Load the Data

In [7]:
```
import pandas as pd
import zipfile
import os

# Function to load CSV from a ZIP file with multiple files
def load_csv_from_zip(zip_path, csv_filename):
    with zipfile.ZipFile(zip_path, 'r') as z:
        # Extract and read the specific CSV file
        with z.open(csv_filename) as f:
            return pd.read_csv(f)

# Define the relative path to the datasets folder
datasets_path = os.path.join('..', 'Datasets')

# Load datasets from zipped CSV files specifying the correct CSV filenames
df_gb = load_csv_from_zip(os.path.join(datasets_path, 'GBvideos.csv.zip'), 'GBvideos.csv')
df_us = load_csv_from_zip(os.path.join(datasets_path, 'USvideos.csv.zip'), 'USvideos.csv')

# Add a new column 'location' in each data file
df_gb['location'] = 'Great Britain'
df_us['location'] = 'USA'

# Merge 5 files into 1
merged_df = pd.concat([df_gb, df_us], ignore_index=True)

# Check the first few rows of the merged DataFrame
print(merged_df.head())
```

```
   dislikes  comment_count                                     thumbnail_link  \
0     10247           9479  https://i.ytimg.com/vi/Jw1Y-zhQURU/default.jpg (https://i.ytimg.com/vi/Jw1Y-zhQURU/default.jpg)
1      2294           2757  https://i.ytimg.com/vi/3s1rvMFUweQ/default.jpg (https://i.ytimg.com/vi/3s1rvMFUweQ/default.jpg)
2     43420         125882  https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg (https://i.ytimg.com/vi/n1WpP7iowLc/default.jpg)
3        12             37  https://i.ytimg.com/vi/PUTEiSjKwJU/default.jpg (https://i.ytimg.com/vi/PUTEiSjKwJU/default.jpg)
4         2             30  https://i.ytimg.com/vi/rHwDegptbI4/default.jpg (https://i.ytimg.com/vi/rHwDegptbI4/default.jpg)

   comments_disabled  ratings_disabled  video_error_or_removed  \
0              False             False                   False
1              False             False                   False
2              False             False                   False
3              False             False                   False
4              False             False                   False

                                         description       location
0  Click here to continue the story and make your...  Great Britain
```

Check Missing Values

In [8]:
```
# Check for missing values in the merged DataFrame
print("Missing values")
print(merged_df.isnull().sum())
```

```
Missing values
video_id                    0
trending_date               0
title                       0
channel_title               0
category_id                 0
publish_time                0
tags                        0
views                       0
likes                       0
dislikes                    0
comment_count               0
thumbnail_link              0
comments_disabled           0
ratings_disabled            0
video_error_or_removed      0
description              1182
location                    0
dtype: int64
```

In [9]:
```
df = merged_df.dropna()
```

In [10]:
```
# Check for missing values in the merged DataFrame
print("Missing values")
print(df.isnull().sum())
```

```
Missing values
video_id                  0
trending_date             0
title                     0
channel_title             0
category_id               0
publish_time              0
tags                      0
views                     0
likes                     0
dislikes                  0
comment_count             0
thumbnail_link            0
comments_disabled         0
ratings_disabled          0
video_error_or_removed    0
description               0
location                  0
dtype: int64
```

