Machine Learning from Scratch: Kaggle Most Streamed Spotify Songs 2023

Machine learning _CSCI_6364_82
Assignment 1

Bhavya Sree Gudiseva
20th October 2023

1. INTRODUCTION

The "Most Streamed Spotify Songs 2023" dataset offers a fascinating insight into the preferences of music listeners on the world's most popular streaming platform. The dataset comprises a range of features about songs, and the goal of this assignment is to predict the number of streams a song has garnered based on these features. This report will outline the steps taken to preprocess the data, implement a machine learning algorithm from first principles, and evaluate its performance.

2. DATA PREPROCESSING

2.1. Handling Missing Values

Upon inspecting the dataset with 24 features and 953 rows, missing values were found in 'in_shazam_charts' (50 values) and 'key' (95 values).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               953 non-null    object
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
 17  danceability_%        953 non-null    int64
 18  valence_%             953 non-null    int64
 19  energy_%              953 non-null    int64
 20  acousticness_%        953 non-null    int64
 21  instrumentalness_%    953 non-null    int64
 22  liveness_%            953 non-null    int64
 23  speechiness_%         953 non-null    int64
dtypes: int64(17), object(7)
memory usage: 178.8+ KB
```

Observations:

- The 'key' column is categorical with 'C#' as the most frequent value and 'D#' the least.

- 'in_shazam_charts' is numerical, and its distribution is right-skewed after any non-numeric entries were converted to NaN.

```
Track Attributes: Index(['track_name', 'artist(s)_name', 'artist_count', 'released_year',
       'released_month', 'released_day', 'in_spotify_playlists',
       'in_spotify_charts', 'streams', 'in_apple_playlists', 'in_apple_charts',
       'in_deezer_playlists', 'in_deezer_charts', 'in_shazam_charts', 'bpm',
       'key', 'mode', 'danceability_%', 'valence_%', 'energy_%',
       'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%'],
      dtype='object')
Total number of features: 24
Total number of rows: 953
Track information (Track Attribute, Null Values Count):
[('track_name', 0), ('artist(s)_name', 0), ('artist_count', 0), ('released_year', 0), ('released_month', 0), ('released_day', 0), ('in_spotif
y_playlists', 0), ('in_spotify_charts', 0), ('streams', 0), ('in_apple_playlists', 0), ('in_apple_charts', 0), ('in_deezer_playlists', 0),
('in_deezer_charts', 0), ('in_shazam_charts', 50), ('bpm', 0), ('key', 95), ('mode', 0), ('danceability_%', 0), ('valence_%', 0), ('energy
_%', 0), ('acousticness_%', 0), ('instrumentalness_%', 0), ('liveness_%', 0), ('speechiness_%', 0)]
```

Methodology:

1. 'key' column:

- Used mode imputation with 'C#'.

- Confirmed no more missing values post-imputation.

```
Unique Keys, Their Frequencies, and Number of Null Values:
Key: C#, Frequency: 120, Null Value: False
Key: G, Frequency: 96, Null Value: False
Key: nan, Frequency: 95, Null Value: True
Key: G#, Frequency: 91, Null Value: False
Key: F, Frequency: 89, Null Value: False
Key: B, Frequency: 81, Null Value: False
Key: D, Frequency: 81, Null Value: False
Key: A, Frequency: 75, Null Value: False
Key: F#, Frequency: 73, Null Value: False
Key: E, Frequency: 62, Null Value: False
Key: A#, Frequency: 57, Null Value: False
Key: D#, Frequency: 33, Null Value: False
```

```
Unique Keys, Their Frequencies, and Number of Null Values:
Key: C#, Frequency: 215, Null Value: False
Key: G, Frequency: 96, Null Value: False
Key: G#, Frequency: 91, Null Value: False
Key: F, Frequency: 89, Null Value: False
Key: B, Frequency: 81, Null Value: False
Key: D, Frequency: 81, Null Value: False
Key: A, Frequency: 75, Null Value: False
Key: F#, Frequency: 73, Null Value: False
Key: E, Frequency: 62, Null Value: False
Key: A#, Frequency: 57, Null Value: False
Key: D#, Frequency: 33, Null Value: False
```

2. 'in_shazam_charts' column:

- Chose median imputation due to the right-skewed distribution.

- Replaced all missing values with the median.

Post-imputation, both columns were saved separately for reference. This approach efficiently addressed the missing values, ensuring a complete dataset for subsequent analyses and model training.

2.2. Handling Outliers

Outliers can skew machine learning models, especially in your music streaming dataset. Properly managing them ensures meaningful insights and robust models.

Observations:

1. 'Streams': Right-skewed, indicating most songs have fewer streams with a few exceptions having many.

2. Platform-Specific Columns: Like 'Streams', they have right-skewed distributions, with few songs having high visibility.

3. 'Released_Year': Potential outliers observed.

4. 'In_Deezer_Charts' and 'In_Shazam_Chart': Right-skewed; few songs dominate.

5. 'BPM': Outliers might represent unrealistic song tempos.

Approach for Outliers:

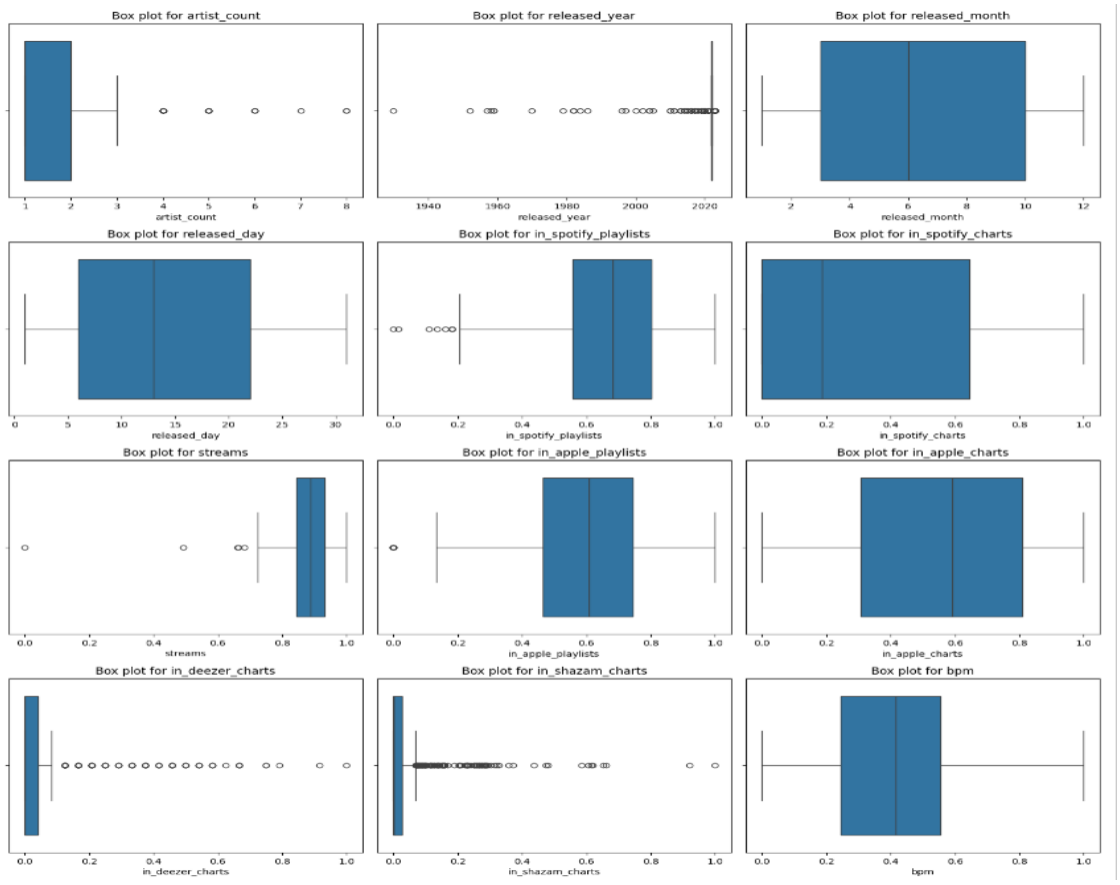
- Used the Interquartile Range (IQR) method to identify outliers in columns like 'streams', platform-specific features, 'released_year', 'bpm', etc.

