```
!pip install openai==0.28
```

```
Requirement already satisfied: openai==0.28 in /usr/local/lib/python3.11/dist-packages (0.28.0)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.11/dist-packages (from openai==0.28) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from openai==0.28) (4.67.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from openai==0.28) (3.11.15)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.20->openai==0.28) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.20->openai==0.28) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.20->openai==0.28) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.20->openai==0.28) (2025.1
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (6.4.2)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->openai==0.28) (1.19.0)
```

```
!pip install pandas numpy matplotlib scikit-learn transformers
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.51.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.30.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.30.0->transform
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.30.0-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.31)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import transformers
```

```
users_df = pd.read_csv('users.csv')
fusers_df = pd.read_csv('fusers.csv')
```

```
print(users_df.head())
print(fusers_df.head())
```

```
           id               name      screen_name  statuses_count  \
0  1502026416    TASUKU HAYAKAWA         0918Bask            2177
1  2492782375              ro_or         1120Roll            2660
2   293212315           bearclaw         14KBBrown            1254
3   191839658  pocahontas farida      wadespeters          202968
4  3020965143           Ms Kathy  191a5bd05da04dc              82
```

```
   followers_count  friends_count  favourites_count  listed_count  \
0              208            332               265             1
1              330            485              3972             5
2              166            177              1185             0
3             2248            981             60304           101
4               21             79                 5             0

                      url lang  ... notifications  \
0                     NaN   ja  ...           NaN
1                     NaN   ja  ...           NaN
2                     NaN   en  ...           NaN
3  http://t.co/rGV0HIJGsu   en  ...           NaN
4                     NaN   en  ...           NaN

                                      description  contributors_enabled  \
0                             15years ago X.Lines24                   NaN
1  保守見習い地元大好き人間。 経済学、電工、仏教を勉強中、ちなDeではいかんのか？ (*^◯^*)                   NaN
2                    Let me see what your best move is!                   NaN
3  20. menna: #farida #nyc and the 80s actually y...                   NaN
4                                     Cosmetologist                   NaN

   following                      created_at            timestamp  \
0        NaN  Tue Jun 11 11:20:35 +0000 2013  2013-06-11 13:20:35
1        NaN  Tue May 13 10:37:57 +0000 2014  2014-05-13 12:37:57
2        NaN  Wed May 04 23:30:37 +0000 2011  2011-05-05 01:30:37
3        NaN  Fri Sep 17 14:02:10 +0000 2010  2010-09-17 16:02:10
4        NaN  Fri Feb 06 04:10:49 +0000 2015  2015-02-06 05:10:49

            crawled_at              updated test_set_1 test_set_2
0  2015-05-02 06:41:46  2016-03-15 15:53:47          0          0
1  2015-05-01 17:20:27  2016-03-15 15:53:48          0          0
2  2015-05-01 18:48:28  2016-03-15 15:53:48          0          0
3  2015-05-01 13:55:16  2016-03-15 15:53:48          0          0
4  2015-05-02 01:17:32  2016-03-15 15:53:48          0          0

[5 rows x 42 columns]
          id                 name   screen_name  statuses_count  \
0   80479674              YI YUAN     yi_twitts              29
1   82487179       Marcos Perez C   marcos_peca            1408
2  105830531        curti lorenzo  curtilorenzo              39
3  114488344  ruben dario toscano     gatito2710              59
4  123222267          Malek Khalaf   MalekKhalaf             987

   followers_count  friends_count  favourites_count  listed_count  \
0               19            255                 1             0
1              208            866               138             0
2               59            962                 8             0
3                7             49                 4             0
4               60            521                61             1

                      created_at                                    url  \
```

```
print(users_df.columns)
users_df.shape
```

```
Index(['id', 'name', 'screen_name', 'statuses_count', 'followers_count',
       'friends_count', 'favourites_count', 'listed_count', 'url', 'lang',
       'time_zone', 'location', 'default_profile', 'default_profile_image',
       'geo_enabled', 'profile_image_url', 'profile_banner_url',
       'profile_use_background_image', 'profile_background_image_url_https',
       'profile_text_color', 'profile_image_url_https',
       'profile_sidebar_border_color', 'profile_background_tile',
       'profile_sidebar_fill_color', 'profile_background_image_url',
       'profile_background_color', 'profile_link_color', 'utc_offset',
       'is_translator', 'follow_request_sent', 'protected', 'verified',
       'notifications', 'description', 'contributors_enabled', 'following',
       'created_at', 'timestamp', 'crawled_at', 'updated', 'test_set_1',
       'test_set_2'],
      dtype='object')
(3474, 42)
```

```
print(users_df.isnull().sum())
```

```
id                                       0
name                                     1
screen_name                              0
statuses_count                           0
followers_count                          0
friends_count                            0
favourites_count                         0
listed_count                             0
url                                   2208
lang                                     0
```

```
time_zone                              999
location                              1109
default_profile                       2442
default_profile_image                 3461
geo_enabled                           1319
profile_image_url                        0
profile_banner_url                     309
profile_use_background_image           390
profile_background_image_url_https       0
profile_text_color                       0
profile_image_url_https                  0
profile_sidebar_border_color             0
profile_background_tile               2167
profile_sidebar_fill_color               0
profile_background_image_url             0
profile_background_color                 0
profile_link_color                       0
utc_offset                             999
is_translator                         3473
follow_request_sent                   3474
protected                             3396
verified                              3463
notifications                         3474
description                            379
contributors_enabled                  3474
following                             3474
created_at                               0
timestamp                                0
crawled_at                               0
updated                                  0
test_set_1                               0
test_set_2                               0
dtype: int64
```

```
users_df.shape
```

```
(3474, 42)
```

```
import pandas as pd

def clean_social_media_features(df):
    # Columns to drop
    drop_cols = [
        'profile_text_color', 'profile_sidebar_border_color',
        'profile_background_color', 'profile_link_color',
        'profile_sidebar_fill_color', 'profile_background_tile',
        'profile_use_background_image', 'profile_background_image_url',
        'profile_image_url', 'test_set_1', 'test_set_2',
        'contributors_enabled', 'follow_request_sent', 'notifications', 'following'
    ]

    df = df.drop(columns=drop_cols, errors='ignore')

    # Convert datetime columns and remove timezone
    df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce').dt.tz_localize(None)
    df['crawled_at'] = pd.to_datetime(df['crawled_at'], errors='coerce').dt.tz_localize(None)
    df['updated'] = pd.to_datetime(df['updated'], errors='coerce').dt.tz_localize(None)

    # Calculate account age and days since update
    df['account_age_days'] = (df['crawled_at'] - df['created_at']).dt.days
    df['days_since_update'] = (df['crawled_at'] - df['updated']).dt.days

    # Drop original date columns after transformation
    df = df.drop(columns=['timestamp', 'crawled_at', 'updated'], errors='ignore')

    # Create binary features for URL presence
    df['has_profile_url'] = df['url'].notna().astype(int)
    df['has_banner'] = df['profile_banner_url'].notna().astype(int)

    # Drop original URL columns after transformation
    df = df.drop(columns=['url', 'profile_banner_url'], errors='ignore')

    return df


clean_social_media_features(users_df)
```

```
<ipython-input-8-bb545f15773d>:17: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `da
  df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce').dt.tz_localize(None)
```

	id	name	screen_name	statuses_count	followers_count	friends_count	favourites_count	listed_count	lang	
0	1502026416	TASUKU HAYAKAWA	0918Bask	2177	208	332	265	1	ja	
1	2492782375	ro_or	1120Roll	2660	330	485	3972	5	ja	
2	293212315	bearclaw	14KBBrown	1254	166	177	1185	0	en	I (U
3	191839658	pocahontas farida	wadespeters	202968	2248	981	60304	101	en	
4	3020965143	Ms Kathy	191a5bd05da04dc	82	21	79	5	0	en	
...	...	...	...	...	...	...	...	...	...	
3469	205218909	Alejandro	zombiemaster999	315	94	597	36	4	es	
3470	2874966164	Zubair Niazi	zubairniaziPTI	4099	5378	1238	471	6	en	
3471	2980901837	Zuhazuu	zuhazuu1	199	18	136	6	0	en	
3472	121122678	zveljka	zveljka	2609	41	263	121	0	en	Eurc
3473	2910276853	// karime //	cypherjimin	20997	498	109	12105	5	en	(U

3474 rows × 26 columns

```
users_df = clean_social_media_features(users_df)
```

```
<ipython-input-8-bb545f15773d>:17: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `da
  df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce').dt.tz_localize(None)
```

```
print(users_df.isnull().sum())
```

```
id                                    0
name                                  1
screen_name                           0
statuses_count                        0
followers_count                       0
friends_count                         0
favourites_count                      0
listed_count                          0
lang                                  0
time_zone                           999
location                           1109
default_profile                    2442
default_profile_image              3461
geo_enabled                        1319
profile_background_image_url_https    0
profile_image_url_https               0
utc_offset                          999
is_translator                      3473
protected                          3396
```

```
verified                              3463
description                            379
created_at                               0
account_age_days                         0
days_since_update                        0
has_profile_url                          0
has_banner                               0
dtype: int64
```

```
def handle_null_values(df):
    # 1. Boolean Features - Fill with False (0)
    boolean_cols = [
        'default_profile', 'default_profile_image', 'geo_enabled',
        'is_translator', 'protected', 'verified'
    ]
    df[boolean_cols] = df[boolean_cols].fillna(0)

    # 2. Location and Time Features
    df['time_zone'] = df['time_zone'].fillna('unknown')
    df['location'] = df['location'].fillna('unknown')
    df['utc_offset'] = df['utc_offset'].fillna(0)  # Fill with 0 for unknown timezone offset

    # 3. Text Features
    df['description'] = df['description'].fillna('')  # Empty string for missing descriptions

    # 4. Name - Fill with screen_name if name is null
    df['name'] = df['name'].fillna(df['screen_name'])

    # 5. Create profile completeness features
    df['profile_fields_filled'] = df[[
        'description', 'location', 'time_zone',
        'default_profile', 'geo_enabled'
    ]].notna().mean(axis=1)

    # 6. Create engagement ratio features (these can help identify bots)
    df['followers_friends_ratio'] = df['followers_count'] / (df['friends_count'] + 1)  # Add 1 to avoid division by zero
    df['statuses_per_day'] = df['statuses_count'] / (df['account_age_days'] + 1)

    # 7. Verify all nulls are handled
    remaining_nulls = df.isnull().sum()
    if remaining_nulls.sum() > 0:
        print("Remaining null values:")
        print(remaining_nulls[remaining_nulls > 0])

    return df

# Apply the null value handling
users_df = handle_null_values(users_df)

# Verify results and show summary statistics
print("\nDataset shape:", users_df.shape)
print("\nNull values after cleaning:")
print(users_df.isnull().sum()[users_df.isnull().sum() > 0])  # Only show columns with nulls

# Show some key statistics
print("\nProfile completeness statistics:")
print(users_df['profile_fields_filled'].describe())

print("\nEngagement metrics:")
print("Followers/Friends ratio statistics:")
print(users_df['followers_friends_ratio'].describe())
print("\nStatuses per day statistics:")
print(users_df['statuses_per_day'].describe())
```

```
Dataset shape: (3474, 29)

Null values after cleaning:
Series([], dtype: int64)

Profile completeness statistics:
count    3474.0
mean        1.0
std         0.0
min         1.0
25%         1.0
50%         1.0
75%         1.0
```

```
max         1.0
Name: profile_fields_filled, dtype: float64

Engagement metrics:
Followers/Friends ratio statistics:
count    3474.000000
mean        3.351967
std        27.293197
min         0.030702
25%         0.584243
50%         1.019096
75%         1.560396
max      1014.166667
Name: followers_friends_ratio, dtype: float64

Statuses per day statistics:
count    3474.000000
mean       17.096568
std        41.377983
min         0.002600
25%         2.008608
50%         6.478300
75%        17.549258
max      1208.072202
Name: statuses_per_day, dtype: float64
```

```
print(users_df.isnull().sum())
```

```
id                                  0
name                                0
screen_name                         0
statuses_count                      0
followers_count                     0
friends_count                       0
favourites_count                    0
listed_count                        0
lang                                0
time_zone                           0
location                            0
default_profile                     0
default_profile_image               0
geo_enabled                         0
profile_background_image_url_https  0
profile_image_url_https             0
utc_offset                          0
is_translator                       0
protected                           0
verified                            0
description                         0
created_at                          0
account_age_days                    0
days_since_update                   0
has_profile_url                     0
has_banner                          0
profile_fields_filled               0
followers_friends_ratio             0
statuses_per_day                    0
dtype: int64
```

```
fusers_df.columns
```

```
Index(['id', 'name', 'screen_name', 'statuses_count', 'followers_count',
       'friends_count', 'favourites_count', 'listed_count', 'created_at',
       'url', 'lang', 'time_zone', 'location', 'default_profile',
       'default_profile_image', 'geo_enabled', 'profile_image_url',
       'profile_banner_url', 'profile_use_background_image',
       'profile_background_image_url_https', 'profile_text_color',
       'profile_image_url_https', 'profile_sidebar_border_color',
       'profile_background_tile', 'profile_sidebar_fill_color',
       'profile_background_image_url', 'profile_background_color',
       'profile_link_color', 'utc_offset', 'is_translator',
       'follow_request_sent', 'protected', 'verified', 'notifications',
       'description', 'contributors_enabled', 'following', 'updated'],
      dtype='object')
```

```
print("Columns in users_df but not in fusers_df:", set(users_df.columns) - set(fusers_df.columns))
print("Columns in fusers_df but not in users_df:", set(fusers_df.columns) - set(users_df.columns))
```

```
Columns in users_df but not in fusers_df: {'statuses_per_day', 'days_since_update', 'has_banner', 'account_age_days', 'followers_friends
Columns in fusers_df but not in users_df: {'profile_background_tile', 'notifications', 'profile_background_color', 'profile_use_backgrou
```

```
def clean_fusers_social_media_features(df):
    # Columns to drop
    drop_cols = [
        'profile_text_color', 'profile_sidebar_border_color',
        'profile_background_color', 'profile_link_color',
        'profile_sidebar_fill_color', 'profile_background_tile',
        'profile_use_background_image', 'profile_background_image_url',
        'profile_image_url', 'test_set_1', 'test_set_2',
        'contributors_enabled', 'follow_request_sent', 'notifications', 'following'
    ]

    df = df.drop(columns=drop_cols, errors='ignore')

    # Create binary features for URL presence
    df['has_profile_url'] = df['url'].notna().astype(int)
    df['has_banner'] = df['profile_banner_url'].notna().astype(int)

    # Drop original URL columns after transformation
    df = df.drop(columns=['url', 'profile_banner_url'], errors='ignore')

    return df


fusers_df = clean_fusers_social_media_features(fusers_df)
```

```
print("Columns in users_df but not in fusers_df:", set(users_df.columns) - set(fusers_df.columns))
print("Columns in fusers_df but not in users_df:", set(fusers_df.columns) - set(users_df.columns))
```

```
Columns in users_df but not in fusers_df: {'statuses_per_day', 'days_since_update', 'account_age_days', 'followers_friends_ratio', 'prof
Columns in fusers_df but not in users_df: {'updated'}
```

```
users_df=users_df.drop(columns=["days_since_update","account_age_days","statuses_per_day"])
fusers_df=fusers_df.drop(columns=["updated"])
```

```
print("Columns in users_df but not in fusers_df:", set(users_df.columns) - set(fusers_df.columns))
print("Columns in fusers_df but not in users_df:", set(fusers_df.columns) - set(users_df.columns))
```

```
Columns in users_df but not in fusers_df: {'followers_friends_ratio', 'profile_fields_filled'}
Columns in fusers_df but not in users_df: set()
```

```
print(fusers_df.isnull().sum())
```

```
id                                    0
name                                  0
screen_name                           0
statuses_count                        0
followers_count                       0
friends_count                         0
favourites_count                      0
listed_count                          0
created_at                            0
lang                                  0
time_zone                          3016
location                            575
default_profile                     317
default_profile_image              3345
geo_enabled                        3212
profile_background_image_url_https    0
profile_image_url_https               0
utc_offset                         3016
is_translator                      3351
protected                          3351
verified                           3351
description                        1073
has_profile_url                       0
has_banner                            0
dtype: int64
```

```
def handle_fusers_null_values(df):
    # 1. Boolean Features - Fill with False (0)
    boolean_cols = [
        'default_profile', 'default_profile_image', 'geo_enabled',
        'is_translator', 'protected', 'verified'
    ]
    df[boolean_cols] = df[boolean_cols].fillna(0)

    # 2. Location and Time Features
    df['time_zone'] = df['time_zone'].fillna('unknown')
    df['location'] = df['location'].fillna('unknown')
    df['utc_offset'] = df['utc_offset'].fillna(0)  # Fill with 0 for unknown timezone offset

    # 3. Text Features
    df['description'] = df['description'].fillna('')  # Empty string for missing descriptions

    # 4. Name - Fill with screen_name if name is null
    df['name'] = df['name'].fillna(df['screen_name'])

    # 5. Create profile completeness features
    df['profile_fields_filled'] = df[[
        'description', 'location', 'time_zone',
        'default_profile', 'geo_enabled'
    ]].notna().mean(axis=1)

    # 6. Create engagement ratio features (these can help identify bots)
    df['followers_friends_ratio'] = df['followers_count'] / (df['friends_count'] + 1)  # Add 1 to avoid division by zero

    # 7. Verify all nulls are handled
    remaining_nulls = df.isnull().sum()
    if remaining_nulls.sum() > 0:
        print("Remaining null values:")
        print(remaining_nulls[remaining_nulls > 0])

    return df

# Apply the null value handling
fusers_df = handle_fusers_null_values(fusers_df)

# Verify results and show summary statistics
print("\nDataset shape:", fusers_df.shape)
print("\nNull values after cleaning:")
print(fusers_df.isnull().sum()[fusers_df.isnull().sum() > 0])  # Only show columns with nulls

# Show some key statistics
print("\nProfile completeness statistics:")
print(fusers_df['profile_fields_filled'].describe())

print("\nEngagement metrics:")
print("Followers/Friends ratio statistics:")
print(fusers_df['followers_friends_ratio'].describe())
```

```
Dataset shape: (3351, 26)

Null values after cleaning:
Series([], dtype: int64)

Profile completeness statistics:
count    3351.0
mean        1.0
std         0.0
min         1.0
25%         1.0
50%         1.0
75%         1.0
max         1.0
Name: profile_fields_filled, dtype: float64

Engagement metrics:
Followers/Friends ratio statistics:
count    3351.000000
mean        0.046191
std         0.082908
min         0.000000
25%         0.027085
50%         0.041237
75%         0.054187
max         3.093333
```

```
Name: followers_friends_ratio, dtype: float64
```

```
fusers_df.shape
```

```
(3351, 26)
```

```
users_df.shape
```

```
(3474, 26)
```

```
print("Columns in users_df but not in fusers_df:", set(users_df.columns) - set(fusers_df.columns))
print("Columns in fusers_df but not in users_df:", set(fusers_df.columns) - set(users_df.columns))
```

```
Columns in users_df but not in fusers_df: set()
Columns in fusers_df but not in users_df: set()
```

```
def engineer_features(df):
    # 1. User Activity Ratios
    df['tweets_to_followers_ratio'] = df['statuses_count'] / (df['followers_count'] + 1)
    df['followers_to_friends_ratio'] = df['followers_count'] / (df['friends_count'] + 1)
    df['favorites_per_tweet'] = df['favourites_count'] / (df['statuses_count'] + 1)
    df['lists_per_follower'] = df['listed_count'] / (df['followers_count'] + 1)

    # 2. Name and Description Analysis
    df['screen_name_length'] = df['screen_name'].str.len()
    df['name_length'] = df['name'].str.len()
    df['description_length'] = df['description'].str.len()
    df['has_numbers_in_name'] = df['screen_name'].str.contains('\d').astype(int)
    df['has_special_chars'] = df['screen_name'].str.contains('[^a-zA-Z0-9_]').astype(int)

    # 3. Profile Completeness
    profile_fields = ['description', 'location', 'time_zone', 'geo_enabled']
    df['profile_completeness'] = df[profile_fields].notna().mean(axis=1)

    return df

# Apply feature engineering to both datasets separately
users_df = engineer_features(users_df)
fusers_df = engineer_features(fusers_df)
```

```
# Compare feature distributions between normal and fraudulent accounts
new_features = [
    'tweets_to_followers_ratio', 'followers_to_friends_ratio',
    'favorites_per_tweet', 'lists_per_follower',
    'screen_name_length', 'name_length', 'description_length',
    'has_numbers_in_name', 'has_special_chars', 'profile_completeness'
]

# Print summary statistics for both datasets
print("\nFeature statistics for normal accounts (users_df):")
print(users_df[new_features].describe())
```

```
Feature statistics for normal accounts (users_df):
       tweets_to_followers_ratio  followers_to_friends_ratio  \
count                3474.000000                 3474.000000   
mean                   31.472533                    3.351967   
std                    58.152294                   27.293197   
min                     0.019807                    0.030702   
25%                     6.911017                    0.584243   
50%                    16.505416                    1.019096   
75%                    36.521027                    1.560396   
max                  1178.478261                 1014.166667   

       favorites_per_tweet  lists_per_follower  screen_name_length  \
count          3474.000000         3474.000000         3474.000000   
mean              0.677606            0.015076           10.862982   
std               2.267411            0.024220            2.541494   
min               0.000000            0.000000            3.000000   
25%               0.066677            0.000000            9.000000   
50%               0.250318            0.006112           11.000000   
75%               0.645490            0.020075           13.000000   
```

```
max             95.287313           0.283988           15.000000

       name_length  description_length  has_numbers_in_name  \
count  3474.000000         3474.000000          3474.000000
mean     10.354347           63.778066             0.206390
std       4.785187           50.882751             0.404772
min       1.000000            0.000000             0.000000
25%       6.000000           20.000000             0.000000
50%      11.000000           51.000000             0.000000
75%      14.000000          106.750000             0.000000
max      20.000000          160.000000             1.000000

       has_special_chars  profile_completeness
count             3474.0                3474.0
mean                 0.0                   1.0
std                  0.0                   0.0
min                  0.0                   1.0
25%                  0.0                   1.0
50%                  0.0                   1.0
75%                  0.0                   1.0
max                  0.0                   1.0
```

```
print("\nFeature statistics for fraudulent accounts (fusers_df):")
print(fusers_df[new_features].describe())
```

```
Feature statistics for fraudulent accounts (fusers_df):
       tweets_to_followers_ratio  followers_to_friends_ratio  \
count                3351.000000                 3351.000000
mean                    3.960837                    0.046191
std                    21.616046                    0.082908
min                     0.000000                    0.000000
25%                     1.285714                    0.027085
50%                     1.850000                    0.041237
75%                     2.571429                    0.054187
max                   979.000000                    3.093333

       favorites_per_tweet  lists_per_follower  screen_name_length  \
count          3351.000000         3351.000000         3351.000000
mean              0.021025            0.001769           11.854372
std               0.228886            0.015371            2.646481
min               0.000000            0.000000            4.000000
25%               0.000000            0.000000           10.000000
50%               0.000000            0.000000           12.000000
75%               0.000000            0.000000           14.000000
max              11.372881            0.333333           15.000000

       name_length  description_length  has_numbers_in_name  \
count  3351.000000         3351.000000          3351.000000
mean     12.921516           55.506714             0.180245
std       2.969314           56.347519             0.384449
min       1.000000            0.000000             0.000000
25%      11.000000            0.000000             0.000000
50%      13.000000           37.000000             0.000000
75%      15.000000          105.000000             0.000000
max      20.000000          160.000000             1.000000

       has_special_chars  profile_completeness
count             3351.0                3351.0
mean                 0.0                   1.0
std                  0.0                   0.0
min                  0.0                   1.0
25%                  0.0                   1.0
50%                  0.0                   1.0
75%                  0.0                   1.0
max                  0.0                   1.0
```

```
def merge_datasets(users_df, fusers_df):
    # Add labels before merging
    users_df['is_fraudulent'] = 0
    fusers_df['is_fraudulent'] = 1

    # Merge datasets
    merged_df = pd.concat([users_df, fusers_df], axis=0, ignore_index=True)

    return merged_df

# Merge if you want to proceed with modeling
merged_df = merge_datasets(users_df, fusers_df)

print("\nShape of merged dataset:", merged_df.shape)
```

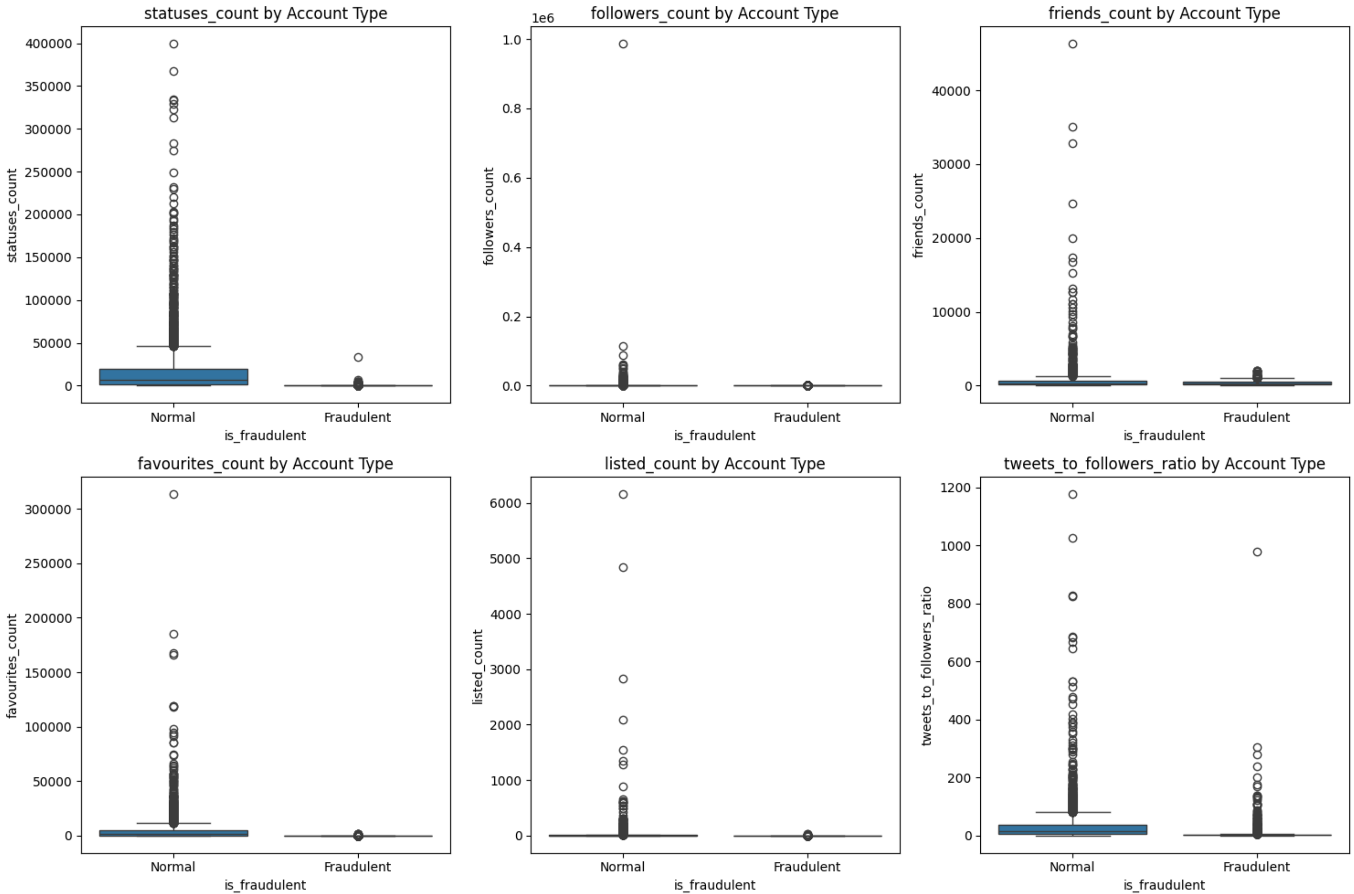
```
print("Number of fraudulent accounts:", merged_df['is_fraudulent'].sum())
print("Percentage of fraudulent accounts: {:.2f}%".format(
    merged_df['is_fraudulent'].mean() * 100))
```

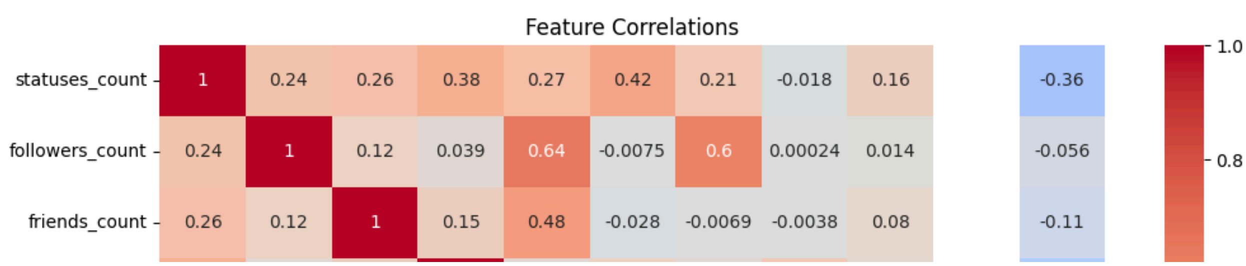
```
Shape of merged dataset: (6825, 37)
Number of fraudulent accounts: 3351
Percentage of fraudulent accounts: 49.10%
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

def perform_eda(df):
    # 1. Basic Distribution Analysis
    print("Dataset Shape:", df.shape)
    print("\nClass Distribution:")
    print(df['is_fraudulent'].value_counts(normalize=True) * 100)

    # 2. Numerical Features Analysis
    numerical_features = [
        'statuses_count', 'followers_count', 'friends_count',
        'favourites_count', 'listed_count', 'tweets_to_followers_ratio',
        'followers_to_friends_ratio', 'favorites_per_tweet',
        'lists_per_follower', 'profile_completeness'
    ]

    # Create distribution plots
    plt.figure(figsize=(15, 10))
    for i, feature in enumerate(numerical_features[:6], 1):
        plt.subplot(2, 3, i)
        sns.boxplot(x='is_fraudulent', y=feature, data=df)
        plt.title(f'{feature} by Account Type')
        plt.xticks([0, 1], ['Normal', 'Fraudulent'])
    plt.tight_layout()
    plt.show()

    # 3. Correlation Analysis
    plt.figure(figsize=(12, 8))
    correlation_matrix = df[numerical_features + ['is_fraudulent']].corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
    plt.title('Feature Correlations')
    plt.show()

    # 4. Profile Characteristics
    categorical_features = ['default_profile', 'geo_enabled', 'verified',
                          'has_profile_url', 'has_banner']

    plt.figure(figsize=(15, 5))
    for i, feature in enumerate(categorical_features, 1):
        plt.subplot(1, 5, i)
        df.groupby('is_fraudulent')[feature].mean().plot(kind='bar')
        plt.title(f'{feature} by Account Type')
        plt.xticks([0, 1], ['Normal', 'Fraudulent'])
    plt.tight_layout()
    plt.show()

    # 5. Statistical Summary
    print("\nNumerical Features Summary for Normal Accounts:")
    print(df[df['is_fraudulent']==0][numerical_features].describe())
    print("\nNumerical Features Summary for Fraudulent Accounts:")
    print(df[df['is_fraudulent']==1][numerical_features].describe())

    # 6. Feature Importance (using simple statistical measures)
    feature_importance = {}
    for feature in numerical_features:
        normal_mean = df[df['is_fraudulent']==0][feature].mean()
        fraud_mean = df[df['is_fraudulent']==1][feature].mean()
        importance = abs(normal_mean - fraud_mean) / (normal_mean + fraud_mean)
        feature_importance[feature] = importance

    print("\nFeature Discrimination Power:")
    for feature, importance in sorted(feature_importance.items(),
                                    key=lambda x: x[1], reverse=True):
        print(f"{feature}: {importance:.4f}")
```

```
# Perform EDA
perform_eda(merged_df)
```

```
Dataset Shape: (6825, 37)

Class Distribution:
is_fraudulent
0    50.901099
1    49.098901
Name: proportion, dtype: float64
```






```
#FEATURE SELECTION
def select_important_features(merged_df):
    # A. Calculate feature correlations
    numerical_cols = merged_df.select_dtypes(include=['float64', 'int64']).columns
    correlation_matrix = merged_df[numerical_cols].corr()

    # B. Important features we want to keep
    important_features = [
        # User Identity Features
        'name', 'screen_name', 'description',

        # Activity Metrics
        'followers_count', 'friends_count', 'statuses_count',
        'followers_to_friends_ratio', 'tweets_to_followers_ratio',

        # Profile Features
        'verified', 'profile_completeness',
        'default_profile', 'geo_enabled',

        # Target Variable
        'is_fraudulent'
```

```
    ]

    # C. Select these features from merged_df
    selected_df = merged_df[important_features]

    return selected_df
```

```
# 2. DATA SPLITTING
from sklearn.model_selection import train_test_split

# First select features
selected_df = select_important_features(merged_df)

# Then split the data
train_df, test_df = train_test_split(
    selected_df,
    test_size=0.2,
    stratify=selected_df['is_fraudulent'],
    random_state=42
)

print("Dataset shapes:")
print(f"Selected features dataset: {selected_df.shape}")
print(f"Training set: {train_df.shape}")
print(f"Test set: {test_df.shape}")
```

```
Dataset shapes:
Selected features dataset: (6825, 13)
Training set: (5460, 13)
Test set: (1365, 13)
```

```
Numerical Features Summary for Normal Accounts:
       statuses_count  followers_count  friends_count  favourites_count  \
count     3474.000000      3474.000000    3474.000000       3474.000000
mean     16958.220207      1393.219632     633.242372       4669.620322
std      30696.286104     17216.664524    1600.962972      11527.566663
```

```
# 3. Verify our data
print("\nFeatures selected:")
print(selected_df.columns.tolist())

print("\nClass distribution in selected data:")
print(selected_df['is_fraudulent'].value_counts(normalize=True))
```

```
Features selected:
['name', 'screen_name', 'description', 'followers_count', 'friends_count', 'statuses_count', 'followers_to_friends_ratio', 'tweets_to_fc

Class distribution in selected data:
is_fraudulent
0    0.509011
1    0.490989
Name: proportion, dtype: float64
```

```
mean      19.496546                31.472533                 3.351967
std      157.740969                58.152294                27.293197
min        0.000000                 0.019807                 0.030702
25%        0.000000                 6.911017                 0.584243
50%        2.000000                16.505416                 1.019096
75%        8.000000                36.521027                 1.560396
max     6166.000000              1178.478261              1014.166667

       favorites_per_tweet  lists_per_follower  profile_completeness
count          3474.000000         3474.000000                3474.0
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Assuming 'selected_df' is your DataFrame
# Define features (X) and target (y)
X = selected_df.drop(columns=['name', 'screen_name', 'description', 'is_fraudulent'])  # Drop target and unnecessary columns
y = selected_df['is_fraudulent']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.97
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
       favorites_per_tweet  lists_per_follower  profile_completeness
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
mean   0.021025    0.001769    1.0
std    0.228886    0.015371    0.0
min    0.000000    0.000000    1.0
25%    0.000000    0.000000    1.0
50%    0.000000    0.000000    1.0
75%    0.000000    0.000000    1.0
max   11.372881    0.333333    1.0
```

```
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an XGBoost classifier
xgboost_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Fit the model
xgboost_model.fit(X_train, y_train)

# Make predictions
y_pred_xgboost = xgboost_model.predict(X_test)

# Evaluate the model
accuracy_xgboost = accuracy_score(y_test, y_pred_xgboost)
print(f'XGBoost Accuracy: {accuracy_xgboost:.2f}')
```

```
XGBoost Accuracy: 0.99
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [22:55:43] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Reshape input data to be 3D [samples, time steps, features]
X_train_reshaped = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))
X_test_reshaped = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))

# Create LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(50, activation='relu', input_shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])))
lstm_model.add(Dropout(0.2))
lstm_model.add(Dense(1, activation='sigmoid'))  # For binary classification

# Compile the model
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Fit the model
lstm_model.fit(X_train_reshaped, y_train, epochs=50, batch_size=32)

# Make predictions and evaluate LSTM
y_pred_prob_lstm = lstm_model.predict(X_test_reshaped)
y_pred_lstm = (y_pred_prob_lstm > 0.5).astype(int)
accuracy_lstm = accuracy_score(y_test, y_pred_lstm)
print(f'LSTM Accuracy: {accuracy_lstm:.2f}')
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argume
  super().__init__(**kwargs)
Epoch 1/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 12s 9ms/step - accuracy: 0.8703 - loss: 0.5947
Epoch 2/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.8776 - loss: 0.3418
```

```
Epoch 3/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.8825 - loss: 0.2989
Epoch 4/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.8887 - loss: 0.2885
Epoch 5/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9006 - loss: 0.2580
Epoch 6/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9207 - loss: 0.2271
Epoch 7/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9201 - loss: 0.2052
Epoch 8/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9215 - loss: 0.2008
Epoch 9/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9286 - loss: 0.1824
Epoch 10/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9312 - loss: 0.1739
Epoch 11/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9443 - loss: 0.1574
Epoch 12/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9442 - loss: 0.1479
Epoch 13/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9329 - loss: 0.1666
Epoch 14/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9461 - loss: 0.1379
Epoch 15/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9507 - loss: 0.1339
Epoch 16/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9484 - loss: 0.1338
Epoch 17/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9538 - loss: 0.1239
Epoch 18/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9512 - loss: 0.1293
Epoch 19/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9559 - loss: 0.1175
Epoch 20/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9551 - loss: 0.1224
Epoch 21/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9583 - loss: 0.1132
Epoch 22/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9587 - loss: 0.1150
Epoch 23/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9620 - loss: 0.1061
Epoch 24/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9579 - loss: 0.1165
Epoch 25/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9584 - loss: 0.1082
Epoch 26/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9651 - loss: 0.0985
Epoch 27/50
171/171 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9687 - loss: 0.0957
```

```
merged_df.head()
```

	id	name	screen_name	statuses_count	followers_count	friends_count	favourites_count	listed_count	lang	time_zo
0	1502026416	TASUKU HAYAKAWA	0918Bask	2177	208	332	265	1	ja	unknc
1	2492782375	ro_or	1120Roll	2660	330	485	3972	5	ja	Tol
2	293212315	bearclaw	14KBBrown	1254	166	177	1185	0	en	East Time (& Cana
3	191839658	pocahontas farida	wadespeters	202968	2248	981	60304	101	en	Greenla
4	3020965143	Ms Kathy	191a5bd05da04dc	82	21	79	5	0	en	unknc

5 rows × 37 columns

```
merged_df.is_fraudulent.value_counts()
```

	count
is_fraudulent	
0	3474
1	3351

```
def generate_user_description(user):
    description = []

    # Basic user information
    name_info = f"{user['name']} (@{user['screen_name']})"
    description.append(name_info)

    # Activity level
    if user['statuses_count'] > 10000:
        activity = "very active"
    elif user['statuses_count'] > 1000:
        activity = "moderately active"
    else:
        activity = "less active"

    activity_info = f"is a {activity} user with {user['statuses_count']} tweets"
    description.append(activity_info)

    # Follower/Following dynamics
    social_info = f"has {user['followers_count']} followers and follows {user['friends_count']} accounts"
    description.append(social_info)

    # Engagement metrics
    if user['favourites_count'] > 0:
        engagement = f"has liked {user['favourites_count']} posts"
        description.append(engagement)

    # Location if available
    if user['location'] and user['location'] != 'unknown':
        location = f"is located in {user['location']}"
        description.append(location)

    # User's self description if available
    if user['description'] and len(str(user['description']).strip()) > 0:
        bio = f"describes themselves as: {user['description']}"
        description.append(bio)

    # Account verification status
    if user['verified']:
        description.append("is a verified account")

    # Join date
    created_at = pd.to_datetime(user['created_at'])
    join_info = f"joined Twitter on {created_at.strftime('%B %d, %Y')}"
    description.append(join_info)

    # Combine all parts into a coherent statement
    return " | ".join(description)

print("Normal User Profiles:")
print("=" * 100)
for _, user in users_df.head().iterrows():
    print(generate_user_description(user))
    print("-" * 100)

print("\nFraudulent User Profiles:")
print("=" * 100)
for _, user in fusers_df.head().iterrows():
    print(generate_user_description(user))
    print("-" * 100)
```

```
Normal User Profiles:
====================================================================================================
TASUKU HAYAKAWA (@0918Bask) | is a moderately active user with 2177 tweets | has 208 followers and follows 332 accounts | has liked 265
----------------------------------------------------------------------------------------------------
ro_or (@1120Roll) | is a moderately active user with 2660 tweets | has 330 followers and follows 485 accounts | has liked 3972 posts | i
----------------------------------------------------------------------------------------------------
```

```
bearclaw (@14KBBrown) | is a moderately active user with 1254 tweets | has 166 followers and follows 177 accounts | has liked 1185 posts
----------------------------------------------------------------------------------------------------
pocahontas farida (@wadespeters) | is a very active user with 202968 tweets | has 2248 followers and follows 981 accounts | has liked 60
----------------------------------------------------------------------------------------------------
Ms Kathy (@191a5bd05da04dc) | is a less active user with 82 tweets | has 21 followers and follows 79 accounts | has liked 5 posts | is l
----------------------------------------------------------------------------------------------------

Fraudulent User Profiles:
====================================================================================================
YI YUAN (@yi_twitts) | is a less active user with 29 tweets | has 19 followers and follows 255 accounts | has liked 1 posts | is located
----------------------------------------------------------------------------------------------------
Marcos Perez C (@marcos_peca) | is a moderately active user with 1408 tweets | has 208 followers and follows 866 accounts | has liked 13
----------------------------------------------------------------------------------------------------
curti lorenzo (@curtilorenzo) | is a less active user with 39 tweets | has 59 followers and follows 962 accounts | has liked 8 posts | i
----------------------------------------------------------------------------------------------------
ruben dario toscano  (@gatito2710) | is a less active user with 59 tweets | has 7 followers and follows 49 accounts | has liked 4 posts
----------------------------------------------------------------------------------------------------
Malek Khalaf (@MalekKhalaf) | is a less active user with 987 tweets | has 60 followers and follows 521 accounts | has liked 61 posts | i
----------------------------------------------------------------------------------------------------
```

```python
import os

# ✅ Set your API key directly
os.environ['OPENAI_API_KEY'] = "sk-proj-fefdc8A2woi3tXYuWvnJ0tTH49jSxyckTIYuKhURifHW0Hm5tjydcmPkDd1Y76-86Xolot120AT3BlbkFJPPi_ITH8Ucz0YJgC8c

# ✅ Then fetch it like you're already doing
api_key = os.getenv('OPENAI_API_KEY')

if not api_key:
    raise ValueError("Please set the OPENAI_API_KEY environment variable")

print("API key successfully set ✅")
```

```
API key successfully set ✅
```

```python
import pandas as pd
import openai
import json
from tqdm import tqdm
import time
from sklearn.metrics import accuracy_score, classification_report
import os

def format_user_profile(user):
    """Format a user profile into a clear text description for the API."""
    profile = f"""User Profile Analysis:
Name: {user['name']} (@{user['screen_name']})
Activity: {user['statuses_count']} tweets
Network: {user['followers_count']} followers, following {user['friends_count']} users
Engagement: {user['favourites_count']} likes, listed in {user['listed_count']} lists
Profile Info: {user['description'] if pd.notna(user['description']) else 'No description'}
Location: {user['location'] if pd.notna(user['location']) else 'Not specified'}
Account Creation: {user['created_at']}
Language: {user['lang']}"""
    return profile

def classify_profile(profile, api_key):
    openai.api_key = api_key

    try:
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {
                    "role": "system",
                    "content": (
                        "You are an AI model trained to detect fraudulent Twitter accounts. "
                        "You will be given a user profile description. "
                        "Respond ONLY with the digit 0 (for genuine) or 1 (for fraudulent). "
                        "No explanation. No extra words."
                    )
                },
                {
                    "role": "user",
                    "content": f"Classify this user profile:\n\n{profile}\n\nIs this account fraudulent?\nRespond only with 0 or 1."
                }
```

```
            ],
            temperature=0,
            max_tokens=5
        )
        result = response.choices[0].message.content.strip()
        # Normalize and validate result
        if result in ["0", "1"]:
            return "genuine" if result == "0" else "fraudulent"
        else:
            print(f"Unexpected API response: {result}")
            return None
    except Exception as e:
        print(f"Error in API call: {e}")
        return None


def evaluate_classifier(df, api_key):
    sample_df = merged_df.copy()  # Use full dataset

    true_labels = []
    predicted_labels = []

    for _, user in tqdm(sample_df.iterrows(), total=len(sample_df)):
        profile = generate_user_description(user)
        prediction = classify_profile(profile, api_key)

        if prediction:
            true_labels.append("genuine" if user['is_fraudulent'] == 0 else "fraudulent")
            predicted_labels.append(prediction)

        time.sleep(1)

    accuracy = accuracy_score(true_labels, predicted_labels)
    report = classification_report(true_labels, predicted_labels)

    return {
        'accuracy': accuracy,
        'classification_report': report,
        'true_labels': true_labels,
        'predicted_labels': predicted_labels
    }


if __name__ == "__main__":
    # Load the data
    merged_df = pd.read_csv('merged_profiles.csv')

    # Get API key from environment variable
    api_key = os.getenv('OPENAI_API_KEY')
    if not api_key:
        raise ValueError("Please set the OPENAI_API_KEY environment variable")

    # Evaluate on a sample of profiles
    print("Starting evaluation...")
    results = evaluate_classifier(merged_df, api_key)

    print("\nResults:")
    print(f"Accuracy: {results['accuracy']:.2f}")
    print("\nDetailed Classification Report:")
    print(results['classification_report'])

    # Save results
    with open('openai_classification_results.json', 'w') as f:
        json.dump({
            'accuracy': results['accuracy'],
            'classification_report': results['classification_report'],
            'predictions': {
                'true_labels': results['true_labels'],
                'predicted_labels': results['predicted_labels']
            }
        }, f, indent=2)
```

```
Starting evaluation...
100%|██████████| 6825/6825 [2:35:22<00:00,  1.37s/it]
Results:
Accuracy: 0.67
```

```
Detailed Classification Report:
              precision    recall  f1-score   support

  fraudulent       0.78      0.46      0.58      3351
     genuine       0.63      0.88      0.73      3474

    accuracy                           0.67      6825
   macro avg       0.70      0.67      0.65      6825
weighted avg       0.70      0.67      0.65      6825
```

Start coding or generate with AI.

Start coding or generate with AI.