Drop Unnecessary Columns

In [11]:
```
#drop columns needed
merged_df.drop(columns=['thumbnail_link', 'video_id','comments_disabled','ratings_disabled','video_error_or_re

print(merged_df.head())
```

```
  trending_date                                              title  \
0      17.14.11      John Lewis Christmas Ad 2017 - #MozTheMonster
1      17.14.11          Taylor Swift: …Ready for It? (Live) - SNL
2      17.14.11         Eminem - Walk On Water (Audio) ft. Beyoncé
3      17.14.11  Goals from Salford City vs Class of 92 and Fri...
4      17.14.11  Dashcam captures truck's near miss with child ...

                channel_title  category_id              publish_time  \
0                  John Lewis           26  2017-11-10T07:38:29.000Z
1         Saturday Night Live           24  2017-11-12T06:24:44.000Z
2                  EminemVEVO           10  2017-11-10T17:00:03.000Z
3  Salford City Football Club           17  2017-11-13T02:30:38.000Z
4            Cute Girl Videos           25  2017-11-13T01:45:13.000Z

                                                tags     views   likes  \
0  christmas|"john lewis christmas"|"john lewis"|...   7224515   55681
1  SNL|"Saturday Night Live"|"SNL Season 43"|"Epi...   1053632   25561
2  Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In...  17158579  787420
3  Salford City FC|"Salford City"|"Salford"|"Clas...     27833     193
4                                            [none]      0815      20
```

Text Preprocessing

In [12]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import re

# Get the list of default English stopwords
stop_words = set(stopwords.words('english'))

# Function to remove stopwords and clean text
def clean_text(text):
    # Lowercase the text
    text = text.lower()

    # Remove non-alphabetical characters (retain only letters and spaces)
    text = re.sub(r'[^a-z\s]', '', text)

    # Split text into words
    words = text.split()

    # Remove stopwords
    remove_stopwords = [word for word in words if word not in stop_words]

    # Join the cleaned words back into a string
    new_text = ' '.join(remove_stopwords)

    return new_text
    data = {'title','description','text'}

# Apply the clean_text function to the 'title' column in merged_df1
merged_df['new_text'] = merged_df['title'].apply(clean_text)

# Display the cleaned DataFrame
print(merged_df)
```

```
                                          description       location  \
0      Click here to continue the story and make your...  Great Britain
1      Musical guest Taylor Swift performs …Ready for...  Great Britain
2      Eminem's new track Walk on Water ft. Beyoncé i...  Great Britain
3      Salford drew 4-4 against the Class of 92 and F...  Great Britain
4      Dashcam captures truck's near miss with child ...  Great Britain
...                                                  ...            ...
79860     The Cat Who Caught the Laser - Aaron's Animals            USA
79861                                                NaN            USA
79862  I had so much fun transforming Safiyas hair in...            USA
79863  How Black Panther Should Have EndedWatch More ...            USA
79864  Call of Duty: Black Ops 4 Multiplayer raises t...            USA

                                          new_text
0                  john lewis christmas ad mozthemonster
1                           taylor swift ready live snl
2                      eminem walk water audio ft beyonc
3      goals salford city vs class friends peninsula ...
4         dashcam captures trucks near miss child norway
```

In [13]:
```
# Check the data types of each column
print(merged_df.dtypes)
```

```
trending_date    object
title            object
channel_title    object
category_id       int64
publish_time     object
tags             object
views             int64
likes             int64
dislikes          int64
comment_count     int64
description      object
location         object
new_text         object
dtype: object
```

Split the Dataset into Train and Test by 80/20

In [14]:
```
from sklearn.model_selection import train_test_split

X = merged_df.drop(columns=['views'])  # Drop 'views' from features to get X
y = merged_df['views']
# Assuming you have a dataset with features X and target y
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42)

train = pd.DataFrame(X_train)
train['views'] = y_train.values

test = pd.DataFrame(X_test)
test['views'] = y_test.values
```

(New) Feature Engineering

- Remove irrelavent features(Time-Based features, `Days Since Published`)
- Create `Basic Engagement Ratio Analysis`
- Create `Time Based Metrics Analysis`

Basic Engagement Ratio Analysis

In [15]:
```python
import pandas as pd

# Convert trending_date and publish_time to datetime
# Convert trending_date and publish_time to timezone-naive datetime
def prepare_datetime_columns(train):
    train['trending_date'] = pd.to_datetime(train['trending_date'], format='%y.%d.%m', errors='coerce')
    train['publish_time'] = pd.to_datetime(train['publish_time'], errors='coerce')

