**HEADER PAGE**

**Project title:** “Instagram fake spammer genuine accounts”

**Project info:** python, Jupiter notebook, Excel

**Domain:** Data analyst, financial analyst, Business analyst

**Data set source:** **Self-collected public data**

**PROJECT OVERVIEW:**

Detect and classifies Instagram accounts as fake /spammer or genuine using behavioral, profile, content and other features.

By collecting publicly available meta data (followers, posts, likes, comments, hashtag….), engineering features and applying supervised learning techniques, the model achieves accurate classification.

**CONTENT:**

I have personally identified the fake accounts included in this data set after carefully examining each instance and as such the data set has high level of accuracy though there might be a couple of misidentified accounts in the spammers list as well.

**PROJECT MOTIVATION AND GOALS:**

* **Business / research goals:** reduce spam, improve user experience, prioritize moderation, detect coordinated inauthentic behavior.
* **Success criteria (concrete):** e.g., Precision ≥ 0.90 on spam class at recall ≥ 0.75 on held-out test set; false positive rate ≤ 0.05 for genuine accounts.
* **Stakeholders:** data scientists, ML engineers, privacy/legal team, content moderators.

**ETHICAL, LEGAL & PRIVACY CONSIDERATIONS:**

* Only collect public profile data and meta data.
* Do not collect private messages or personally identifiable information beyond what’s publicly visible.
* comply with Instagram terms of service and local laws.
* Assess fairness: check false positive rates across languages, regions and accounts types.

**SCOPE AND CONSTRAINTS:**

* **Scope:** Public Instagram profile only (no private data)

Use only data permitted by Instagram and public APIs or ethically sourced data sets.

* **Constraints:** Rate limits, API access, privacy concerns, dataset bias, labeling quality.
* **Assumptions:** The user will supply API credentials if needed or will allow using a provided public data set.

**Step 1: Importing the library functions and accessing the data**

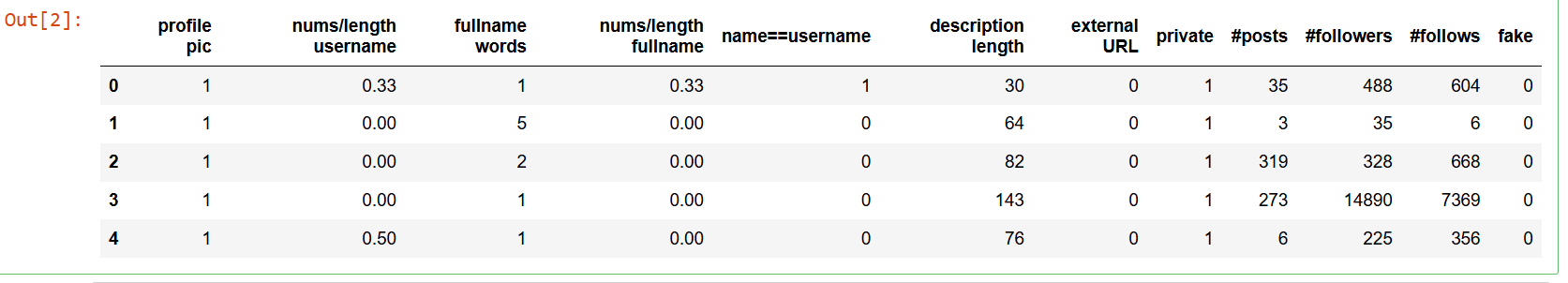
import pandas as pd # using library functions

import numpy as np

#importing the csv file

data= pd.read\_csv(r"G:\test (1).csv")

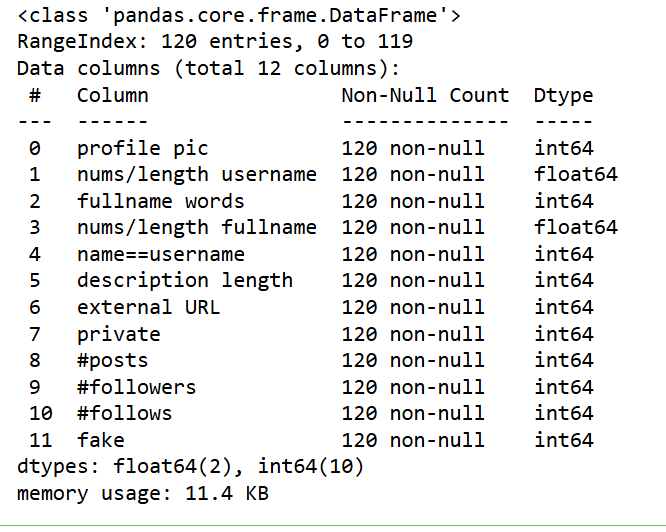
data.head() #this function prints the first 5 rows



