**Code Requried:**

import instaloader

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

# Function to fetch profile details

def fetch\_profile\_details(username):

L = instaloader.Instaloader()

profile = instaloader.Profile.from\_username(L.context, username)

profile\_details = {

"username": profile.username,

"full\_name": profile.full\_name,

"biography": profile.biography,

"external\_url": profile.external\_url,

"followers\_count": profile.followers,

"following\_count": profile.followees,

"posts\_count": profile.mediacount,

}

return profile\_details

# List of public Instagram usernames

usernames = [

'natgeo', 'nasa', 'therock', 'beyonce',

'virat.kohli', 'cristiano', 'kimkardashian',

'leomessi', 'taylorswift', 'justinbieber',

'arianagrande', 'badgalriri', 'katyperry',

'shakira', 'oprah', 'jlo'

]

# Initialize an empty list to store profile details

profiles\_data = []

# Fetch details for each username

for username in usernames:

try:

profile\_details = fetch\_profile\_details(username)

profiles\_data.append(profile\_details)

except instaloader.exceptions.ProfileNotExistsException:

print(f"Profile {username} does not exist.")

except Exception as e:

print(f"An error occurred while fetching data for {username}: {e}")

# Convert the list of dictionaries to a pandas DataFrame

profiles\_df = pd.DataFrame(profiles\_data)

# Save the DataFrame to a CSV file

profiles\_df.to\_csv('public\_instagram\_profiles\_details.csv', index=False)

print("Profile data saved to public\_instagram\_profiles\_details.csv")

profiles\_df

# Display the first few rows of the dataframe

print(profiles\_df.head())

# Visualize the Data

# Set the aesthetic style of the plots

sns.set\_style("whitegrid")

# Bar plot of followers count

plt.figure(figsize=(10, 6))

sns.barplot(x='followers\_count', y='username', data=profiles\_df.sort\_values(by='followers\_count', ascending=False))

plt.title('Followers Count of Public Instagram Accounts')

plt.xlabel('Followers Count')

plt.ylabel('Username')

plt.show()

# Bar plot of following count

plt.figure(figsize=(10, 6))

sns.barplot(x='following\_count', y='username', data=profiles\_df.sort\_values(by='following\_count', ascending=False))

plt.title('Following Count of Public Instagram Accounts')

plt.xlabel('Following Count')

plt.ylabel('Username')

plt.show()

# Scatter plot of followers vs. following

plt.figure(figsize=(10, 6))

sns.scatterplot(x='following\_count', y='followers\_count', data=profiles\_df, hue='username', s=100)

plt.title('Followers vs. Following Count')

plt.xlabel('Following Count')

plt.ylabel('Followers Count')

plt.legend(bbox\_to\_anchor=(1.05, 1), loc=2)

plt.show()

# Analyze the Data

# Descriptive statistics

print(profiles\_df.describe())

# Correlation analysis

correlation\_matrix = profiles\_df[['followers\_count', 'following\_count', 'posts\_count']].corr()

# Plot the heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

plt.title('Correlation Matrix')

plt.show()

# Simple Machine Learning Model

# Prepare the Data

X = profiles\_df[['following\_count', 'posts\_count']]

y = profiles\_df['followers\_count']

# 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)

# Train the Model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Evaluate the Model

y\_pred = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Absolute Error: {mae}')

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Visualize the Model's Predictions

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.7)

plt.title('True vs. Predicted Followers Count')

plt.xlabel('True Followers Count')

plt.ylabel('Predicted Followers Count')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.show()