Influencers Data Analysis

Internship Project Report

1. Introduction

This project analyzes social media influencers data to understand popularity, engagement, and performance metrics such as followers, likes, posts, and influence score.

2. Objective

• Analyze influencer ranking and engagement
• Study relationship between followers and likes
• Predict influencer influence score using machine learning

3. Dataset Description

Dataset file name: **Influencers Data.csv**. The dataset contains 200 records with influencer-related metrics.

4. Tools Used

Python, Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, VS Code

5. Data Preprocessing

Missing values were removed, textual numeric values were converted to numeric format, and relevant features were selected for modeling.

6. Exploratory Data Analysis

EDA included followers distribution analysis, likes versus followers scatter plots, and engagement rate visualization.

7. Machine Learning Model

A Linear Regression model was used to predict influencer influence score.

8. Results & Evaluation

Model performance was evaluated using Mean Squared Error (MSE) and R^2 Score.

9. Conclusion

The analysis provides insights into how followers, posting behavior, and average likes affect influencer influence score.

10. Code Appendix (Python Implementation)

```
# Import Libraries
import pandas as pd
import numpy as np
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_squared_error, r2_score

# Load dataset
df = pd.read_csv("Influencers Data.csv")

# Missing Values
print(df.isnull().sum())
df = df.dropna()

# Data Cleaning And Preprocessing
def convert_to_number(value):
    if isinstance(value, str):
        value = value.replace(',', '')
        if 'k' in value:
            return float(value.replace('k','')) * 1_000
        elif 'm' in value:
            return float(value.replace('m','')) * 1_000_000
        elif 'b' in value:
            return float(value.replace('b','')) * 1_000_000_000
    return value

df['followers'] = df['followers'].apply(convert_to_number)
df['avg_likes'] = df['avg_likes'].apply(convert_to_number)
df['posts'] = df['posts'].apply(convert_to_number)

# Followers Distribution
plt.figure(figsize=(6,4))
sns.histplot(df['followers'], bins=30, kde=True)
plt.title("Followers Distribution")
plt.show()

# Likes vs Followers
plt.figure(figsize=(6,4))
sns.scatterplot(x='followers', y='avg_likes', data=df)
plt.title("Likes vs Followers")
plt.show()

# Engagement Rate Analysis
plt.figure(figsize=(6,4))
sns.boxplot(y='60_day_eng_rate', data=df)
plt.title("Engagement Rate Distribution")
plt.show()

# Features and target
X = df[['followers', 'posts', 'avg_likes']]
y = df['influence_score']

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

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Evaluation
print("MSE:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))

# Actual vs Predicted Plot
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Influence Score")
```

```
plt.ylabel("Predicted Influence Score")
plt.title("Actual vs Predicted Influence Score")
plt.show()
```