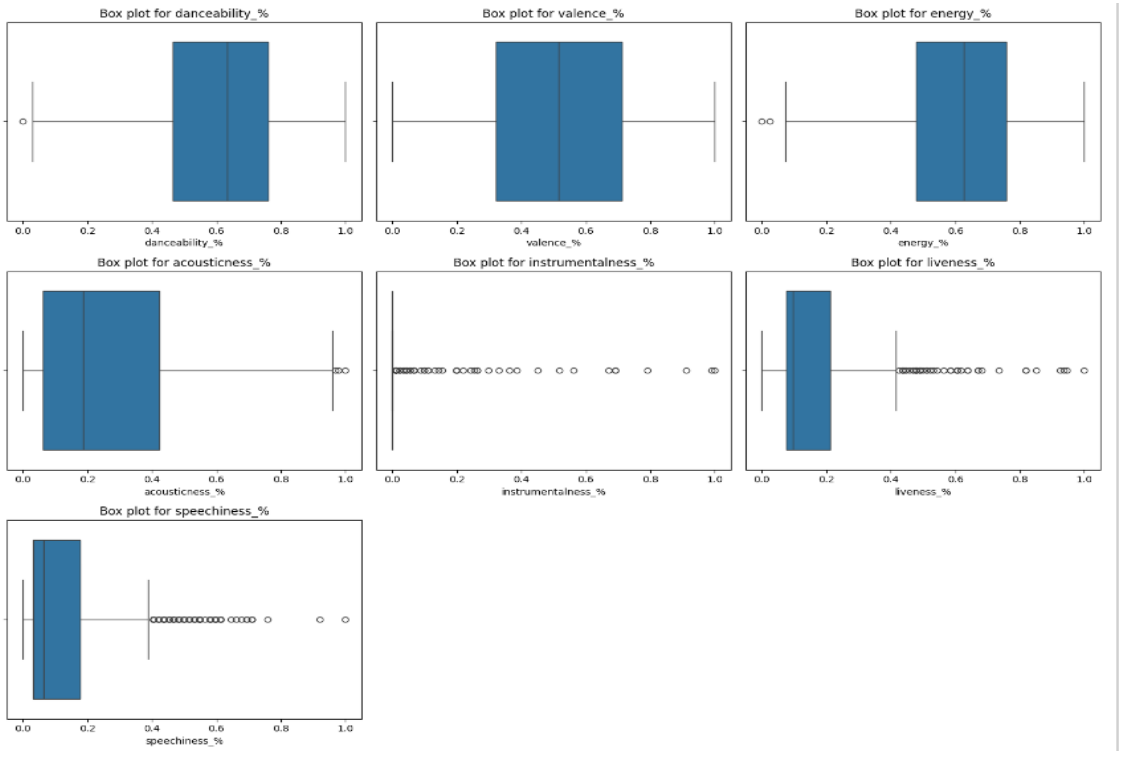
Handling Outliers:

1. 'BPM': Removed 5 outliers entirely due to their importance.

2. For right-skewed columns like 'streams' and platform-specific features: Applied log transformation (`np.log1p`) for symmetry.

