Project Title	**Instagram fake spammer genuine accounts**
Tools	ML, Python, Tableau Desktop, SQL, Excel
Domain	Data Analyst, Finance Analyst, Business Analyst
Project Difficulties level	intermediate

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set]

About Dataset

Context

Fakes and spammers are a major problem on all social media platforms, including Instagram.

This is the subject of my final-year project in which I set out to find ways of detecting them using machine learning.

In this dataset fake and spammer are interchangeable terms.

Content

I have personally identified the spammer/fake accounts included in this dataset after carefully examining each instance and as such the dataset has high level of accuracy

though there might be a couple of misidentified accounts in the spammers list as well.
The dataset has been collected using a crawler from 15-19, March 2019.

Inspiration

This dataset could be further improved in quantity and quality measures, but how much accuracy can it achieve?
Possible ways of using the models to tackle the problem?

Example: You can get the basic idea how you can create a project from here

Project: Instagram Account Classification – Fake vs. Genuine Accounts

---

Step 1: Data Preparation

Import Libraries and Load Dataset

python

code

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix
```

```python
# Load the dataset
data = pd.read_csv('instagram_data.csv')  # replace with the path to your dataset
data.head()
```

Columns Overview:

- **profile_pic**: Binary (1 if the profile has a picture, 0 otherwise).
- **nums/length username**: Number of characters or numbers in the username.
- **fullname words**: Number of words in the full name.
- **nums/length fullname**: Character length and number count in the full name.
- **name == username**: Binary (1 if the username and full name are identical, 0 otherwise).
- **description length**: Character length of the bio/description.
- **external URL**: Binary (1 if an external URL is present, 0 otherwise).
- **private**: Binary (1 if the profile is private, 0 otherwise).
- **#posts**: Number of posts.
- **#followers**: Number of followers.
- **#follows**: Number of accounts followed.
- **fake**: Target label (1 if fake, 0 if genuine).

Step 2: Exploratory Data Analysis (EDA)

Initial Data Check

python

code

```python
# Check for missing values
data.isnull().sum()
```

```
# Basic statistics of each column
data.describe()
```

Distribution of Target Variable

python

code

```
# Plotting the distribution of fake and genuine accounts
sns.countplot(x='fake', data=data)
plt.title("Distribution of Fake vs Genuine Accounts")
plt.show()
```

Correlation Analysis

Check how features are correlated with each other and the target label:

python

code

```
# Correlation matrix
correlation = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()
```

Visualization of Key Features

Profile Picture (Fake vs. Genuine):

python

code

```
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:

python

code

```
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:

python

code

```
sns.boxplot(x='fake', y='#posts', data=data)
plt.title("Posts Count in Fake vs Genuine Accounts")
plt.show()
```

-

Step 3: Data Preprocessing

Feature Engineering

- Convert categorical features into numeric formats, if any.
- Scale or normalize features if necessary.

python

code

```
# Example: Feature Scaling (Optional)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_features = scaler.fit_transform(data.drop('fake',
axis=1))
scaled_data = pd.DataFrame(scaled_features,
columns=data.columns[:-1])
scaled_data['fake'] = data['fake']
```

Step 4: Model Building

We'll use a Random Forest classifier for this binary classification task due to its effectiveness in handling imbalanced data and feature importance analysis.

python

code

```
# Split data into training and test sets
```

```
X = scaled_data.drop('fake', axis=1)
y = scaled_data['fake']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Build Random Forest Model
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Feature Importance Plot
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
sns.barplot(y=X.columns[indices], x=importances[indices],
palette='viridis')
plt.show()
```

Step 5: Model Evaluation

Predictions and Metrics

python

code

```
# Make predictions
y_pred = model.predict(X_test)
```

```python
# Evaluate the model
print("Classification Report:\n", classification_report(y_test,
y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Visualize Confusion Matrix

python

code

```python
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_predictions(y_test, y_pred,
display_labels=['Genuine', 'Fake'], cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

Step 6: Interpretation and Insights

- **Feature Importance**: Analyze the top features contributing to the classification.
    - For example, if **profile_pic** and **#followers** are among the most important features, this can indicate that fake accounts often lack profile pictures and tend to have suspiciously high or low follower counts.
- **Model Performance**: Evaluate the model accuracy, precision, recall, and F1-score from the classification report to understand how well it distinguishes between fake and genuine accounts.

Step 7: Future Improvements

To enhance the model, consider:

- Using more complex models like Gradient Boosting or XGBoost.
- Tuning hyperparameters with GridSearchCV.
- Applying techniques to handle imbalanced classes if fake accounts are less frequent.

Example: You can get the basic idea how you can create a project from here

Sample Project code and output

```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
```

```python
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/instagram-fake-spammer-genuine-accounts/train.csv
/kaggle/input/instagram-fake-spammer-genuine-accounts/test.csv
```

In [2]:

```python
df_train = pd.read_csv('/kaggle/input/instagram-fake-spammer-genuine-accounts/train.csv')
```

In [3]:

```
df_train.head()
```

Out[3]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
0	1	0.27	0	0.0	0	53	0	0	32	1000	955	0
1	1	0.00	2	0.0	0	44	0	0	286	2740	533	0
2	1	0.10	2	0.0	0	0	0	1	13	159	98	0
3	1	0.00	1	0.0	0	82	0	0	679	414	651	0

4	1	0.00	2	0.0	0	0	0	1	6	151	126	0

In [4]:

```
df_train.shape
```

Out[4]:

```
(576, 12)
```

In [5]:

```
df_train.info()
```

```
<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
```

In [6]:

```
df_train.describe()
```

Out[6]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
co	576.000	576.000	576.000	576.000	576.00	576.000	576.000	576.000	576.000	5.760000	576.000	576.000

unt	000	000	000	000	0000	000	000	000	000	e+02	000	000
mean	0.701389	0.163837	1.460069	0.036094	0.034722	22.623264	0.116319	0.381944	107.489583	8.530724e+04	508.381944	0.500000
std	0.458047	0.214096	1.052601	0.125121	0.183234	37.702987	0.320886	0.486285	402.034431	9.101485e+05	917.981239	0.500435
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000
25%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.900000e+01	57.500000	0.000000
50	1.00000	0.00000	1.00000	0.00000	0.0000	0.00000	0.00000	0.00000	9.00000	1.505000	229.500	0.50000

%	0	0	0	0	00	0	0	0	0	e+02	000	0
75%	1.000000	0.310000	2.000000	0.000000	0.000000	34.000000	0.000000	1.000000	81.500000	7.160000e+02	589.500000	1.000000
max	1.000000	0.920000	12.000000	1.000000	1.000000	150.000000	1.000000	1.000000	7389.000000	1.533854e+07	7500.000000	1.000000

In [7]:

```
df_train.isnull().sum()
```

Out[7]:

```
profile pic             0
nums/length username    0
fullname words          0
nums/length fullname    0
name==username          0
description length      0
external URL            0
private                 0
```

```
#posts                  0
#followers              0
#follows                0
fake                    0

dtype: int64
```

In [8]:

```python
df_train['fake'].value_counts()
```

Out[8]:

```
fake
0    288
1    288

Name: count, dtype: int64
```

In [9]:

```python
df_train.nunique()
```

Out[9]:

```
profile pic               2
nums/length username     54
fullname words            9
nums/length fullname     25
```

```
name==username          2
description length    104
external URL            2
private                 2
#posts                193
#followers            372
#follows              400
fake                    2

dtype: int64
```

In [10]:

```
df_train.corr()
```

Out[10]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followers	#follows	fake
profile	1.0000	-0.3640	0.2132	-0.1317	-0.124	0.3678	0.2367	0.1147	0.1695	0.0611	0.1948	-0.637

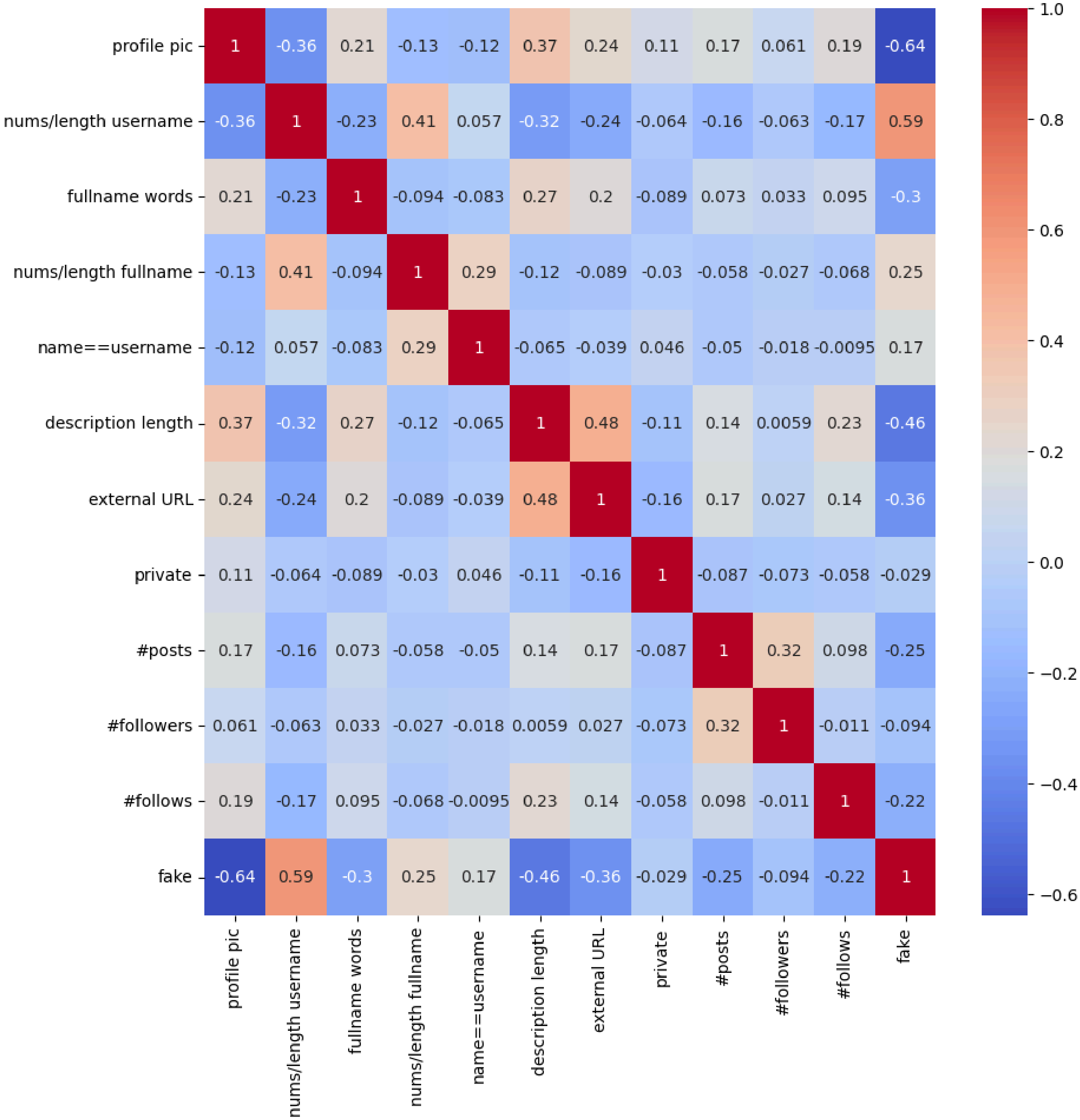
pic	00	87	95	56	903	92	29	32	70	37	33	315
nums/length username	-0.364087	1.000000	-0.225472	0.408567	0.056890	-0.321170	-0.237125	-0.063713	-0.157442	-0.062785	-0.172413	0.587687
fullname words	0.213295	-0.225472	1.000000	-0.094348	-0.082969	0.272522	0.196562	-0.089070	0.073350	0.033225	0.094855	-0.298793
nums/length fullname	-0.131756	0.408567	-0.094348	1.000000	0.291149	-0.117521	-0.088724	-0.030030	-0.057716	-0.027035	-0.067971	0.246782
name==username	-0.124903	0.056890	-0.082969	0.291149	1.000000	-0.064814	-0.039232	0.046084	-0.049808	-0.017761	-0.009529	0.170695
description	0.3678	-0.3211	0.2725	-0.1175	-0.064814	1.0000	0.4823	-0.110	0.1448	0.0059	0.2265	-0.460

length	92	70	22	21		00	13	329	24	29	61	825
external URL	0.236729	-0.237125	0.196562	-0.088724	-0.039232	0.482313	1.000000	-0.162612	0.165008	0.027189	0.142519	-0.362809
private	0.114732	-0.063713	-0.089070	-0.030030	0.046084	-0.110329	-0.162612	1.000000	-0.087495	-0.073473	-0.057542	-0.028586
#posts	0.169570	-0.157442	0.073350	-0.057716	-0.049808	0.144824	0.165008	-0.087495	1.000000	0.321385	0.098225	-0.245355
#followers	0.061137	-0.062785	0.033225	-0.027035	-0.017761	0.005929	0.027189	-0.073473	0.321385	1.000000	-0.011066	-0.093689
#follows	0.194833	-0.172413	0.094855	-0.067971	-0.009529	0.226561	0.142519	-0.057542	0.098225	-0.011066	1.000000	-0.224835

fake	-0.637315	0.587687	-0.298793	0.246782	0.170695	-0.460825	-0.362809	-0.028586	-0.245355	-0.093689	-0.224835	1.000000

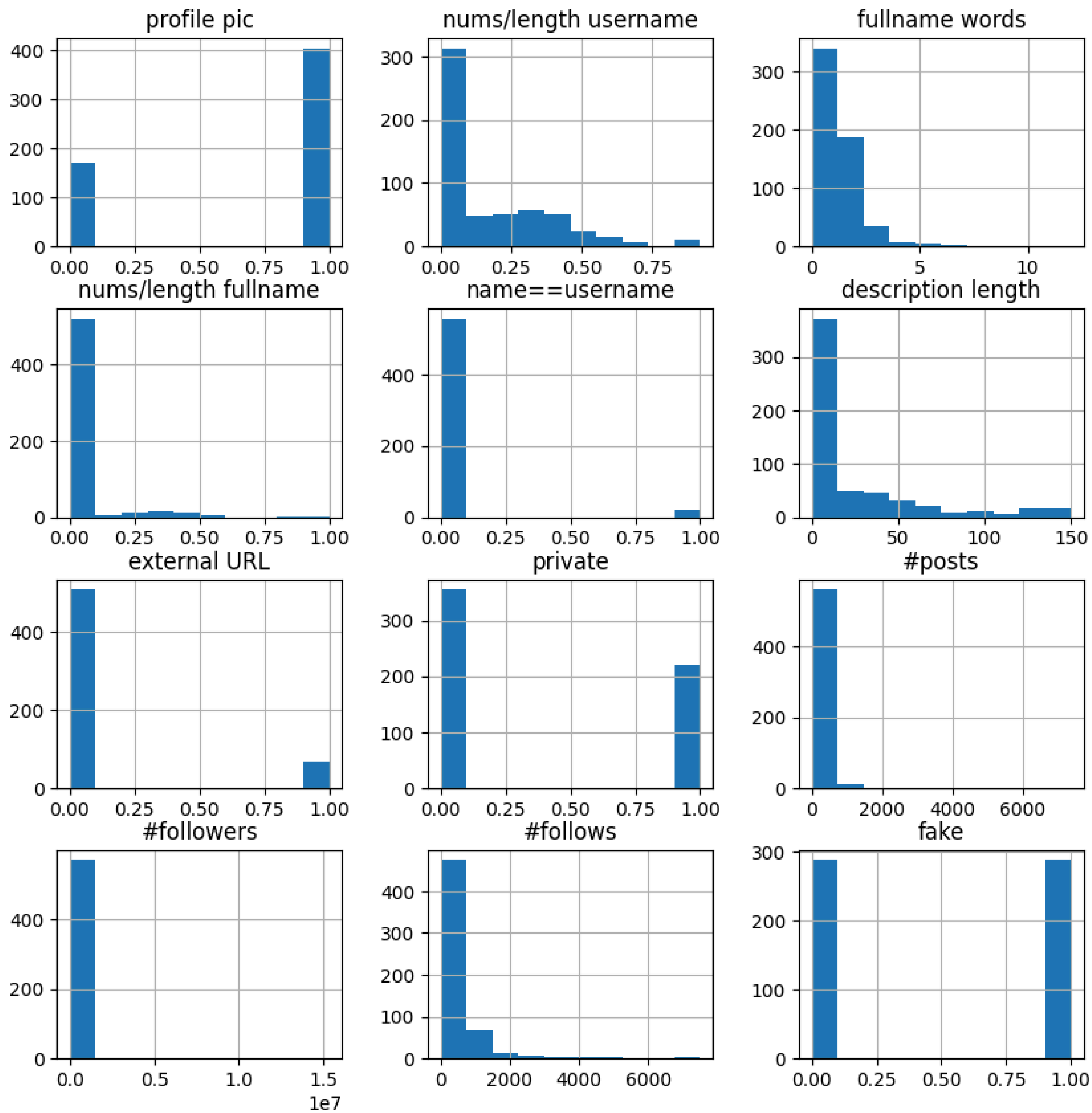
In [11]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
sns.heatmap(df_train.corr(), annot=True, cmap='coolwarm')

plt.show()
```



In [12]:

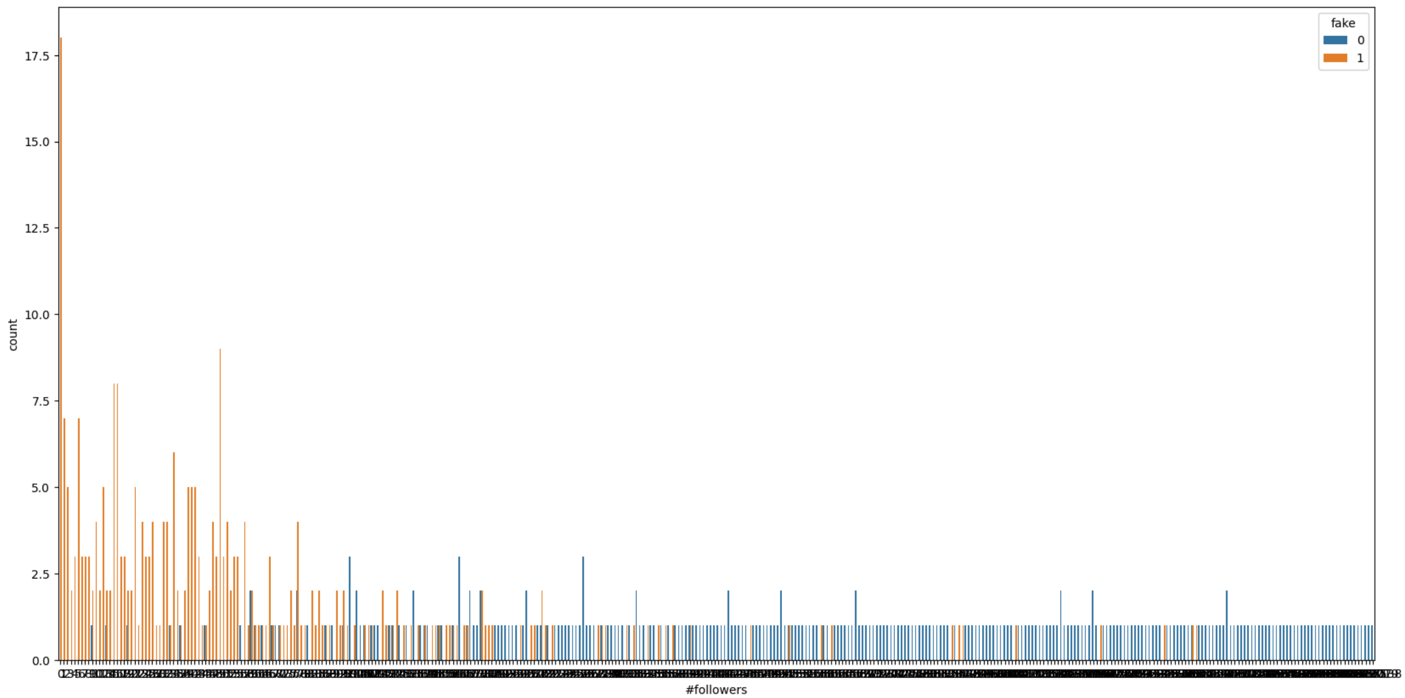
```
df_train.hist(figsize=(10,10))

plt.show()
```



In [13]:

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

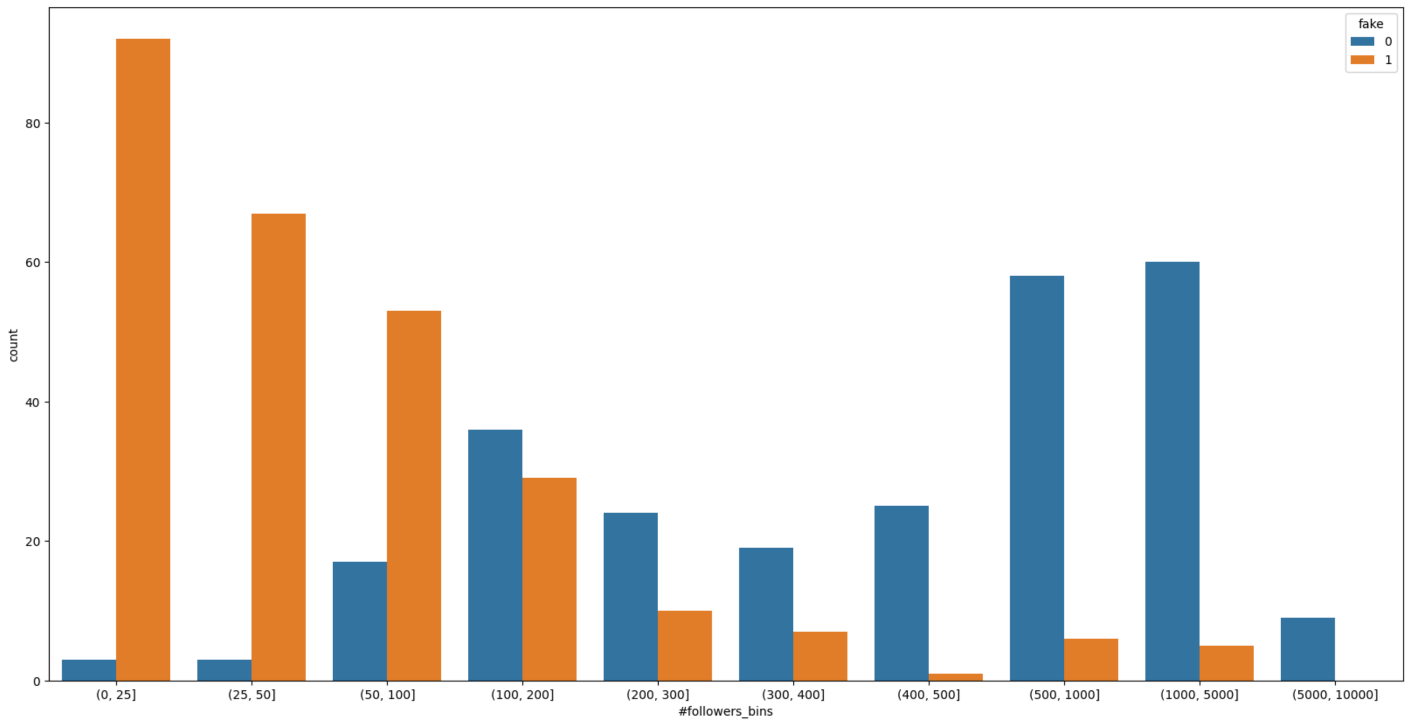
```
sns.countplot(x='#followers', hue='fake', data=df_train)
plt.show()
```



In [14]:

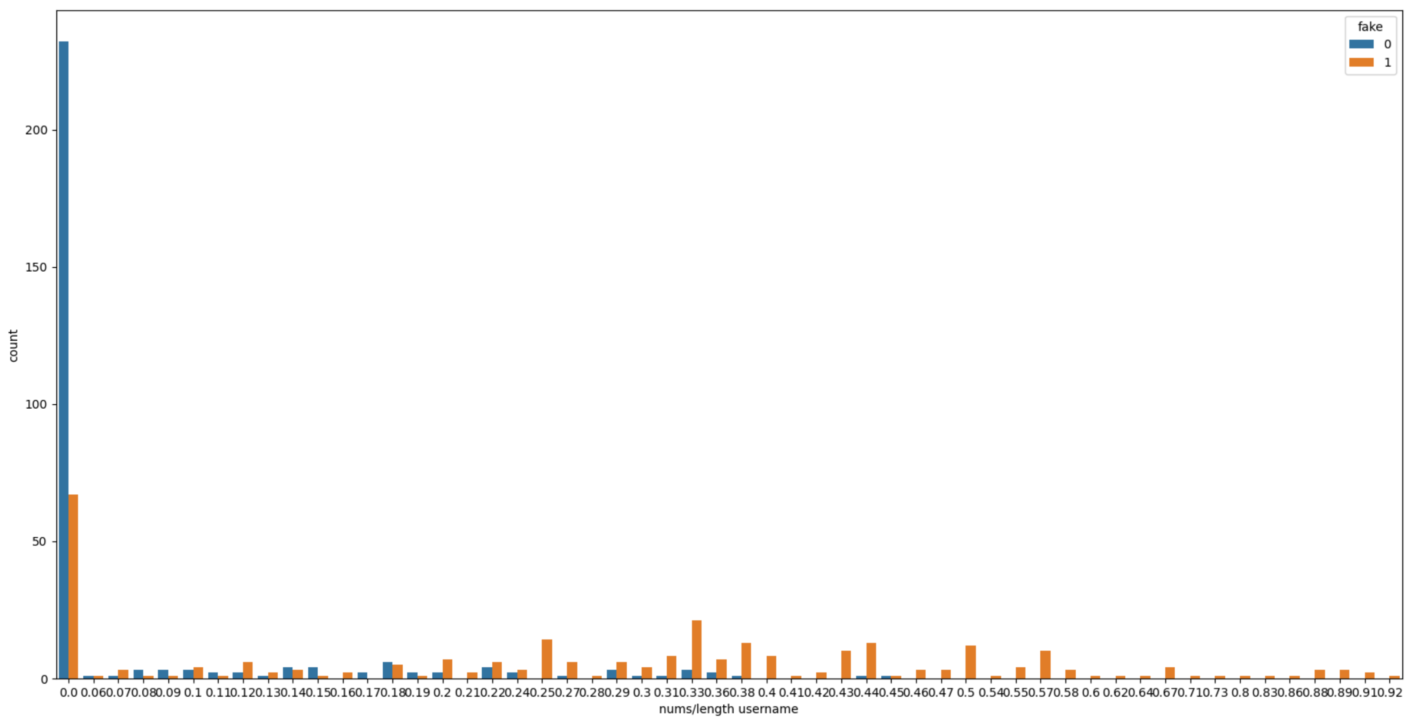
```
# create bins for #followers column
bins = [0, 25,  50, 100, 200, 300, 400, 500, 1000, 5000, 10000]

# cut the #followers column into the bins
df_train['#followers_bins'] = pd.cut(df_train['#followers'],
bins=bins)

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

```
sns.countplot(x='#followers_bins', hue='fake', data=df_train)
plt.show()
```



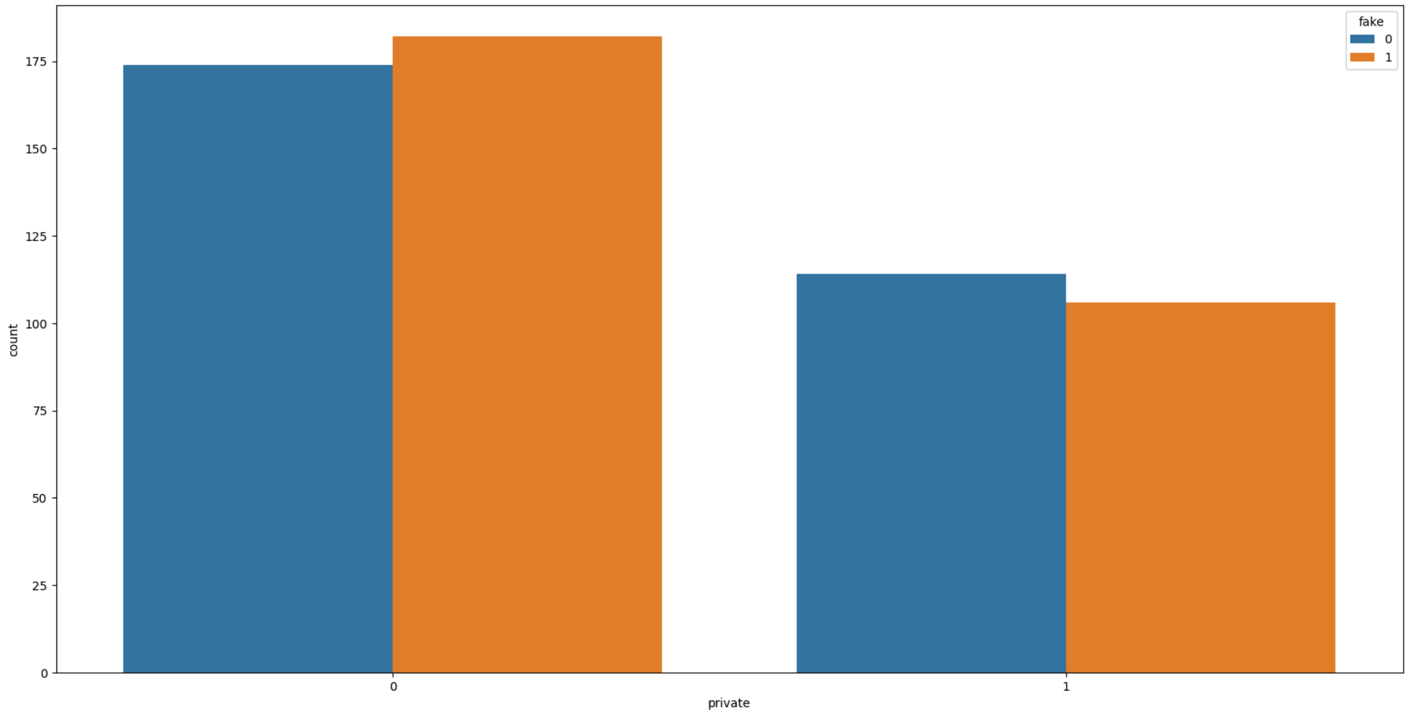
In [15]:

```
# plot nums/length username and show fake value counts
plt.figure(figsize=(20,10))
sns.countplot(x='nums/length username', hue='fake',
data=df_train)
plt.show()
```



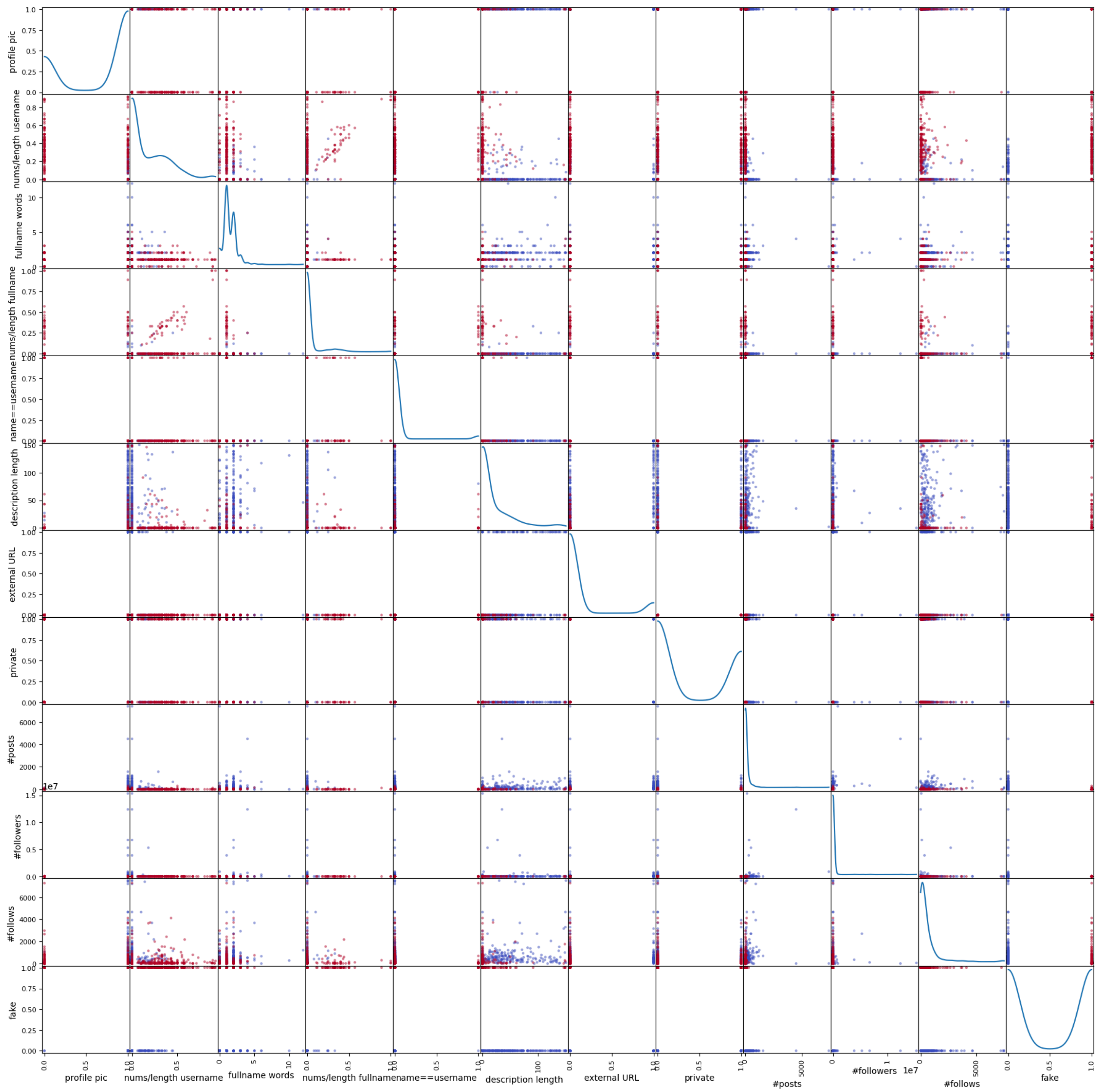
In [16]:

```python
# how is fake distributed amongst private accounts
plt.figure(figsize=(20,10))
sns.countplot(x='private', hue='fake', data=df_train)
plt.show()
```



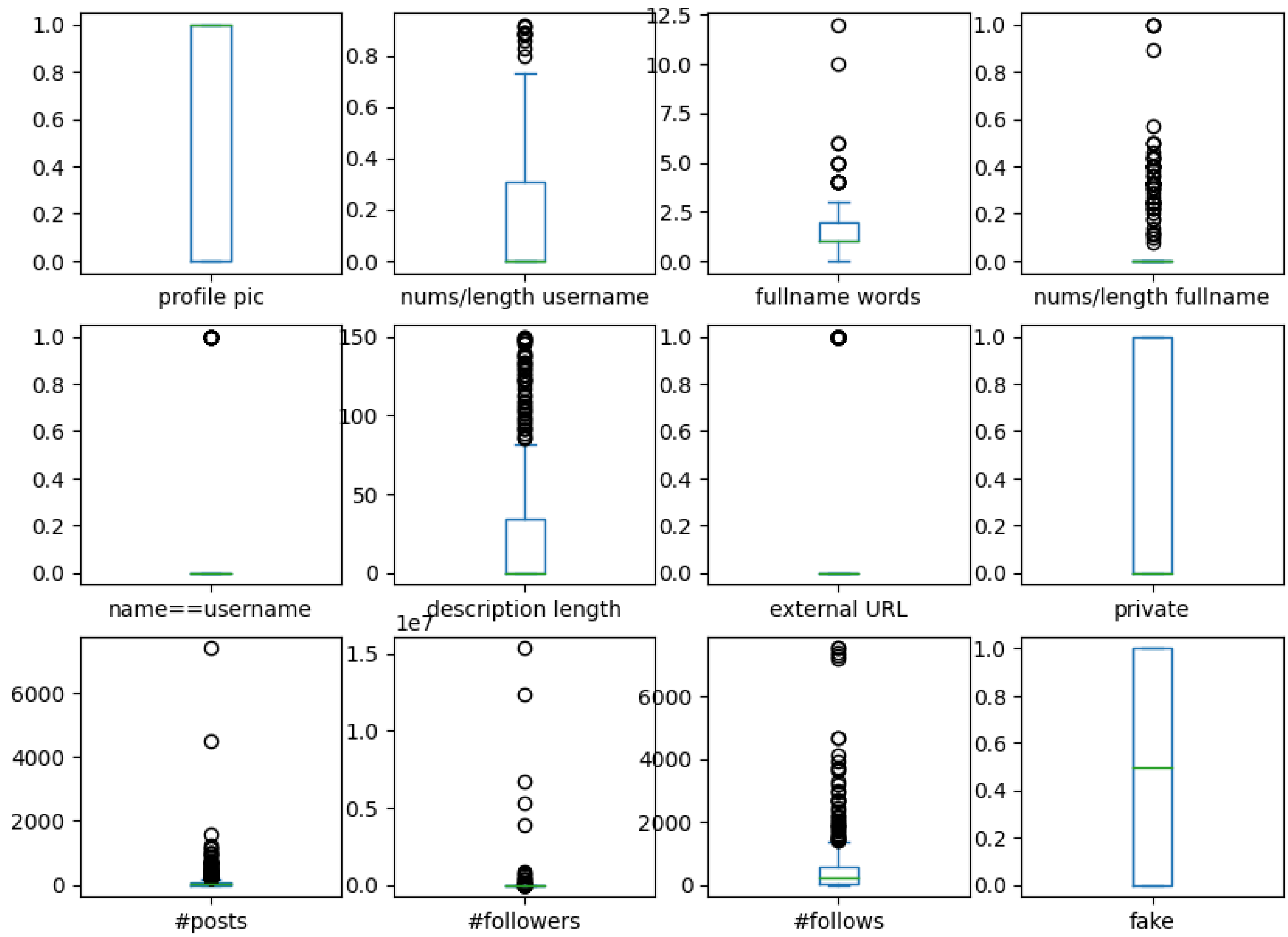
In [17]:

```
from pandas.plotting import scatter_matrix
scatter_matrix(df_train, figsize=(22,22), diagonal='kde',
c=df_train['fake'], cmap='coolwarm')



plt.show()
```



In [18]:

```
df_train.plot(kind='box', subplots=True, layout=(4,4),
figsize=(10,10))


plt.show()
```



In [ ]:

In [19]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
```

```
classification_report
```

In [20]:

```
X = df_train.drop(['#followers_bins', 'fake'], axis=1)
y = df_train['fake']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

In [21]:

```
model = DecisionTreeClassifier()
```

In [22]:

```
print(X_train.dtypes)
print(y_train.dtypes)
```

```
profile pic                int64
nums/length username     float64
fullname words             int64
nums/length fullname     float64
name==username             int64
```

```
description length        int64
external URL              int64
private                   int64
#posts                    int64
#followers                int64
#follows                  int64
dtype: object
int64
```

In [23]:

```
model.fit(X_train, y_train)
```

Out[23]:

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [24]:

```
y_pred = model.predict(X_test)
```

In [25]:

```
accuracy_score(y_test, y_pred)
```

Out[25]:

```
0.8706896551724138
```

In [26]:

```
confusion_matrix(y_test, y_pred)
```

Out[26]:

```
array([[57,  6],
       [ 9, 44]])
```

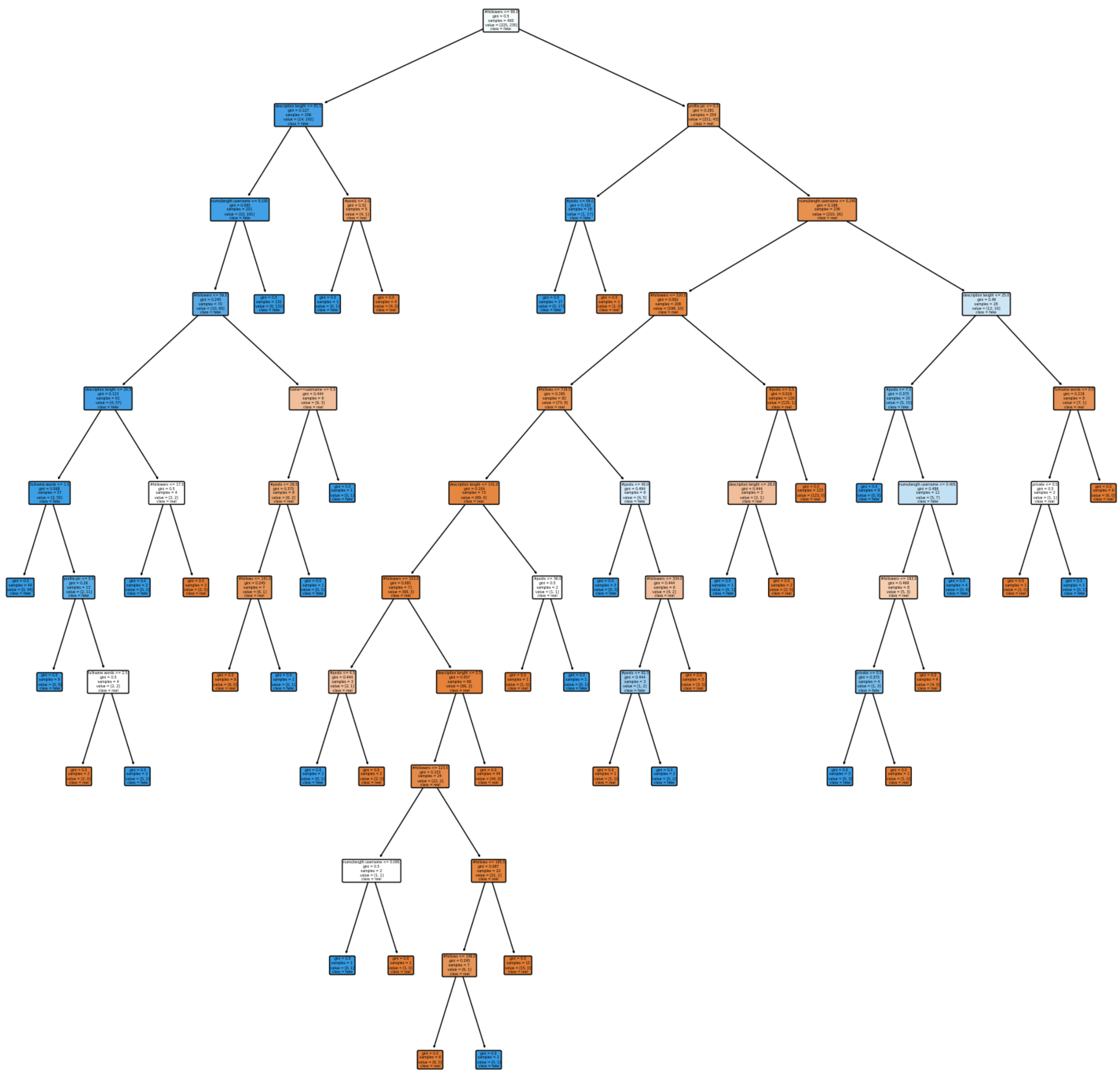
In [27]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        63
           1       0.88      0.83      0.85        53

    accuracy                           0.87       116
   macro avg       0.87      0.87      0.87       116
weighted avg       0.87      0.87      0.87       116
```

In [28]:

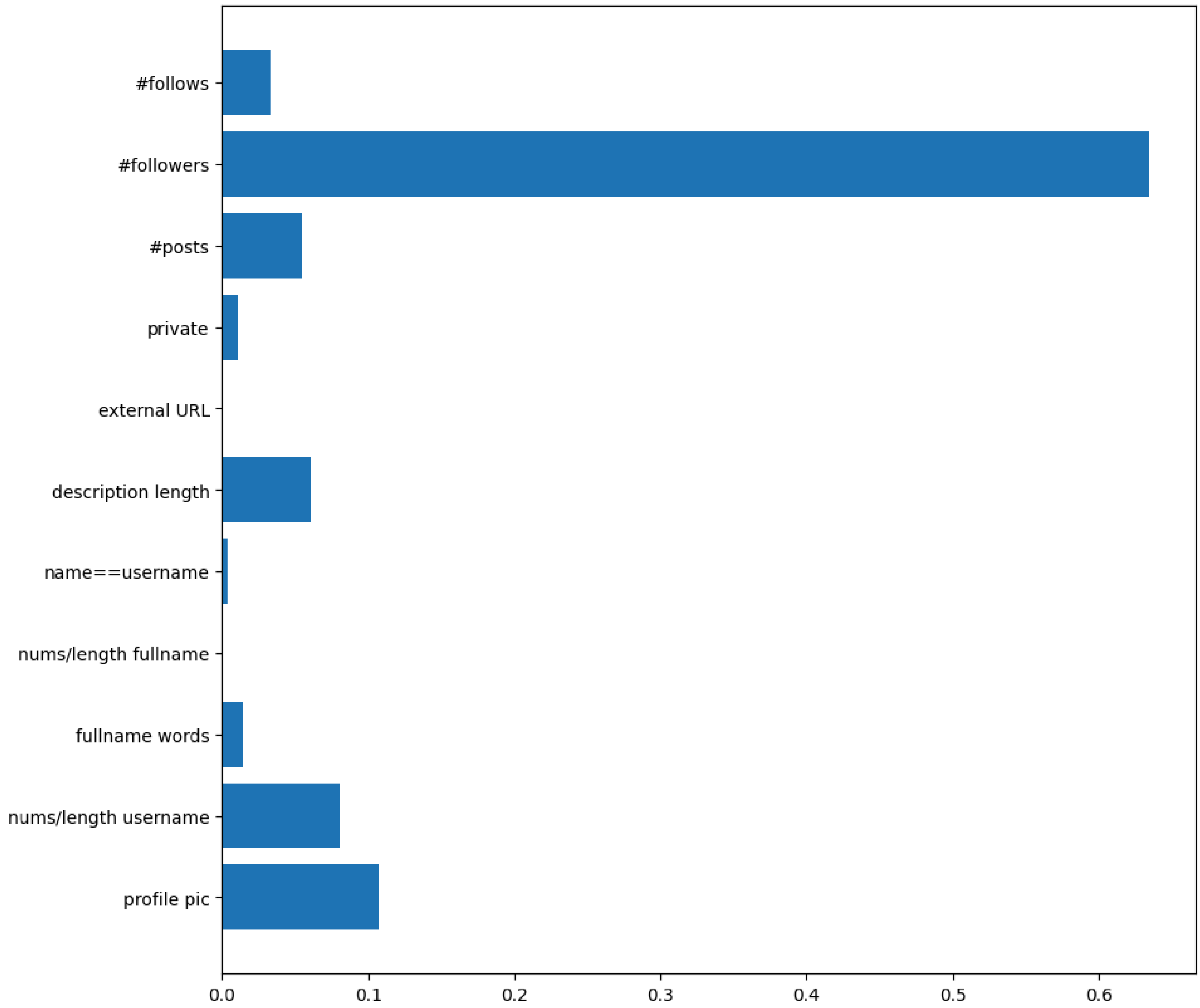
```
# explain model with tree plot
from sklearn import tree
plt.figure(figsize=(20,20))
tree.plot_tree(model, filled=True, feature_names=X.columns,
class_names=['real', 'fake'], rounded=True)

plt.show()
```



In [29]:

```
model.feature_importances_


# plot feature importance
plt.figure(figsize=(10,10))
```

```python
plt.barh(X.columns, model.feature_importances_)
plt.show()
```



In [ ]:

In [30]:

```
df_test = pd.read_csv('/kaggle/input/instagram-fake-spammer-genuine-accounts/test.csv')

df_test.head()
```

Out[30]:

	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

In [31]:

```
X_test = df_test.drop('fake', axis=1)
```

In [32]:

```
y_pred = model.predict(X_test)
```

In [33]:

```
y_pred
```

Out[33]:

```
array([0, 0, 0, 0, 0, 0, 0, 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, 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, 1,
1, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1,

       0, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [34]:

```python
accuracy_score(df_test['fake'], y_pred)
```

Out[34]:

```
0.9416666666666667
```

In [35]:

```python
print(confusion_matrix(df_test['fake'], y_pred))
```

```
[[57  3]
 [ 4 56]]
```

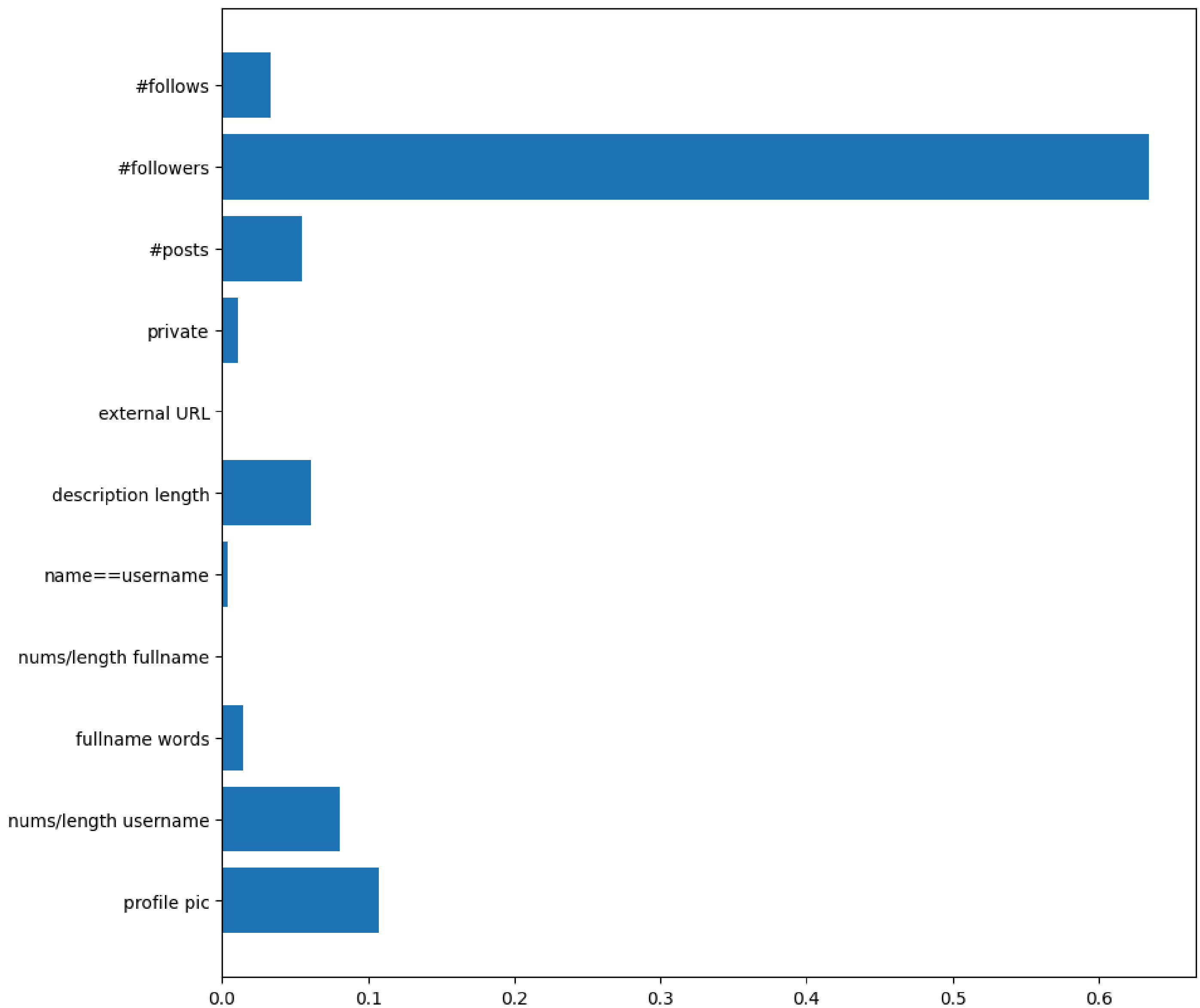
In [36]:

```python
print(classification_report(df_test['fake'], y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94        60
           1       0.95      0.93      0.94        60

    accuracy                           0.94       120
   macro avg       0.94      0.94      0.94       120
weighted avg       0.94      0.94      0.94       120
```

In [37]:

```
plt.figure(figsize=(10,10))
plt.barh(X_test.columns, model.feature_importances_)
plt.show()
```



In [ ]:

Reference link

Example: You can get the basic idea how you can create a project from here

To build an SQL project that identifies Instagram accounts as fake, spam, or genuine based on various indicators, we will create a database schema, write SQL queries for EDA (exploratory data analysis), and build visualizations. This project can be structured to help a business analyst or data scientist with several years of experience to manage, query, and interpret insights from a large dataset.

Project: Instagram Account Classification (SQL-Based)

Objective

The main goal is to analyze Instagram account data and classify accounts as fake or genuine based on characteristics like username patterns, profile picture presence, follower/following counts, and other user activity metrics.

Step 1: Database Schema Design

We'll create a table called `instagram_accounts` with the following columns:

sql
code
```sql
CREATE TABLE instagram_accounts (
    account_id INT PRIMARY KEY,
    profile_pic BOOLEAN,               -- 1 if a profile pic exists, 0 otherwise
    username_length INT,               -- Number of characters in the username
    username_nums INT,                 -- Number of digits in the username
    fullname_words INT,                -- Number of words in
```

```
the full name
    fullname_length INT,                -- Total character
length of the full name
    name_equals_username BOOLEAN,       -- 1 if name matches
username, 0 otherwise
    description_length INT,             -- Character count of
bio description
    external_url BOOLEAN,               -- 1 if an external URL
exists, 0 otherwise
    private BOOLEAN,                    -- 1 if the account is
private, 0 if public
    post_count INT,                     -- Number of posts by
the account
    follower_count INT,                 -- Number of followers
    following_count INT,                -- Number of accounts
followed by the user
    fake BOOLEAN                        -- 1 if account is fake,
0 otherwise
);
```

Step 2: Data Ingestion

Assuming you have a dataset in CSV format, you can load it using SQL tools like PostgreSQL's `COPY` command or with SQL scripts that import data from structured files.

sql

code

```sql
COPY instagram_accounts FROM '/path/to/instagram_data.csv'
DELIMITER ',' CSV HEADER;
```

Step 3: EDA with SQL Queries

1. Basic Data Overview

sql

code

```sql
-- Check basic statistics
SELECT
    COUNT(*) AS total_accounts,
    SUM(CASE WHEN fake = 1 THEN 1 ELSE 0 END) AS
total_fake_accounts,
    SUM(CASE WHEN fake = 0 THEN 1 ELSE 0 END) AS
total_genuine_accounts
FROM instagram_accounts;
```

2. Analyzing Profile Picture Distribution

sql

code

```sql
-- Percentage of fake vs. genuine accounts with a profile
picture
SELECT
    profile_pic,
```

```
    fake,
    COUNT(*) AS count,
    ROUND((COUNT(*) * 100.0 / (SELECT COUNT(*) FROM
instagram_accounts)), 2) AS percentage
FROM instagram_accounts
GROUP BY profile_pic, fake;
```

3. Followers and Following Analysis

Accounts with very high follower-to-following ratios are often indicative of spam accounts. We can analyze this by calculating ratios and plotting distributions.

sql

code

```
-- Calculate follower-following ratio
SELECT
    account_id,
    follower_count,
    following_count,
    ROUND(CAST(follower_count AS FLOAT) /
NULLIF(following_count, 0), 2) AS follower_following_ratio,
    fake
FROM instagram_accounts;
```

4. Username and Full Name Patterns

Analyzing usernames and full names, especially focusing on accounts where names match usernames or where usernames have unusual lengths, can help identify suspicious accounts.

sql

code

```sql
-- Average username length and presence of numbers in usernames for fake vs. genuine accounts
SELECT
    fake,
    AVG(username_length) AS avg_username_length,
    AVG(username_nums) AS avg_username_nums
FROM instagram_accounts
GROUP BY fake;
```

5. Private vs. Public Accounts

This query identifies how many private accounts are labeled as fake or genuine.

sql

code

```sql
SELECT
    private,
    fake,
    COUNT(*) AS count
FROM instagram_accounts
GROUP BY private, fake;
```

Step 4: Visualization with SQL and BI Tools

You can export the summarized data from SQL and visualize it using BI tools such as Tableau or Power BI. Here's what each visualization could represent:

- **Profile Picture Analysis**: Use a bar chart to show the proportion of accounts with and without profile pictures across fake and genuine accounts.
- **Follower/Following Ratio**: Visualize the follower-to-following ratio as a box plot, with separate plots for fake and genuine accounts to observe any significant differences.
- **Username and Full Name Patterns**: Display the average username length and the average number of numbers in usernames for fake and genuine accounts using a grouped bar chart.
- **Private vs Public Analysis**: Create a stacked bar chart to show the count of private and public accounts classified as fake or genuine.

Step 5: Classification Insights

Based on the SQL analysis, we can infer key indicators of fake accounts, such as:

- Fake accounts may frequently lack profile pictures or have very short usernames with unusual character patterns.
- They often have low follower-to-following ratios.
- Fake accounts might have a high likelihood of private profiles.

Step 6: Next Steps for Model Building

If further machine learning modeling is required:

1. **Data Export**: Export the processed SQL data to a CSV file for machine learning modeling.
2. **Feature Engineering**: Use the derived statistics (e.g., follower-following ratios)

as additional features in a machine learning model.

3. **Modeling**: Build a classification model using Python and libraries like `scikit-learn` to predict fake accounts.

Sample Code for Exporting Data

sql

code

```
COPY (
    SELECT
        account_id, profile_pic, username_length,
username_nums, fullname_words,
        fullname_length, name_equals_username,
description_length, external_url,
        private, post_count, follower_count, following_count,
fake
    FROM instagram_accounts
) TO '/path/to/output.csv' DELIMITER ',' CSV HEADER;
```