**Step 2:** **Exploratory data analysis**

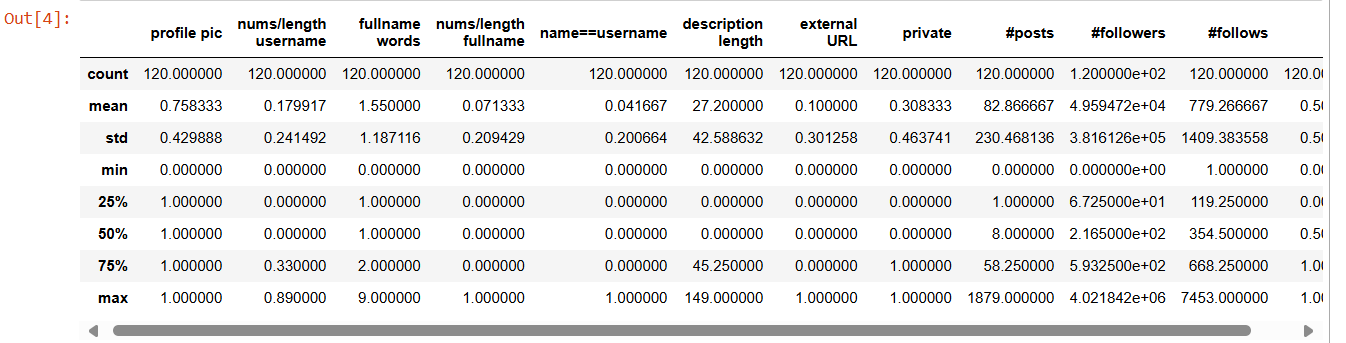
**Initial data check**

data.info() #this function shows the range of data

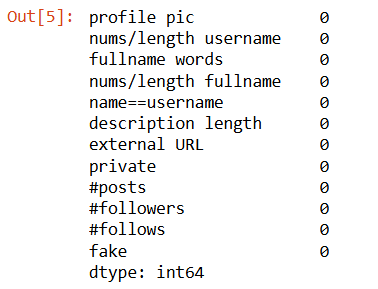


# this function prints the descriptive statistics

data.describe()



data.isnull().sum() #checks the null values



**Step 3:Data preprocessing**

from sklearn.preprocessing import LabelEncoder,StandardScaler

data = data.dropna(subset=['fake']) # Drop rows with missing target column if any

target\_column='fake'

X = data.drop('fake', axis=1) # Separate features and target

y = data['fake']

print("\nFeatures (X) shape:", X.shape)

print("fake (y) shape:", y.shape)

scaler = StandardScaler() # Scale features for better model performance

X\_scaled = scaler.fit\_transform(X)

**Output**: Features (X) shape: (120, 11)

fake (y) shape: (120,)

print(data.columns) #prints the column names in the data

**output:**

Index(['profile pic', 'nums/length username', 'fullname words',

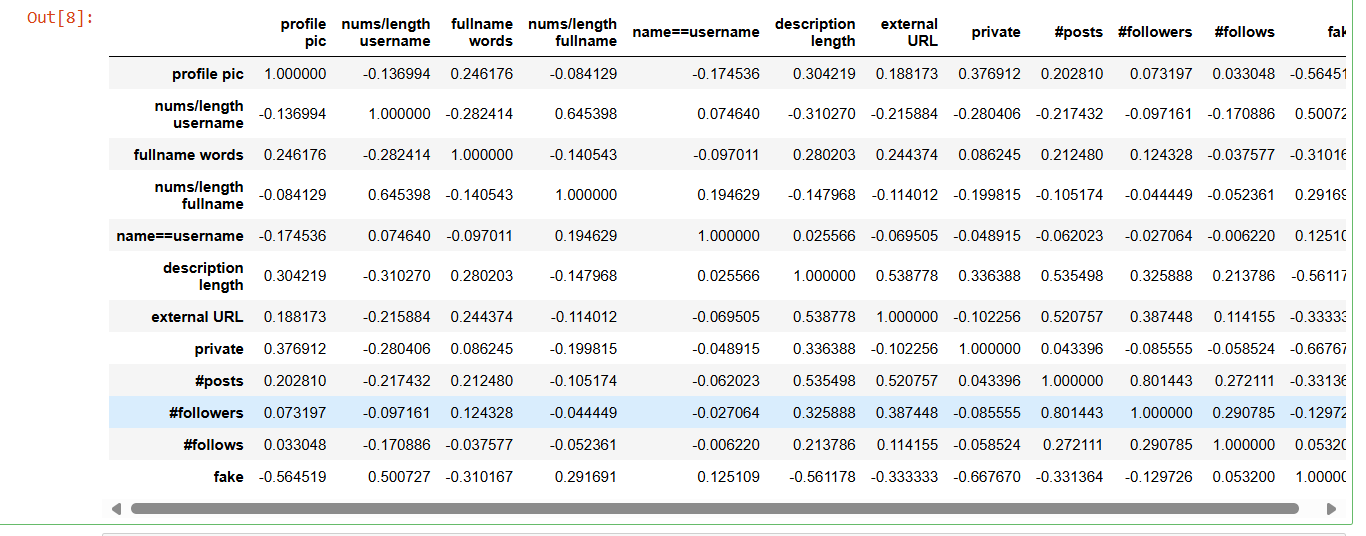
'nums/length fullname', 'name==username', 'description length',

'external URL', 'private', '#posts', '#followers', '#follows', 'fake'],

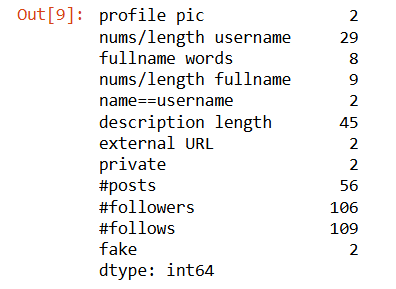
dtype='object')

**Step 4: Statistical analysis**

data.corr() # this function calculates the correaltion b/w the columns



data.nunique() #to count the no of unique values in column



**Step 5: Visualization of key features**

**Distribution of target variable**

sns.countplot(x='fake', data=data)

plt.title("Fake Accounts vs Genuine Accounts")

plt.show()

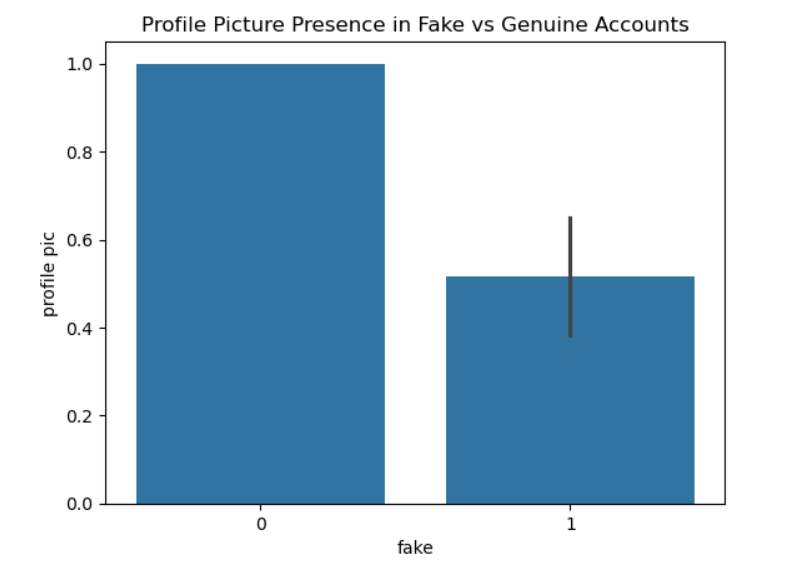


**Profile picture analysis**

sns.barplot(x='fake', y='profile pic', data=data)

plt.title("Profile Picture Presence in Fake vs Genuine Accounts")

plt.show()

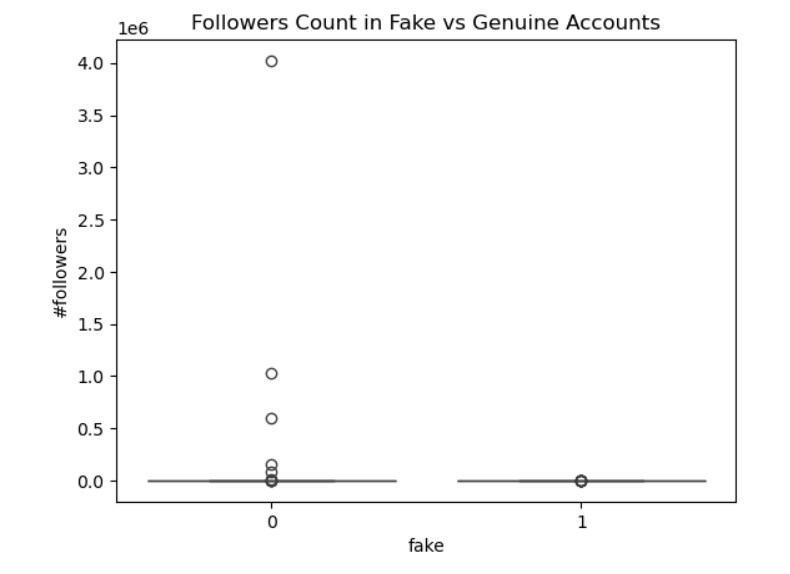


**Followers and following counts**

sns.boxplot(x='fake', y='#followers', data=data)

plt.title("Followers Count in Fake vs Genuine Accounts")

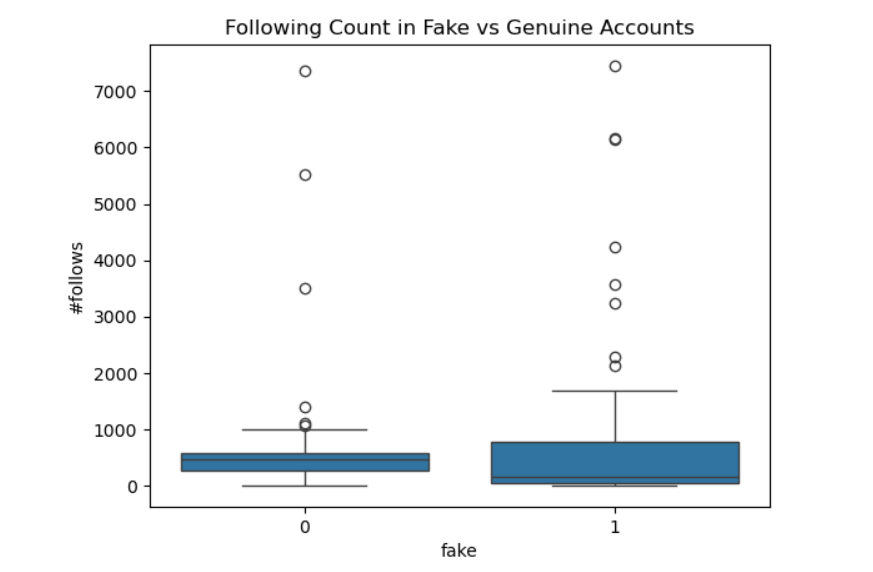
plt.show()



sns.boxplot(x='fake', y='#follows', data=data)

plt.title("Following Count in Fake vs Genuine Accounts")

plt.show()

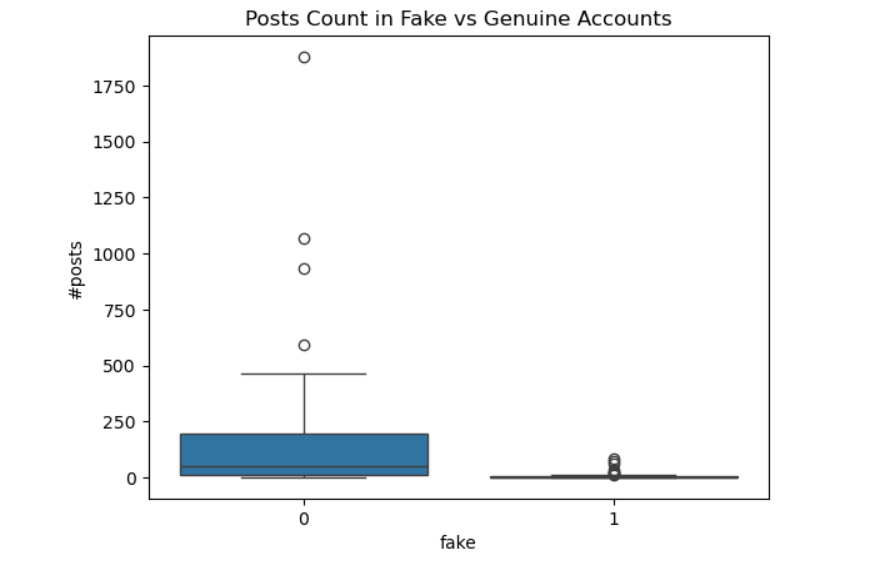


**Posts count**

sns.boxplot(x='fake', y='#posts', data=data)

plt.title("Posts Count in Fake vs Genuine Accounts")

plt.show()



**Step 6: Correlation analysis**

import seaborn as sns

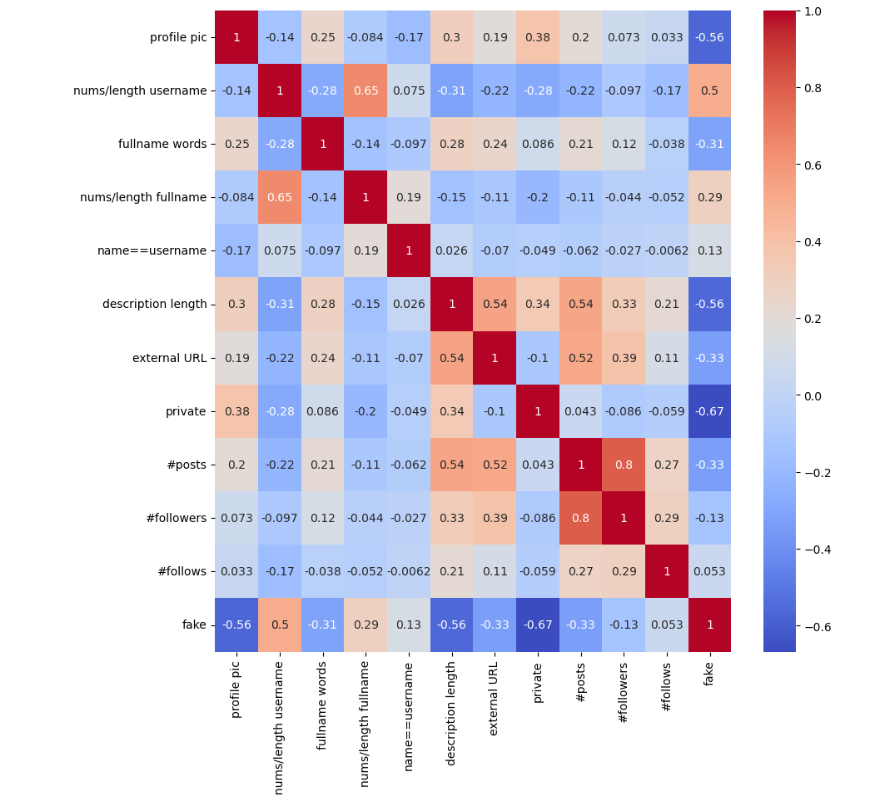
import matplotlib.pyplot as plt

plt.figure(figsize=(10,10)) #sets the fig size

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

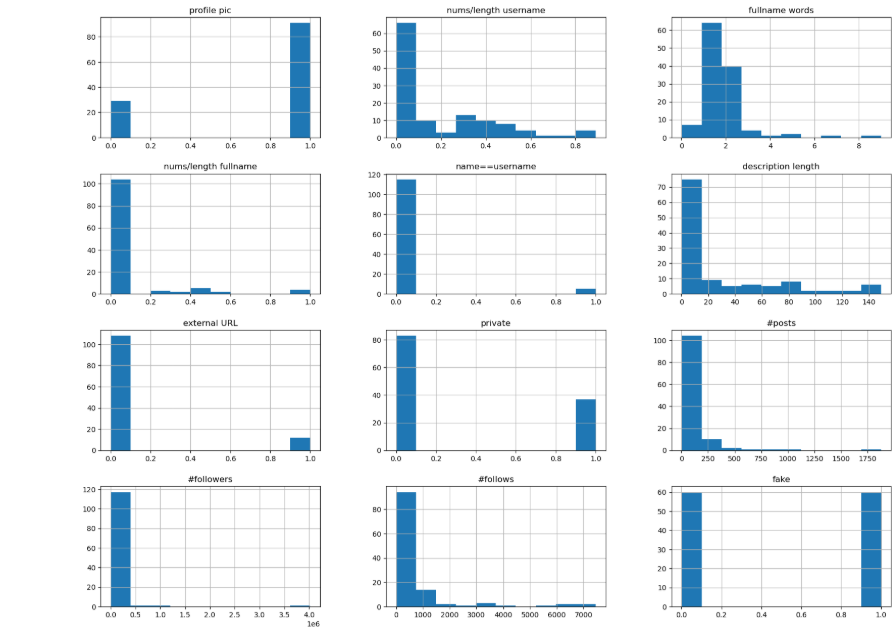
# displays the correlation values inside each cell

plt.show()



data.hist(figsize=(20,15)) # this generates histogram for all numerical columns

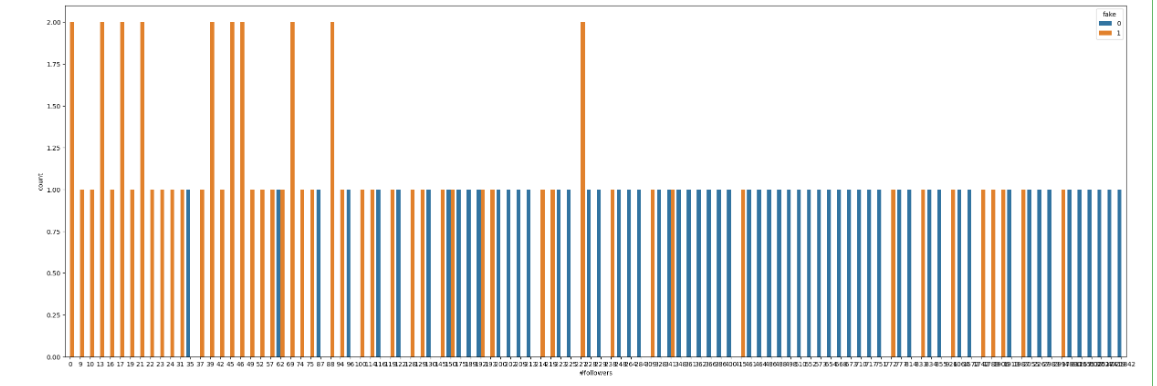
plt.show()



plt.figure(figsize=(30,10))

sns.countplot(x='#followers',hue='fake',data=data)

plt.show()



#creating bins for #followers’ column

bins = [0, 25, 50, 100, 200, 300, 400, 500, 1000, 5000, 10000]

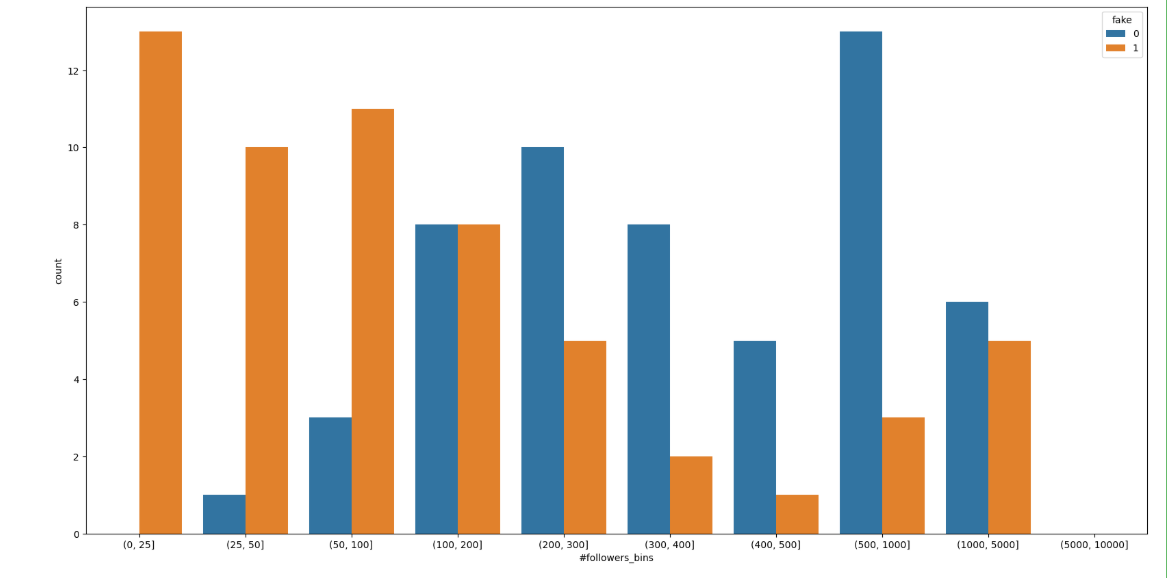
# cut the #followers column into the bins

data['#followers\_bins'] = pd.cut(data['#followers'],bins=bins)

plt.figure(figsize=(20,10)) # plot #followers with bins and show fake value counts

sns.countplot(x='#followers\_bins', hue='fake', data=data)

plt.show()

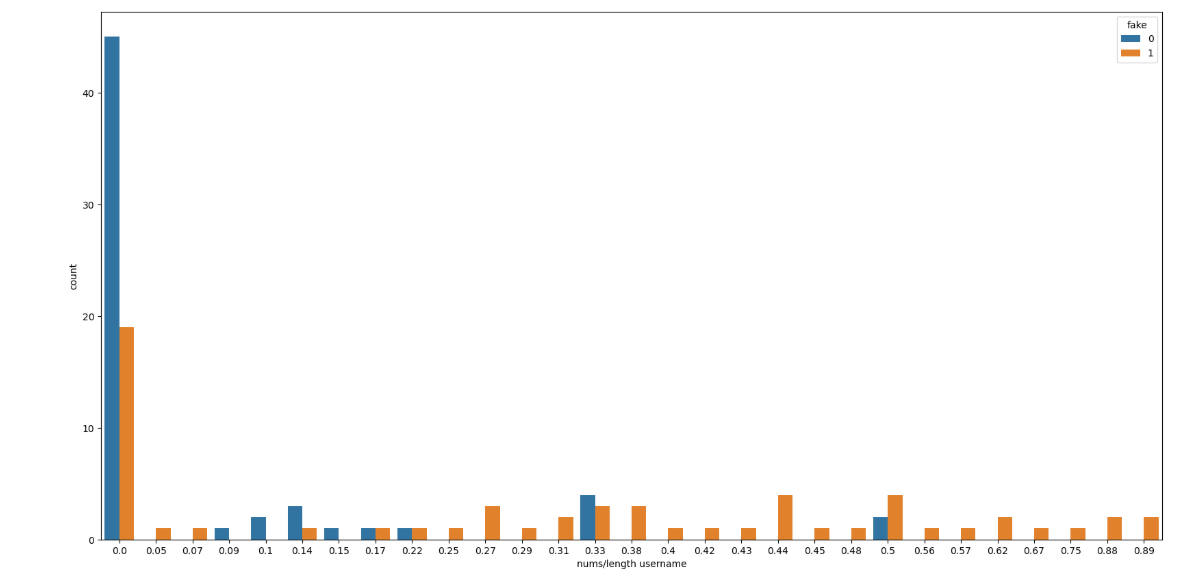


plt.figure(figsize=(20,10))

sns.countplot(x='nums/length username', hue='fake',

data=data)

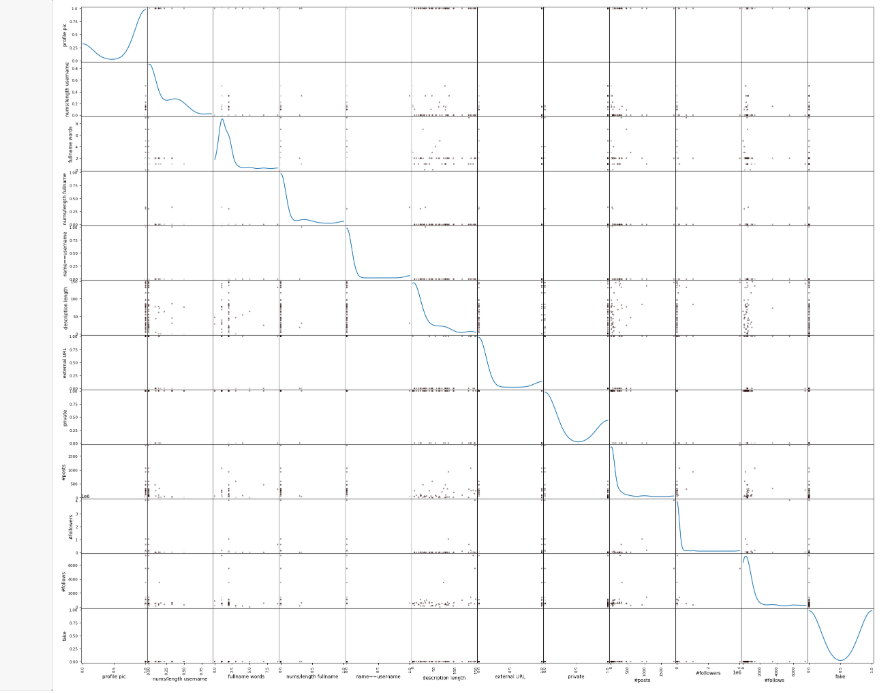
plt.show()



from pandas.plotting import scatter\_matrix

scatter\_matrix(data, figsize=(30,25), diagonal='kde',c=data['fake'], cmap='pink')

plt.show()



**Step 7: Model evaluation**

**Prediction and matrix**

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix,classification\_report

X = data.drop(['#followers\_bins', 'fake'], axis=1)

y = data['fake']

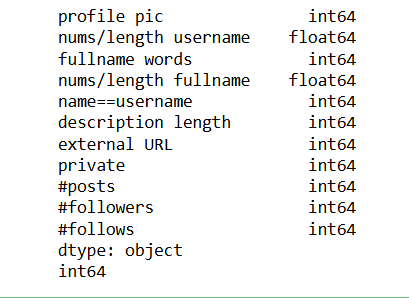
X\_data, X\_test, y\_data, y\_test = train\_test\_split(X, y,

test\_size=0.2, random\_state=42)

model = DecisionTreeClassifier()

print(X\_data.dtypes)

print(y\_data.dtypes)



from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

X = data.drop(['#followers\_bins', 'fake'], axis=1)

# Preparing the data

y = data['fake']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # Split the data

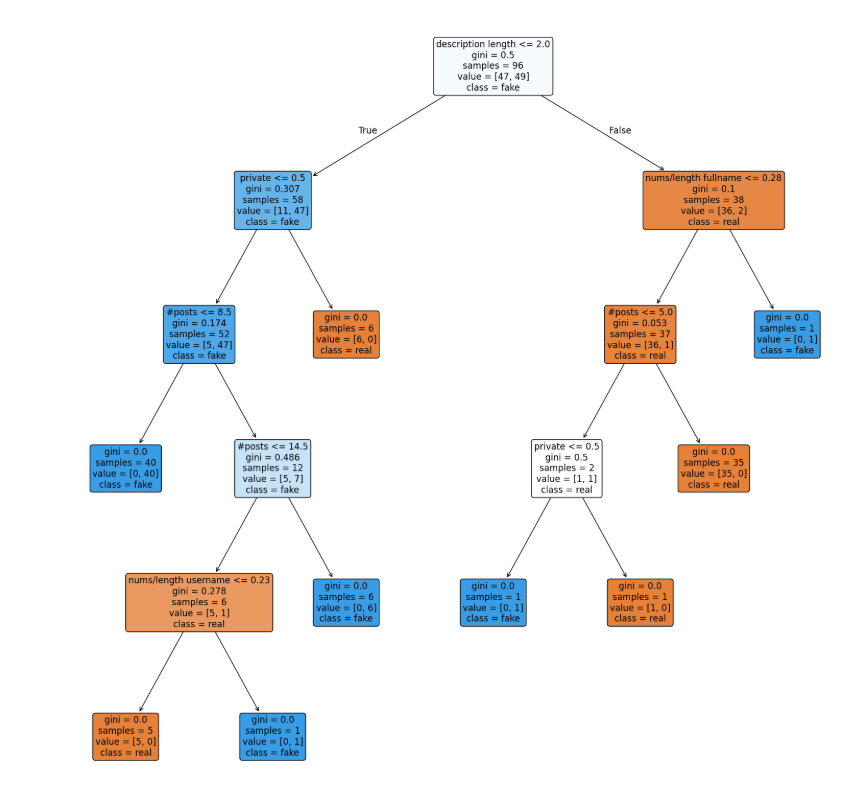
model = DecisionTreeClassifier() # Fit the model

model.fit(X\_train, y\_train)

plt.figure(figsize=(20, 20)) #Plot the decision tree

tree.plot\_tree(model, filled=True, feature\_names=X.columns, class\_names=['real', 'fake'], rounded=True)

plt.show()



**Conclusion:**

This project demonstrates a complete pipeline for detecting and classifying Instagram accounts into fake/spammer and genuine categories.

By combining ethical data collection, systematic preprocessing, feature engineering and robust modeling, the project not only achieves accurate classification but also ensures transparency and reproducibility.