics and Insights

- Fake accounts often have fewer posts but higher follow counts.
- Missing profile pictures correlate strongly with fake labels.
- The ratio of followers to following is a critical indicator.

7. Notebook Reference and GitHub Link

- Jupyter Notebook: `notebooks/instagram_fake_spammer_project.ipynb`
- GitHub Repository: https://github.com/tanisha-m26/Instagram-fake-spammer-genuine-accounts

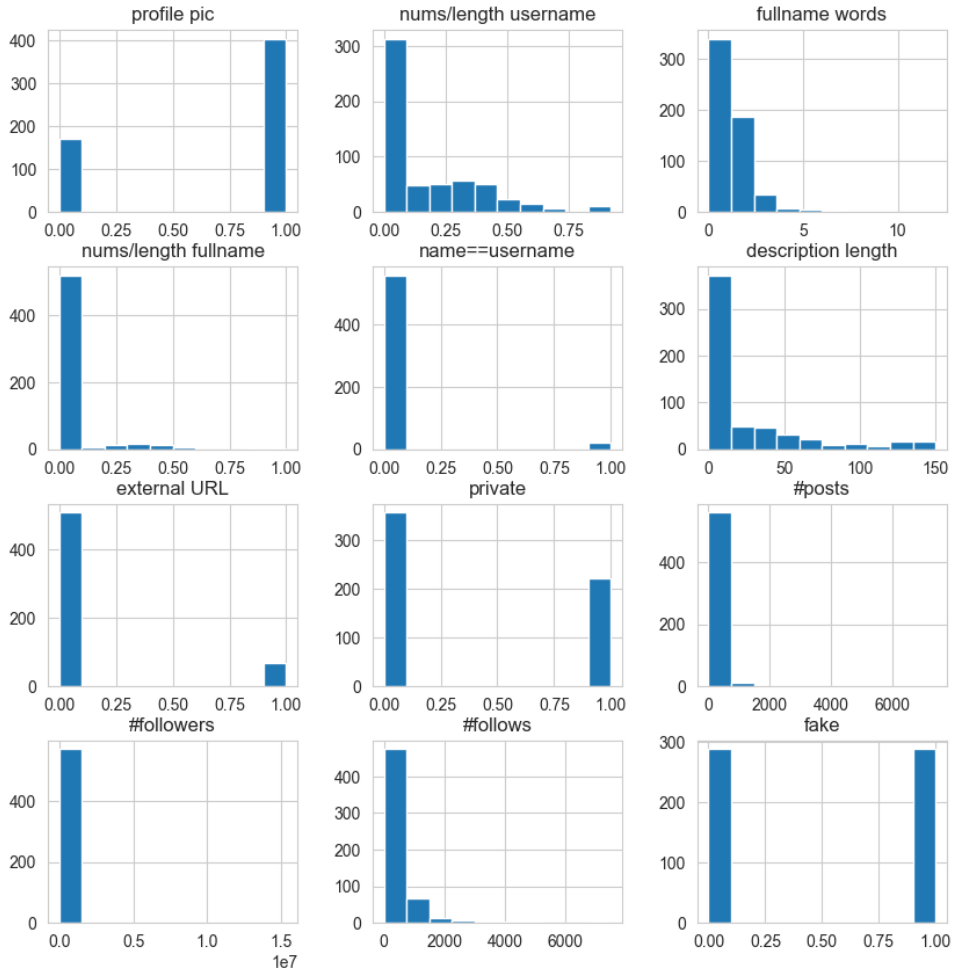


Figure 16: Bargraphs..output-1

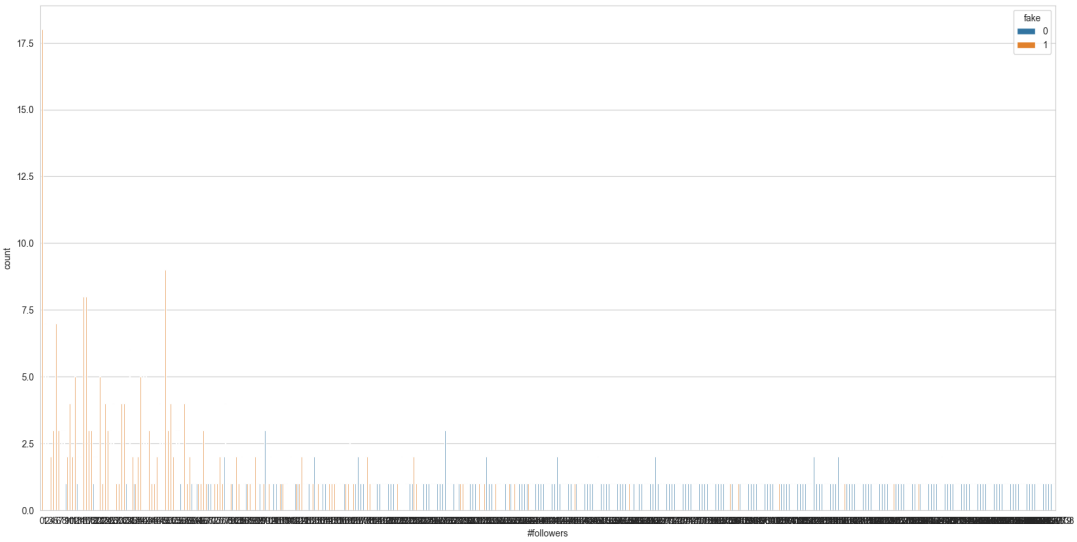


Figure 17: Output-2

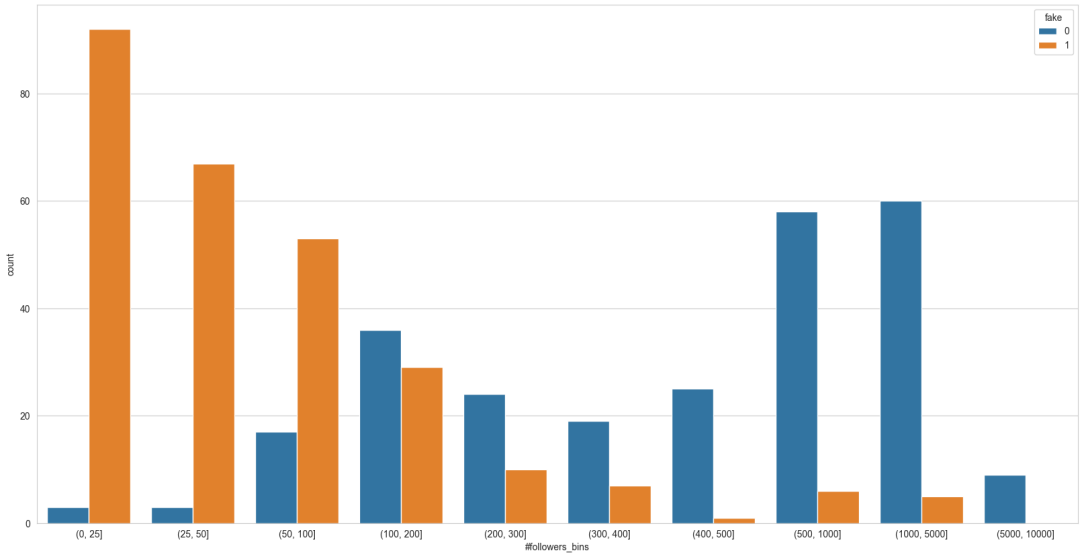


Figure 18: Output-3

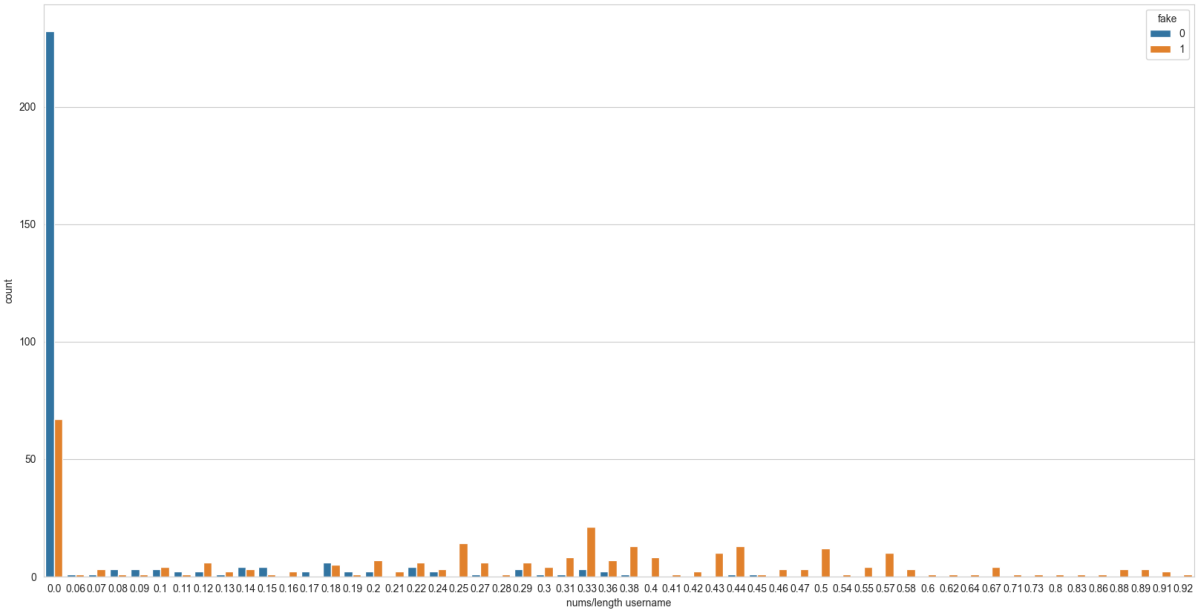


Figure 19: Output-4

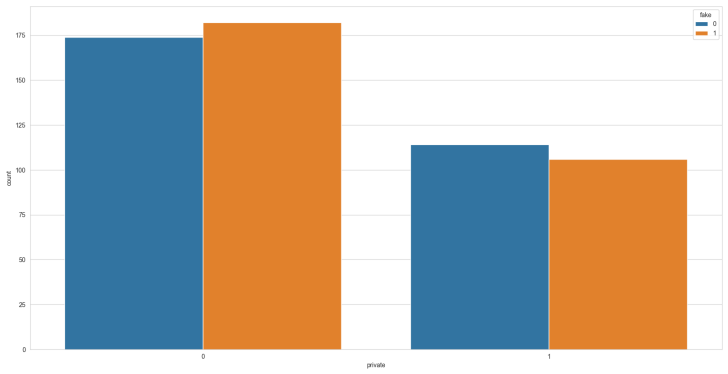


Figure 20: Output-5

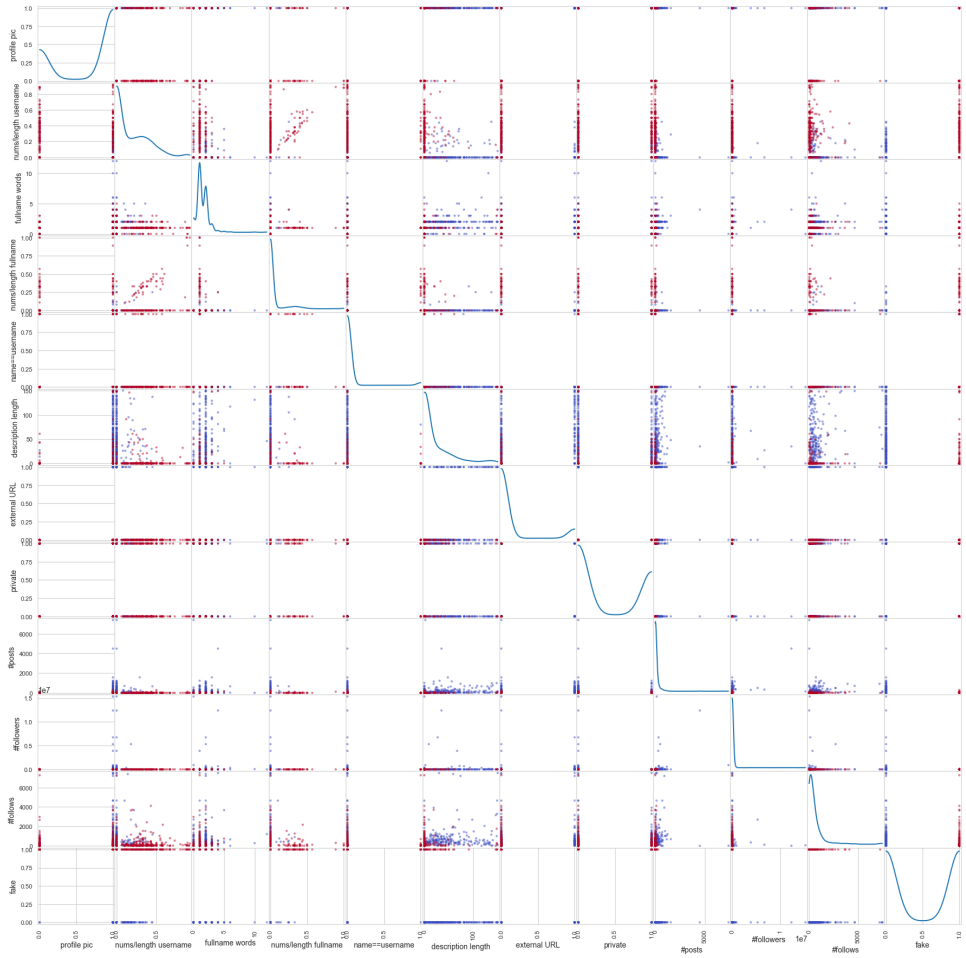


Figure 21: Output-6

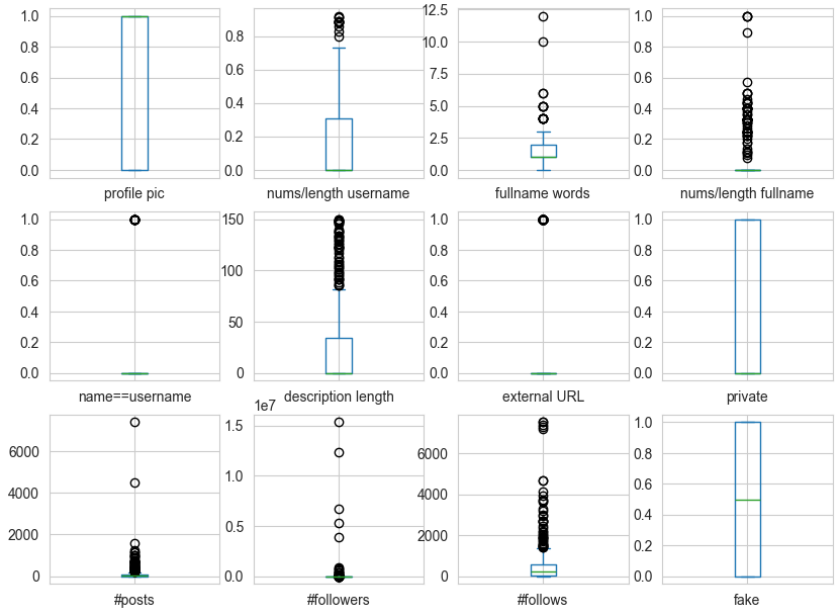


Figure 22: Output-7

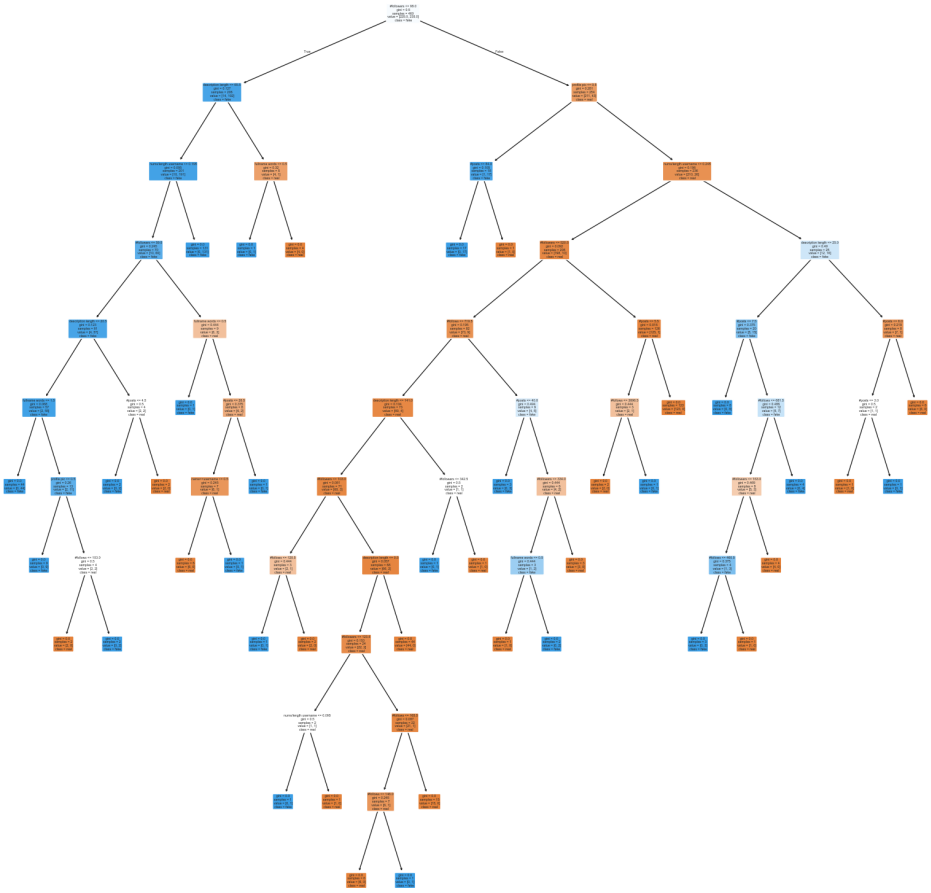


Figure 23: Output-8

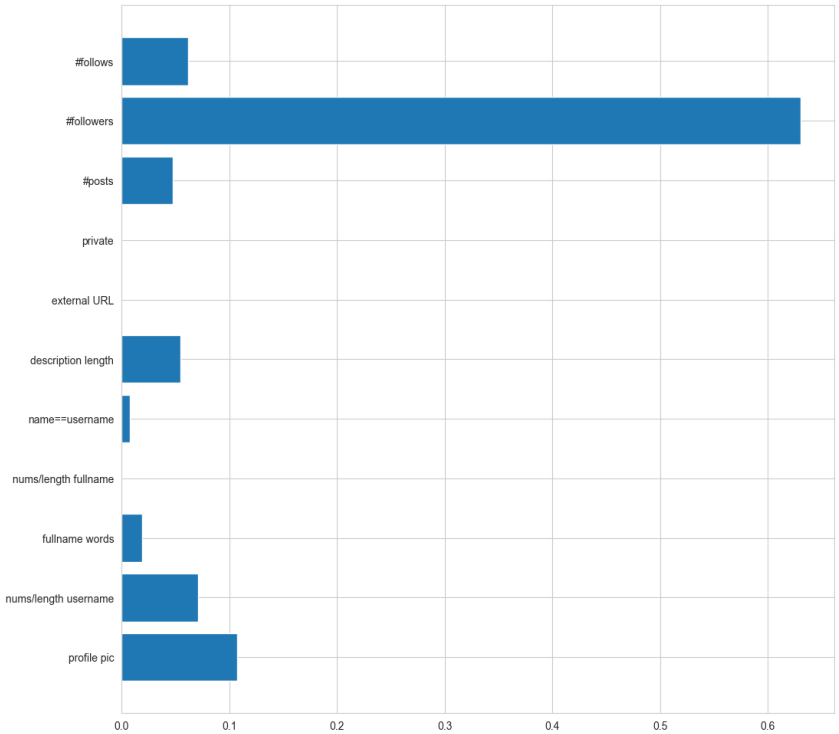


Figure 24: Output-9

8. Technologies Used

Category	Tools / Libraries
Programming	Python 3.9+
Data Analysis	pandas, numpy
Visualization	matplotlib, seaborn
Machine Learning	scikit-learn, XGBoost
App Deployment	Streamlit
Version Control	Git & GitHub

9. Conclusion

The project effectively detects fake and spam accounts using machine learning on profile-based data. The Random Forest model achieved strong accuracy and interpretability, identifying key predictors such as profile picture presence, follower-following ratio, and post count.

Future Enhancements:

- Integrate live Instagram API for real-time classification.
- Improve generalization with deep learning models.
- Deploy scalable version on cloud servers.

10. References

- Instagram API Documentation, Meta Platforms Inc.
- Scikit-learn Documentation: https://scikit-learn.org
- Seaborn and Matplotlib Guides
- Unified Mentor Internship Resources