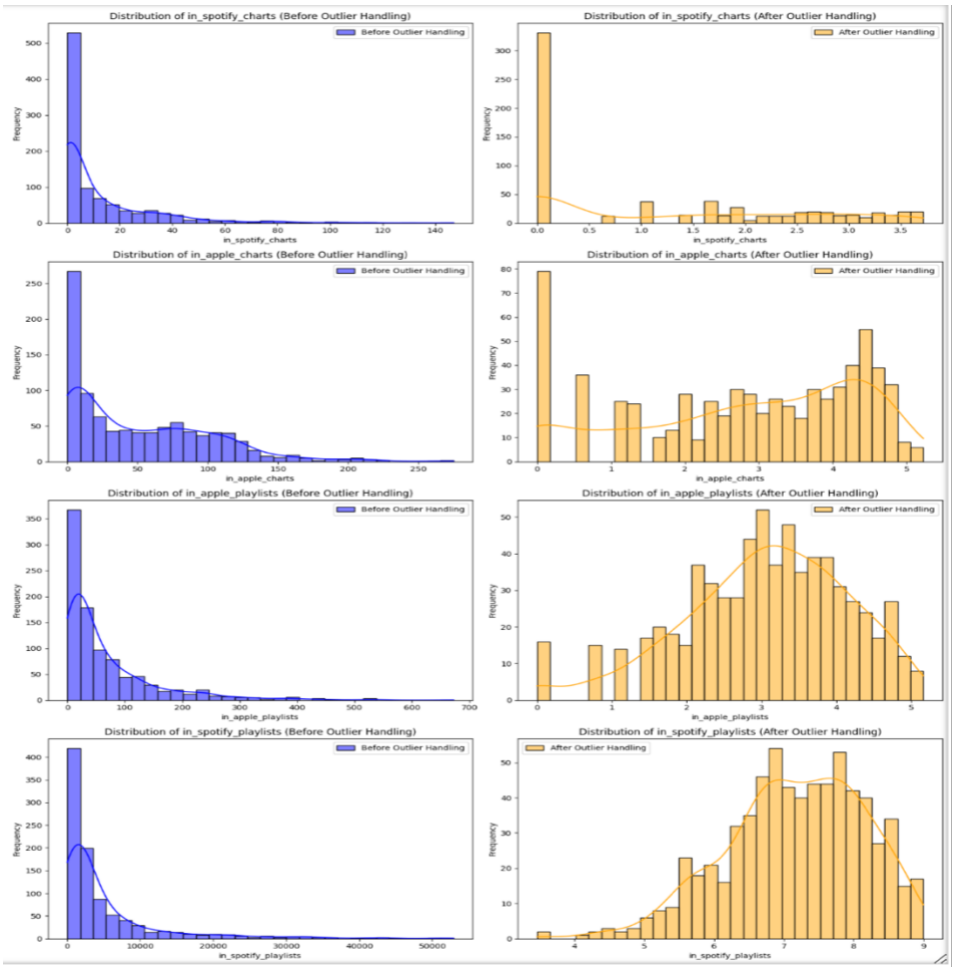
3. Decided to retain outliers in 'in_shazam_charts', 'in_deezer_charts', and 'released_year'.





Comparative Analysis:

- Post-outlier handling, visualizing the distributions confirms the changes. Summary statistics further emphasize the transformation's effectiveness.



Conclusion:

Tackling outliers is vital. In your dataset, the structured approach to manage outliers paves the way for reliable analytics and model training.

2.3. Scaling of Data

Data scaling is vital for machine learning as algorithms can prioritize features based on their numerical ranges. Scaling ensures every attribute impacts the model equally.

Approach:

1. Min-Max Scaling: Chosen for its ability to transform features to a range, usually [0, 1], using the formula: (value - min) / (max - min).

2. Features for Scaling: Metrics related to streaming platforms, song characteristics like BPM, energy, danceability, and speechiness were selected.

3. Visualization: Distributions of each feature were compared before and after scaling. The goal was to confirm that the inherent distribution remains consistent post-scaling.

4. Backup and Replacement: A dataset backup was created before substituting original features with scaled versions.

Observations:

- Features after scaling were bounded between [0, 1].

- Distributions' shapes remained consistent pre- and post-scaling.

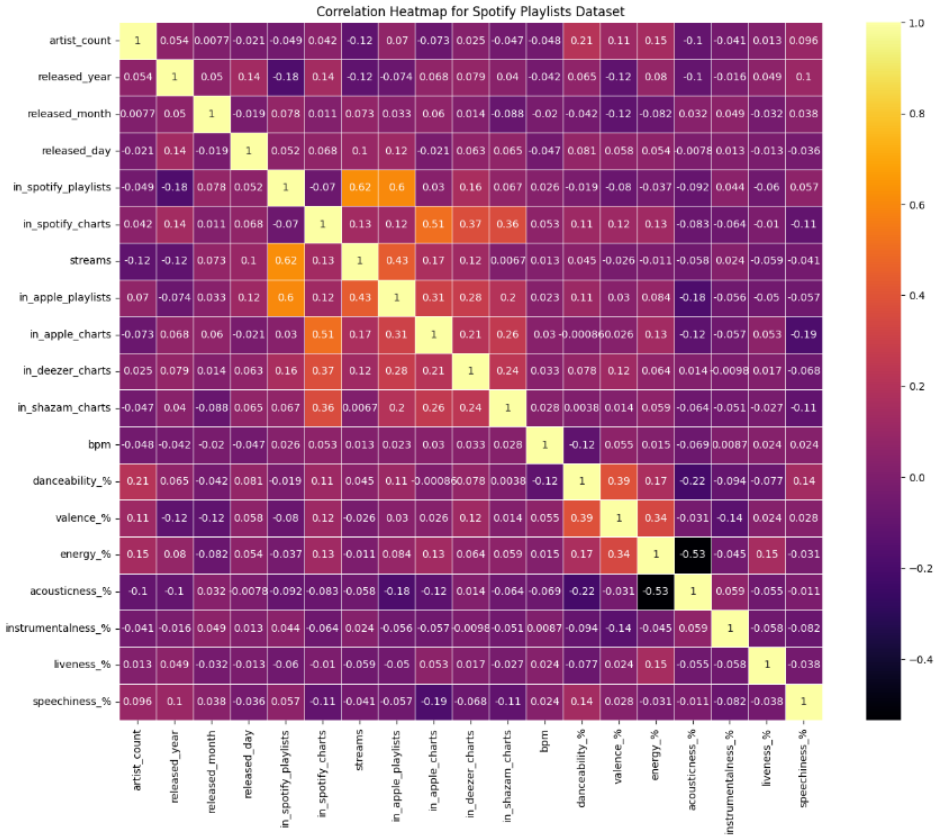
- Having a backup is crucial for potential reanalysis or comparisons.

Conclusion:

Scaling, especially Min-Max, ensures consistent feature impact on algorithms while retaining inherent data distributions.

2.4. Correlation Analysis

Correlation analysis reveals the linear relationship between quantitative variables, helping pinpoint key features influencing the target variable, vital for predictive modeling.



Approach:

1. Numeric Columns Selection: Using `select_dtypes`, only 'float64' and 'int64' data types were considered, ensuring relevant correlations.

2. Heatmap Visualization: The correlation matrix was visualized with a heatmap. Dark to bright colors indicate weak to strong correlations, with `annot=True` displaying exact coefficients.3. Top Features Correlated with 'streams': Absolute correlation values with 'streams' were sorted to identify strongly related features.

Observations:

- Strong Correlations: 'in_spotify_playlists' and 'in_apple_playlists' had high positive correlations with 'streams', suggesting songs on more playlists achieve higher streams.

- Medium Correlations: Chart appearances on platforms like 'in_apple_charts', 'in_spotify_charts', and 'in_deezer_charts' moderately correlate with 'streams'.

- Other Observations: Release timing, like 'released_year', 'released_day', and 'released_month', showed some correlation with 'streams', suggesting potential influence on streaming success.

- Lower Correlations: Attributes like 'danceability_%', 'speechiness_%', and 'energy_%' had minimal correlation with 'streams', indicating a lesser impact on streaming success.

Conclusion:

Playlist and chart appearances strongly influence streaming numbers, while some features play minor roles. This understanding is pivotal for feature selection in predictive modeling.

3. MACHINE LEARNING ALGORITHM IMPLEMENTATION

In this section, we explore predictive analytics by implementing machine learning algorithms. Through this, we aim to harness patterns in our data to build predictive models that can make informed decisions related to our dataset.

3.1 Algorithm Choice:

Linear Regression: A fundamental choice for understanding linear relationships between features and the target. Initially, without regularization, the model had higher error, hinting at overfitting.

3.2 Cost Function & Optimization:

MSE with L2 Regularization & Gradient Descent: To combat overfitting, L2 (Ridge) regularization was introduced, making the model less sensitive to individual data fluctuations. The combined cost function of Mean Squared Error and L2 was minimized using Gradient Descent.

Ordinary Least Squares (OLS): An initial attempt using OLS yielded a large error, suggesting it might not be apt for this dataset or its assumptions weren't met.

```
Iteration 0, Loss: 0.7543118265650477
Iteration 100, Loss: 0.0014014953510547363
Iteration 200, Loss: 0.001039002170445439
Iteration 300, Loss: 0.0007702669219544011
Iteration 400, Loss: 0.0005710393567346875
Iteration 500, Loss: 0.00042334149065181934
Iteration 600, Loss: 0.0003138452990913059
Iteration 700, Loss: 0.00023267001684633437
Iteration 800, Loss: 0.00017249051330708024
Iteration 900, Loss: 0.0001278762841221188
Iteration 1000, Loss: 9.480141097249343e-05
Iteration 1100, Loss: 7.028126899428036e-05
Iteration 1200, Loss: 5.210319889520981e-05
Iteration 1300, Loss: 3.862684003805803e-05
Iteration 1400, Loss: 2.8636106860281445e-05
Iteration 1500, Loss: 2.1229451213340382e-05
Iteration 1600, Loss: 1.5738508066706373e-05
Iteration 1700, Loss: 1.1667783292020696e-05
Iteration 1800, Loss: 8.64994104730589e-06
Iteration 1900, Loss: 6.412655964653931e-06
```

Conclusion:

The Linear Regression model, refined with L2 regularization and optimized with Gradient Descent, balanced simplicity with predictive strength, making it a sound choice for this data.

4. MODEL EVALUATION

Evaluating the performance of our machine learning model is essential to ensure its accuracy and reliability when faced with new data.

4.1 Dataset Split:

We split our data into 80% for training and 20% for testing, providing a real-world evaluation environment. The predictors include playlist inclusions and musical attributes, with 'streams' as the target. Using `numpy`, we randomized and split the data.

4.2 Training the Model:

We employed a Linear Regression model enhanced with L2 regularization (Ridge Regression) to prevent overfitting. The model "learns" by using Gradient Descent to minimize prediction errors.

4.3 Performance Evaluation:

To assess the model's accuracy on the testing set, several metrics were used:

- MSE: Calculates average squared difference between predictions and actuals.
- RMSE: Provides a unit-consistent view of the error.
- MAE: Offers an average absolute difference, not overly punishing large errors.
- R^2 score: Measures how well the model explains the variance in the target.

```
Train Data:
MSE: 0.0035953659845403977
RMSE: 0.059961370769357814
MAE: 0.03755567309190075
R^2 Score: 0.38523338081431435

Test Data:
MSE: 0.0014893011927136924
RMSE: 0.03859146528332
MAE: 0.03173903608482878
R^2 Score: 0.5588776464621223
```

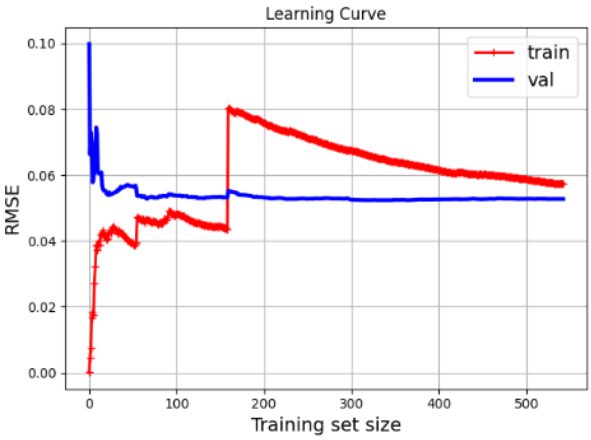
Overall, these metrics give a holistic view of the model's capacity to predict song streams.

5. ANALYSIS: Unveiling the Insights

Using a data-driven approach is not just about applying advanced models but translating intricate results into actionable strategies. In the Analysis section, we interpret the results to uncover patterns and relationships that can guide future strategies.

5.1. Bias-Variance Trade-off:

Bias-variance is central to understanding model performance.



Key Insights:

- With a small training set, the model overfits, evident from the gap between training and validation errors.

- As training data increases, the model generalizes better.

Approach:

1. Start with a smaller data subset.

2. Track model performance as data increases.

3. Identify where both errors converge, suggesting optimal model complexity.

5.2 Model Selection:

Balancing model complexity is essential: too complex risks overfitting, too simple may underfit.

Key Insights:

- The Linear Regression with L2 regularization was selected after an OLS model attempt.

- The model's R^2 score of 0.4464 means it explains about 44.64% of the target variability, a notable achievement considering the complexities influencing stream counts.

Approach:

1. Start with a basic model (OLS).

2. Incrementally adjust complexity, like introducing L2 regularization.

3. Compare different models and finalize the one with the best balance.

4. Ensure the model performs well on unseen data.

6. CHALLENGES & TRAINING

Challenges:

1. Data Understanding: Determining significant predictors for 'streams' due to the multifaceted nature of the music streaming domain.

2. Model Complexity: Addressing overfitting where models excelled on training data but faltered on new data.

3. Bias-Variance Trade-off: Balancing bias and variance, evident when examining learning curves.

4. Parameter Tuning: Iterative fine-tuning of hyperparameters like regularization strength.

5. Model Evaluation: Deciphering which metric (MSE, RMSE, MAE, R^2) was most crucial for this specific problem.

6. Algorithm Selection: Implementing gradient descent-based linear regression and monitoring for convergence.

Learnings:

1. Regularization's Role: L2 regularization effectively controlled model complexity and curbed overfitting.

2. Visualization's Impact: The learning curve visually represented model behavior across training sizes.

3. Iterative Nature: Machine learning requires multiple iterations for model refinement.

4. Linear Regression Insights: Gained practical understanding of linear regression's intricacies.

5. Holistic Evaluation: Using multiple metrics provided a rounded assessment of performance.

6. Unseen Data's Importance: Validating unseen data underscored the model's generalizability.

7. CONCLUSION

Main Findings:

1. Data & Features: Key predictors for 'streams' included 'in_spotify_playlists', 'in_apple_playlists', and 'bpm'.

2. Model: Linear regression with L2 regularization was selected.

3. Bias-Variance Insight: The learning curve offered insights on model generalization.

Performance:

Evaluation Metrics:

Train		Data:
MSE:		0.0035953659845403977
RMSE:		0.059961370769357814
MAE:		0.03755567309190075
R^2	Score:	0.38523338081431435

Test		Data:
MSE:		0.0014893011927136924
RMSE:		0.03859146528332
MAE:		0.03173903608482878

R^2 Score: 0.5588776464621223

Suggestions:

1. Feature Engineering: Explore feature interactions for intricate patterns.

2. Model Exploration: Test advanced models like Random Forests or Neural Networks.

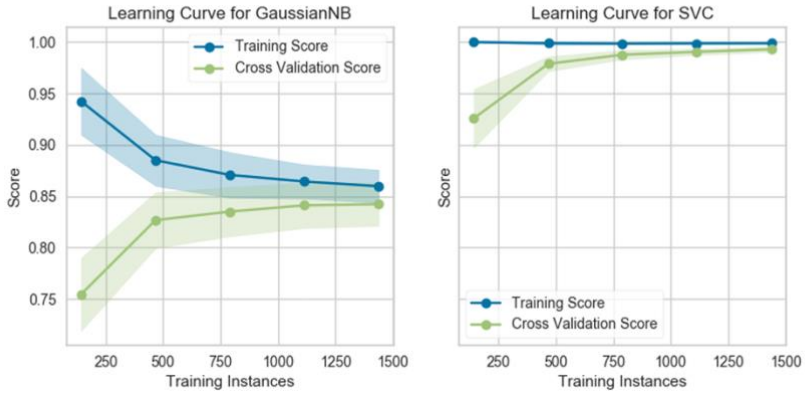
3. Hyperparameter Tuning: Utilize tools like Grid Search for optimization.

4. Ensemble Approaches: Combine predictions from different models for enhanced performance.

5. Data Augmentation: Source more data to enhance generalization.

6. Residual Analysis: Examine residuals to pinpoint model weaknesses.

7. Feedback Integration: Use expert feedback for model refinement.



QUESTION 2

Bias-Variance Tradeoff

A. Based on the provided graph, we can analyze the bias-variance tradeoff for each model:

GaussianNB:

- The training score starts high, around 0.95, and gradually decreases as the dataset size increases.
- The cross-validation score starts much lower, around 0.78, and slowly rises as the dataset size increases.
- The two curves are getting closer to each other but still remain fairly distant even at 1500 training instances.

SVC:
- The training score is consistently high, remaining close to 1.0 regardless of the dataset size.
- The cross-validation score starts at approximately 0.8 and steadily increases, getting very close to the training score as the dataset size increases.

For the optimal balance between bias and variance:
- GaussianNB: This model doesn't seem to achieve an optimal balance in the provided range. The training and cross-validation scores are still relatively far apart even at 1500 training instances. However, the point where the distance between the two curves is the smallest (which could be considered as approaching the optimal balance) seems to be around 1500 instances. A larger dataset might further improve the balance.

- SVC: The optimal balance is achieved faster, around the 1250 training instances mark. At this point, the cross-validation score is very close to the training score, suggesting that the model has a good tradeoff between bias and variance.

In summary, for the GaussianNB, the optimal balance might be achieved with a larger dataset beyond 1500 instances. For the SVC, it's around 1250 training instances.

B. GaussianNB:

a. Small Dataset Size (e.g., 250 data points):
- Training Score: Close to 0.95

- Cross Validation Score: Around 0.78

The large gap between the training score and the cross-validation score indicates high variance (overfitting) because the model performs very well on the training data but poorly on the validation data.

b. Large Dataset Size (e.g., 1000+ data points):
- Training Score: Close to 0.90
- Cross Validation Score: Around 0.83

The gap between the training score and the cross-validation score reduces as the dataset size increases. However, the model still has a noticeable gap, indicating it's still in the high variance regime, but it's moving towards a better bias-variance tradeoff.

SVC:
a. Small Dataset Size (e.g., 250 data points):
- Training Score: Close to 1.0
- Cross Validation Score: Around 0.80

The very high training score compared to the cross-validation score indicates high variance (overfitting). The model is almost perfectly fitting the training data but not performing as well on the validation data.

b. Large Dataset Size (e.g., 1000+ data points):
- Training Score: Close to 1.0
- Cross Validation Score: Close to 0.95
As the dataset size increases, the cross-validation score improves significantly, coming close to the training score. This suggests the model is nearing the optimal regime, where it has a good balance between bias and variance.

Summary:
- GaussianNB:
  a. Small Dataset Size: High variance
  b. Large Dataset Size: High variance (but moving towards optimal)
- SVC:
  a. Small Dataset Size: High variance
  b. Large Dataset Size: Optimal

C. Modifying a model's complexity based on its performance:

1. High Bias:

- Model too simple, not fitting data well.

- Solutions: Use complex models, add/engineer features, reduce regularization, or longer training.

2. High Variance:

- Model too complex, overfits training data.

- Solutions: Simplify model, reduce features, increase regularization, prune trees, get more data, use dropout (for neural nets), or use ensemble methods.

In essence, diagnose model issues (bias or variance) and adjust complexity, features, or regularization accordingly.

D. Analysis based on the provided graph:

1. GaussianNB: The training score drops, and the validation score rises with more data, with both converging slowly. Further data may give minor improvements.

2. SVC: High and stable training score. The validation score rises closer to the training score with more data. More data might offer notable benefits.

In essence, both models may benefit from more data, but SVC shows a stronger potential for improvement

E. Underfitting in a learning curve:

For an underfitting model (or a model with high bias), the training and validation scores would be relatively close to each other but both scores would be low, indicating that the model is not capturing the underlying patterns of the data. This means the model performs poorly on both the training and validation datasets. The learning curve for such a scenario would show the training and validation scores converging to a suboptimal value as the number of training instances increases. Please find below the learning curve: