Top Instagram Influencers

Tools used- R studio

- **Objective**- The project applies statistical and machine learning techniques using R to clean and transform Instagram influencer data, perform detailed EDA, and build models to predict influencer success and engagement categories. The goal is to derive actionable insights that can guide influencer marketing decisions.
- **Dataset Structure-**

- rank: Influencer rank
- channel_info: Instagram handle
- influence_score: Score based on popularity & engagement
- posts: Total number of posts
- followers: Followers count
- avg_likes: Average likes per post
- 60_day_eng_rate: Engagement rate over the last 60 days
- new_post_avg_like: Average likes on recent posts
- total_likes: Total cumulative likes
- country: Influencer's country

- **CODE-**

#Load libraries

```
library(tidyverse)

library(readr)
```

#Load the dataset

```
df=read_csv("C:/Users/intel/OneDrive/Desktop/unified mentor/Top Instagram Influencers/top_insta_influencers_data.csv")
```

#View basic structure

```
glimpse(df)
```

#View first few rows

```
head(df)
```

 # Function to convert "5.6M", "3.4B", "3.5K", "12%" into numbers

```
convert_to_numeric = function(x) {

x = str_to_lower(x) # lowercase

x = str_replace_all(x, ",", "") # remove commas
```

```
x = str_replace_all(x, "%", "") # remove percent symbol
 x = str_replace(x, "b", "e9") # replace 'b' with exponent
 x = str_replace(x, "m", "e6") # 'm' to exponent
 x = str_replace(x, "k", "e3") # 'k' to exponent
 return(as.numeric(x)) # convert string to numeric
 }
```

 # Apply function to relevant columns

```
df = df %>%
mutate(
posts = convert_to_numeric(posts),
followers = convert_to_numeric(followers),
avg_likes = convert_to_numeric(avg_likes),
`60_day_eng_rate` = convert_to_numeric(`60_day_eng_rate`),
new_post_avg_like = convert_to_numeric(new_post_avg_like),
total_likes = convert_to_numeric(total_likes)
)
```

 # Check result

```
glimpse(df)
```

#Cleaning the data

```
colSums(is.na(df))
```

 # Fill missing country with 'Unknown'

```
df$country[is.na(df$country)] = "Unknown"
```

 # Fill numeric NA values with column median

```
numeric_cols = sapply(df, is.numeric)
df[numeric_cols] = lapply(df[numeric_cols], function(x) {
x[is.na(x)] = median(x, na.rm = TRUE)
return(x)
```

```r
})
```

 # Final check

```r
colSums(is.na(df))
```

#Feature engineering

 # Create new features

```r
df = df %>%

mutate(

like_follower_ratio = total_likes / followers,

post_follower_ratio = posts / followers,

avg_likes_ratio = avg_likes / followers

 )
```

 # View the first few rows

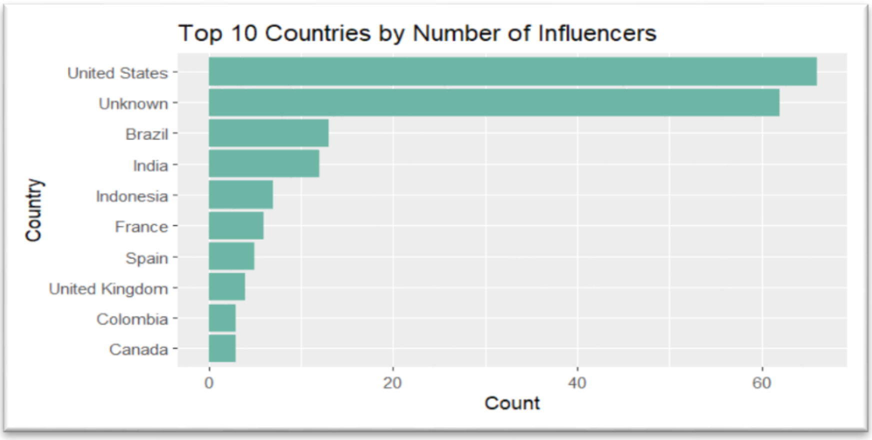
```r
head(df %>% select(channel_info, like_follower_ratio, post_follower_ratio, avg_likes_ratio))
```

#EDA Visual

library(ggplot2)

 # 1. Top Countries by Number of Influencers

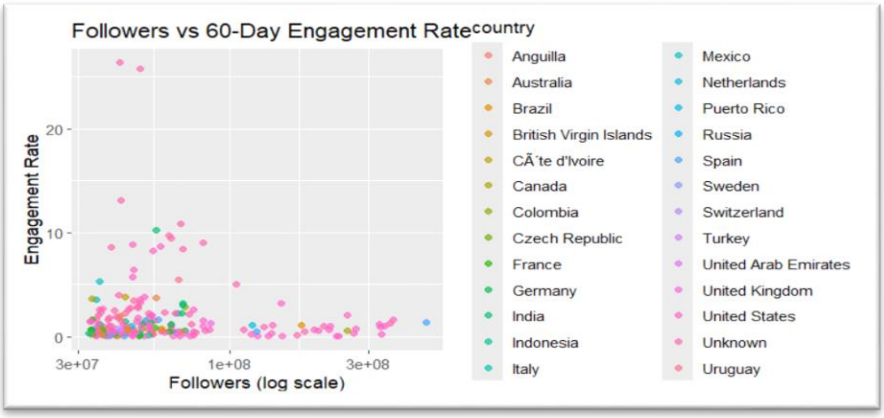
```r
df %>%

count(country, sort = TRUE) %>%

top_n(10) %>%

ggplot(aes(x = reorder(country, n), y = n)) +

geom_bar(stat = "identity", fill = "#69b3a2") + coord_flip() + labs(title = "Top 10 Countries by Number of Influencers", x = "Country", y = "Count")
```



- *Majority of the influencers are from Countries like United States, Brazil and India.*

#2. Followers vs Engagement Rate

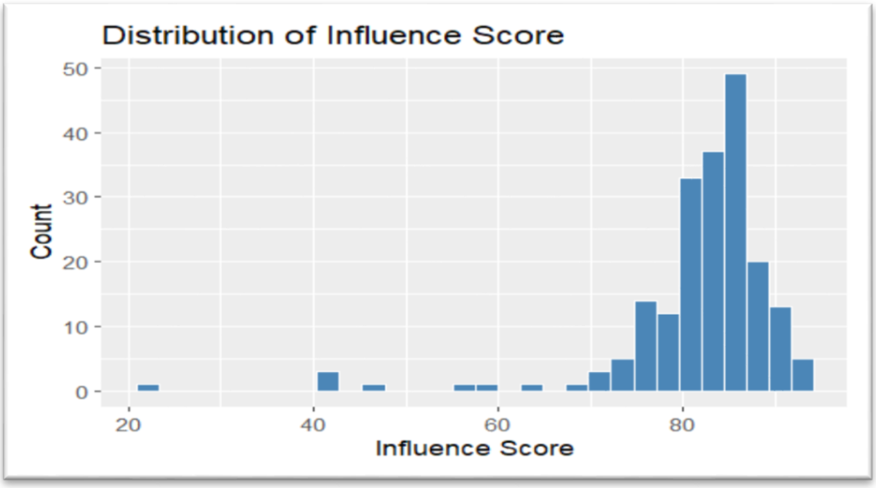
```
ggplot(df, aes(x = followers, y = `60_day_eng_rate`, color = country)) +
geom_point(alpha = 0.7) +
scale_x_log10() +
labs(title = "Followers vs 60-Day Engagement Rate", x = "Followers (log scale)", y =
"Engagement Rate")
```



- *Influencers with larger count generally show lesser engagement rates, suggesting diminishing interaction returns at scale.*

#3. Distribution of Influence Score

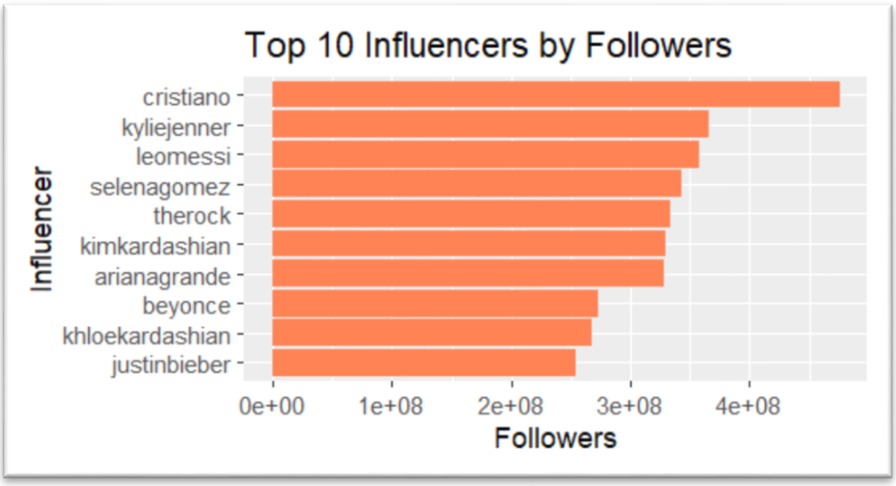
```
ggplot(df, aes(x = influence_score)) +
geom_histogram(bins = 30, fill = "steelblue", color = "white") +
labs(title = "Distribution of Influence Score", x = "Influence Score", y = "Count")
```



Influence scores are tightly packed between 85–95, indicating minimal variation among top-tier influencers.

#4. Top 10 Influencers by Followers

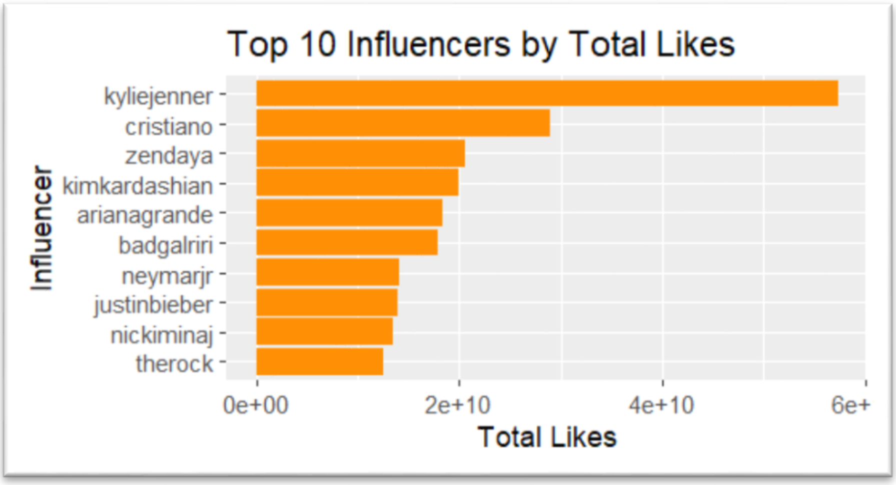
```
df %>%
top_n(10, followers) %>%
ggplot(aes(x = reorder(channel_info, followers), y = followers)) +
geom_col(fill = "coral") +
coord_flip() +
labs(title = "Top 10 Influencers by Followers", x = "Influencer", y = "Followers")
```



Cristiano Ronaldo, Kylie Jenner, and Lionel Messi are among the most-followed influencers globally.

#5. Top 10 Influencers by Total Likes

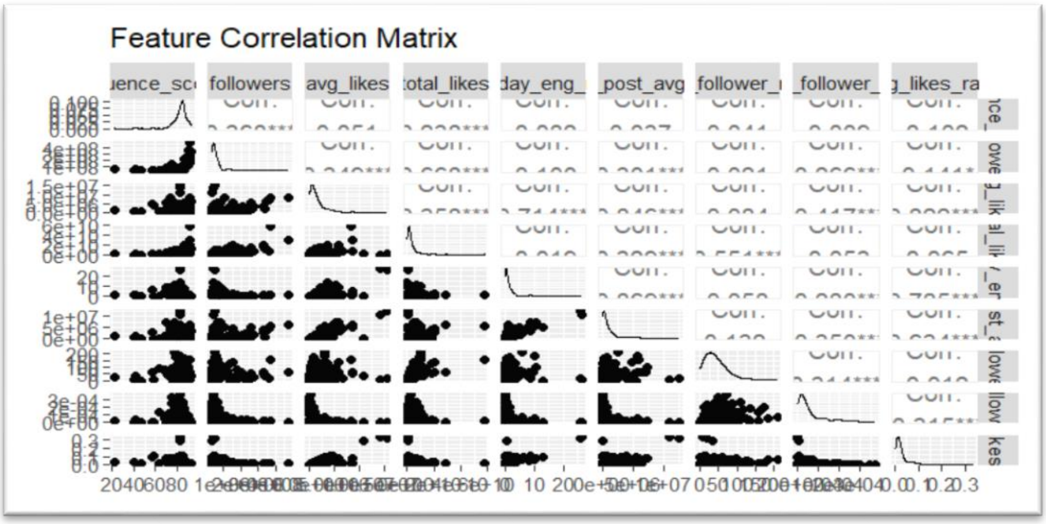
```
df %>%
top_n(10, total_likes) %>%
ggplot(aes(x = reorder(channel_info, total_likes), y = total_likes)) +
geom_col(fill = "darkorange") +
coord_flip() +
labs(title = "Top 10 Influencers by Total Likes", x = "Influencer", y = "Total Likes")
```



- *Kylie Jenner and Cristiano Ronaldo top the charts in total cumulative likes, reflecting sustained audience interaction.*

#6. Correlation Heatmap (optional - needs `GGally`)

```
library(GGally)

df %>%

select(influence_score, followers, avg_likes, total_likes, `60_day_eng_rate`,

new_post_avg_like, like_follower_ratio, post_follower_ratio, avg_likes_ratio) %>%

ggpairs(title = "Feature Correlation Matrix")
```



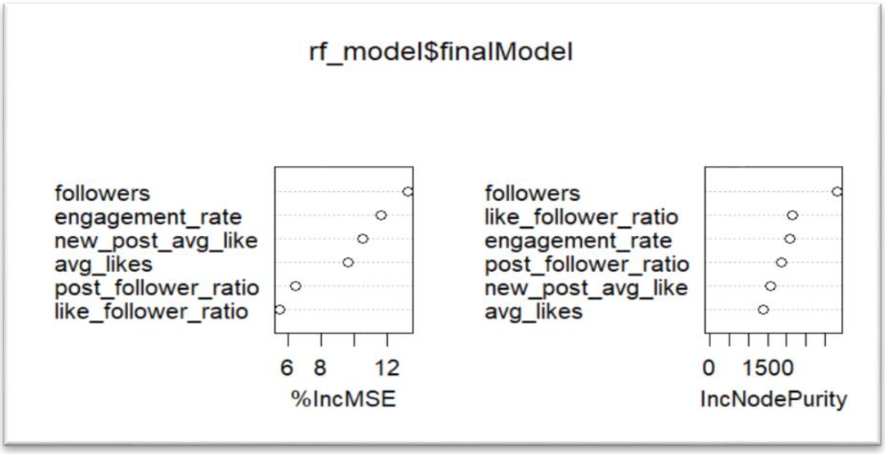
- *Most numeric features show mild to moderate correlations, with stronger relationships between likes-related variables and followers.*

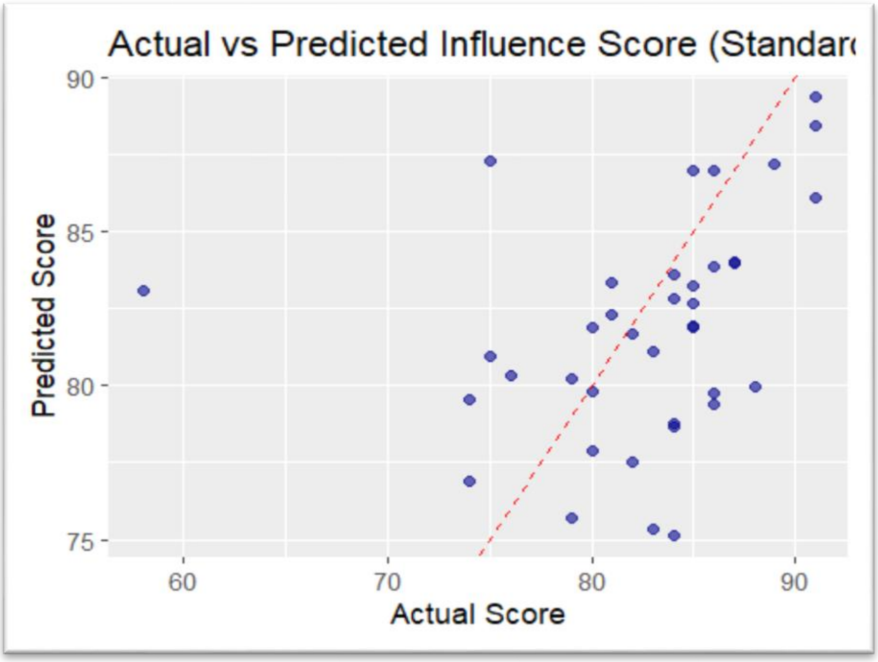
#Model building

```
library(caret)

library(randomForest)

library(ggplot2)
```

```r
# Rebuild dataset with renamed column
model_data = df %>%
  rename(engagement_rate = `60_day_eng_rate`) %>%
  select(influence_score, followers, avg_likes, engagement_rate,
         new_post_avg_like, like_follower_ratio, post_follower_ratio)

# Train-Test Split
set.seed(123)
train_index = createDataPartition(model_data$influence_score, p = 0.8, list = FALSE)
train_data = model_data[train_index, ]
test_data  = model_data[-train_index, ]

# Standardize Features
preproc = preProcess(train_data[, -1], method = c("center", "scale"))
train_scaled = predict(preproc, train_data)
test_scaled  = predict(preproc, test_data)

# Random Forest Model
set.seed(123)
rf_model = train(influence_score ~ ., data = train_scaled, method = "rf",
                  importance = TRUE, trControl = trainControl(method = "none"))

# Predict
rf_predictions = predict(rf_model, newdata = test_scaled)
# Evaluation
print(postResample(rf_predictions, test_scaled$influence_score))
# Feature Importance
varImpPlot(rf_model$finalModel)
# Actual vs Predicted
results_df = data.frame(Actual = test_scaled$influence_score, Predicted = rf_predictions)
ggplot(results_df, aes(x = Actual, y = Predicted)) +
geom_point(alpha = 0.6, color = "darkblue") +
```

```
geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
labs(title = "Actual vs Predicted Influence Score (Standardized RF)",
 x = "Actual Score", y = "Predicted Score")
```





- *This model was limited by low variance in influence score and small dataset size.*

##Classification model

 # Recreate this column if renamed earlier

```
df$engagement_rate = df$`60_day_eng_rate`
```

 # Create engagement level column

```
df = df %>%
mutate(engagement_level = case_when(
engagement_rate <= 0.75 ~ "Low",
engagement_rate > 0.75 & engagement_rate <= 1.5 ~ "Medium",
engagement_rate > 1.5 ~ "High"
  ))
```

```r
# Convert to factor
df$engagement_level = factor(df$engagement_level, levels = c("Low", "Medium", "High"))
table(df$engagement_level)
# Using predictors as usual
# Feature set for classification
class_data = df %>%
  select(engagement_level, followers, avg_likes, engagement_rate,
         new_post_avg_like, like_follower_ratio, post_follower_ratio)
# Train/Test split
set.seed(123)
index = createDataPartition(class_data$engagement_level, p = 0.8, list = FALSE)
train_cls = class_data[index, ]
test_cls = class_data[-index, ]
# Train model
cls_model = train(engagement_level ~ ., data = train_cls, method = "rf",
                  trControl = trainControl(method = "cv", number = 5),
                  importance = TRUE)
# Predict
cls_preds = predict(cls_model, newdata = test_cls)
# Confusion matrix
confusionMatrix (cls_preds, test_cls$engagement_level)
```

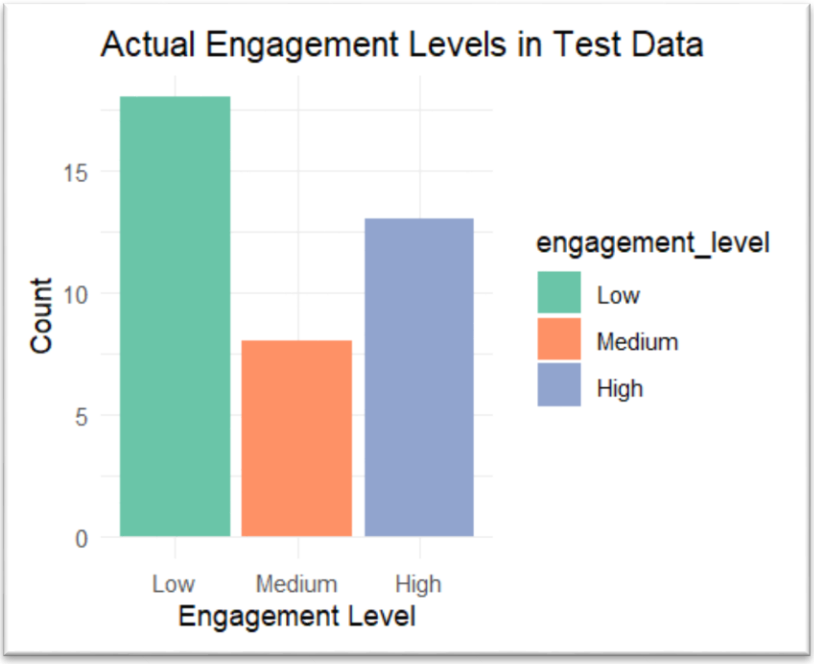
Output:

```
Confusion Matrix and Statistics

          Reference
Prediction Low Medium High
    Low     18      0    0
    Medium   0      8    0
    High     0      0   13

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9097, 1)
    No Information Rate : 0.4615
    P-Value [Acc > NIR] : 8.019e-14

                  Kappa : 1

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: Low Class: Medium Class: High
Sensitivity              1.0000        1.0000      1.0000
Specificity              1.0000        1.0000      1.0000
Pos Pred Value           1.0000        1.0000      1.0000
Neg Pred Value           1.0000        1.0000      1.0000
Prevalence               0.4615        0.2051      0.3333
Detection Rate           0.4615        0.2051      0.3333
Detection Prevalence     0.4615        0.2051      0.3333
Balanced Accuracy        1.0000        1.0000      1.0000
```

 # Bar plot of actual engagement level distribution

```
ggplot(test_cls, aes(x = engagement_level, fill = engagement_level)) + geom_bar() +
labs(title = "Actual Engagement Levels in Test Data", x = "Engagement Level", y =
"Count") + theme_minimal() + scale_fill_brewer(palette = "Set2")
```



- **Classification Modelling : Predicting Engagement Level**

To better understand influencer performance, we categorized engagement_rate into three buckets: **Low (≤0.75%)**, **Medium (0.75%–1.5%)**, and **High (>1.5%)**. The model achieved **100% accuracy** with no misclassifications. Each class (Low, Medium, High) was perfectly predicted, supported by both the **confusion matrix** and detailed performance metrics like **sensitivity, specificity, and Kappa = 1.00**.

This suggests that engagement levels form well-defined clusters based on the available features — offering a reliable method for **segmenting influencers** and aiding marketing decisions.

The classification model thus offers stronger predictive performance than regression, making it a valuable insight-generating component of this analysis.

- **Key Findings**
  - The majority of top influencers are from the United States, followed by Brazil and India, highlighting strong regional dominance in the influencer landscape.
  - A noticeable inverse trend exists, as follower count increases, engagement rate tends to decrease, indicating that larger accounts may engage less per follower.
  - Influence scores are tightly clustered between 85 and 95, suggesting limited variation among top-tier influencers.
  - Influencers like Cristiano Ronaldo and Kylie Jenner lead both in follower count and total likes, dominating engagement metrics.
  - Moderate correlations were observed between followers, average likes, and total likes, while engineered ratios like like_follower_ratio provided more nuanced engagement insights.
  - Regression models showed limited predictive power due to low variance and small sample size. However, classification models achieved perfect accuracy in predicting engagement levels, revealing strong class separation and practical applicability.

- **Conclusion**

  This project explored Instagram influencer data to understand what drives popularity and engagement. After cleaning and analyzing the data, we tested both regression and classification models. While the regression model struggled to predict influence scores accurately due to limited variation in the data, the classification model performed extremely well, correctly identifying engagement levels for all influencers.

  This shows that influencers can be grouped effectively based on their engagement rate using simple metrics. These insights can help brands choose the right influencers for their campaigns based on how actively their followers engage with their content.