    # Remove timezone information to make them timezone-naive
    train['trending_date'] = train['trending_date'].dt.tz_localize(None)
    train['publish_time'] = train['publish_time'].dt.tz_localize(None)

    return train


# Basic Engagement Ratios
def create_engagement_ratios(train):
    epsilon = 1e-10  # Prevents division by zero

    train['like_view_ratio'] = train['likes'] / (train['views'] + epsilon)
    train['comment_like_ratio'] = train['comment_count'] / (train['likes'] + epsilon)
    train['dislike_view_ratio'] = train['dislikes'] / (train['views'] + epsilon)
    train['comment_view_ratio'] = train['comment_count'] / (train['views'] + epsilon)
    train['total_engagement_ratio'] = (train['likes'] + train['dislikes'] + train['comment_count']) / (train['
    train['like_dislike_ratio'] = train['likes'] / (train['dislikes'] + epsilon)

    # Normalized Engagement Scores
    train['normalized_likes'] = (train['likes'] - train['likes'].mean()) / train['likes'].std()
    train['normalized_views'] = (train['views'] - train['views'].mean()) / train['views'].std()

    # Category-specific engagement ratios
    train['category_like_view_ratio'] = train.groupby('category_id')['like_view_ratio'].transform('mean')
    train['relative_category_engagement'] = train['like_view_ratio'] / (train['category_like_view_ratio'] + ep

    # Engagement rate percentiles
    train['like_view_percentile'] = train['like_view_ratio'].rank(pct=True)
    train['comment_like_percentile'] = train['comment_like_ratio'].rank(pct=True)

    return train

# Create engagement level categories
def create_engagement_categories(train):
    train['like_view_category'] = pd.qcut(train['like_view_ratio'], q=5, labels=['Very Low', 'Low', 'Medium',
    train['comment_like_category'] = pd.qcut(train['comment_like_ratio'], q=5, labels=['Very Low', 'Low', 'Med

    # Combined Engagement Score
    train['engagement_score'] = (train['like_view_percentile'] + train['comment_like_percentile']) / 2
    train['engagement_category'] = pd.qcut(train['engagement_score'], q=5, labels=['Very Low', 'Low', 'Medium'

    return train

# Time-based engagement metrics
def create_time_based_engagement(train):
    epsilon = 1e-10
    train['hours_to_trend'] = (train['trending_date'] - train['publish_time']).dt.total_seconds() / 3600

    return train


# Putting it all together
def create_all_engagement_features(train):
    train = prepare_datetime_columns(train)
    train = create_engagement_ratios(train)
    train = create_engagement_categories(train)
    train = create_time_based_engagement(train)
    return train
```

In [16]:
```
train = create_all_engagement_features(train)
print(train)
```

```
      trending_date                                              title  \
23604    2018-03-14                  Marshmello & Anne-Marie: Friends
25630    2018-03-24  Kirby Star Allies' Surprising HD Rumble Secret...
68698    2018-04-20  Stephen A.: Kevin Hart 'got his feelings hurt'...
39559    2017-11-17                            How to be an Aquarius
62877    2018-03-16  Charlie Puth - Done For Me (feat. Kehlani) [Of...
...             ...                                                ...
6265     2017-12-15         Adrienne’s Full Performance of 'The Gift'
54886    2018-02-03  THIS BRA MAKES YOU STRONGER! (Weird As Seen on...
76820    2018-05-30  Why You Should Wake Up at 4:30 AM Every Day, A...
860      2017-11-18  NBA Countdown debates if Ben Simmons is the be...
15795    2018-02-03    Sam Smith Will Only Do Karaoke to Fifth Harmony

                             channel_title  category_id  \
23604  The Tonight Show Starring Jimmy Fallon           23
25630                            GameXplain           20
68698                                  ESPN           17
39559                              Sailor J           24
62877                           Charlie Puth           10
```

Time-Based Metrics Analysis

In [17]:
```
train['publish_weekday'] = train['publish_time'].dt.dayofweek
train['is_weekend'] = train['publish_weekday'].isin([5,6]).astype(int)

print(train)
```

```
6265   Adrienne brings the house down performing a si...  ...
54886  Here are some weird As Seen on TV Products tha...  ...
76820  With a busy schedule, Jocko Willink finds time...  ...
860    The NBA Countdown crew debates if Philadelphia...  ...
15795  Sam Smith chats about a wild night in Sydney, ...  ...

      relative_category_engagement like_view_percentile  \
23604                     0.756425             0.562042
25630                     0.750198             0.462092
68698                     0.440029             0.091529
39559                     1.994679             0.853166
62877                     2.832422             0.982017
...                            ...                  ...
6265                      1.752947             0.801822
54886                     0.635839             0.327694
76820                     0.444818             0.351734
860                       0.551411             0.127739
15795                     0.762011             0.565697

       comment_like_percentile  like_view_category  comment_like_category  \
```

TF-IDF Feature

description Column

In [18]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# Ensure the 'description' column exists in the DataFrame
if 'description' in train.columns:
    # Assuming 'description' column contains the text data
    text_data = train['description'].fillna('')  # Handle missing values

    # Check if text_data is iterable, not a single string
    if isinstance(text_data, pd.Series):
        # Initialize the TF-IDF Vectorizer
        tfidf_vectorizer = TfidfVectorizer(max_features=100, stop_words='english')

        # Fit and transform the text data to generate the TF-IDF matrix
        tfidf_matrix = tfidf_vectorizer.fit_transform(text_data)

        # Convert the sparse matrix into a DataFrame for easier manipulation
        tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

        # Function to get top N features per row based on TF-IDF score
        def get_top_tfidf_features(row, features, top_n=5):
            top_indices = np.argsort(row)[::-1][:top_n]  # Get the indices of the top n features
            top_features = [(features[i], row[i]) for i in top_indices]  # Get feature names and scores
            return top_features

        # Apply the function to each row in the TF-IDF matrix
        top_tfidf_features = [get_top_tfidf_features(row, tfidf_vectorizer.get_feature_names_out(), top_n=5)
                              for row in tfidf_matrix.toarray()]

        # Add the top TF-IDF features as a new column in the original DataFrame
        train['top_tfidf_features'] = top_tfidf_features

        # Display the entire first 5 rows of the DataFrame including the top TF-IDF features
        print(train.head(5))
    else:
        print("The 'description' column should be a pandas Series.")
else:
    print("The DataFrame does not contain a 'description' column.")
```

```
68698              Very High          0.451801                 Low
39559                   High          0.818240           Very High
62877                 Medium          0.761449           Very High

       hours_to_trend  publish_weekday  is_weekend  \
23604      153.999167                2           0
25630      187.999722                4           0
68698       57.074722                1           0
39559       34.508889                2           0
62877        7.961944                3           0

                                  top_tfidf_features
23604  [(jimmy, 0.6973441834478303), (nbc, 0.47929685...
25630  [(patreon, 0.5185912254067347), (com, 0.407894...
68698  [(http, 0.6127313163416526), (youtube, 0.33885...
39559  [(ll, 0.5094338648331312), (don, 0.47521836732...
62877  [(nhttp, 0.4915874738421168), (com, 0.44499439...

[5 rows x 33 columns]
```

tags Column

In [19]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# Ensure the 'description' column exists in the DataFrame
if 'tags' in train.columns:
    # Assuming 'description' column contains the text data
    text_data = train['tags'].fillna('')  # Handle missing values

    # Check if text_data is iterable, not a single string
    if isinstance(text_data, pd.Series):
        # Initialize the TF-IDF Vectorizer
        tfidf_vectorizer = TfidfVectorizer(max_features=100, stop_words='english')

        # Fit and transform the text data to generate the TF-IDF matrix
        tfidf_matrix = tfidf_vectorizer.fit_transform(text_data)

        # Convert the sparse matrix into a DataFrame for easier manipulation
        tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

        # Function to get top N features per row based on TF-IDF score
        def get_top_tfidf_features(row, features, top_n=5):
            top_indices = np.argsort(row)[::-1][:top_n]  # Get the indices of the top n features
            top_features = [(features[i], row[i]) for i in top_indices]  # Get feature names and scores
            return top_features

        # Apply the function to each row in the TF-IDF matrix
        top_tfidf_features = [get_top_tfidf_features(row, tfidf_vectorizer.get_feature_names_out(), top_n=5)
                              for row in tfidf_matrix.toarray()]

        # Add the top TF-IDF features as a new column in the original DataFrame
        train['top_tfidf_features'] = top_tfidf_features

        # Display the entire first 5 rows of the DataFrame including the top TF-IDF features
        print(train.head(5))
    else:
        print("The 'description' column should be a pandas Series.")
else:
    print("The DataFrame does not contain a 'description' column.")
```

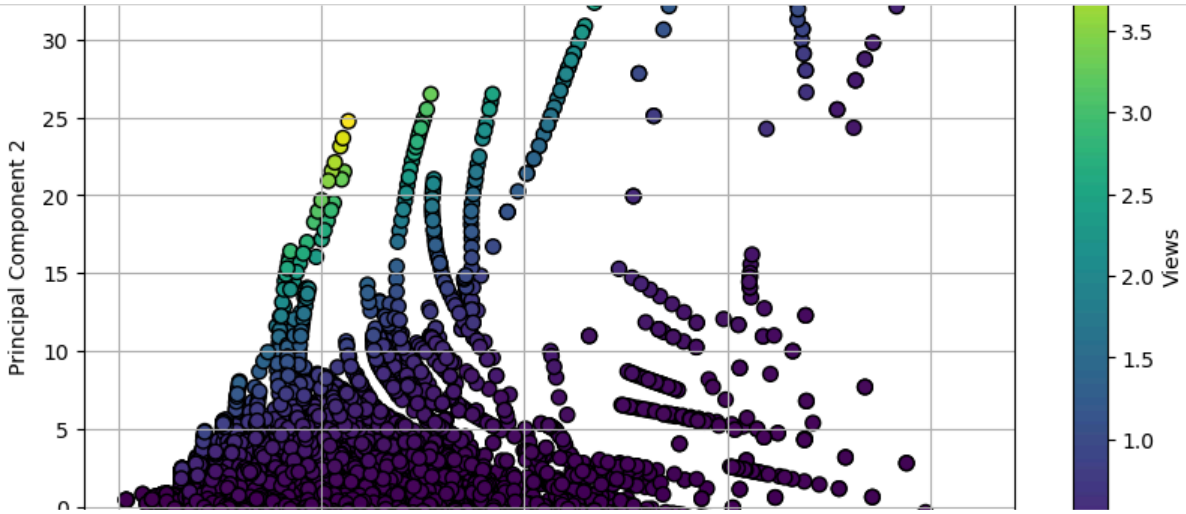
```
      trending_date                                              title  \
23604    2018-03-14                        Marshmello & Anne-Marie: Friends
25630    2018-03-24  Kirby Star Allies' Surprising HD Rumble Secret...
68698    2018-04-20  Stephen A.: Kevin Hart 'got his feelings hurt'...
39559    2017-11-17                                 How to be an Aquarius
62877    2018-03-16  Charlie Puth - Done For Me (feat. Kehlani) [Of...

                              channel_title  category_id  \
23604  The Tonight Show Starring Jimmy Fallon           23
25630                            GameXplain           20
68698                                  ESPN           17
39559                              Sailor J           24
62877                           Charlie Puth           10

             publish_time                                               tags  \
23604 2018-03-07 14:00:03  The Tonight Show|"Jimmy Fallon"|"Marshmello"|"...
25630 2018-03-16 04:00:01  Kirby|"Kirby Star Allies"|"Dedede"|"Meta Knigh...
68698 2018-04-17 14:55:31  espn|"dwyane wade"|"dwayne wade"|"d wade"|"76e...
39559 2017-11-15 13:29:28                 Zodiac|"makeup"|"comedy"|"aquarius"
```

Dimension Reduction-PCA

In [20]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

non_numeric_cols = ['publish_time', 'title', 'channel_title', 'tags', 'description', 'location', 'trending_dat
X_train_model = train.drop(columns=non_numeric_cols + ['views']).select_dtypes(include=[float, int])
X_test_model = test.drop(columns=non_numeric_cols + ['views']).select_dtypes(include=[float, int])

X_test_model = X_test_model.reindex(columns=X_train_model.columns, fill_value=0)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_model)
X_test_scaled = scaler.transform(X_test_model)

print("Missing values in X_train_model:\n", X_train_model.isna().sum())
print("Missing values in X_test_model:\n", X_test_model.isna().sum())
```

```
Missing values in X_train_model:
 category_id                      0
likes                             0
dislikes                          0
comment_count                     0
like_view_ratio                   0
comment_like_ratio                0
dislike_view_ratio                0
comment_view_ratio                0
total_engagement_ratio            0
like_dislike_ratio                0
normalized_likes                  0
normalized_views                  0
category_like_view_ratio          0
relative_category_engagement      0
like_view_percentile              0
comment_like_percentile           0
engagement_score                  0
hours_to_trend                    0
```

In [21]:
```
import matplotlib.pyplot as plt
# Apply PCA (Reduce to n components to capture 95% of variance)
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Visualize the PCA results (Plot only the first two components)
plt.figure(figsize=(10, 6))
plt.scatter(X_train_pca[:, 0], X_train_pca[:, 1], c=y_train, cmap='viridis', edgecolor='k', s=50)
plt.colorbar(label='Views')
plt.title('PCA of YouTube Data (Train Set)', weight='bold')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()

# Explained variance for all components selected by PCA
explained_variance = pca.explained_variance_ratio_
print("Explained Variance per component:")
for i, variance in enumerate(explained_variance, start=1):
    print(f"PC{i}: {variance:.2%}")
```



Model Refitting

Winning Model-XGBoost

In [22]:
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd
```

In [23]:
```
# Define a function for calculating model metrics
def calculate_metrics(model, X_train, y_train, X_test, y_test):
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)

    # Calculate RMSE and R^2 for training and test sets
    train_rmse = np.sqrt(mean_squared_error(y_train, train_preds))
    test_rmse = np.sqrt(mean_squared_error(y_test, test_preds))
    train_r2 = r2_score(y_train, train_preds)
    test_r2 = r2_score(y_test, test_preds)

    return {
        "Train RMSE": train_rmse, "Test RMSE": val_rmse,
        "Train R^2": train_r2, "Test R^2": val_r2
    }
```

In [24]:
```
# Define a function to train the model with specific hyperparameters
def train_xgboost(X_train, y_train, X_test, y_test, params):
    model = xgb.XGBRegressor(**params, random_state=42)
    model.fit(X_train, y_train)

    # Calculate and return metrics
    metrics = calculate_metrics(model, X_train, y_train, X_test, y_test)
    return model, metrics
```

In [25]:
```
# Define hyperparameter variations
variations = [
    {"learning_rate": 0.05, "n_estimators": 200, "max_depth": 6}# Winning variation
]
```

In [26]:
```
# Initialize a DataFrame to store results for each variation
results = pd.DataFrame(columns=["Variation", "Train RMSE", "Test RMSE", "Train R^2", "Test R^2"])
```

In [27]:
```
# List of columns to drop if they exist
text_columns = ['title', 'channel_title', 'tags', 'description', 'location', 'new_text']
X_train.drop([col for col in text_columns if col in X_train.columns], axis=1, inplace=True)
X_test.drop([col for col in text_columns if col in X_test.columns], axis=1, inplace=True)

# Encode categorical features using one-hot encoding for consistency
categorical_columns = ['engagement_category', 'comment_like_category', 'like-view-category']
X_train = pd.get_dummies(X_train, columns=[col for col in categorical_columns if col in X_train.columns], drop
X_test = pd.get_dummies(X_test, columns=[col for col in categorical_columns if col in X_test.columns], drop_fi

# Drop 'trending_date' and 'publish_time' if they exist
for date_col in ['trending_date', 'publish_time']:
    if date_col in X_train.columns:
        X_train.drop([date_col], axis=1, inplace=True)
    if date_col in X_test.columns:
        X_test.drop([date_col], axis=1, inplace=True)

# Add any missing features from the provided feature list, excluding already existing columns
required_features = [
    'category_id', 'likes', 'dislikes', 'comment_count', 'like_view_ratio', 'comment_like_ratio',
    'dislike_view_ratio', 'comment_view_ratio', 'total_engagement_ratio', 'like_dislike_ratio',
    'normalized_likes', 'normalized_views', 'category_like_view_ratio', 'relative_category_engagement',
    'like_view_percentile', 'comment_like_percentile', 'engagement_score', 'hours_to_trend',
    'publish_weekday', 'is_weekend'
]
# Add missing features with default values of 0
for feature in required_features:
    if feature not in X_train.columns:
        X_train[feature] = 0
    if feature not in X_test.columns:
        X_test[feature] = 0

# Ensure X_test has the same columns as X_train
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# Verify that X_train and X_test now have the same columns
print("X_train columns:", X_train.columns)
print("X_test columns:", X_test.columns)
```

```
X_train columns: Index(['category_id', 'likes', 'dislikes', 'comment_count', 'like_view_ratio',
       'comment_like_ratio', 'dislike_view_ratio', 'comment_view_ratio',
       'total_engagement_ratio', 'like_dislike_ratio', 'normalized_likes',
       'normalized_views', 'category_like_view_ratio',
       'relative_category_engagement', 'like_view_percentile',
       'comment_like_percentile', 'engagement_score', 'hours_to_trend',
       'publish_weekday', 'is_weekend'],
      dtype='object')
X_test columns: Index(['category_id', 'likes', 'dislikes', 'comment_count', 'like_view_ratio',
       'comment_like_ratio', 'dislike_view_ratio', 'comment_view_ratio',
       'total_engagement_ratio', 'like_dislike_ratio', 'normalized_likes',
       'normalized_views', 'category_like_view_ratio',
       'relative_category_engagement', 'like_view_percentile',
       'comment_like_percentile', 'engagement_score', 'hours_to_trend',
       'publish_weekday', 'is_weekend'],
      dtype='object')
```

In [28]:
```
def calculate_metrics(model, X_train, y_train, X_test, y_test):
    # Predictions
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)

    # Calculate metrics
    train_rmse = np.sqrt(mean_squared_error(y_train, train_preds))
    test_rmse = np.sqrt(mean_squared_error(y_test, test_preds))

    train_r2 = r2_score(y_train, train_preds)
    test_r2 = r2_score(y_test, test_preds)

    return {
        "Train RMSE": train_rmse,
        "Test RMSE": test_rmse,  # Changed from val_rmse to test_rmse
        "Train R^2": train_r2,
        "Test R^2": test_r2
    }
```

In [29]:
```
# Create an empty DataFrame if it isn't already
results = pd.DataFrame()

# Train models for each variation and record results
for i, params in enumerate(variations):
    model, metrics = train_xgboost(X_train, y_train, X_test, y_test, params)

    # Create a DataFrame with the metrics for this variation
    result_row = pd.DataFrame({
        "Variation": [f"Variation {i + 1}"],
        **metrics
    })

    # Concatenate the new row to the results DataFrame
    results = pd.concat([results, result_row], ignore_index=True)
```

In [30]:
```
# Display the comparison table
print("XGBoost Model with Previous Winning Variation:")
print(results)
```

```
XGBoost Model with Previous Winning Variation:
     Variation    Train RMSE     Test RMSE  Train R^2  Test R^2
0  Variation 1  3.494705e+06  4.616764e+06     0.9404  0.901259
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: