Top Instagram Influencers Data Analytics

Objective

- Importing the required libraries and the dataset
- Data Cleaning and Preprocessing
- Exploratory Data Analysis
- Data Visualization
- Feature Engineering
- Modeling
- Prediction Visualization

Overview

The dataset includes 200 top Instagram influencers with attributes like:

- rank: Ranking by followers
- channel_info: Instagram username
- influence_score: Calculated metric based on visibility & engagement
- followers, posts, avg_likes, total_likes
- 60_day_eng_rate: Recent engagement rate
- new_post_avg_like: Avg likes on recent posts
- country: Origin country

Data Cleaning and Preprocessing

Importing required Libraries and the Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv(
    r"C:\Users\91931\Downloads\Intern\Dataset\top_insta_influencers_data.csv"
)
```

Data Cleaning and Preprocessing

```
data.drop_duplicates(inplace=True)

data.isnull().value_counts()

replace = {"b": "e9", "m": "e6", "k": "e3", "%": ""}
convert_column = [
    "total_likes",
    "posts",
    "followers",
    "avg_likes",
    "60_day_eng_rate",
    "new_post_avg_like",
]
data[convert_column] = data[convert_column].replace(replace,
regex=True).astype(float)
data[convert_column]
```

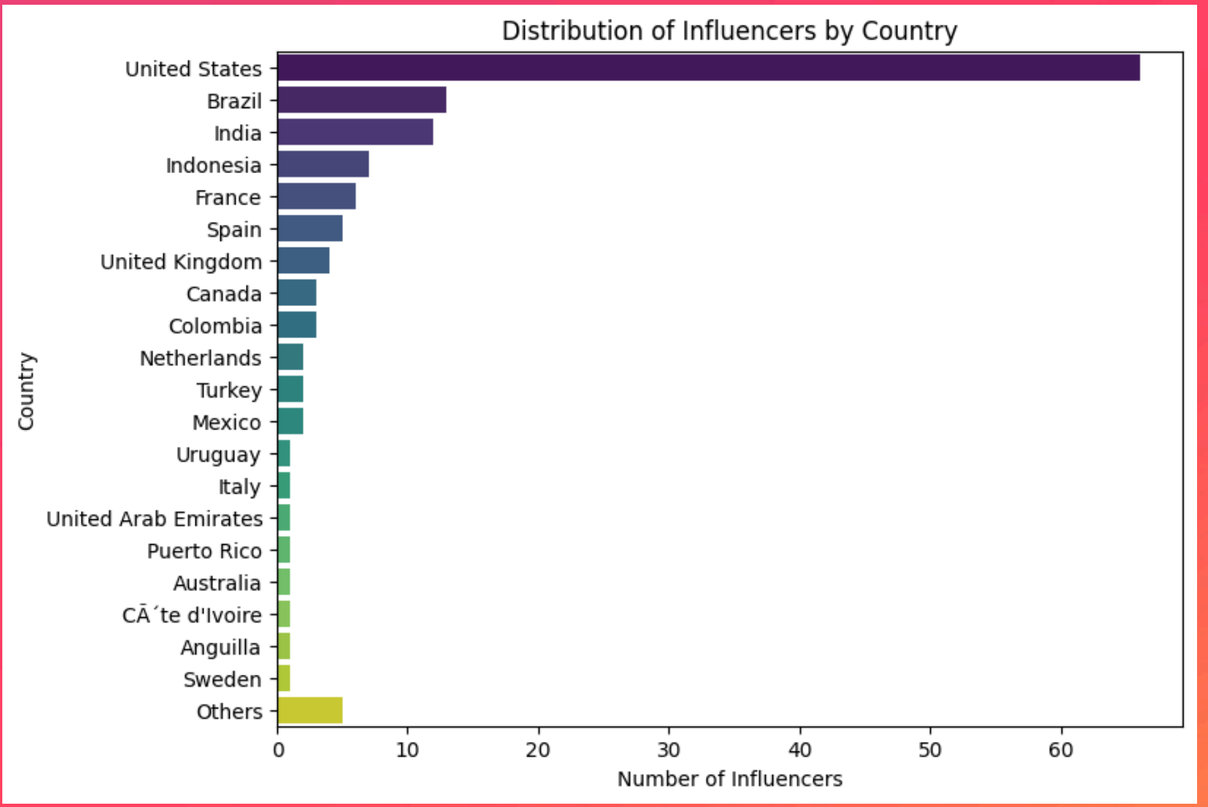
Exploratory Data Analysis

Exploratory Data Analysis

```
country = data["country"].value_counts()
country

import seaborn as sns

plt.figure(figsize=(15, 8))
plt.title("Influencers on Instagram from nations")
sns.countplot(x=data["country"])
plt.xticks(rotation=90)

# Filtering influencers with over 1 million followers
highest_follower = data[data["followers"] > 1_000_000]
highest_follower

high_engagement = data[data["60_day_eng_rate"] > 5]
high_engagement

top10_er = data.drop(
    [
        "rank",
        "influence_score",
        "posts",
        "avg_likes",
        "new_post_avg_like",
        "total_likes",
        "country",
    ],
    axis=1,
)
top10_er.head(10)
```

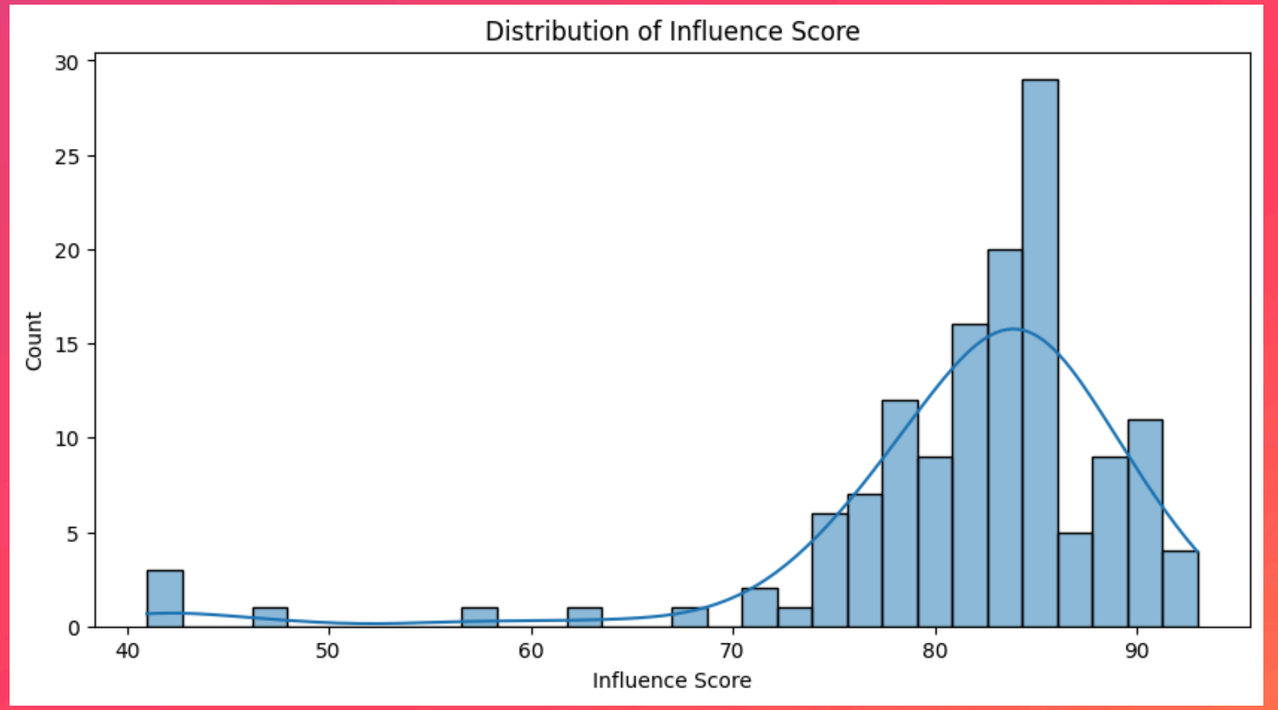
Data Visualization

```python
country = data["country"].value_counts()[:20].to_list()
name_countries = data["country"].value_counts().index[:20].to_list()
name_countries.append("Others")
max20 = sum(country)
others = len(data) - max20
country.append(others)

plt.figure(figsize=(8, 6))
sns.barplot(x=country, y=name_countries, palette="viridis")
plt.title("Distribution of Influencers by Country")
plt.xlabel("Number of Influencers")
plt.ylabel("Country")
plt.show()
```



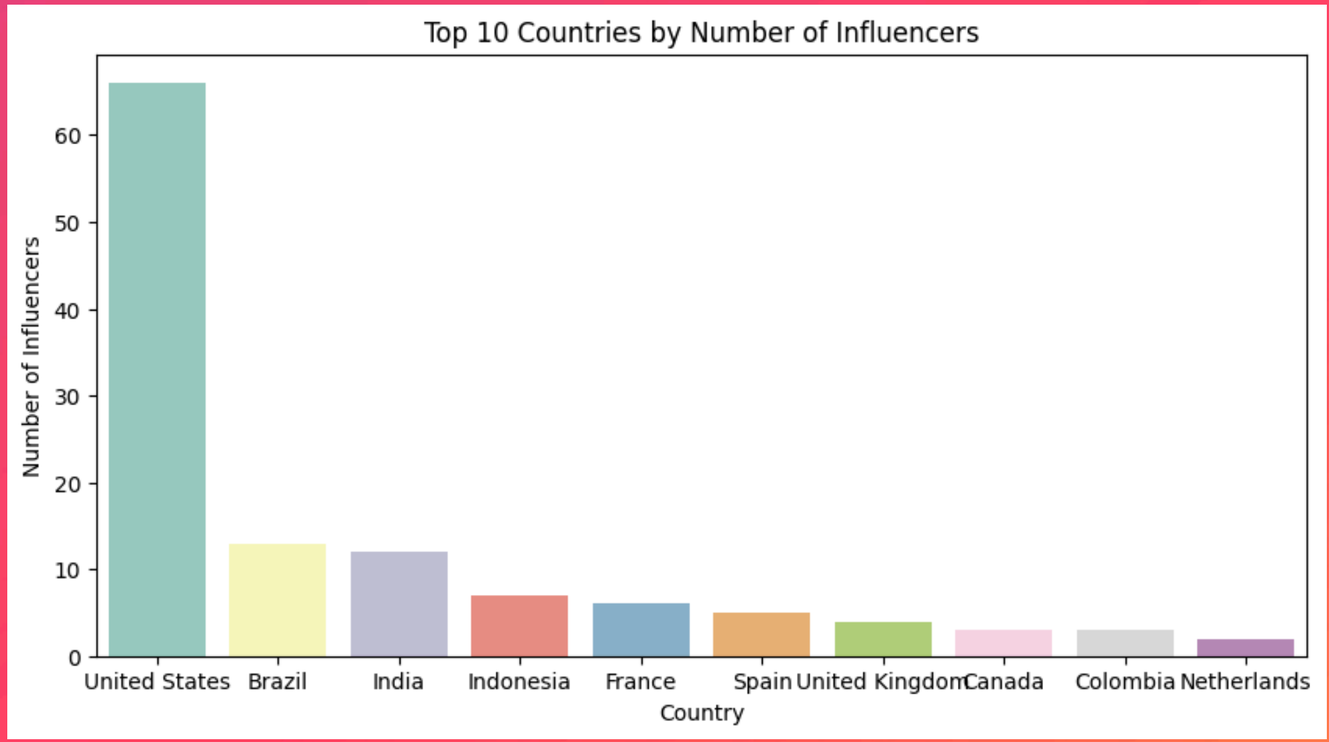
Data Visualization

```python
# Distribution of Influence Score

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



Data Visualization

```python
# Most Active Countries

top_countries = data["country"].value_counts().head(10)
plt.figure(figsize=(10, 5))
sns.barplot(
    x=top_countries.index,
    y=top_countries.values,
    hue=top_countries.index,
    legend=False,
    palette="Set3",
)
plt.title("Top 10 Countries by Number of Influencers")
plt.xlabel("Country")
plt.ylabel("Number of Influencers")
plt.show()
```



Feature Engineering

```
# To improve our model's performance, let's create a few additional features
that might be helpful predictors.

data["like_follower_ratio"] = data["total_likes"] / data["followers"]
data["post_follower_ratio"] = data["posts"] / data["followers"]
data["avg_likes_ratio"] = data["avg_likes"] / data["followers"]
```

Model Building

```
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

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

Random Forest Regressor

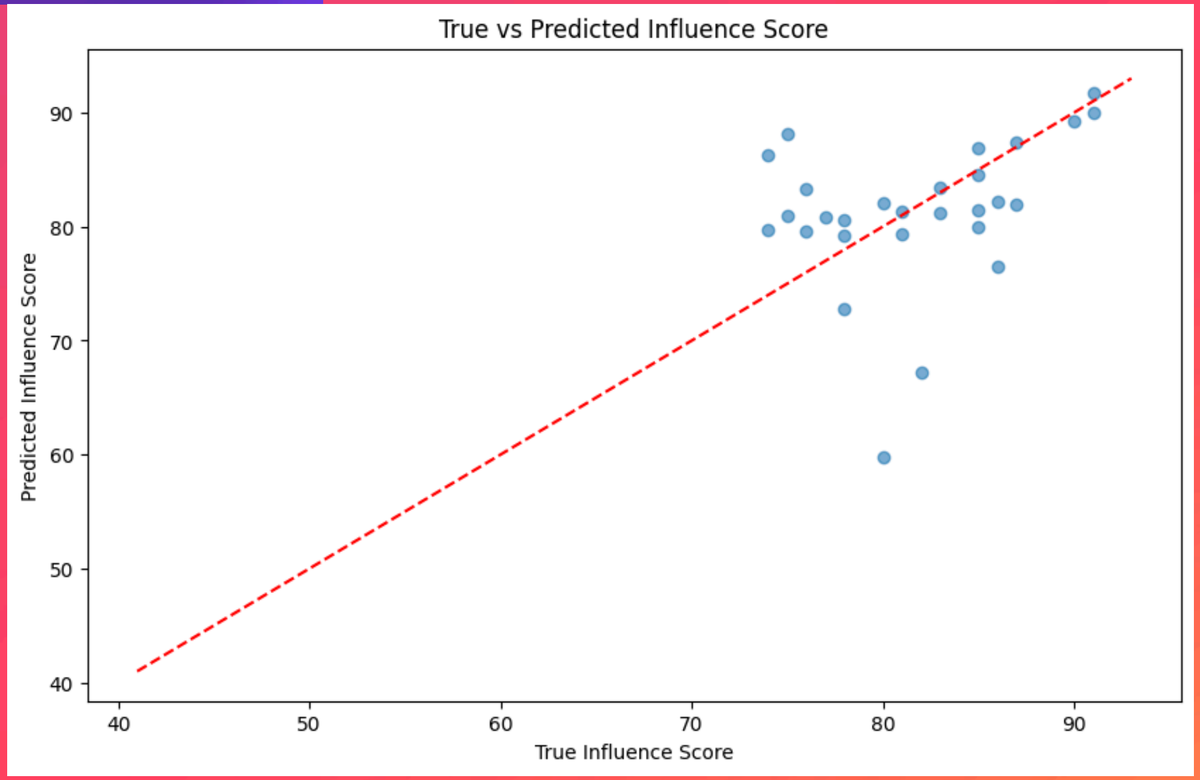
```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

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

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

Random Forest Regressor

```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}")

# Output
# Mean Squared Error: 47.18427499999999
# R^2 Score: -0.821661082385384
```

Visualizing Predictions

```
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()
```



Final Observations and Summary

Top Influential Factors: Feature importance analysis indicates which features most influence the influence_score.

Model Performance: With the achieved R^2 score, assess the accuracy of the modelin 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.