Instagram Influencers Analytics Project Report

This report provides a comprehensive analysis of top Instagram influencers using data-driven techniques.

The objective was to understand the distribution, engagement patterns, and key performance indicators of influencers on Instagram through visualization and predictive modeling.

Dataset Overview

The dataset includes 200 Instagram influencers with fields like followers, posts, likes, engagement rate, and country.

Missing values in the 'country' column were imputed using backfill and forward fill techniques.

Metric units like 'k', 'm', and 'b' were standardized into numerical formats for analysis.

Key Visual Insights

1. Distribution of Followers: Most influencers have between 50M to 400M followers.
2. Top Countries: The United States dominates the influencer landscape.
3. Total Likes: Cristiano, Kylie Jenner, and The Rock lead in overall engagement.
4. Engagement Analysis: Average Likes strongly correlate with New Post Average Likes.
5. Country-wise Engagement: Top countries exhibit varied engagement rate distributions.
6. Correlation Heatmap: Influence score correlates best with average likes and total likes.

Predictive Modeling

A Random Forest Regressor was trained to predict an influencer's score based on engagement and content statistics.

Features included: followers, avg likes, engagement rate, new post likes, and derived ratios.

Model Performance:

- Mean Squared Error (MSE): 171.79
- R^2 Score: -0.016 (indicating poor predictive performance)

The model indicates the need for more meaningful or broader features to accurately predict influence.

Recommendations

- Consider adding metrics like comments, shares, and content category for better influence modeling.
- Improve model accuracy by incorporating influencer trends over time.
- Normalize country data and include cultural/geographic factors affecting engagement.
- For marketers, focus on influencers with consistent engagement-to-follower ratios.

In [5]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_excel('insta.xlsx')
df.head()
```

Out[5]:

	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	0.0139	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366.2m	8.3m	0.0162	5.9m	57.4b	United States
2	3	leomessi	90	0.89k	357.3m	6.8m	0.0124	4.4m	6.0b	NaN
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.0097	3.3m	11.5b	United States
4	5	therock	91	6.8k	334.1m	1.9m	0.002	665.3k	12.5b	United States

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
rank                  0
channel_info          0
influence_score       0
posts                 0
followers             0
avg_likes             0
60_day_eng_rate       0
new_post_avg_like     0
total_likes           0
country              62
dtype: int64
```

In [9]:
```
df.shape
```

Out[9]:
```
(200, 10)
```

In [11]:

```
replace = {'b': 'e9', 'm': 'e6', 'k': 'e3', '%': ''}
columns_to_convert = ['total_likes', 'posts', 'followers',
                      'avg_likes', '60_day_eng_rate', 'new_post_avg_like']

# Replace and convert columns
df[columns_to_convert] = df[columns_to_convert].replace(replace, regex=True).astype(float)
```

In [13]:

```
df.head()
```

Out[13]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	cou
0	1	cristiano	92	3300.0	475800000.0	8700000.0	0.0139	6500000.0	2.900000e+10	S
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	0.0162	5900000.0	5.740000e+10	U S
2	3	leomessi	90	890.0	357300000.0	6800000.0	0.0124	4400000.0	6.000000e+09	
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	0.0097	3300000.0	1.150000e+10	U S
4	5	therock	91	6800.0	334100000.0	1900000.0	0.0020	665300.0	1.250000e+10	U S

In [15]:

```
df['country']=df['country'].bfill().ffill()
```

In [17]:

```
df.head()
```

Out[17]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	cou
0	1	cristiano	92	3300.0	475800000.0	8700000.0	0.0139	6500000.0	2.900000e+10	S
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	0.0162	5900000.0	5.740000e+10	U S
2	3	leomessi	90	890.0	357300000.0	6800000.0	0.0124	4400000.0	6.000000e+09	U S
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	0.0097	3300000.0	1.150000e+10	U S
4	5	therock	91	6800.0	334100000.0	1900000.0	0.0020	665300.0	1.250000e+10	U S

In [19]:

```
df.isnull().sum()
```

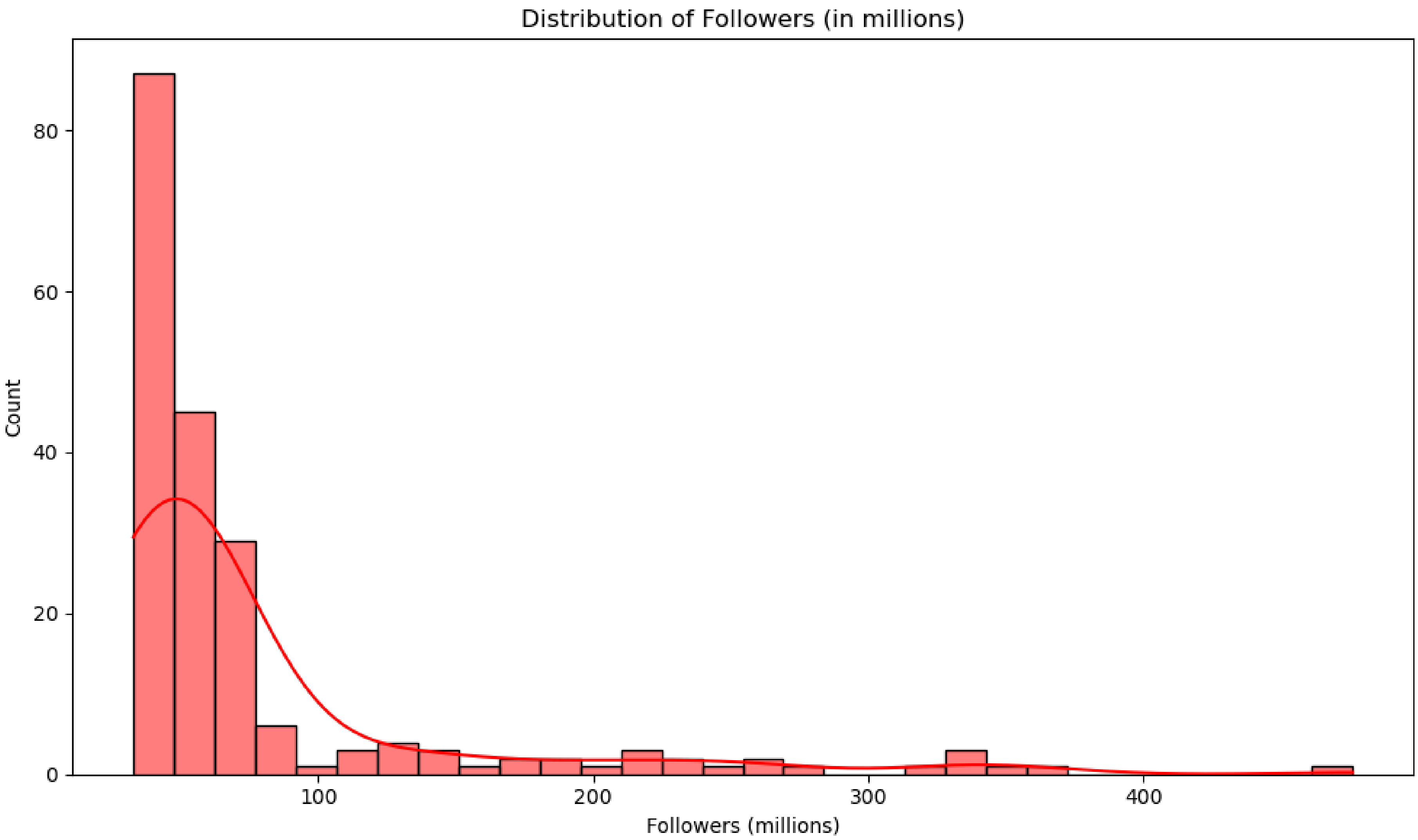
Out[19]:

```
rank                 0
channel_info         0
influence_score      0
posts                0
followers            0
avg_likes            0
60_day_eng_rate      1
new_post_avg_like    0
total_likes          0
country              0
dtype: int64
```

Distribution of Followers

In [21]:

```
plt.figure(figsize=(10, 6))
sns.histplot(df['followers'] / 1e6, bins=30, color='red', kde=True)
plt.title("Distribution of Followers (in millions)")
plt.xlabel("Followers (millions)")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

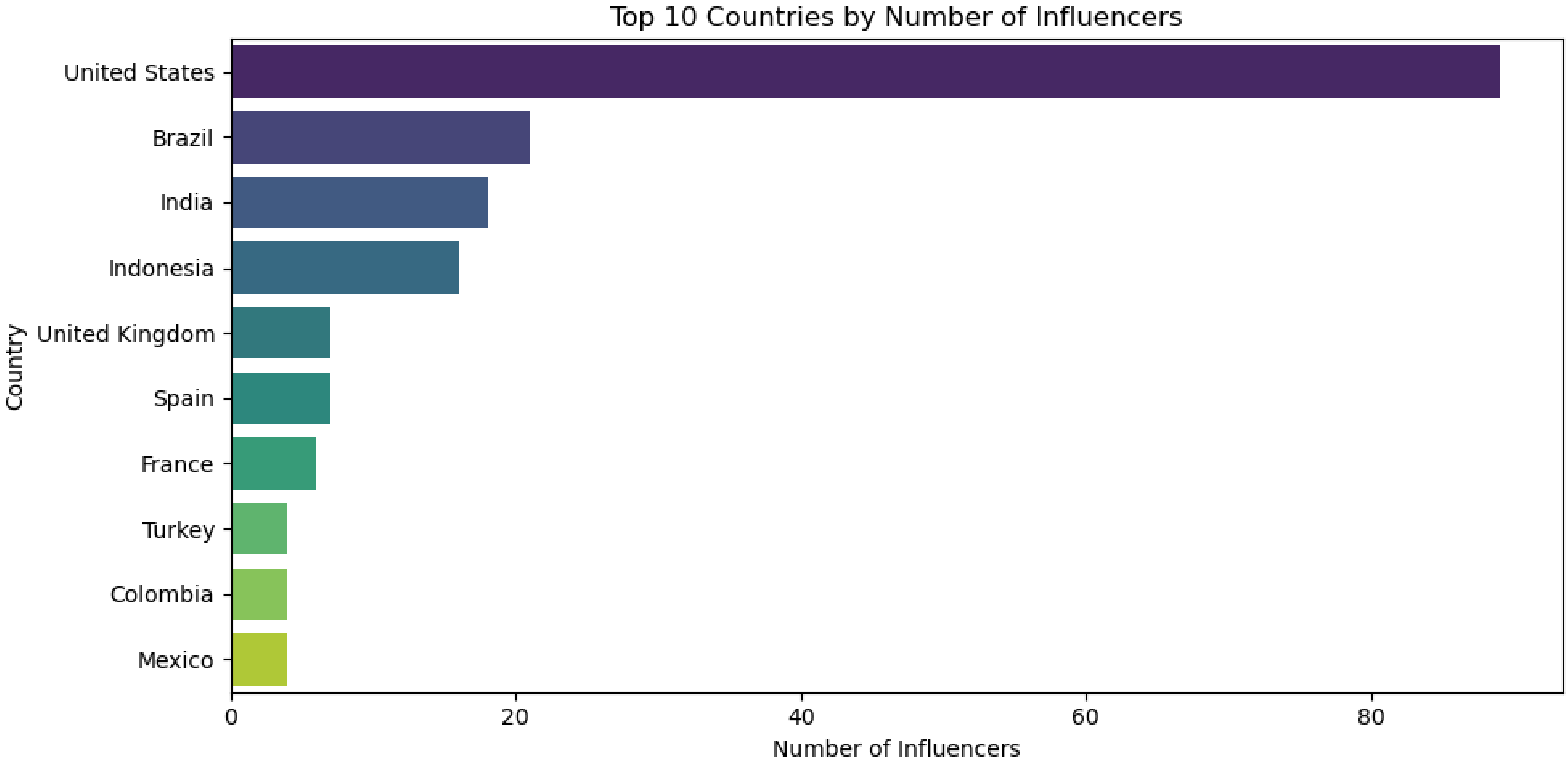


Top 10 Countries By Influencer Count

In [23]:

```
top_countries = df['country'].value_counts().head(10)

plt.figure(figsize=(10, 5))
sns.barplot(x=top_countries.values, y=top_countries.index, palette='viridis')
plt.title("Top 10 Countries by Number of Influencers")
plt.xlabel("Number of Influencers")
plt.ylabel("Country")
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_4822/183342791.py:4: FutureWarning:

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

  sns.barplot(x=top_countries.values, y=top_countries.index, palette='viridis')
```

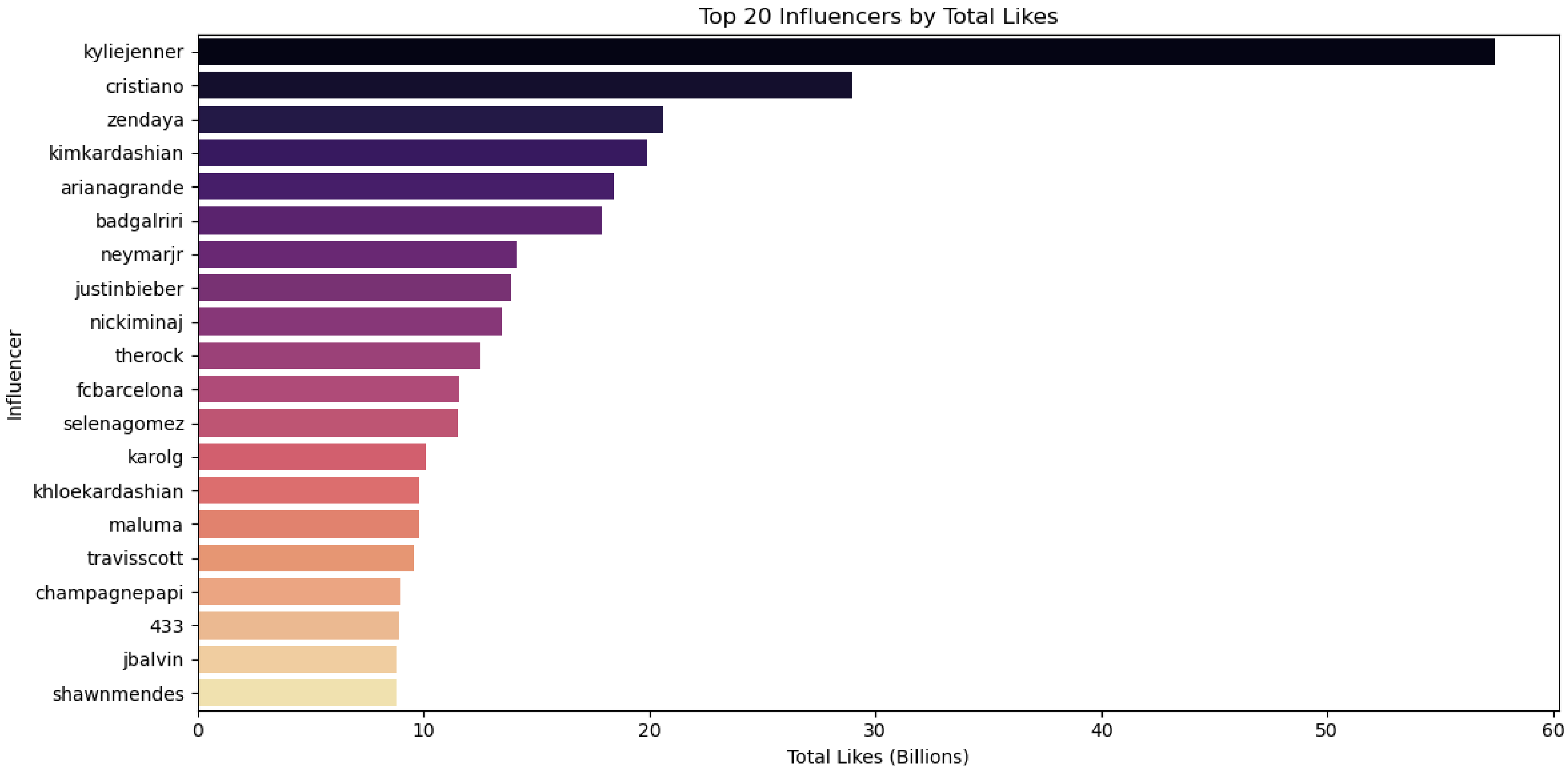


Top 20 Influencers by Total Likes

In [25]:

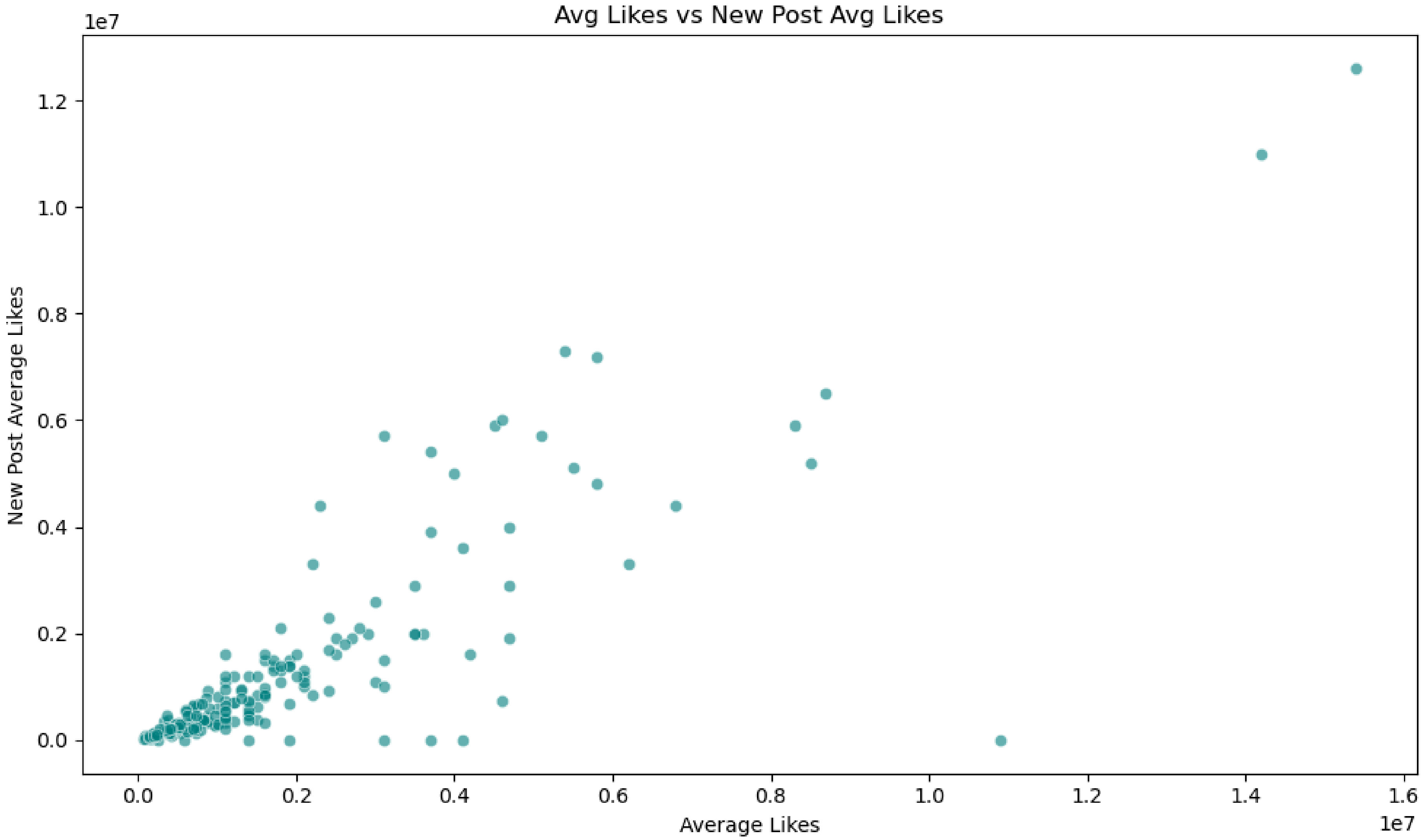
```python
top_liked = df.sort_values(by='total_likes', ascending=False).head(20)

plt.figure(figsize=(12, 6))
sns.barplot(y=top_liked['channel_info'], x=top_liked['total_likes'] / 1e9, palette='magma')
plt.title("Top 20 Influencers by Total Likes")
plt.xlabel("Total Likes (Billions)")
plt.ylabel("Influencer")
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_4822/4048472697.py:4: FutureWarning:

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

  sns.barplot(y=top_liked['channel_info'], x=top_liked['total_likes'] / 1e9, palette='magma')
```



Average Likes vs New Post Average Likes

```
In [27]: plt.figure(figsize=(10, 6))
         sns.scatterplot(data=df, x='avg_likes', y='new_post_avg_like', alpha=0.6, color='teal')
         plt.title("Avg Likes vs New Post Avg Likes")
         plt.xlabel("Average Likes")
         plt.ylabel("New Post Average Likes")
         plt.tight_layout()
         plt.show()
```

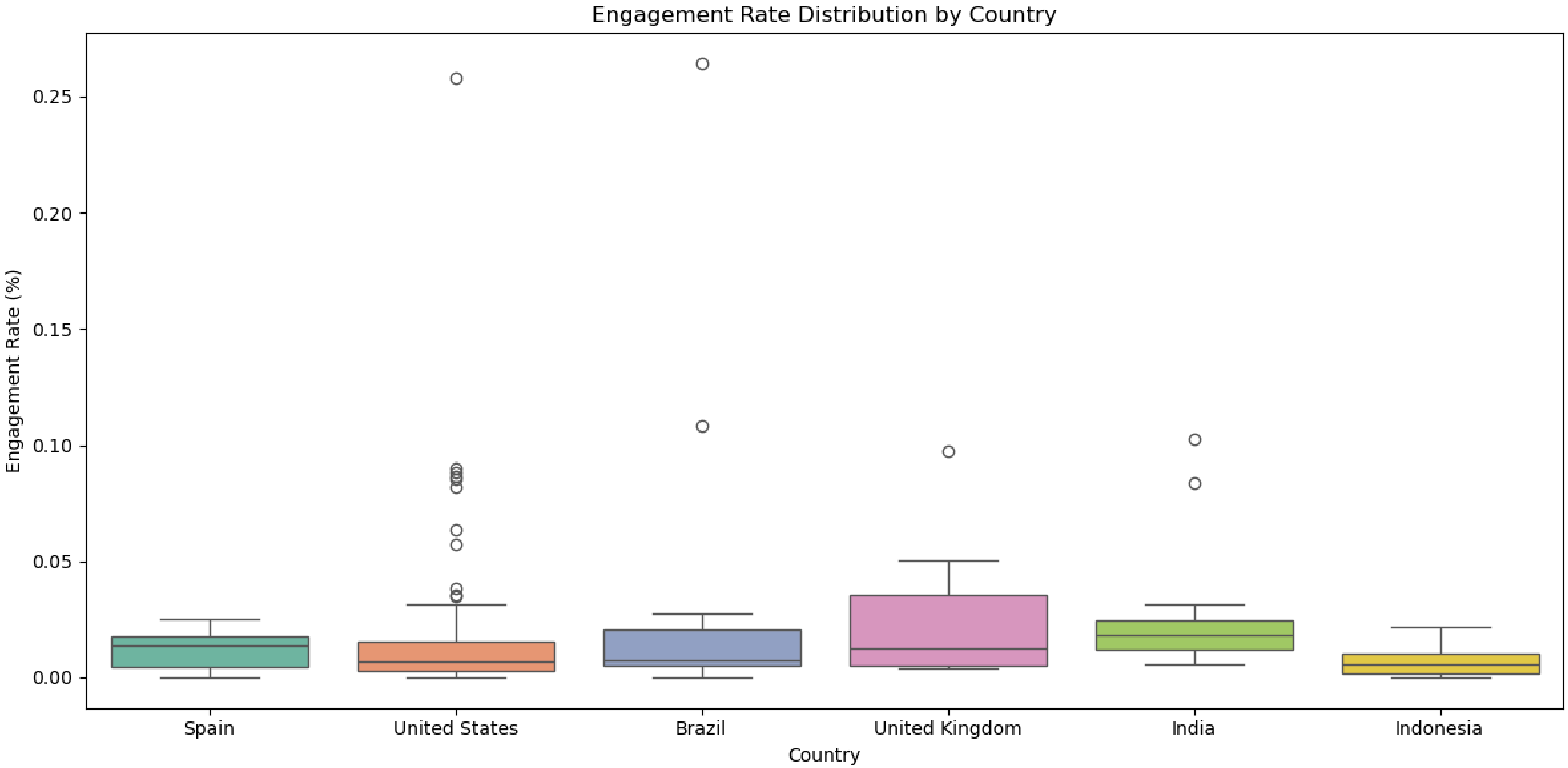


Boxplot: Engagement Rate by Country (Top 6 Only)

```
In [29]: top_countries = df['country'].value_counts().nlargest(6).index
         filtered = df[df['country'].isin(top_countries)]

         plt.figure(figsize=(12, 6))
         sns.boxplot(x='country', y='60_day_eng_rate', data=filtered, palette='Set2')
```

```
plt.title("Engagement Rate Distribution by Country")
plt.xlabel("Country")
plt.ylabel("Engagement Rate (%)")
plt.tight_layout()
plt.show()
```

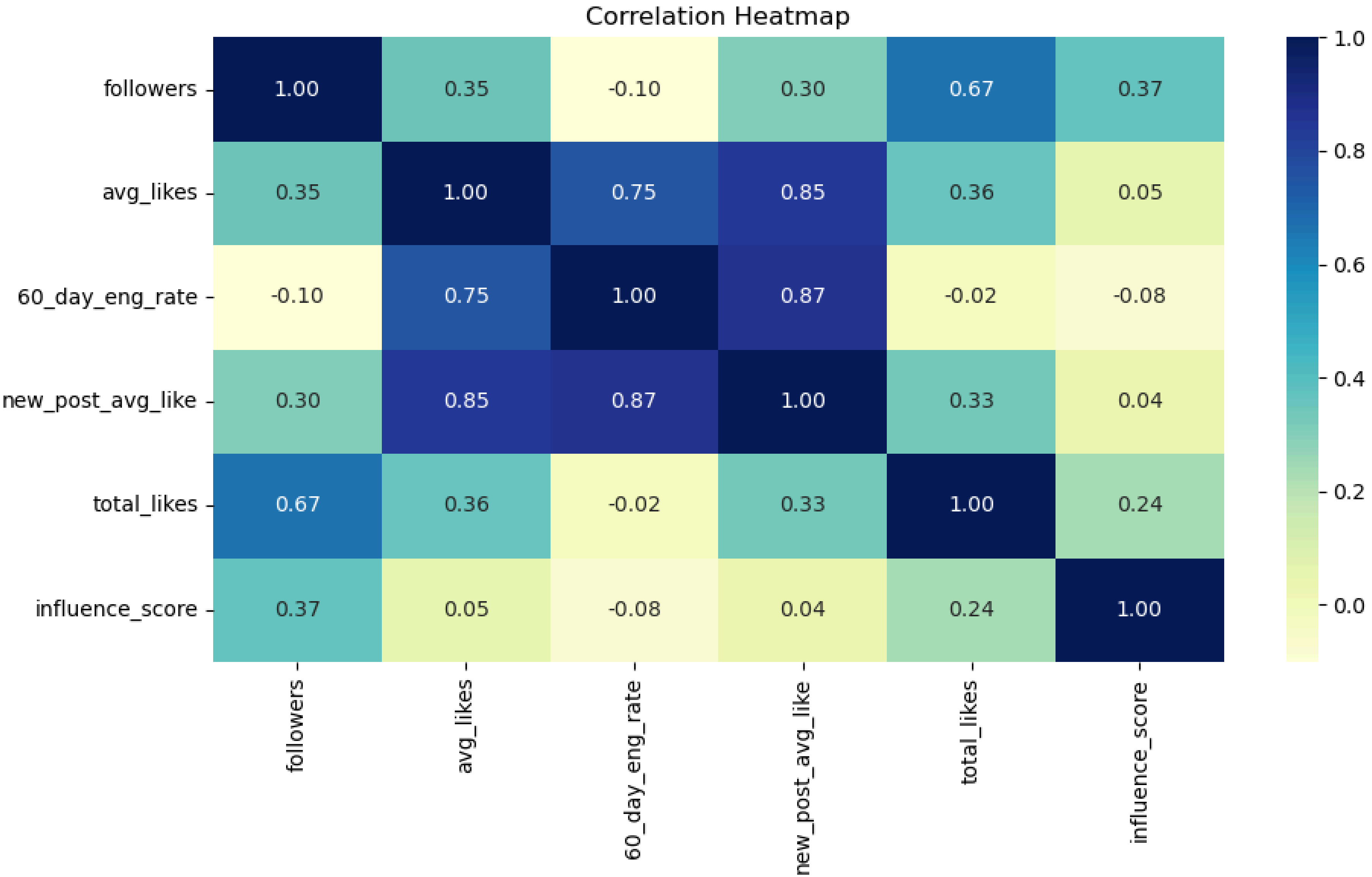
```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_4822/3869602318.py:5: FutureWarning:

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.boxplot(x='country', y='60_day_eng_rate', data=filtered, palette='Set2')
```



Heatmap: Top Features vs Influence Score

In [31]:

```
plt.figure(figsize=(10, 6))
sns.heatmap(df[['followers', 'avg_likes', '60_day_eng_rate', 'new_post_avg_like', 'total_likes', 'influence_score']
            annot=True, cmap='YlGnBu', fmt=".2f")
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()
```



In [47]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [49]:

```
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 [51]:

```
df.head()
```

Out[51]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	cou
0	1	cristiano	92	3300.0	475800000.0	8700000.0	0.0139	6500000.0	2.900000e+10	S
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	0.0162	5900000.0	5.740000e+10	US
2	3	leomessi	90	890.0	357300000.0	6800000.0	0.0124	4400000.0	6.000000e+09	US
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	0.0097	3300000.0	1.150000e+10	US
4	5	therock	91	6800.0	334100000.0	1900000.0	0.0020	665300.0	1.250000e+10	US

In [53]:

```
X = df[['followers', 'avg_likes', '60_day_eng_rate',
        'new_post_avg_like', 'like_follower_ratio',
        'post_follower_ratio']]
y = df['influence_score']
```

In [55]:

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

Out[55]:

```
((160, 6), (40, 6), (160,), (40,))
```

In [57]:

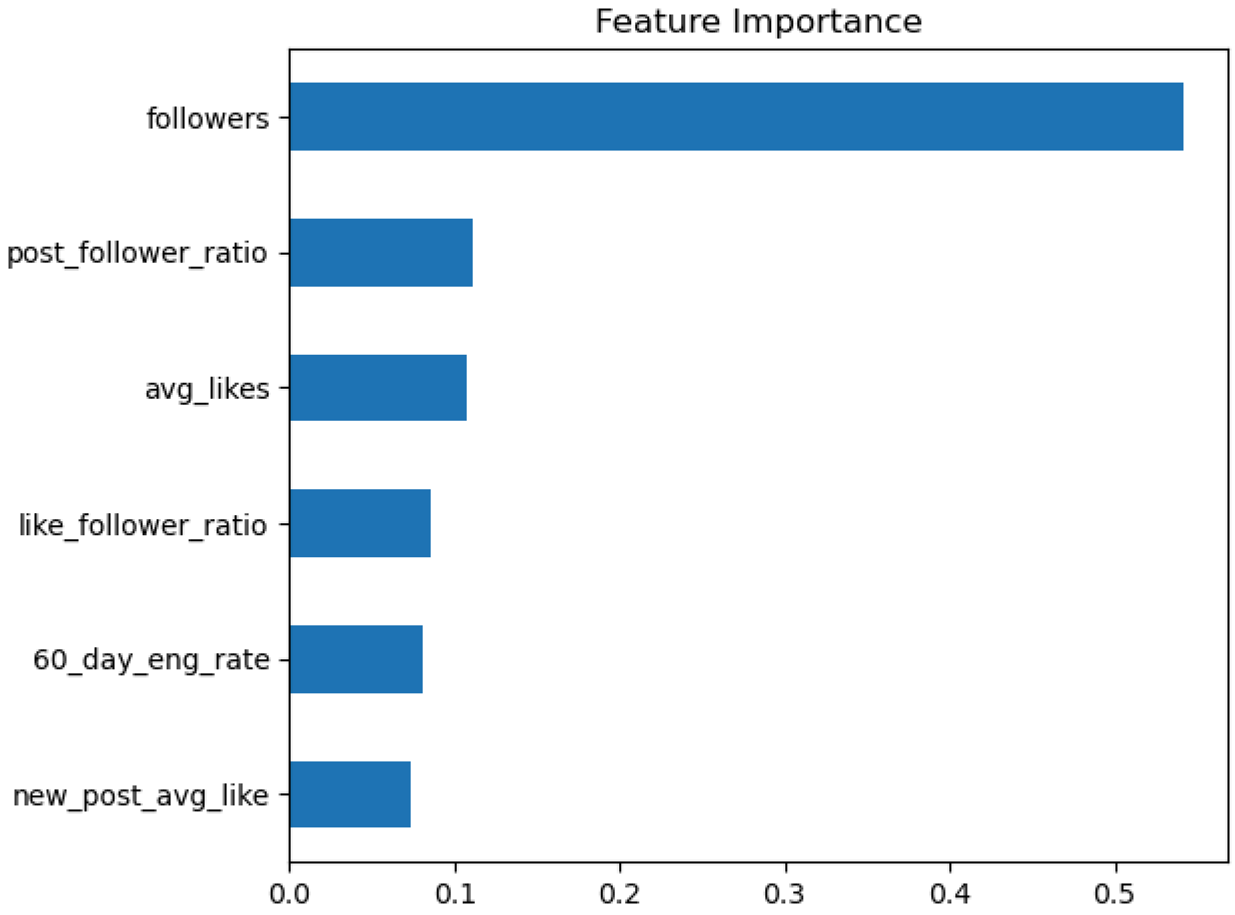
```
# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

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

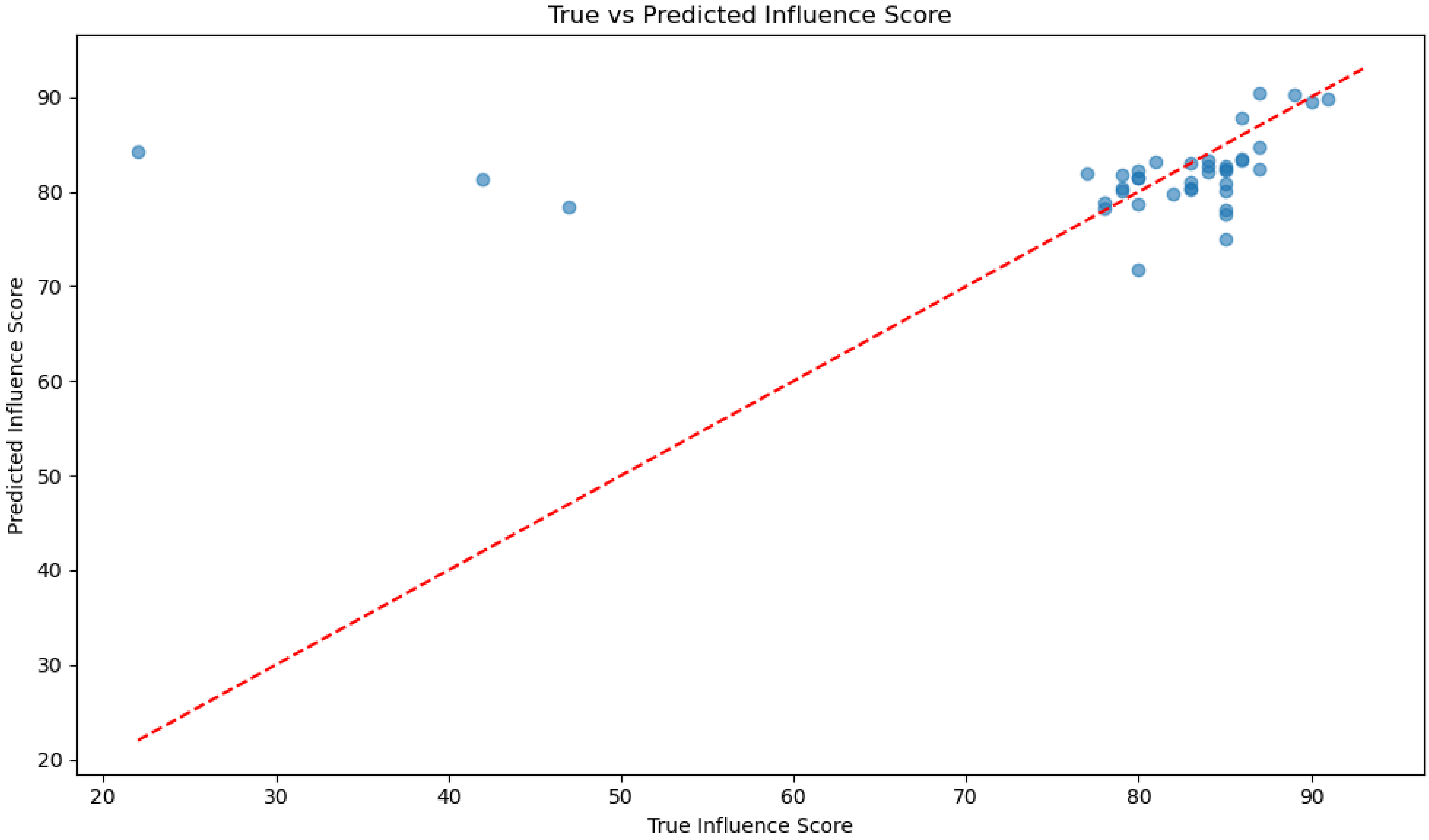
```
# Predict and evaluate
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

```
Mean Squared Error: 171.79258749999997
R² Score: -0.01610707143147594
```

In [59]:

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

# Plot actual vs predicted
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.tight_layout()
plt.show()
```


Feature Importance
followers
post_follower_ratio
avg_likes
like_follower_ratio
60_day_eng_rate
new_post_avg_like
0.0
0.1
0.2
0.3
0.4
0.5


True vs Predicted Influence Score
Predicted Influence Score
True Influence Score
20
30
40
50
60
70
80
90
20
30
40
50
60
70
80
90

```
In [ ]:
```