# Spotify Popularity Regression Analysis

Kiluva James

2025-06-12

## Table of contents



```
import basic libraries
import Sci-kit Learn modules
import SciPy and Statsmodels for statistical analysis (ANOVA, etc.)
import System and Utility modules
```



```
--------------------
Loading the dataset...
--------------------
--------------------
DATA CLEANING
--------------------
adding market counts for each song
Check for non-string types in 'country' column, Before removing non-string rows:
               spotify_id country
0  2plbrEY59IikOBgBGLjaoe     NaN
1  4wJ5Qq0jBN4ajy7ouZIV1c     NaN
2  2CGNAOSuO1MEFCbBRgUzjd     NaN
3  6AI3ezQ4o3HUoP6Dhudph3     NaN
4  6dOtVTDdiauQNBQEDOtlAB     NaN
country
```

```
<class 'str'>      1726125
<class 'float'>      23907
Name: count, dtype: int64
Removing rows where 'country' is not a string,
We also check for pd.NA which is a nullable type for object columns in newer pandas
Converting 'country' column to string type...
After removing non-string rows and converting to string:
               spotify_id country
0  2plbrEY59IikOBgBGLjaoe     nan
1  4wJ5Qq0jBN4ajy7ouZIV1c     nan
2  2CGNAOSuO1MEFCbBRgUzjd     nan
3  6AI3ezQ4o3HUoP6Dhudph3     nan
4  6dOtVTDdiauQNBQEDOtlAB     nan
country
<class 'str'>    1750032
Name: count, dtype: int64
Checking for duplicate rows based on 'spotify_id' and 'country'...
count te no. of artists in each song
converting date columns to datetime format and creating  a new column days out.
change boolean into integers
Standardize duration_ms (convert to minutes)
Removing unneeded columns
Missing values before handling:
daily_rank           0
daily_movement       0
weekly_movement      0
popularity           0
is_explicit          0
danceability         0
energy               0
key                  0
loudness             0
mode                 0
speechiness          0
acousticness         0
instrumentalness     0
liveness             0
valence              0
tempo                0
time_signature       0
market_count         0
artist_count         0
days_out            12
```

```
duration_min         0
dtype: int64
Handle missing values by imputing with mean (for numerical columns)
For categorical columns, you'd typically use mode or a constant
Missing values after handling:
daily_rank          0
daily_movement      0
weekly_movement     0
popularity          0
is_explicit         0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
market_count        0
artist_count        0
days_out            0
duration_min        0
dtype: int64
Arrange the columns
Remove popularity 0
```

==========

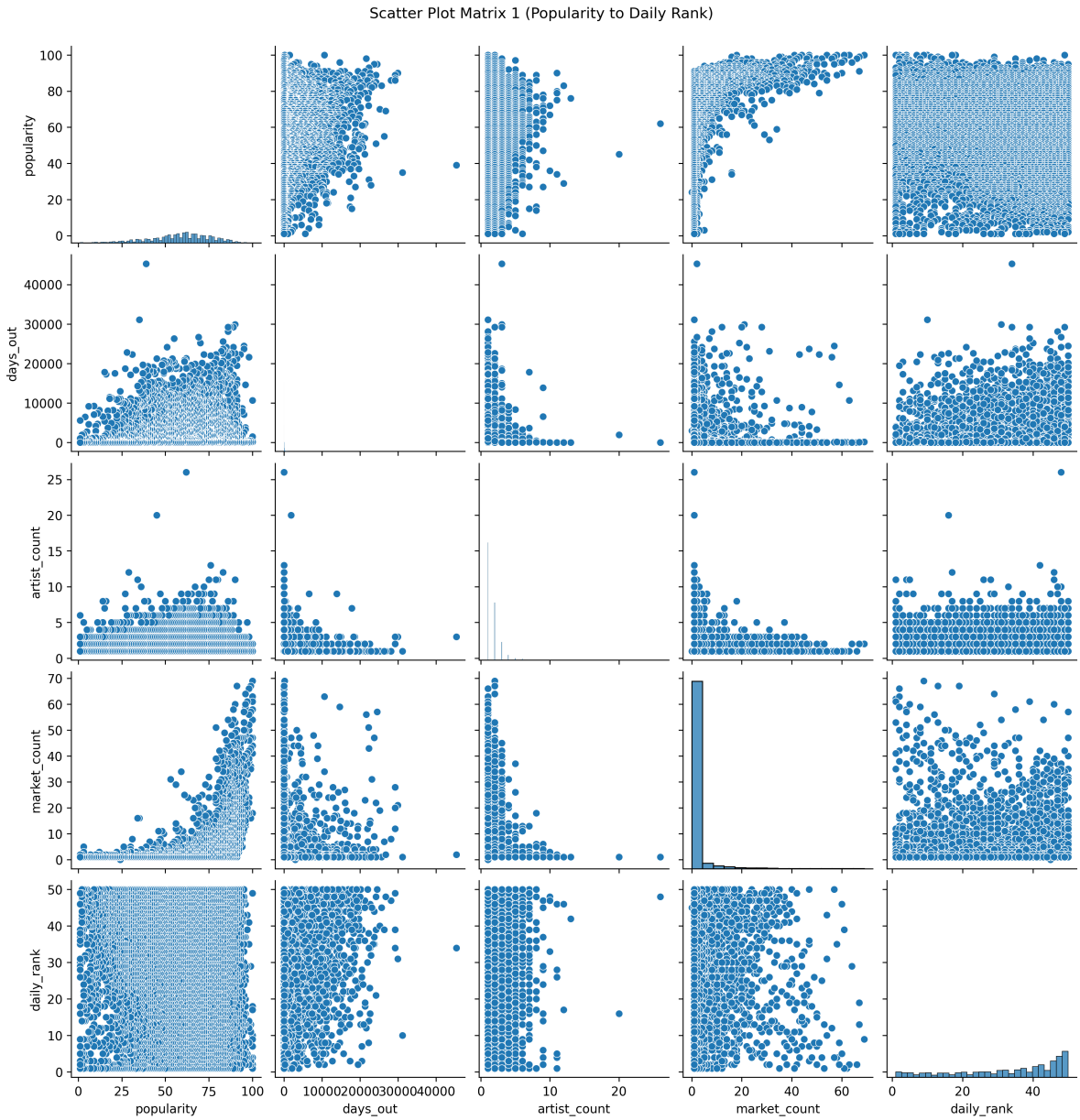
3. DESCRIPTIVE STATISTICS, ANOVA AND CORRELATION. ==========

```
--------------------
--------------------
Function to compute basic statistics
Looping through columns and compute statistics
Descriptive Statistics:  scatter plots of inter-features
                         Mean      Median           SD      Variance  \
popularity          58.216856   60.000000    17.932957  3.215910e+02
days_out           942.045425   39.000000  2718.615353  7.390869e+06
artist_count         1.631012    1.000000     1.000867  1.001735e+00
```

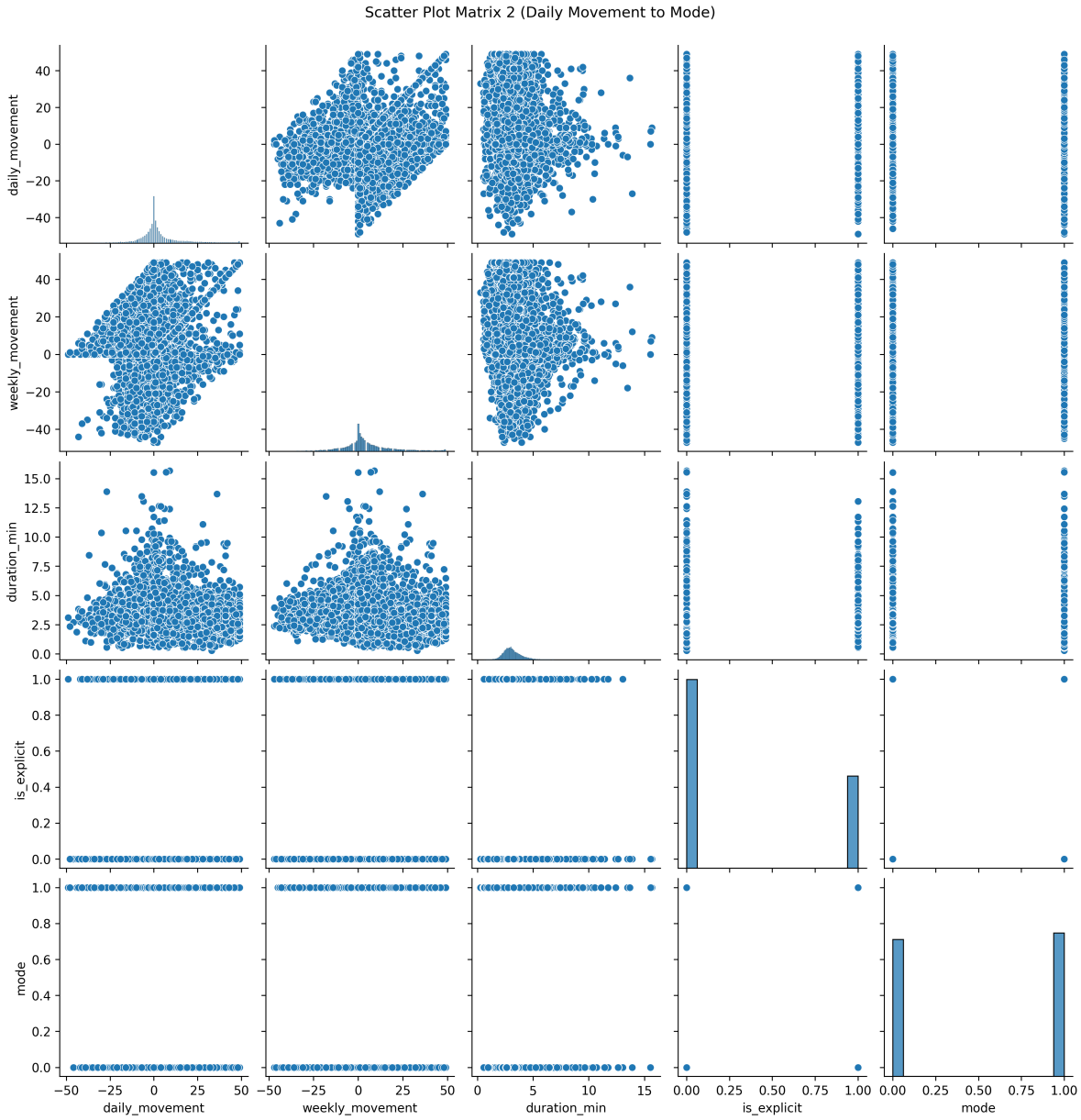
| | | | | |
|---|---|---|---|---|
| market_count | 2.201072 | 1.000000 | 4.725443 | 2.232982e+01 |
| daily_rank | 33.333946 | 38.000000 | 14.713720 | 2.164936e+02 |
| daily_movement | 2.905743 | 0.000000 | 12.264597 | 1.504203e+02 |
| weekly_movement | 4.500620 | 2.000000 | 14.583514 | 2.126789e+02 |
| duration_min | 3.184053 | 3.028083 | 0.985339 | 9.708934e-01 |
| is_explicit | 0.327840 | 0.000000 | 0.469438 | 2.203720e-01 |
| mode | 0.511590 | 1.000000 | 0.499878 | 2.498781e-01 |
| danceability | 0.673112 | 0.691000 | 0.140192 | 1.965372e-02 |
| energy | 0.650164 | 0.668000 | 0.170803 | 2.917380e-02 |
| loudness | -7.076410 | -6.582000 | 3.195620 | 1.021199e+01 |
| speechiness | 0.120909 | 0.069500 | 0.112578 | 1.267391e-02 |
| acousticness | 0.277781 | 0.201000 | 0.251338 | 6.317067e-02 |
| instrumentalness | 0.027357 | 0.000001 | 0.129016 | 1.664518e-02 |
| liveness | 0.182934 | 0.127000 | 0.139376 | 1.942577e-02 |
| valence | 0.534793 | 0.534000 | 0.229416 | 5.263188e-02 |
| tempo | 122.338718 | 121.031000 | 27.866344 | 7.765331e+02 |
| key | 5.372562 | 6.000000 | 3.600920 | 1.296662e+01 |
| time_signature | 3.945649 | 4.000000 | 0.348977 | 1.217853e-01 |

| | IQR |
|---|---|
| popularity | 23.000000 |
| days_out | 221.000000 |
| artist_count | 1.000000 |
| market_count | 0.000000 |
| daily_rank | 23.500000 |
| daily_movement | 7.000000 |
| weekly_movement | 12.000000 |
| duration_min | 0.995108 |
| is_explicit | 1.000000 |
| mode | 1.000000 |
| danceability | 0.191000 |
| energy | 0.231000 |
| loudness | 3.208000 |
| speechiness | 0.122600 |
| acousticness | 0.374350 |
| instrumentalness | 0.000137 |
| liveness | 0.126600 |
| valence | 0.356000 |
| tempo | 40.051500 |
| key | 7.000000 |
| time_signature | 0.000000 |

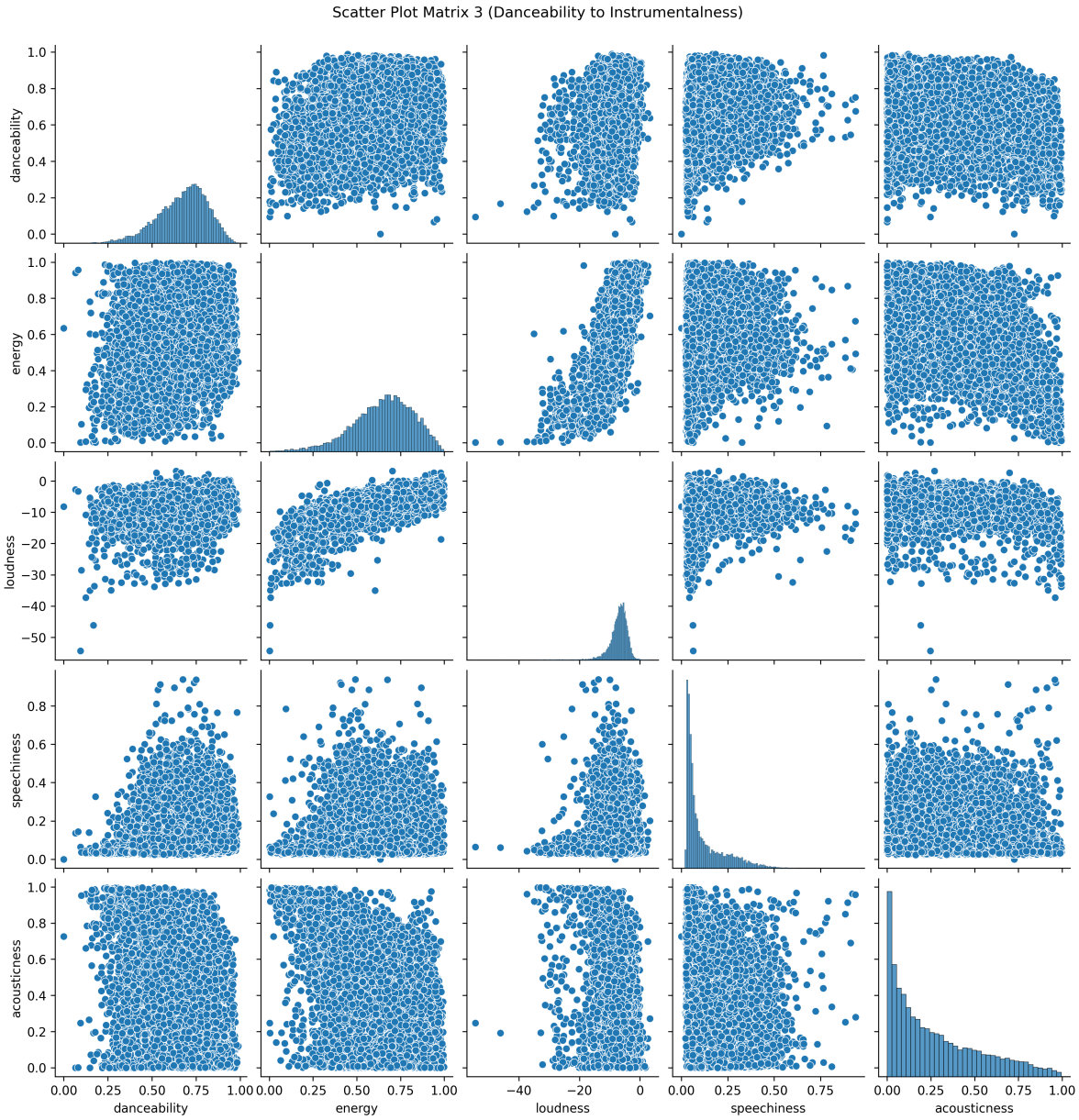
Plot 1: Popularity to daily_rank



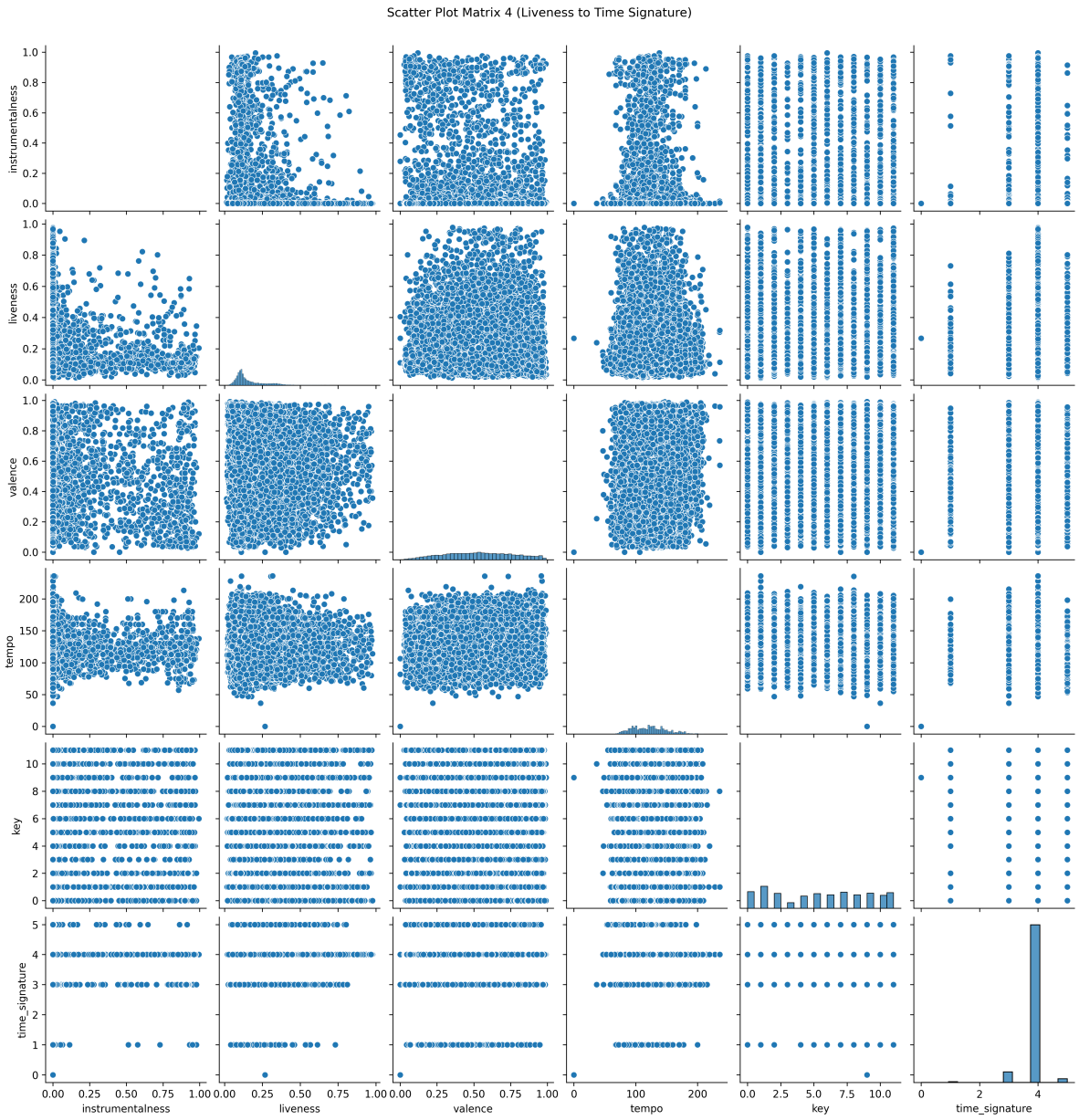

Plot 2: daily_movement to mode



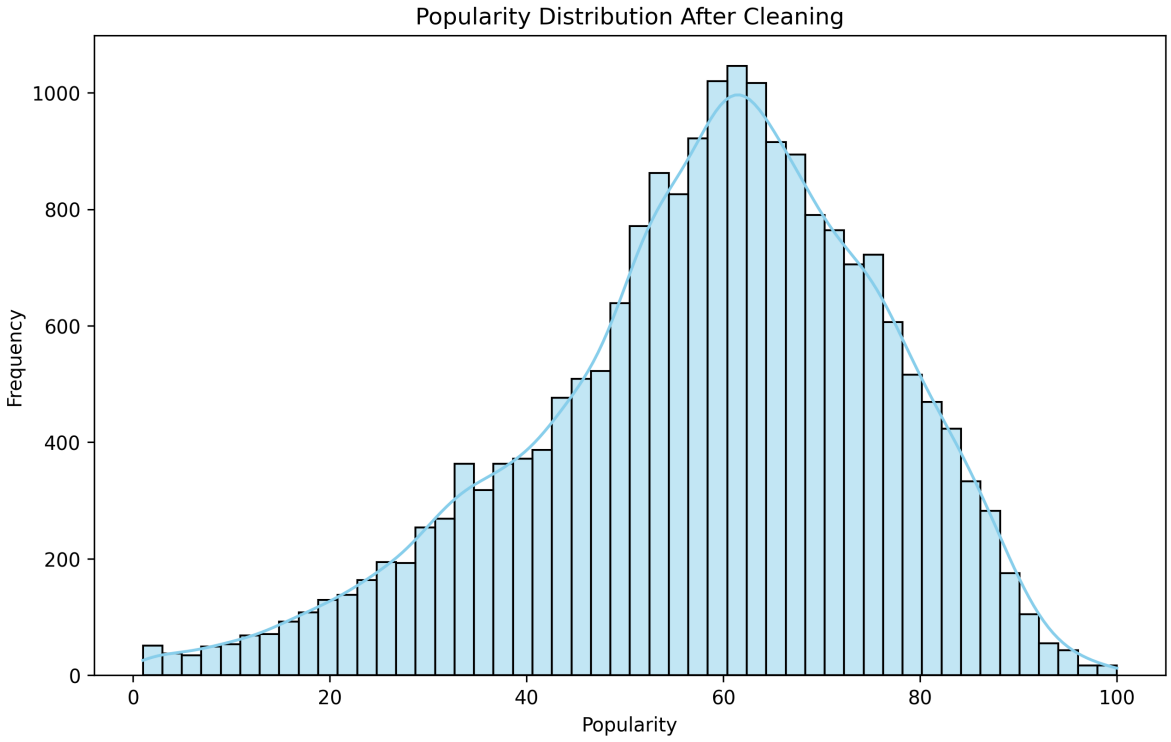

Plot 3: danceability to instrumentalness



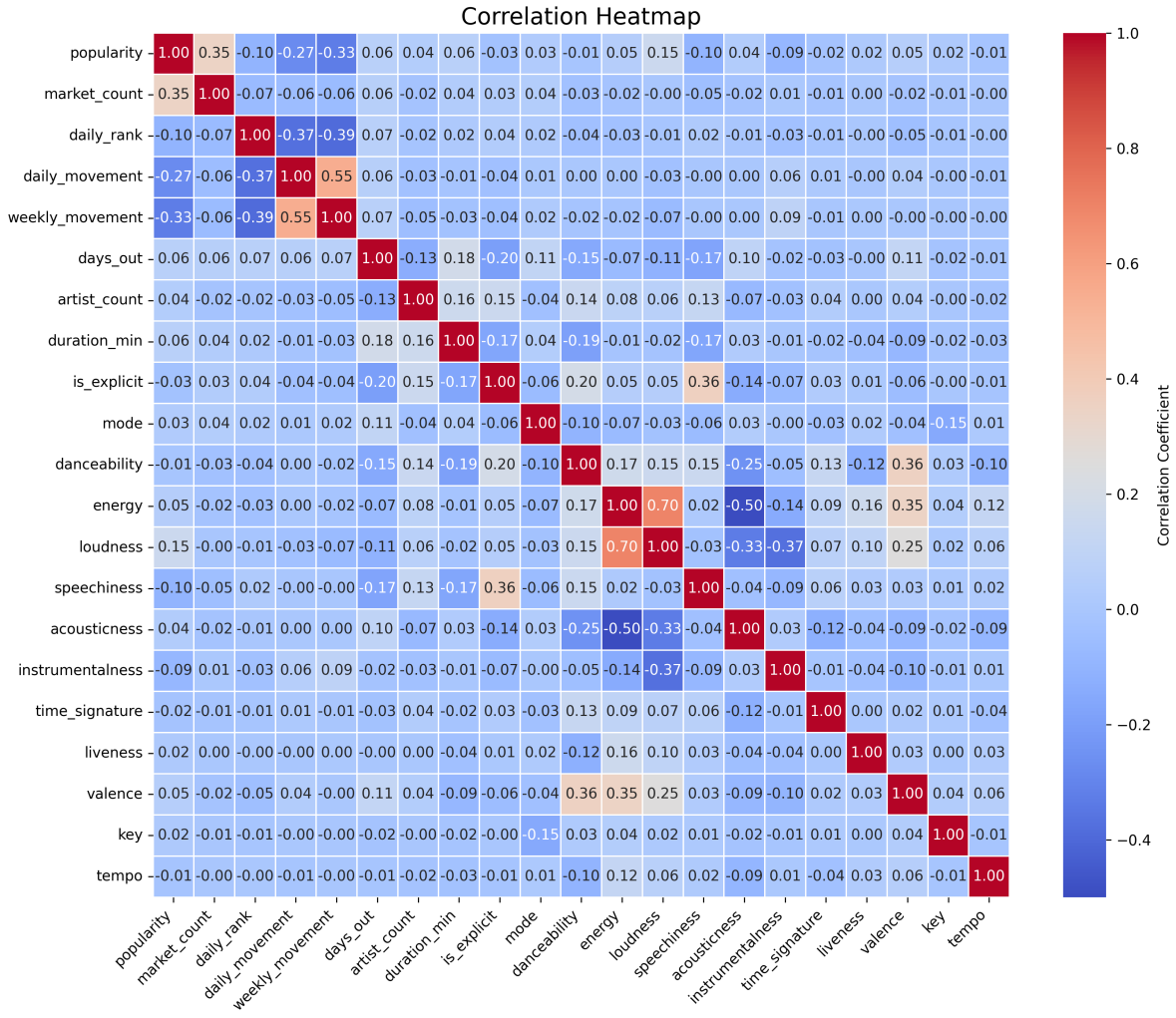

Plot 4: liveness to time_signature




check popularity distribution



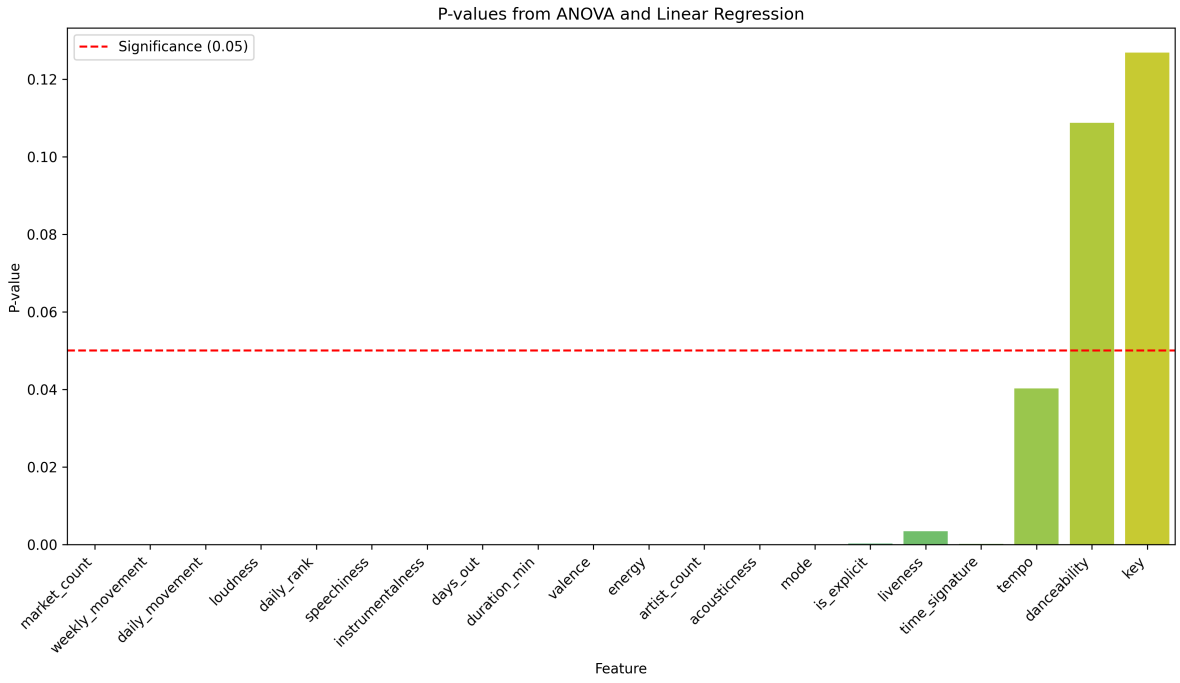

```
---------------------
CORRELATION
---------------------
Select the audio feature columns for correlation analysis
Compute correlation matrix
Plot heatmap with correlation values
```



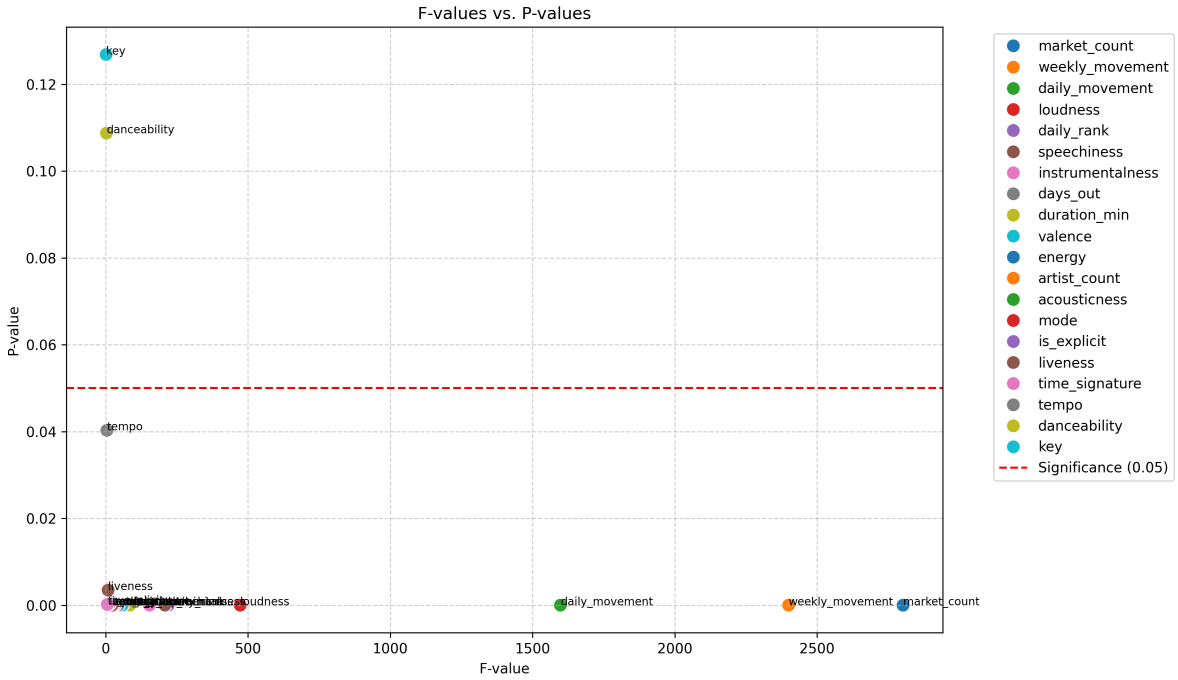

```
--------------------
ANOVA FOR FEATURE IMPORTANCE
--------------------
Performing ANOVA for feature importance, check categorical and continous variables
Create a data frame for variance importance based on ANOVA
ANOVA Feature Importance (Combined):
              Feature      F_Value        P_Value
0        market_count  2802.118963   0.000000e+00
1    weekly_movement   2399.985319   0.000000e+00
2     daily_movement   1598.022570   0.000000e+00
3           loudness    472.545990  1.386045e-103
4         daily_rank    219.741565   1.877037e-49
5        speechiness    208.137340   6.003006e-47
```

| | | | |
|---|---|---|---|
| 6 | instrumentalness | 153.807716 | 3.429673e-35 |
| 7 | days_out | 84.952470 | 3.348759e-20 |
| 8 | duration_min | 84.798300 | 3.619138e-20 |
| 9 | valence | 57.860831 | 2.936628e-14 |
| 10 | energy | 43.021824 | 5.544713e-11 |
| 11 | artist_count | 31.795854 | 1.735529e-08 |
| 12 | acousticness | 25.090714 | 5.515806e-07 |
| 13 | mode | 21.596230 | 3.386430e-06 |
| 14 | is_explicit | 13.087214 | 2.980451e-04 |
| 15 | liveness | 8.543747 | 3.470968e-03 |
| 16 | time_signature | 5.635186 | 1.572127e-04 |
| 17 | tempo | 4.206484 | 4.028262e-02 |
| 18 | danceability | 2.572726 | 1.087359e-01 |
| 19 | key | 1.491359 | 1.268530e-01 |

1. Bar plot of p-values



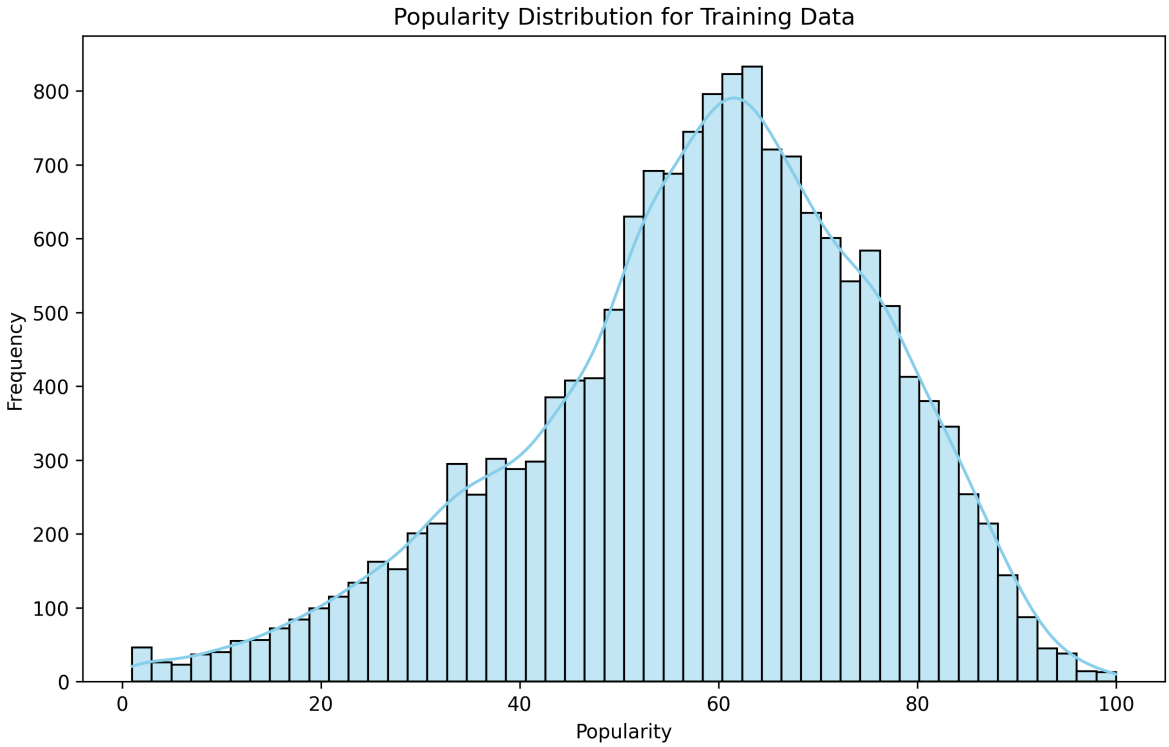

2. Scatter plot of F-values vs. p-values



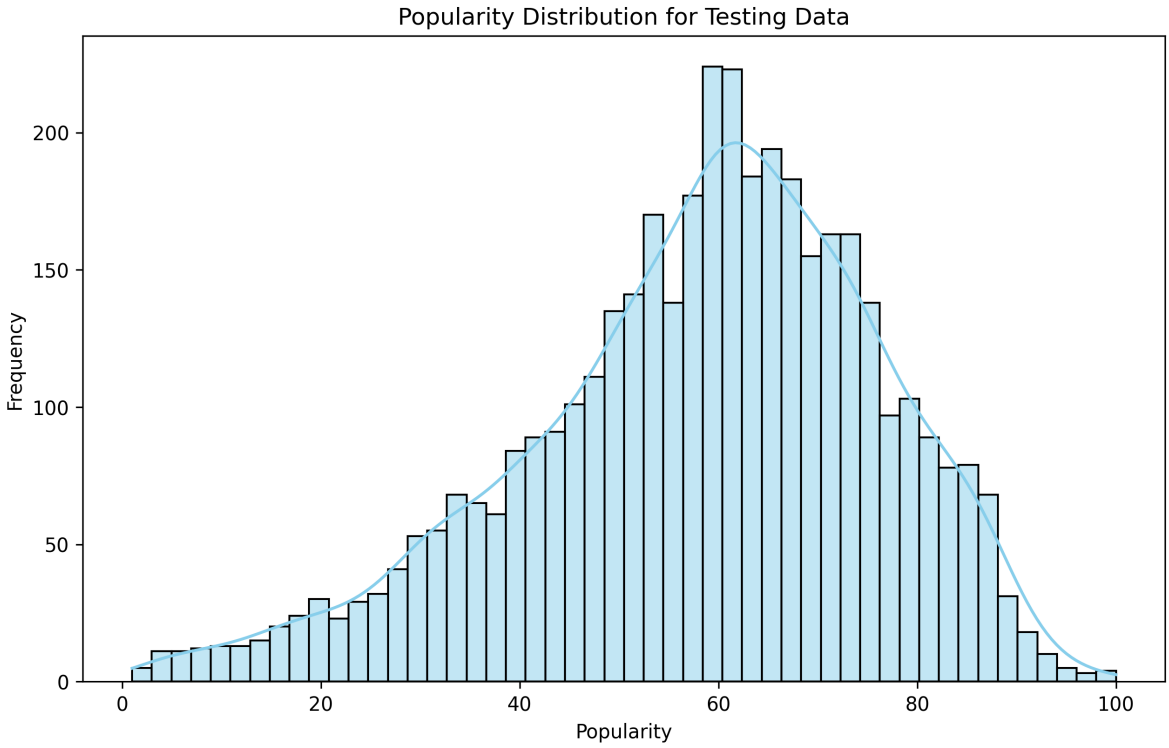

==========

## 4. DATA PATITION AND PREPARATION FOR MODELLING. ==========

```
--------------------
SKLEARN DATA PARTITION
--------------------
Define feature columns and target variable
Split the data into training and testing sets and set random_state for reproducibility
create a data frame of the sets
combinning features and target into data frames
testing popularity distribution for training data
```




testing popularity distribution for testing data




```
--------------------
DATA PREPARATION FOR MODELING
--------------------
Identify numerical and categorical columns
Re-check categorical columns, as they might have been converted to int/float earlier
Create a preprocessing pipeline
Apply preprocessing
Get feature names after one-hot encoding
Convert back to DataFrame
Processed Training Data (First few rows):
       num__days_out  num__artist_count  num__market_count  num__daily_rank  \
13535      -0.224137          -0.630518          -0.253529        -2.196366
9686       -0.334784          -0.630518          -0.253529        -0.431651
15680      -0.342581          -0.630518          -0.253529         0.925823
5165       -0.341096          -0.630518          -0.043337        -0.160156
10436      -0.203715           0.361290          -0.253529        -1.517629

       num__daily_movement  num__weekly_movement  num__duration_min  \
13535            -0.153363             -0.303227           1.067532
9686              0.416652             -0.371464           0.079572
```

```
15680            -0.886238             -0.098517           1.309202
5165              0.823805             -0.644411          -0.046354
10436             0.498082             -0.030280           0.020173

       num__danceability  num__energy  num__loudness  ...  cat__key_7  \
13535           0.089659     0.213958      -0.351809  ...         0.0
9686            0.111187     1.221695       0.693972  ...         1.0
15680          -2.816718    -1.165051      -0.848508  ...         0.0
5165            0.922102     0.703093       1.090321  ...         0.0
10436           1.316795    -1.271128      -1.823603  ...         0.0

       cat__key_8  cat__key_9  cat__key_10  cat__key_11  \
13535         0.0         0.0          0.0          1.0
9686          0.0         0.0          0.0          0.0
15680         0.0         0.0          0.0          0.0
5165          0.0         0.0          0.0          0.0
10436         1.0         0.0          0.0          0.0

       cat__time_signature_0  cat__time_signature_1  cat__time_signature_3  \
13535                    0.0                    0.0                    0.0
9686                     0.0                    0.0                    0.0
15680                    0.0                    0.0                    0.0
5165                     0.0                    0.0                    0.0
10436                    0.0                    0.0                    0.0

       cat__time_signature_4  cat__time_signature_5
13535                    1.0                    0.0
9686                     1.0                    0.0
15680                    1.0                    0.0
5165                     1.0                    0.0
10436                    1.0                    0.0

[5 rows x 37 columns]
Processed Testing Data (First few rows):
       num__days_out  num__artist_count  num__market_count  num__daily_rank  \
9555       -0.303223          -0.630518          -0.253529         0.925823
400        -0.337011           1.353097           5.001254        -0.703145
1233       -0.323645          -0.630518          -0.253529        -0.635272
1948       -0.151362          -0.630518          -0.043337        -1.314008
15621      -0.342210          -0.630518          -0.253529         0.790075

       num__daily_movement  num__weekly_movement  num__duration_min  \
9555             -0.316224             -0.234991          -0.726558
```

```
400              -0.316224             -0.507937          -1.346719
1233             -0.071932             -0.303227          -0.046201
1948             -0.316224             -0.303227           1.919808
15621            -0.967669              0.037956          -0.152627

       num__danceability  num__energy  num__loudness  ...  cat__key_7  \
9555            2.034419    -0.605197      -0.535468  ...         0.0
400            -0.534674     1.374918       0.264489  ...         0.0
1233            0.771401    -0.328216      -1.536439  ...         0.0
1948           -0.541851    -1.365419      -1.448081  ...         0.0
15621          -1.123126     1.422064       0.778544  ...         0.0

       cat__key_8  cat__key_9  cat__key_10  cat__key_11  \
9555          1.0         0.0          0.0          0.0
400           0.0         0.0          0.0          1.0
1233          1.0         0.0          0.0          0.0
1948          0.0         0.0          1.0          0.0
15621         0.0         0.0          0.0          0.0

       cat__time_signature_0  cat__time_signature_1  cat__time_signature_3  \
9555                     0.0                    0.0                    0.0
400                      0.0                    0.0                    0.0
1233                     0.0                    0.0                    0.0
1948                     0.0                    0.0                    0.0
15621                    0.0                    0.0                    0.0

       cat__time_signature_4  cat__time_signature_5
9555                     1.0                    0.0
400                      1.0                    0.0
1233                     1.0                    0.0
1948                     1.0                    0.0
15621                    1.0                    0.0

[5 rows x 37 columns]
Combine processed features with the target variable (for modeling libraries that prefer a si
```

========== 5. MODEL TRAINING: R_F, XGB and GBM. ==========

```
print("---" * 10)
print("RANDOM FOREST MODEL")
print("---" * 10)
```

```
print("**Hyperparameter tuning using GridSearchCV**")

print("Defining the Random Forest model and hyperparameter grid...")
rf_model = RandomForestRegressor(random_state=50, n_jobs=-1)
num_features = train_processed.shape[1]

param_grid_rf = {
    'n_estimators': [200],
    'max_features': [5, 7, 9, 11],
    'min_samples_leaf': [3, 5, 7]
}

print("Performing GridSearchCV for hyperparameter tuning...")
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid_rf,
                              cv=5, n_jobs=-1, scoring='neg_mean_squared_error',
                              verbose=0)
grid_search_rf.fit(train_processed, y_train)

rf_best_model = grid_search_rf.best_estimator_

print("**Random Forest Model Results**")
print(f"  Best Parameters: {grid_search_rf.best_params_}")
print(f"  Best Cross-Validation RMSE: {np.sqrt(-grid_search_rf.best_score_):.4f}")

# Save RF model
joblib.dump(rf_best_model, "rf_model.pkl")
print("Random Forest model saved as 'rf_model.pkl'")

print("**Evaluating Random Forest on Test Set**")
rf_pred = rf_best_model.predict(test_processed)

MAE_rf = mean_absolute_error(y_test, rf_pred)
RMSE_rf = np.sqrt(mean_squared_error(y_test, rf_pred))
R_squared_rf = r2_score(y_test, rf_pred)

print(f"  Mean Absolute Error (MAE): {MAE_rf:.4f}")
print(f"  Root Mean Squared Error (RMSE): {RMSE_rf:.4f}")
print(f"  R-squared (R²): {R_squared_rf:.4f}")

print("**Random Forest - Top 10 Feature Importances**")
importance_df_rf = pd.DataFrame({
    'Feature': train_processed.columns,
```

```
    'Importance': rf_best_model.feature_importances_
})
sorted_importance_df_rf = importance_df_rf.sort_values(by='Importance',
ascending=False).reset_index(drop=True)
# Using to_markdown() for excellent table rendering in Quarto PDF/HTML
print(sorted_importance_df_rf[0:10].to_markdown(index=False))


# --- XGBOOST MODEL ---
print("---" * 10)
print("XGBOOST MODEL")
print("---" * 10)

print("**Hyperparameter tuning using GridSearchCV**")

print("Defining the XGBoost model and hyperparameter grid...")
xgb_model = XGBRegressor(objective='reg:squarederror', random_state=50, n_jobs=-1)

param_grid_xgb = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.05, 0.1], # eta
    'gamma': [0],
    'colsample_bytree': [0.7, 0.9],
    'min_child_weight': [1],
    'subsample': [0.8]
}

print("Performing GridSearchCV for hyperparameter tuning...")
grid_search_xgb = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb,
                               cv=5, n_jobs=-1, scoring='neg_mean_squared_error',
                               verbose=0)
grid_search_xgb.fit(train_processed, y_train)

xgb_best_model = grid_search_xgb.best_estimator_

print("**XGBoost Model Results**")
print(f"  Best Parameters: {grid_search_xgb.best_params_}")
print(f"  Best Cross-Validation RMSE: {np.sqrt(-grid_search_xgb.best_score_):.4f}")

# Save XGBoost model
joblib.dump(xgb_best_model, "xgb_model.pkl")
```

```
print("XGBoost model saved as 'xgb_model.pkl'")

print("**Evaluating XGBoost on Test Set**")
xgb_pred = xgb_best_model.predict(test_processed)

MAE_xgb = mean_absolute_error(y_test, xgb_pred)
RMSE_xgb = np.sqrt(mean_squared_error(y_test, xgb_pred))
R_squared_xgb = r2_score(y_test, xgb_pred)

print(f"  Mean Absolute Error (MAE): {MAE_xgb:.4f}")
print(f"  Root Mean Squared Error (RMSE): {RMSE_xgb:.4f}")
print(f"  R-squared (R²): {R_squared_xgb:.4f}")

print("**XGBoost - Top 10 Feature Importances**")
importance_df_xgb = pd.DataFrame({
    'Feature': train_processed.columns,
    'Importance': xgb_best_model.feature_importances_
})
sorted_importance_df_xgb = importance_df_xgb.sort_values(by='Importance',
ascending=False).reset_index(drop=True)
print(sorted_importance_df_xgb[0:10].to_markdown(index=False))

# --- GBM MODEL ---
print("---" * 10)
print("GBM MODEL")
print("---" * 10)

print("**Hyperparameter tuning using GridSearchCV**")

print("Defining the GBM model and hyperparameter grid...")
gbm_model = GradientBoostingRegressor(random_state=50)

param_grid_gbm = {
    'n_estimators': [100, 200],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.025],
    'min_samples_leaf': [10, 20]
}

print("Performing GridSearchCV for hyperparameter tuning...")
grid_search_gbm = GridSearchCV(estimator=gbm_model, param_grid=param_grid_gbm,
                               cv=5, n_jobs=-1, scoring='neg_mean_squared_error',
```

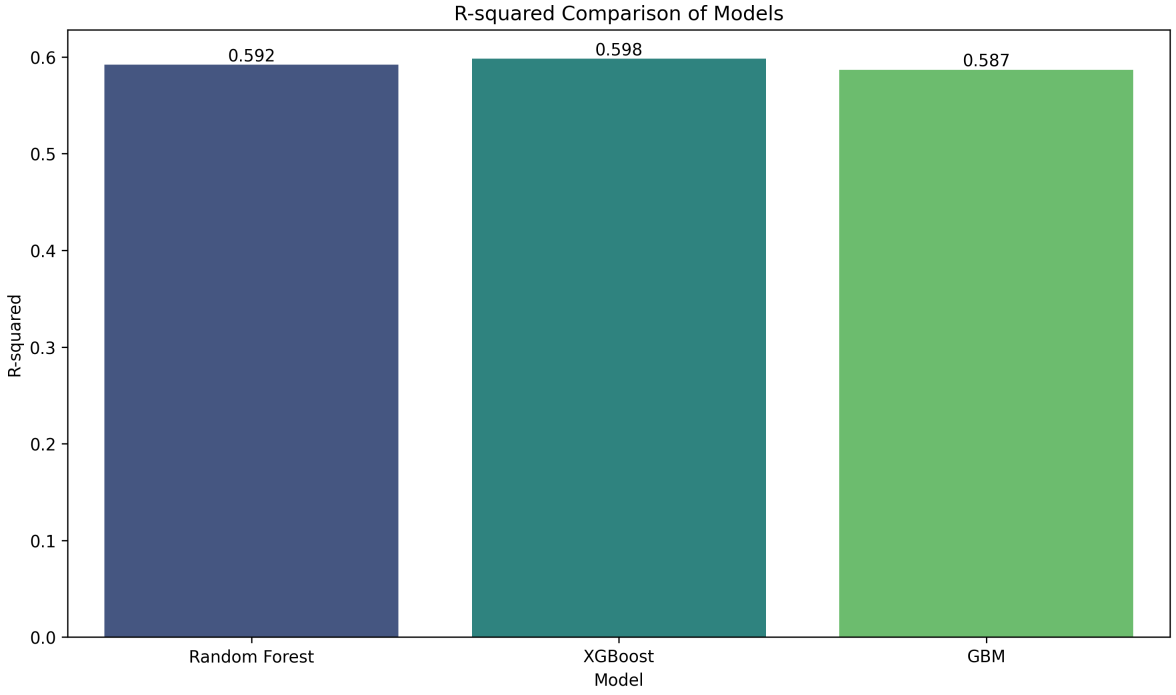
```
                                    verbose=0)
grid_search_gbm.fit(train_processed, y_train)

gbm_best_model = grid_search_gbm.best_estimator_

print("**GBM Model Results**")
print(f"  Best Parameters: {grid_search_gbm.best_params_}")
print(f"  Best Cross-Validation RMSE: {np.sqrt(-grid_search_gbm.best_score_):.4f}")

# Save GBM model
joblib.dump(gbm_best_model, "gbm_model.pkl")
print("GBM model saved as 'gbm_model.pkl'")

print("**Evaluating GBM on Test Set**")
gbm_pred = gbm_best_model.predict(test_processed)

MAE_gbm = mean_absolute_error(y_test, gbm_pred)
RMSE_gbm = np.sqrt(mean_squared_error(y_test, gbm_pred))
R_squared_gbm = r2_score(y_test, gbm_pred)

print(f"  Mean Absolute Error (MAE): {MAE_gbm:.4f}")
print(f"  Root Mean Squared Error (RMSE): {RMSE_gbm:.4f}")
print(f"  R-squared (R²): {R_squared_gbm:.4f}")

print("**GBM - Top 10 Feature Importances**")
importance_df_gbm = pd.DataFrame({
    'Feature': train_processed.columns,
    'Importance': gbm_best_model.feature_importances_
})
sorted_importance_df_gbm = importance_df_gbm.sort_values(by='Importance',
ascending=False).reset_index(drop=True)
print(sorted_importance_df_gbm[0:10].to_markdown(index=False)) # Use 0:10 for top 10

print("="*25)
print("All model training and evaluation complete.")
print("The models and their performance characteristics are printed above.")
print("="*25)
```

```
------------------------------
RANDOM FOREST MODEL
------------------------------
**Hyperparameter tuning using GridSearchCV**
```

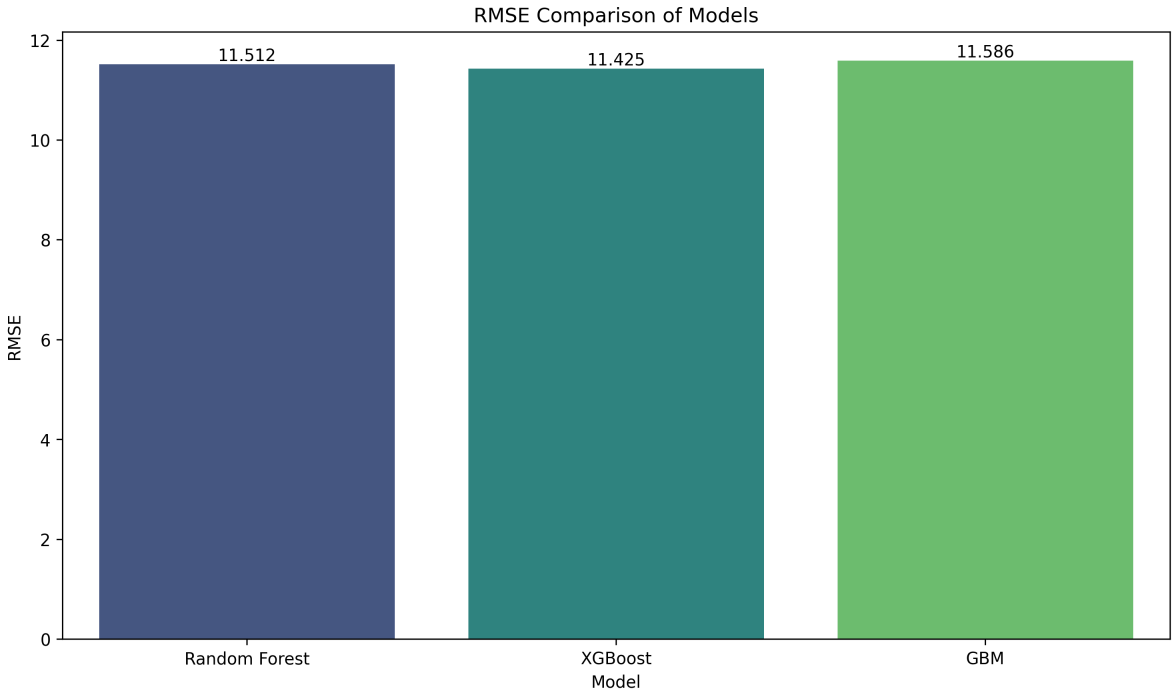
```
Defining the Random Forest model and hyperparameter grid...
Performing GridSearchCV for hyperparameter tuning...
**Random Forest Model Results**
  Best Parameters: {'max_features': 11, 'min_samples_leaf': 3, 'n_estimators': 200}
  Best Cross-Validation RMSE: 11.2798
Random Forest model saved as 'rf_model.pkl'
**Evaluating Random Forest on Test Set**
  Mean Absolute Error (MAE): 8.6988
  Root Mean Squared Error (RMSE): 11.5120
  R-squared (R²): 0.5920
**Random Forest - Top 10 Feature Importances**
| Feature              |   Importance |
|:---------------------|-------------:|
| num__days_out        |    0.26828   |
| num__market_count    |    0.191807  |
| num__daily_movement  |    0.101856  |
| num__weekly_movement |    0.0901116 |
| num__loudness        |    0.0422417 |
| num__daily_rank      |    0.0348682 |
| num__duration_min    |    0.0301496 |
| num__acousticness    |    0.0281824 |
| num__valence         |    0.0275825 |
| num__speechiness     |    0.0272215 |
------------------------------
XGBOOST MODEL
------------------------------
**Hyperparameter tuning using GridSearchCV**
Defining the XGBoost model and hyperparameter grid...
Performing GridSearchCV for hyperparameter tuning...
**XGBoost Model Results**
  Best Parameters: {'colsample_bytree': 0.7, 'gamma': 0, 'learning_rate': 0.05, 'max_depth':
  Best Cross-Validation RMSE: 11.1928
XGBoost model saved as 'xgb_model.pkl'
**Evaluating XGBoost on Test Set**
  Mean Absolute Error (MAE): 8.6070
  Root Mean Squared Error (RMSE): 11.4245
  R-squared (R²): 0.5982
**XGBoost - Top 10 Feature Importances**
| Feature               |   Importance |
|:----------------------|-------------:|
| num__market_count     |    0.267319  |
| num__days_out         |    0.136925  |
| num__weekly_movement  |    0.0652688 |
```

```
| num__daily_movement   |    0.0553407 |
| cat__is_explicit_0    |    0.0266484 |
| num__daily_rank       |    0.0262132 |
| num__loudness         |    0.024811  |
| cat__is_explicit_1    |    0.0244138 |
| cat__time_signature_4 |    0.0189405 |
| num__instrumentalness |    0.0189145 |
------------------------------
GBM MODEL
------------------------------
**Hyperparameter tuning using GridSearchCV**
Defining the GBM model and hyperparameter grid...
Performing GridSearchCV for hyperparameter tuning...
**GBM Model Results**
  Best Parameters: {'learning_rate': 0.025, 'max_depth': 5, 'min_samples_leaf': 20, 'n_estima
  Best Cross-Validation RMSE: 11.3073
GBM model saved as 'gbm_model.pkl'
**Evaluating GBM on Test Set**
  Mean Absolute Error (MAE): 8.7656
  Root Mean Squared Error (RMSE): 11.5860
  R-squared (R²): 0.5867
**GBM - Top 10 Feature Importances**
| Feature               |   Importance |
|:----------------------|-------------:|
| num__days_out         |   0.468381   |
| num__market_count     |   0.267058   |
| num__weekly_movement  |   0.0804574  |
| num__daily_movement   |   0.0771722  |
| num__loudness         |   0.0297597  |
| num__daily_rank       |   0.0214124  |
| num__instrumentalness |   0.00834009 |
| num__acousticness     |   0.00805259 |
| num__duration_min     |   0.00643343 |
| num__energy           |   0.00472631 |
========================
All model training and evaluation complete.
The models and their performance characteristics are printed above.
========================

============================================================
6. MODEL COMPARISON EVALUATION AND PLOTTING. ==============================

--------------------
```

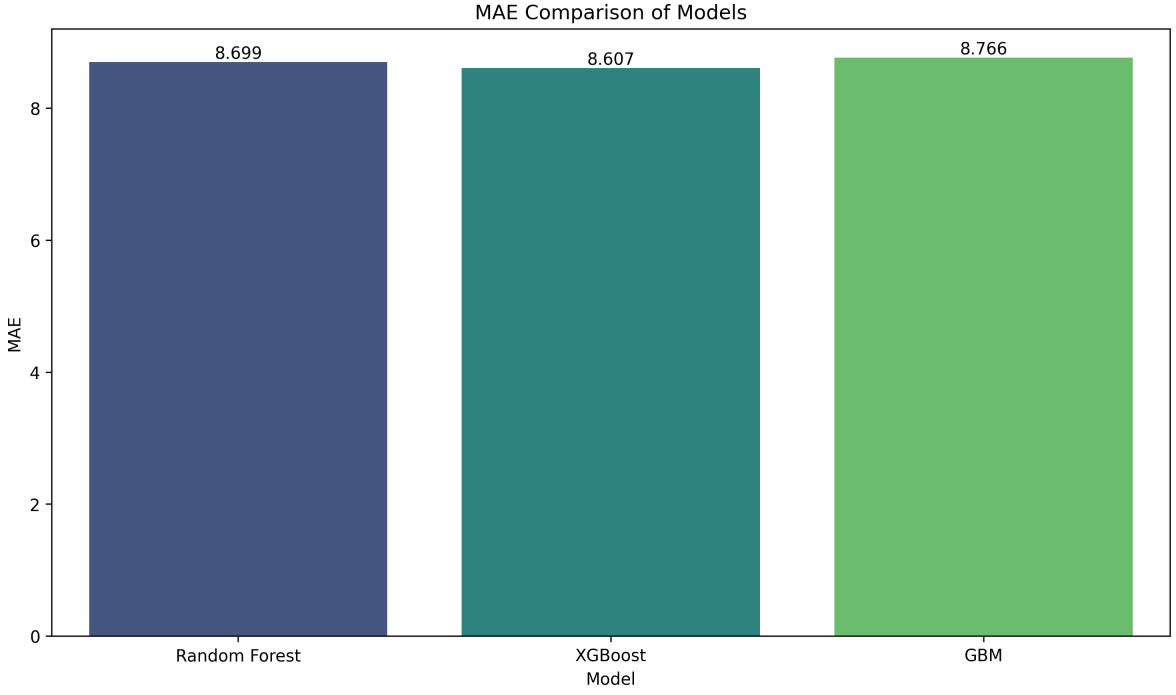
```
MODEL COMPARISON:
--------------------
Create a data frame to compare model performance
Model Performance Comparison:
           Model       MAE       RMSE  R_squared
0  Random Forest  8.698848  11.512042   0.591994
1        XGBoost  8.606954  11.424508   0.598175
2            GBM  8.765571  11.586006   0.586734
Create a bar plot for R-squared comparison
```



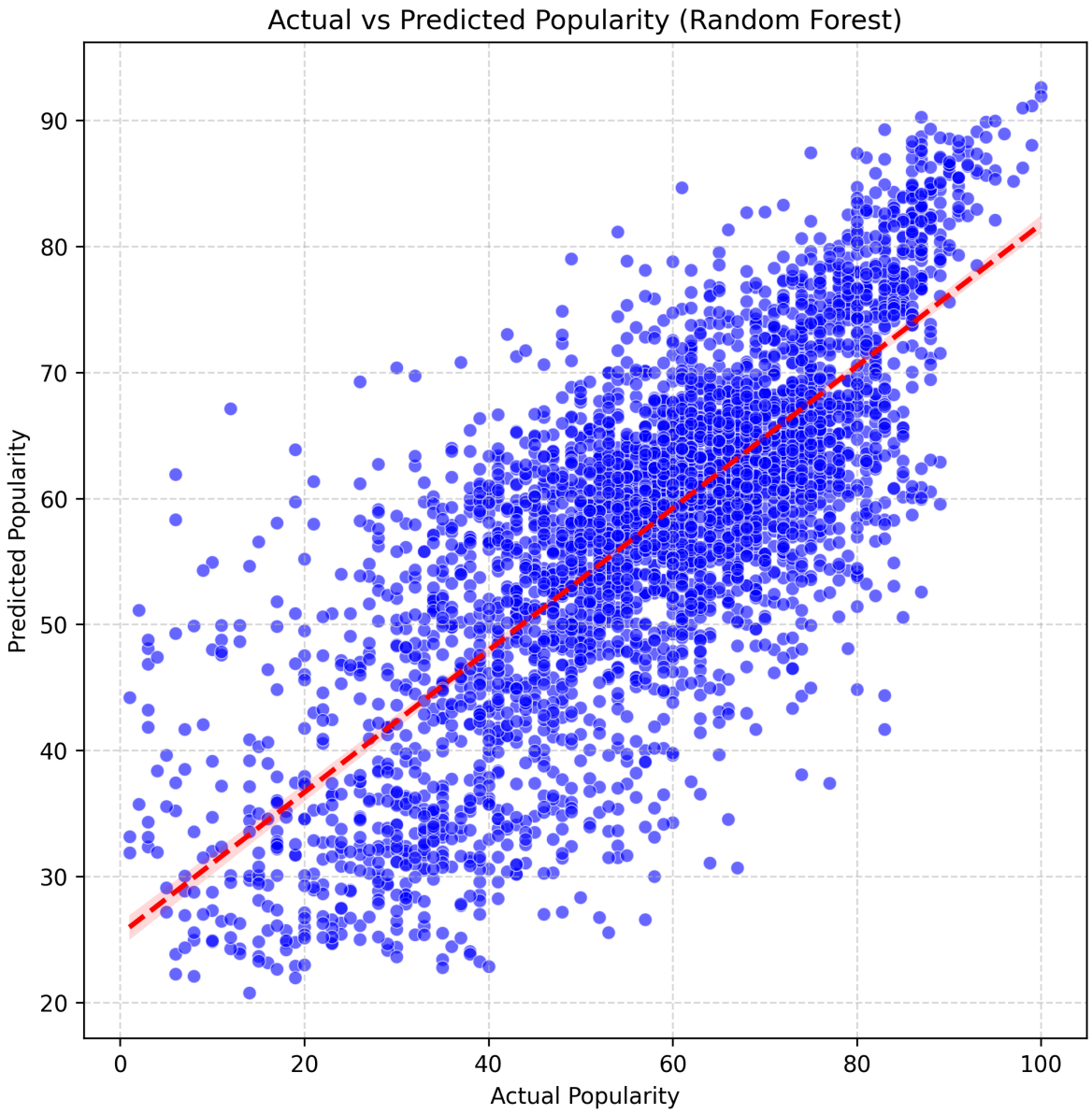

```
Create a bar plot for RMSE comparison
```



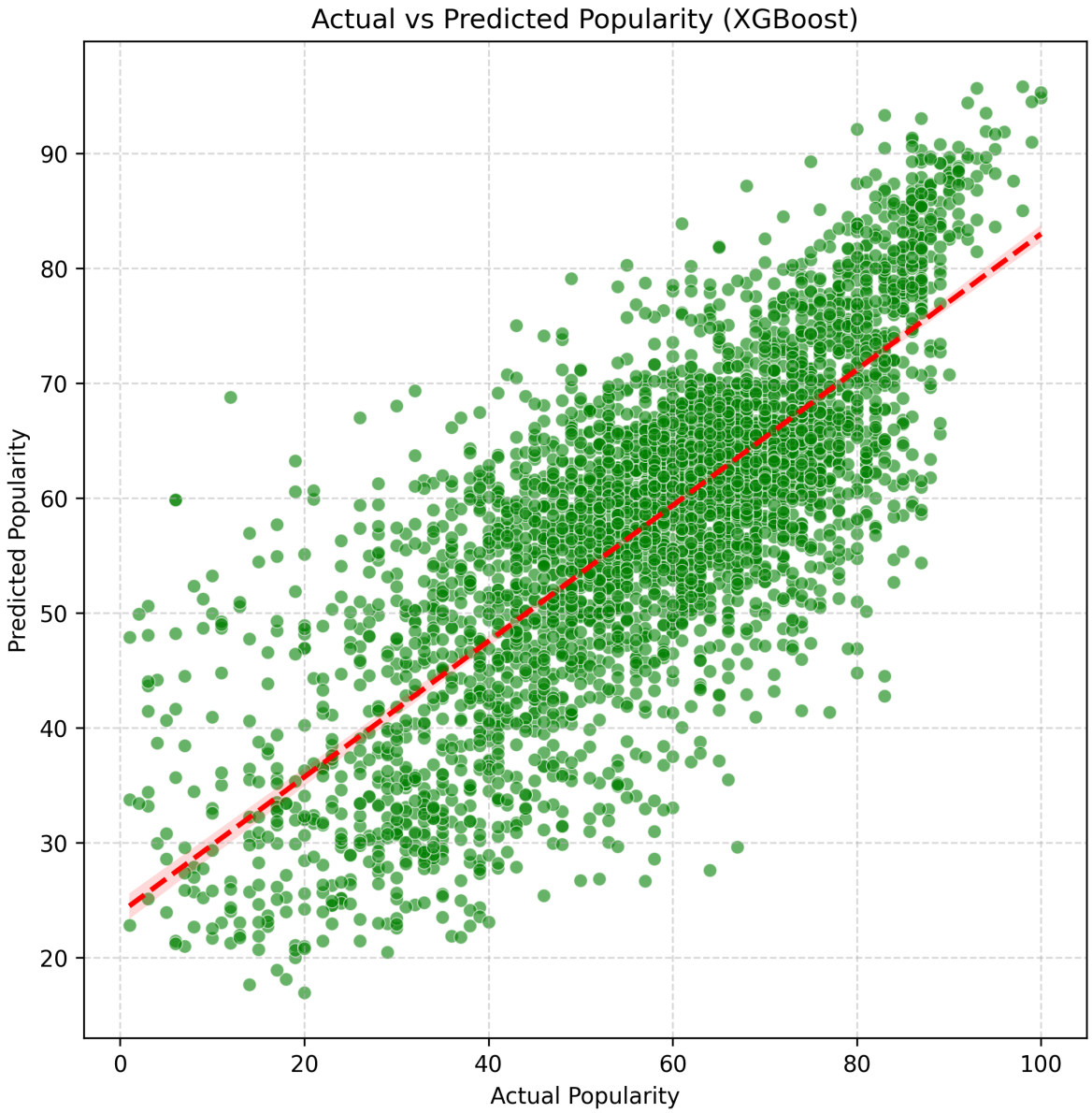

Create a bar plot for MAE comparison



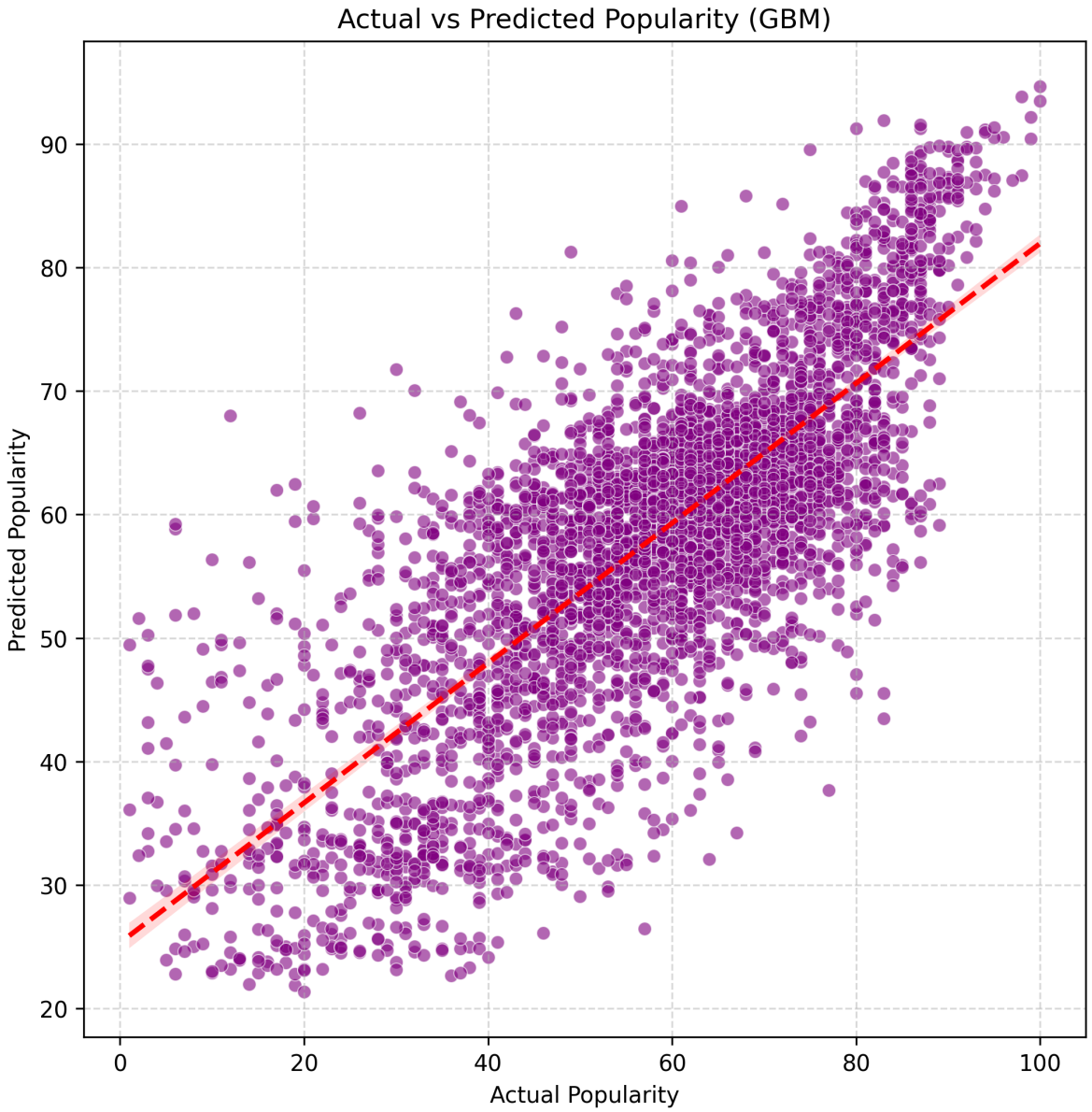
```
--------------------
ACTUAL VS PREDICTED PLOTS:
--------------------
1. Random Forest
```




2. XGBoost


Actual vs Predicted Popularity (XGBoost)
Predicted Popularity
Actual Popularity
0
20
40
60
80
100
20
30
40
50
60
70
80
90

3. GBM




THE END : ALWAYS and FOREVER : -j-k.-