In [194…

```python
                                            # Step 1: Data Loading and Preprocessing

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset (assume CSV format)
df = pd.read_csv('top_insta_influencers_data.csv')
```

In [196…

```python
df
```

Out[196…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%	5.9m	57.4b	United States
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%	4.4m	6.0b	NaN
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%	3.3m	11.5b	United States
4	5	therock	91	6.8k	334.1m	1.9m	0.20%	665.3k	12.5b	United States
...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33.2m	623.8k	1.40%	464.7k	1.4b	United States
196	197	nancyajram	81	3.8k	33.2m	390.4k	0.64%	208.0k	1.5b	France
197	198	luansantana	79	0.77k	33.2m	193.3k	0.26%	82.6k	149.2m	Brazil
198	199	nickjonas	78	2.3k	33.0m	719.6k	1.42%	467.7k	1.7b	United States
199	200	raisa6690	80	4.2k	32.8m	232.2k	0.30%	97.4k	969.1m	Indonesia

200 rows × 10 columns

In [198…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   rank               200 non-null    int64
 1   channel_info       200 non-null    object
 2   influence_score    200 non-null    int64
 3   posts              200 non-null    object
 4   followers          200 non-null    object
 5   avg_likes          200 non-null    object
 6   60_day_eng_rate    200 non-null    object
 7   new_post_avg_like  200 non-null    object
 8   total_likes        200 non-null    object
 9   country            138 non-null    object
dtypes: int64(2), object(8)
memory usage: 15.8+ KB
```

In [200…

```python
df.describe()
```

Out[200…

	rank	influence_score
count	200.000000	200.000000
mean	100.500000	81.820000
std	57.879185	8.878159
min	1.000000	22.000000
25%	50.750000	80.000000
50%	100.500000	84.000000
75%	150.250000	86.000000
max	200.000000	93.000000

In [202…

```python
df.drop_duplicates(inplace=True)
```

In [204…

```python
df
```

Out[204…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%	5.9m	57.4b	United States
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%	4.4m	6.0b	NaN
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%	3.3m	11.5b	United States
4	5	therock	91	6.8k	334.1m	1.9m	0.20%	665.3k	12.5b	United States
...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33.2m	623.8k	1.40%	464.7k	1.4b	United States
196	197	nancyajram	81	3.8k	33.2m	390.4k	0.64%	208.0k	1.5b	France
197	198	luansantana	79	0.77k	33.2m	193.3k	0.26%	82.6k	149.2m	Brazil
198	199	nickjonas	78	2.3k	33.0m	719.6k	1.42%	467.7k	1.7b	United States
199	200	raisa6690	80	4.2k	32.8m	232.2k	0.30%	97.4k	969.1m	Indonesia

200 rows × 10 columns

In [206…

```python
# Handle missing values
# Fill missing numerical values with the median, and categorical with mode:
for column in df.columns:
    if df[column].dtype == 'object':
        df.fillna({column: df[column].mode()[0]}, inplace=True)
    else:
        df.fillna({column: df[column].median()}, inplace=True)
```

In [208…

```python
df
```

Out[208…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%	5.9m	57.4b	United States
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%	4.4m	6.0b	United States
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%	3.3m	11.5b	United States
4	5	therock	91	6.8k	334.1m	1.9m	0.20%	665.3k	12.5b	United States
...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33.2m	623.8k	1.40%	464.7k	1.4b	United States
196	197	nancyajram	81	3.8k	33.2m	390.4k	0.64%	208.0k	1.5b	France
197	198	luansantana	79	0.77k	33.2m	193.3k	0.26%	82.6k	149.2m	Brazil
198	199	nickjonas	78	2.3k	33.0m	719.6k	1.42%	467.7k	1.7b	United States
199	200	raisa6690	80	4.2k	32.8m	232.2k	0.30%	97.4k	969.1m	Indonesia

200 rows × 10 columns

In [210…

```python
# Step 1: Define the conversion function
def convert_shorthand(x):
    if isinstance(x, str):
        x = x.lower().strip()
        try:
            if x.endswith('k'):
                return float(x[:-1]) * 1_000
            elif x.endswith('m'):
                return float(x[:-1]) * 1_000_000
            elif x.endswith('b'):
                return float(x[:-1]) * 1_000_000_000
            else:
                return float(x)
        except ValueError:
            return None
    return x  # return as-is if already a number or None

# Step 2: Apply to the DataFrame
df['followers'] = df['followers'].apply(convert_shorthand)
df['followers'] = df['followers'].fillna(0).astype(int)  # Or Int64 if you want nullable ints

# Repeat if needed for other columns like 'posts', 'total_likes'
```

In [212…

```python
df
```

Out[212…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3.3k	475800000	8.7m	1.39%	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366200000	8.3m	1.62%	5.9m	57.4b	United States
2	3	leomessi	90	0.89k	357300000	6.8m	1.24%	4.4m	6.0b	United States
3	4	selenagomez	93	1.8k	342700000	6.2m	0.97%	3.3m	11.5b	United States
4	5	therock	91	6.8k	334100000	1.9m	0.20%	665.3k	12.5b	United States
...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33200000	623.8k	1.40%	464.7k	1.4b	United States
196	197	nancyajram	81	3.8k	33200000	390.4k	0.64%	208.0k	1.5b	France
197	198	luansantana	79	0.77k	33200000	193.3k	0.26%	82.6k	149.2m	Brazil
198	199	nickjonas	78	2.3k	33000000	719.6k	1.42%	467.7k	1.7b	United States
199	200	raisa6690	80	4.2k	32799999	232.2k	0.30%	97.4k	969.1m	Indonesia

200 rows × 10 columns

In [214…

```python
                              # Step 2: Exploratory Data Analysis (EDA)
```

In [216…

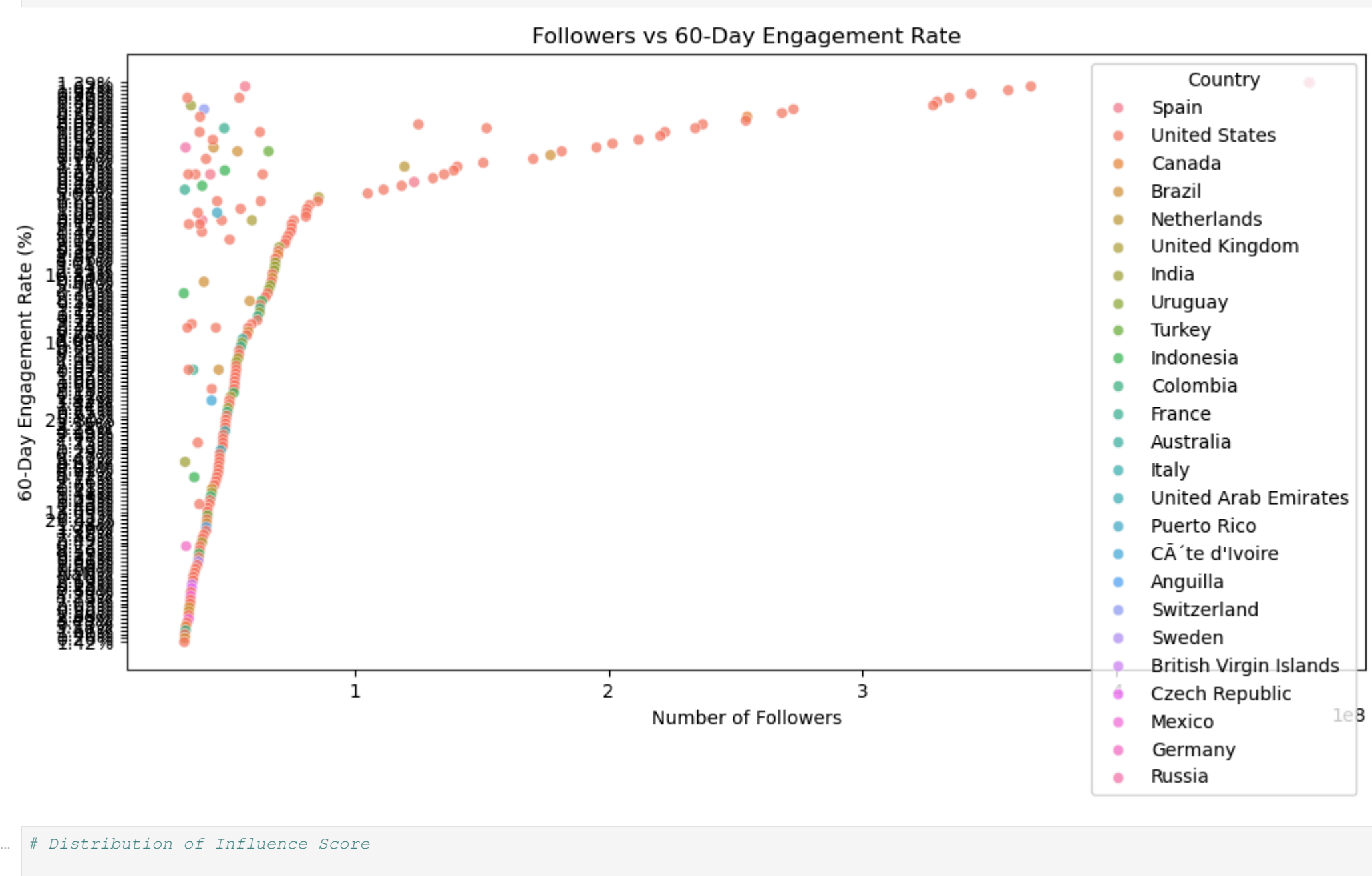
```python
#Display summary statistics for numeric columns
df[['influence_score', 'followers', 'avg_likes','60_day_eng_rate', 'new_post_avg_like']].describe()
```

Out[216…

	influence_score	followers
count	200.000000	2.000000e+02
mean	81.820000	7.740950e+07
std	8.878159	7.368727e+07
min	22.000000	3.280000e+07
25%	80.000000	4.000000e+07
50%	84.000000	5.005000e+07
75%	86.000000	6.890000e+07
max	93.000000	4.758000e+08

In [218…

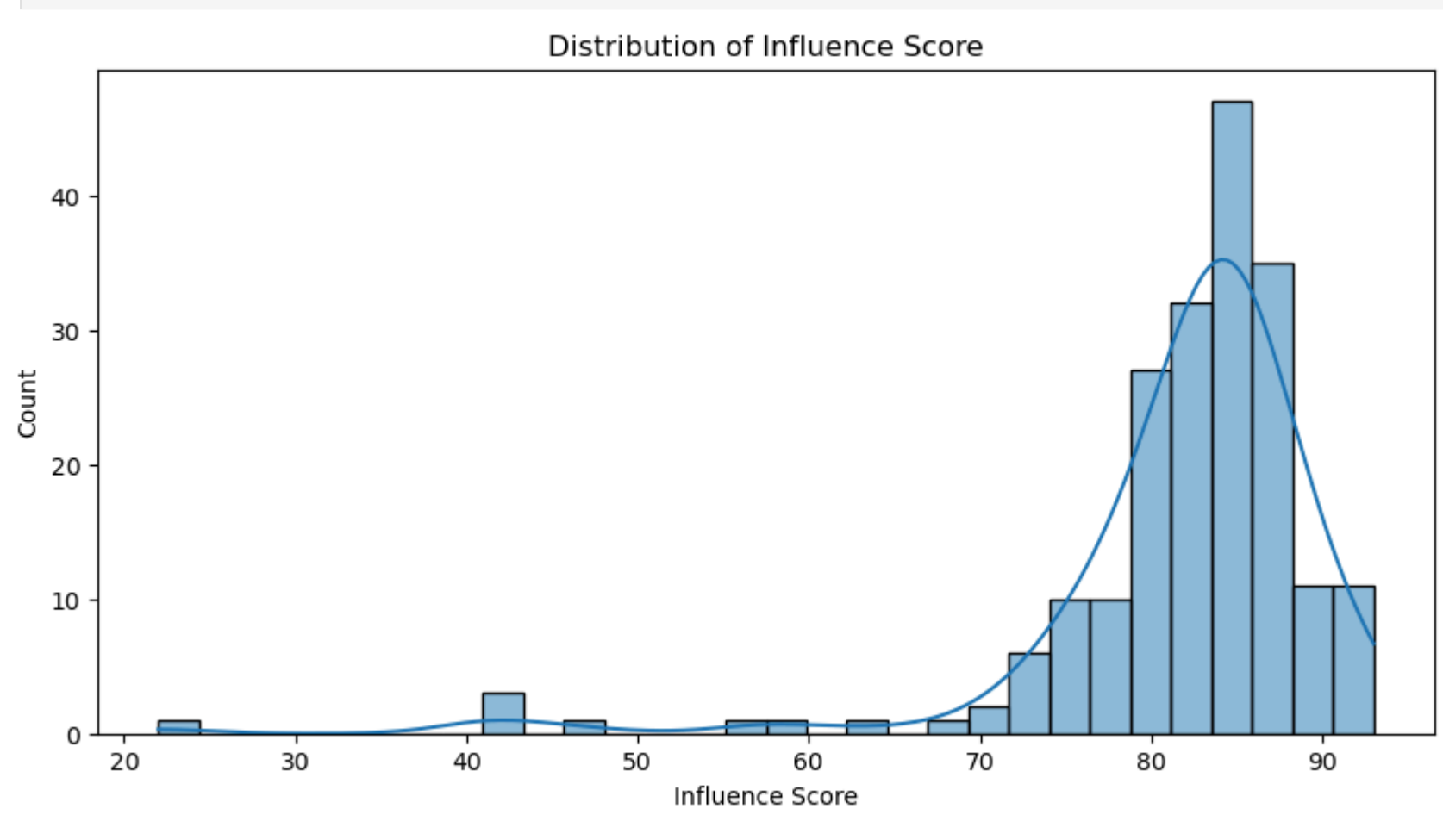
```python
#Relationship between Followers and Engagement
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='followers', y='60_day_eng_rate',
hue='country', alpha=0.7)
plt.title('Followers vs 60-Day Engagement Rate')
plt.xlabel('Number of Followers')
plt.ylabel('60-Day Engagement Rate (%)')
plt.legend(title='Country')
plt.show()
```



In [219…

```python
# Distribution of Influence Score

plt.figure(figsize=(10, 5))
sns.histplot(df['influence_score'], bins=30, kde=True)
plt.title('Distribution of Influence Score')
plt.xlabel('Influence Score')
plt.show()
```



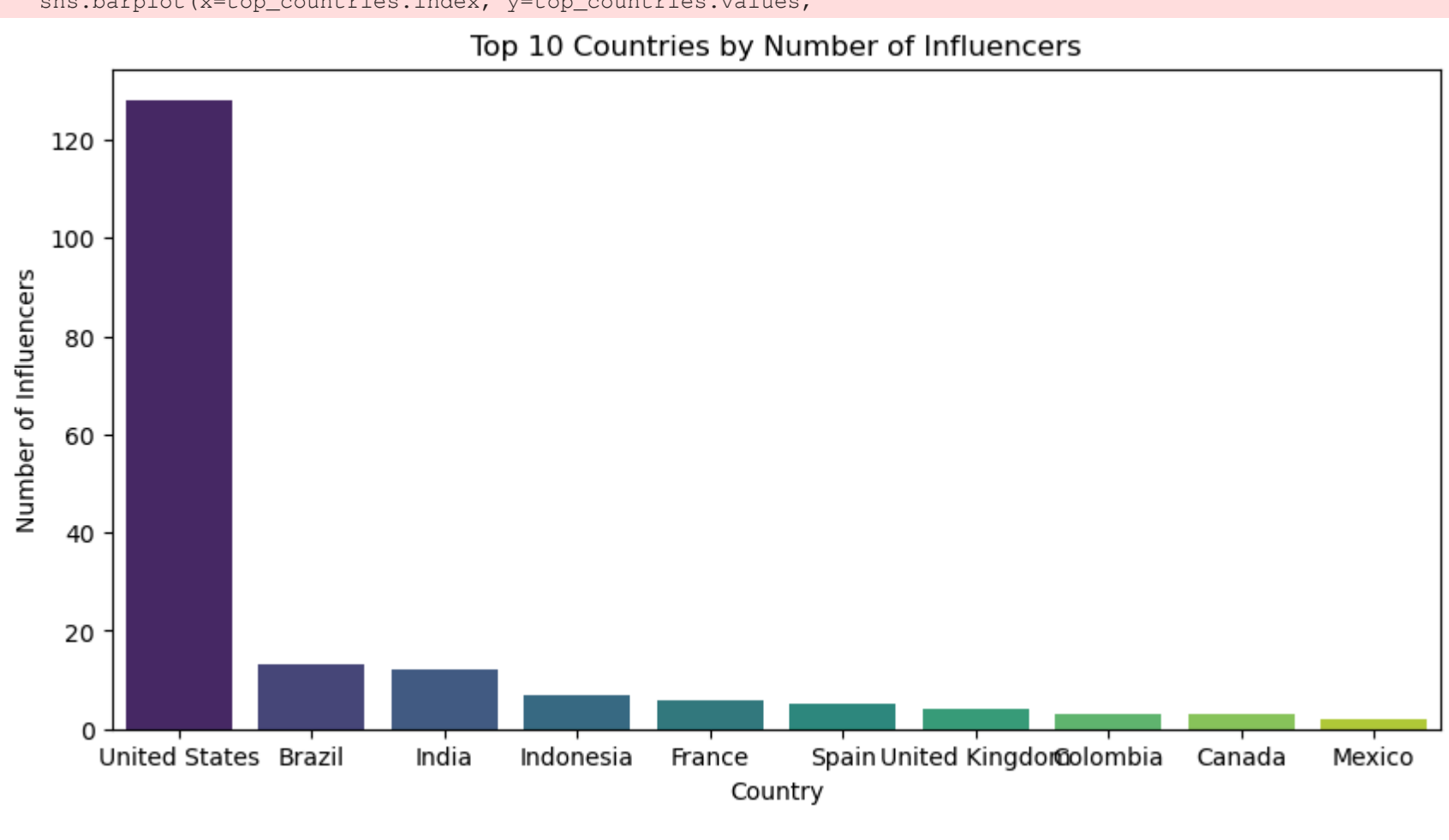
In [221…

```python
# Most Active Countries
top_countries = df['country'].value_counts().head(10)
plt.figure(figsize=(10, 5))
sns.barplot(x=top_countries.index, y=top_countries.values,
palette="viridis")
plt.title('Top 10 Countries by Number of Influencers')
plt.xlabel('Country')
plt.ylabel('Number of Influencers')
plt.show()
```

```
C:\Users\intel\AppData\Local\Temp\ipykernel_30540\1408171692.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_countries.index, y=top_countries.values,
```



In [224…

```python
                              #Step 3: Feature Engineering
```

In [226…

```python
# Creating engagement-related features
def convert_shorthand(x):
    if isinstance(x, str):
        x = x.lower().strip().replace(',', '')  # remove commas like 1,000
        try:
            if x.endswith('k'):
                return float(x[:-1]) * 1_000
            elif x.endswith('m'):
                return float(x[:-1]) * 1_000_000
            elif x.endswith('b'):
                return float(x[:-1]) * 1_000_000_000
            else:
                return float(x)
        except ValueError:
            return None
    return x
columns_to_convert = ['followers', 'total_likes', 'posts', 'avg_likes']

for col in columns_to_convert:
    df[col] = df[col].apply(convert_shorthand)
    df[col] = pd.to_numeric(df[col], errors='coerce')  # just in case
df['like_follower_ratio'] = df['total_likes'] / df['followers']
df['post_follower_ratio'] = df['posts'] / df['followers']
df['avg_likes_ratio'] = df['avg_likes'] / df['followers']
```

In [228…

```python
df
```

Out[228…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country	like_follower_ratio	post_follower_ratio	avg_likes_ratio
0	1	cristiano	92	3300.0	475800000	8700000.0	1.39%	6.5m	2.900000e+10	Spain	60.949979	0.000007	0.018285
1	2	kyliejenner	91	6900.0	366200000	8300000.0	1.62%	5.9m	5.740000e+10	United States	156.744948	0.000019	0.022665
2	3	leomessi	90	890.0	357300000	6800000.0	1.24%	4.4m	6.000000e+09	United States	16.792611	0.000002	0.019032
3	4	selenagomez	93	1800.0	342700000	6200000.0	0.97%	3.3m	1.150000e+10	United States	33.557047	0.000005	0.018092
4	5	therock	91	6800.0	334100000	1900000.0	0.20%	665.3k	1.250000e+10	United States	37.413948	0.000020	0.005687
...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2300.0	33200000	623800.0	1.40%	464.7k	1.400000e+09	United States	42.168675	0.000069	0.018789
196	197	nancyajram	81	3800.0	33200000	390400.0	0.64%	208.0k	1.500000e+09	France	45.180723	0.000114	0.011759
197	198	luansantana	79	770.0	33200000	193300.0	0.26%	82.6k	1.492000e+08	Brazil	4.493976	0.000023	0.005822
198	199	nickjonas	78	2300.0	33000000	719600.0	1.42%	467.7k	1.700000e+09	United States	51.515152	0.000070	0.021806
199	200	raisa6690	80	4200.0	32799999	232200.0	0.30%	97.4k	9.691000e+08	Indonesia	29.545733	0.000128	0.007079

200 rows × 13 columns

In [230…

```python
                              #Step 4: Model Building
```

In [232…

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [234…

```python
# Define feature columns and target variable
X = df[['followers', 'avg_likes', '60_day_eng_rate','new_post_avg_like', 'like_follower_ratio','post_follower_ratio']]
y = df['influence_score']
```

In [236…

```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state=42)
```

In [238…

```python
non_numeric_cols = X_train.select_dtypes(include=['object']).columns
print("Non-numeric columns in X_train:", non_numeric_cols)
def clean_percent_column(col):
    return col.apply(lambda x: float(x.strip('%')) / 100 if isinstance(x, str) and '%' in x else x)

# Clean all object columns with possible %
for col in non_numeric_cols:
    X_train[col] = clean_percent_column(X_train[col])
    X_train[col] = pd.to_numeric(X_train[col], errors='coerce')

    X_test[col] = clean_percent_column(X_test[col])
    X_test[col] = pd.to_numeric(X_test[col], errors='coerce')
print(X_train.dtypes)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
Non-numeric columns in X_train: Index(['60_day_eng_rate', 'new_post_avg_like'], dtype='object')
followers                int64
avg_likes              float64
60_day_eng_rate        float64
new_post_avg_like      float64
like_follower_ratio    float64
post_follower_ratio    float64
dtype: object
```

In [240…

```python
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [242…

```python
# Initialize and train a Random Forest Regressor
model = RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train_scaled, y_train)
```

Out[242…

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [244…

```python
# Predictions and evaluation
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

```
Mean Squared Error: 170.45508750000005
R^2 Score: -0.008196117717948814
```

In [246…

```python
                              #Step 5: Model Interpretation and Feature Importance
```

In [248…

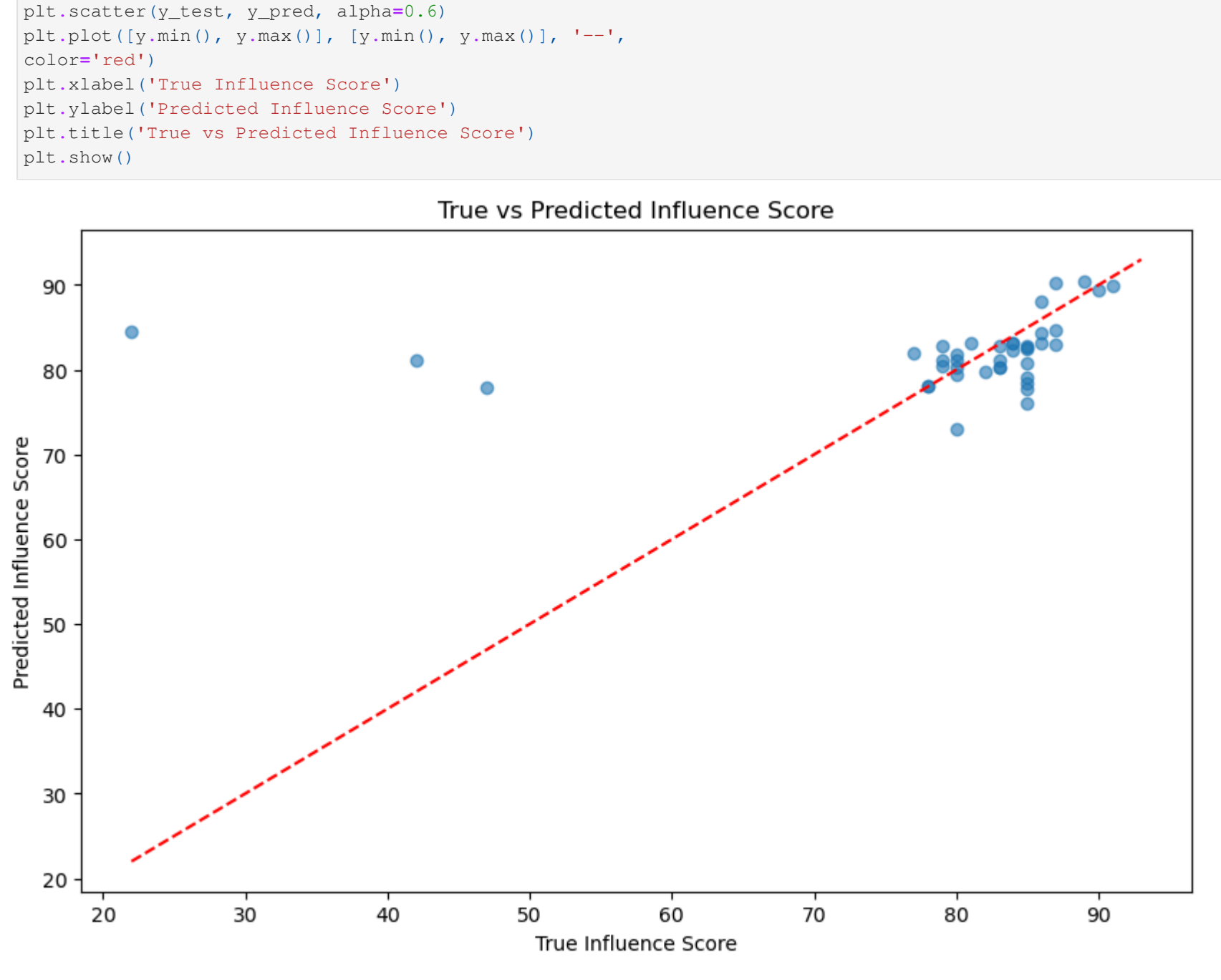
```python
# Display feature importances
feature_importances = pd.Series(model.feature_importances_,
index=X.columns)
feature_importances.sort_values().plot(kind='barh',
title='Feature Importance')
plt.show()
```

In [250…

```python
                              #   Step 6: Visualizing Predictions
```

In [252…

```python
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y.min(), y.max()], [y.min(), y.max()], '--',
color='red')
plt.xlabel('True Influence Score')
plt.ylabel('Predicted Influence Score')
plt.title('True vs Predicted Influence Score')
plt.show()
```



In [254…

```python
                              #Step 7: Final Observations and Summary
```

In [256…

```python
# Top Influential Factors: Feature importance analysis indicates which features most
# influence the influence_score.
# Model Performance: With the achieved R² score, assess the accuracy of the model
# in predicting an influencer's influence score based on follower metrics.
# Business Insights: Using insights on top-engaging influencers by country and
# engagement rates, businesses can strategize influencer collaborations for marketing.
```

In [258…

```python
                              # END - PROJECT #
```

In [ ]: