# Question 1: Machine Learning: Kaggle Most Streamed Spotify Songs

## Objective:

The objective of this report is to document the process and findings of implementing a machine learning algorithm from scratch for target variable Streams of Spotify songs using the "Most Streamed Spotify Songs 2023" dataset. The report aims to provide a comprehensive understanding of the steps taken, challenges encountered, and insights gained throughout the entire process

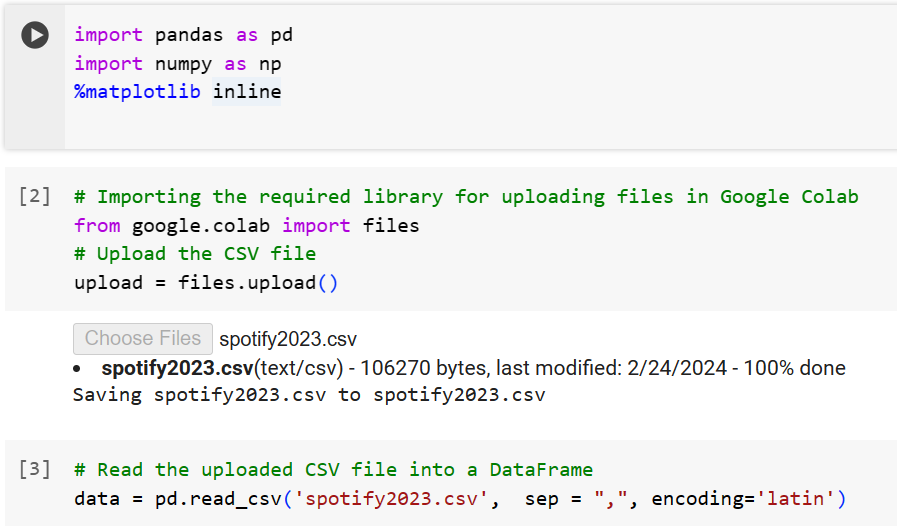
## Google colab link:

<https://colab.research.google.com/drive/10ppl2g_EwGzuzbsNzOUQ25Lrpua6xIDF?usp=sharing>

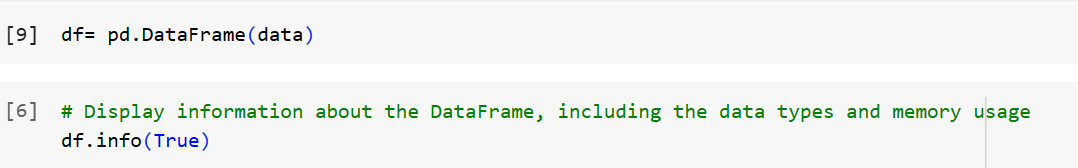
## Tasks:

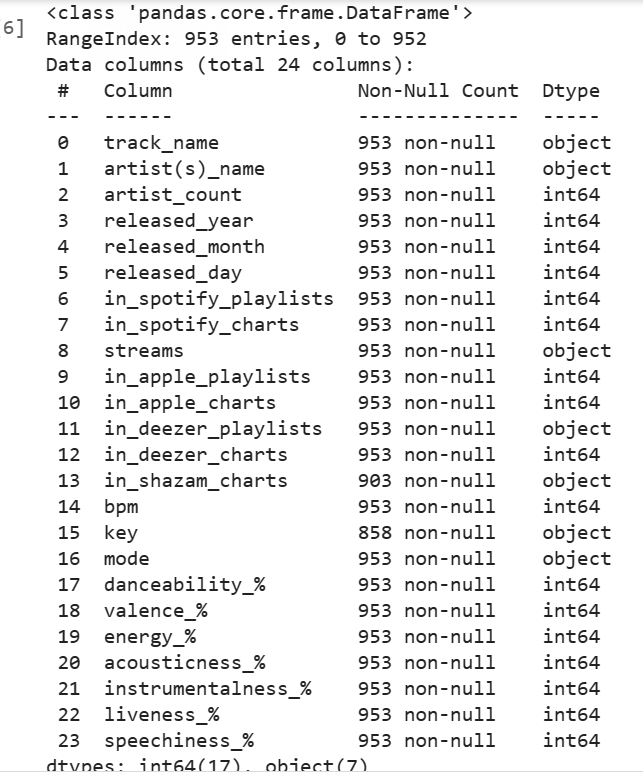
### Data preprocessing

#### 1.1 Loading Data into Google Colab:

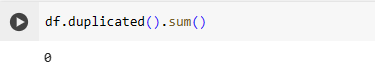


#### 1.2 Displaying and analyzing the dataset

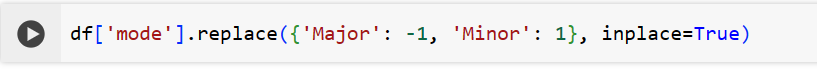




Checking duplicates



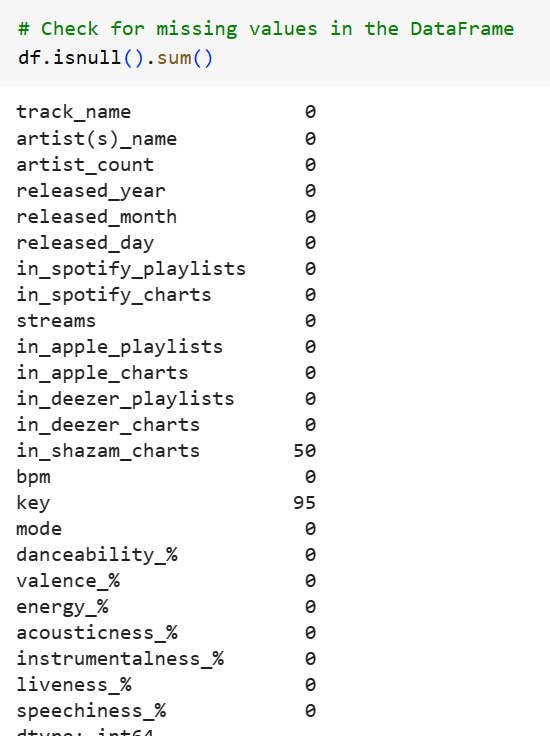
Replacing major and minor in mode column with 1 and -1, so that column is compatible with our model. Converting categorical data to numerical format can sometimes improve the performance of your machine learning models. Algorithms may find it easier to detect patterns and relationships in numerical data compared to categorical data.



#### 1.3 Missing Values

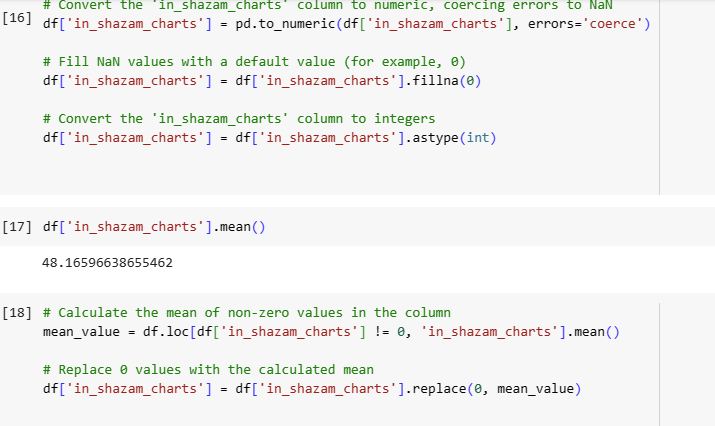
Detecting and properly handling missing values is a critical preprocessing step in machine learning pipelines. It ensures the quality and integrity of the data, enhances model performance, and facilitates reliable and unbiased analyses and predictions

Below we found out that there are missing values for in\_shazam\_charts and key

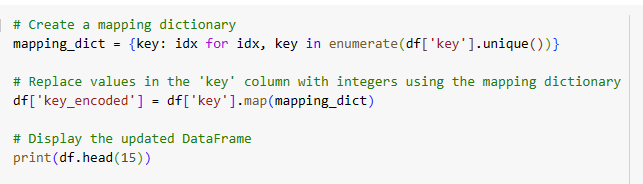


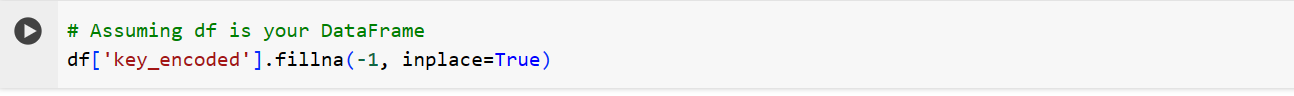
Dealing with Missing values

In\_shazam\_charts column:



Missing values in Key column:





#### 1.4 Detecting and Deleting incorrect value for ‘streams’ column, as streams is our target variable

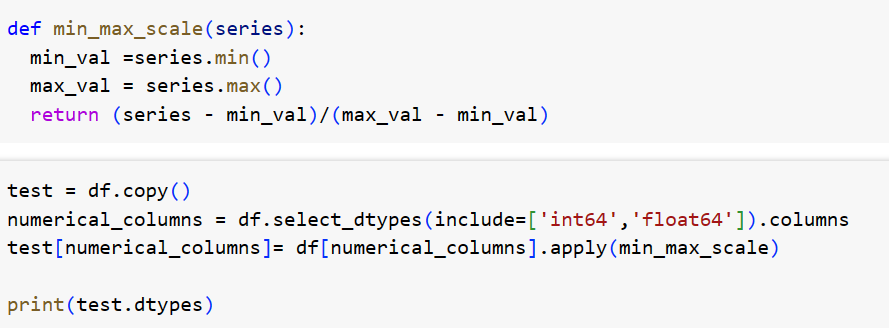
Incorrect values can introduce bias and errors into your analysis, leading to misleading conclusions and decisions. Deleting incorrect values helps mitigate bias and errors, ensuring that our machine learning models provide unbiased and accurate insights.



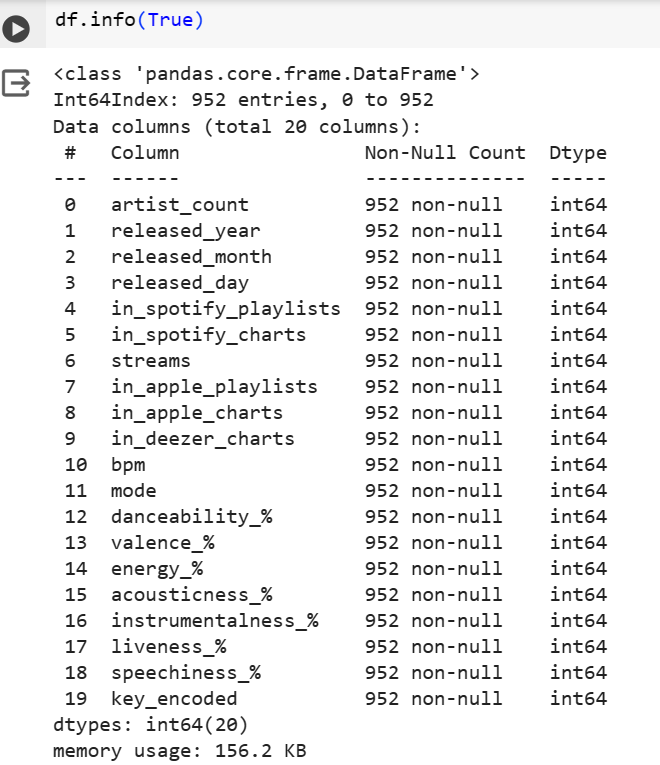
#### 1.5 Scaling

Scaling involves transforming the values of features so that they fall within a similar scale. This prevents features with larger scales from dominating those with smaller scales during the learning process. When using gradient descent-based optimization algorithms, having features on a similar scale can help the algorithm converge more efficiently. Min-Max scaling ensures that the gradients for different features are on a similar scale, leading to smoother and faster convergence.

Using Min\_Max scaling:

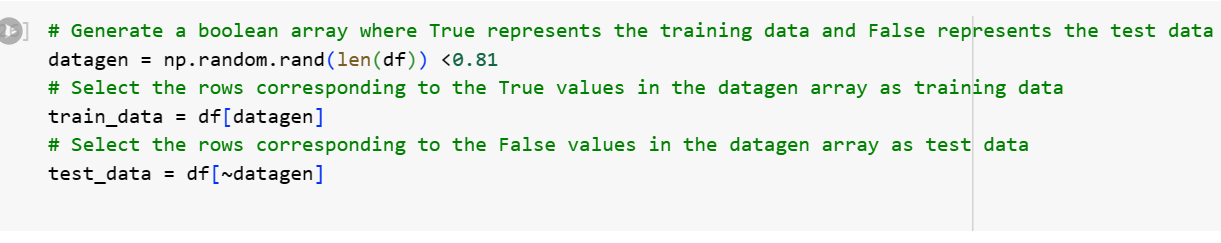


After dropping columns that are not required we will be working on the below columns for modeling



### Splitting data into Training and Test Set

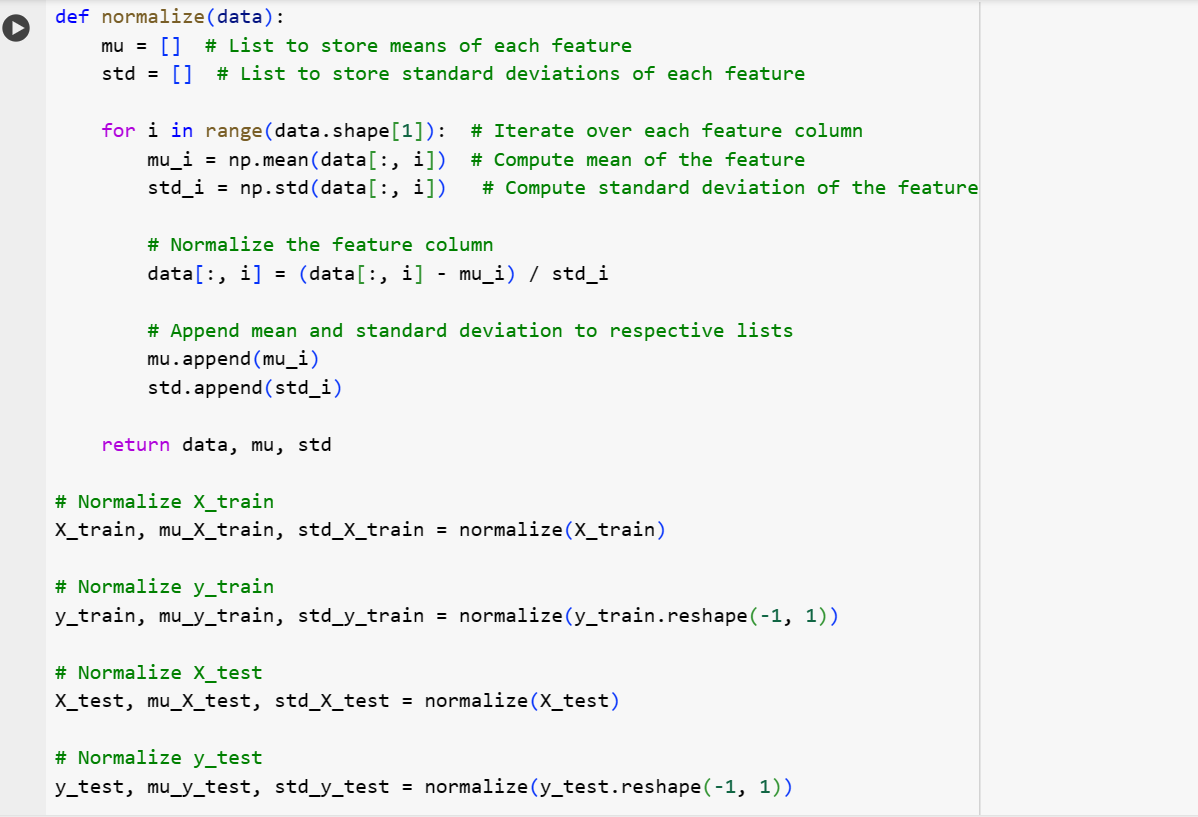
By splitting the dataset into separate training and test sets, we can evaluate the performance of the model on unseen data. The test set serves as a proxy for new, unseen data, allowing us to assess how well the model generalizes to unseen instances.





### Data Normalization

Data normalization is a crucial preprocessing step in machine learning that helps improve algorithm convergence, performance, interpretability, and generalization, while also preventing data leakage and mitigating the effects of skewed distributions. Normalizing the data can improve the performance of machine learning models, particularly for algorithms that rely on distance measures or gradient descent optimization. Models trained on normalized data often generalize better to new, unseen data, leading to more accurate predictions.



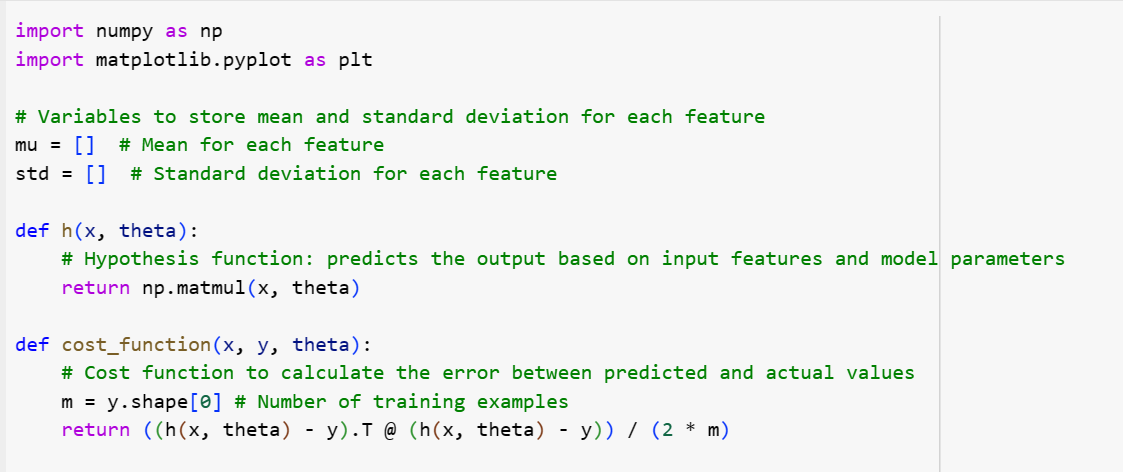
### Model Implementation: Linear Regression Model

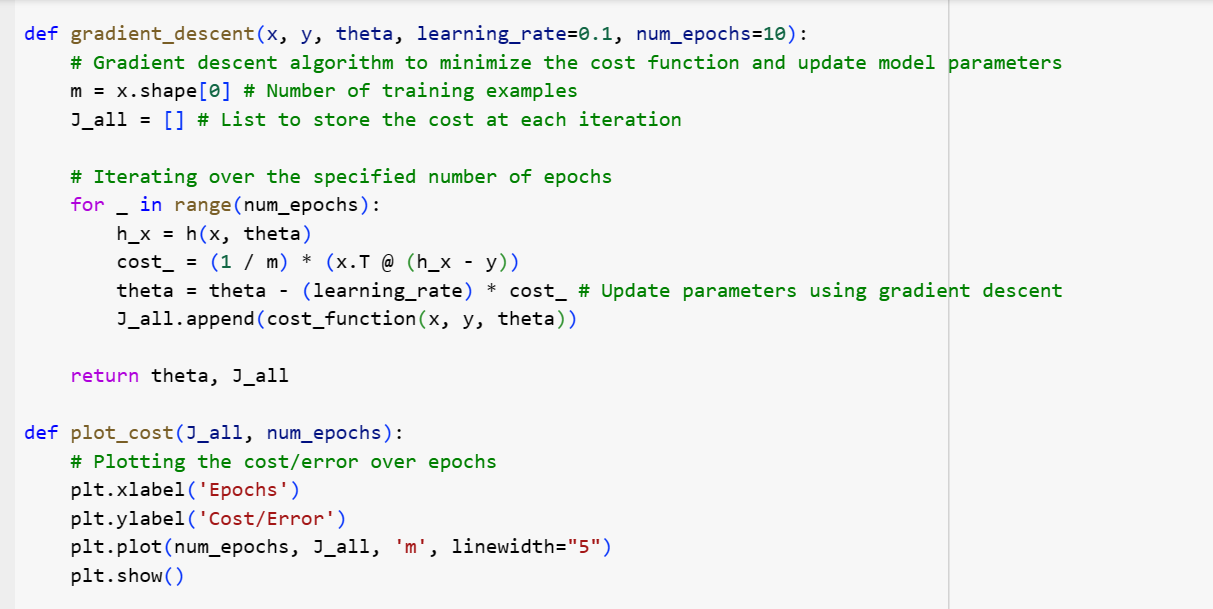
Initializing Parameters: In the first code block, the parameters (theta) of the model are initialized to zeros. These parameters represent the coefficients of the linear regression model.

Cost Function: The cost function measures how well the model's predictions match the actual target values. In linear regression, a common cost function is the mean squared error (MSE), which calculates the average squared difference between predicted and actual values.

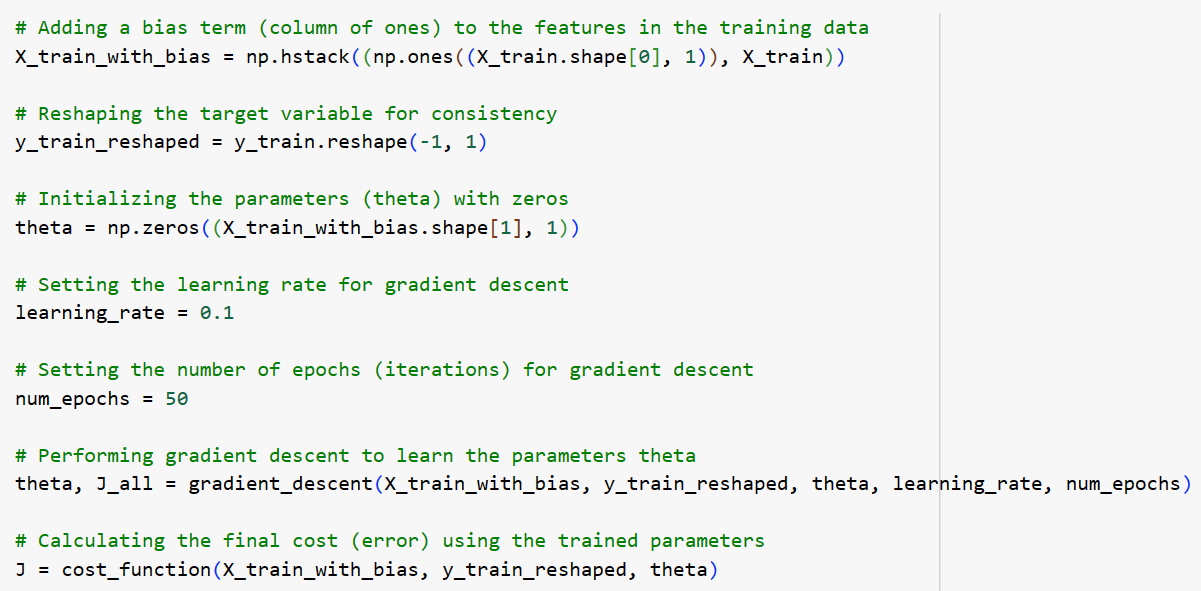
Gradient Descent Iterations: Gradient descent is an iterative optimization algorithm. It starts with initial parameter values and updates them iteratively to minimize the cost function. At each iteration, the gradients of the cost function with respect to the parameters are computed. These gradients indicate the direction and magnitude of the steepest ascent of the cost function.

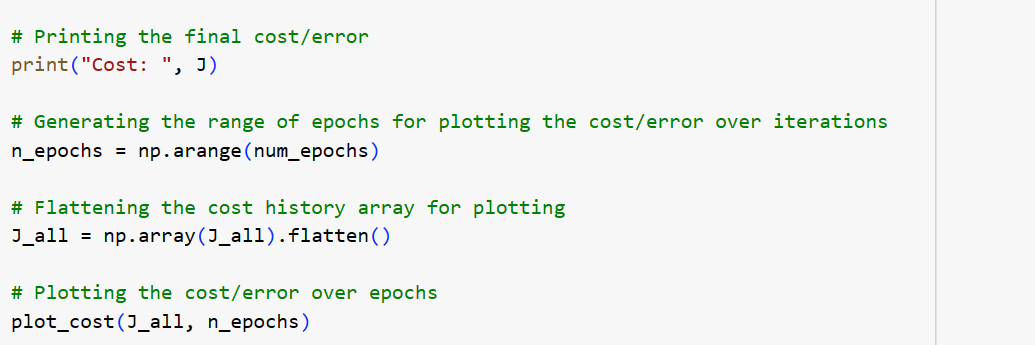
Parameter Update: The parameters (theta) are updated in the direction that reduces the cost function. This update is performed by subtracting a fraction of the gradient multiplied by a learning rate from the current parameter values. The learning rate controls the size of the steps taken during parameter updates.



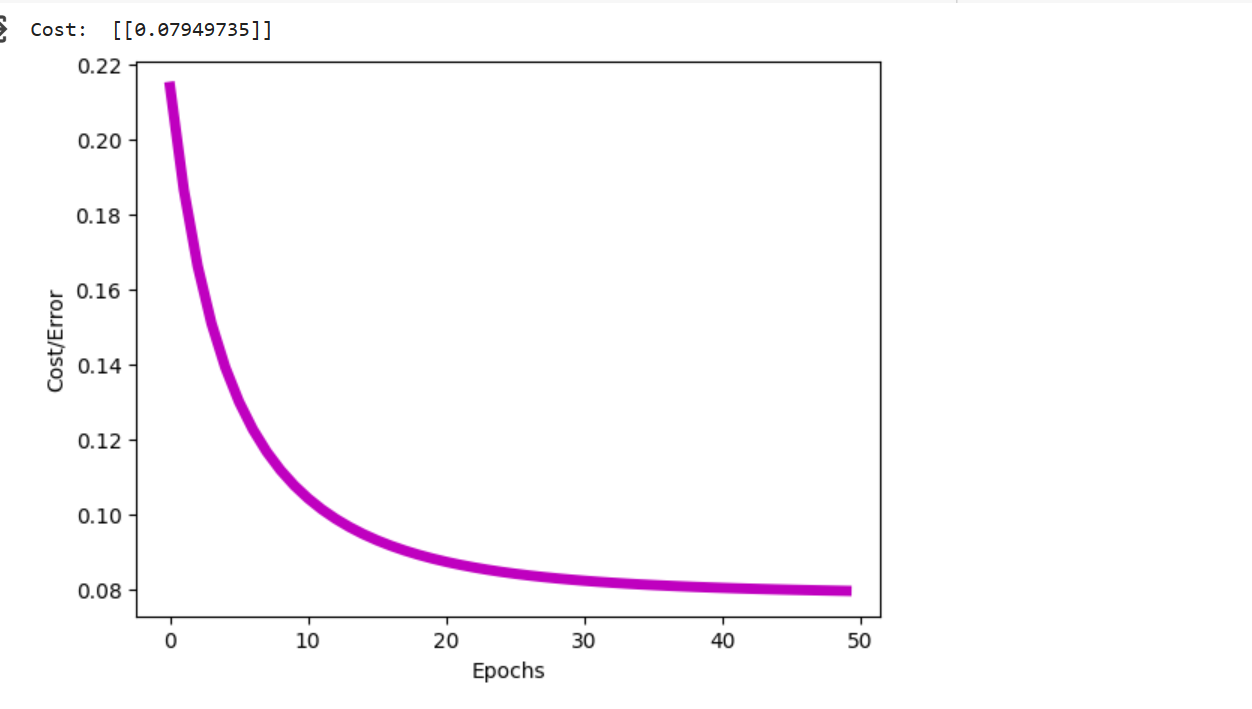


After implementing gradient descent, the model is trained on the training data to learn the optimal parameters. Additionally, the cost function is plotted over iterations to visualize the training progress.



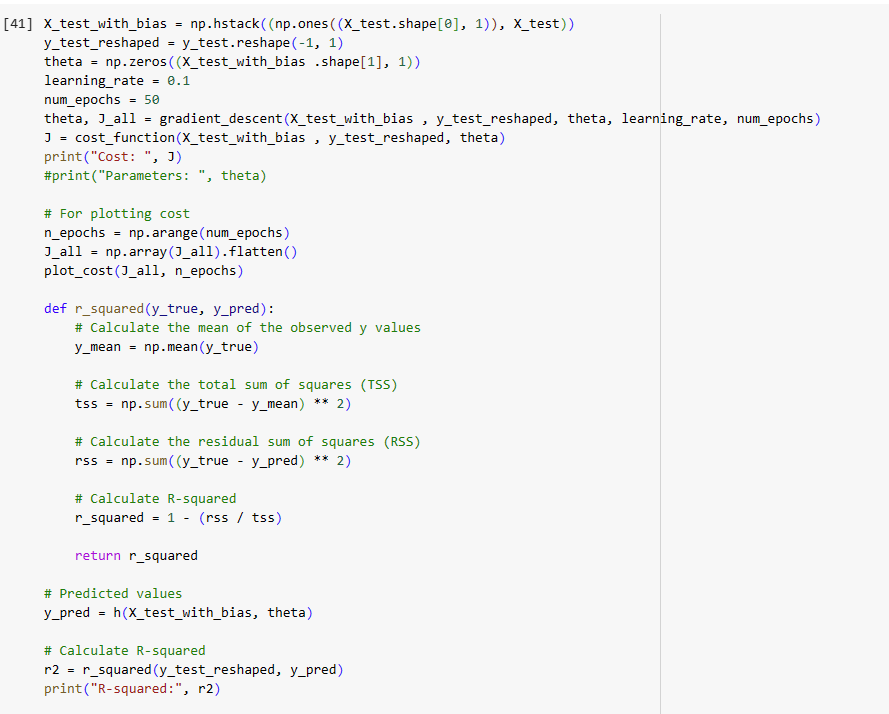


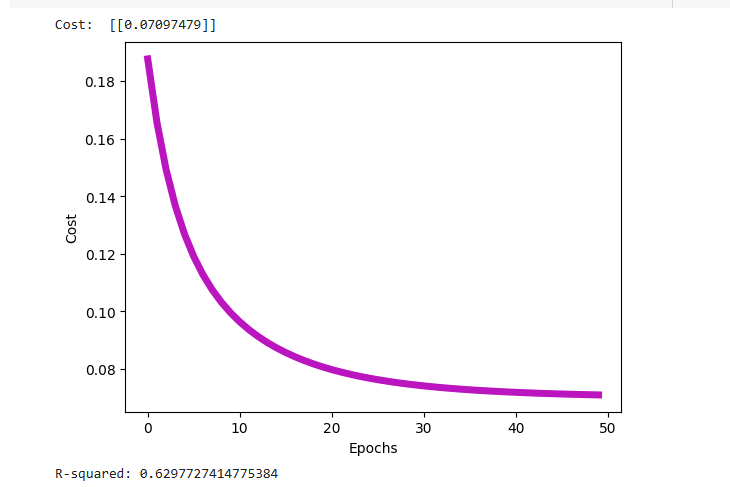
Output: The cost value of approximately 0.07949735 indicates the average squared difference between the predicted stream counts of Spotify songs and the actual stream counts in the training dataset. A lower cost value suggests that the model has achieved a better fit to the training data. Below is the visualization of how error is decreasing with time and epochs.



### Model Evaluation on testing data: R-squared (coefficient of determination)

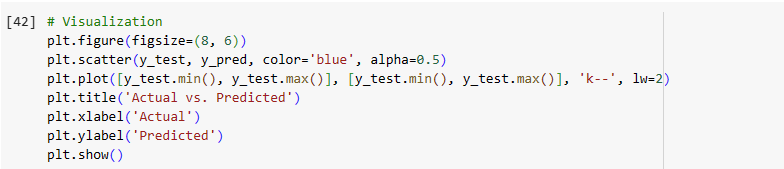
The R-squared (coefficient of determination) is a statistical measure that represents the proportion of the variance in the dependent variable (target) that is predictable from the independent variables (features) in a regression model. It is a measure of how well the model explains the variability of the target variable. It ranges from 0 to 1, where a value closer to 1 indicates that the model explains a larger proportion of the variability in the target variable. A value of 0 indicates that the model does not explain any of the variability.



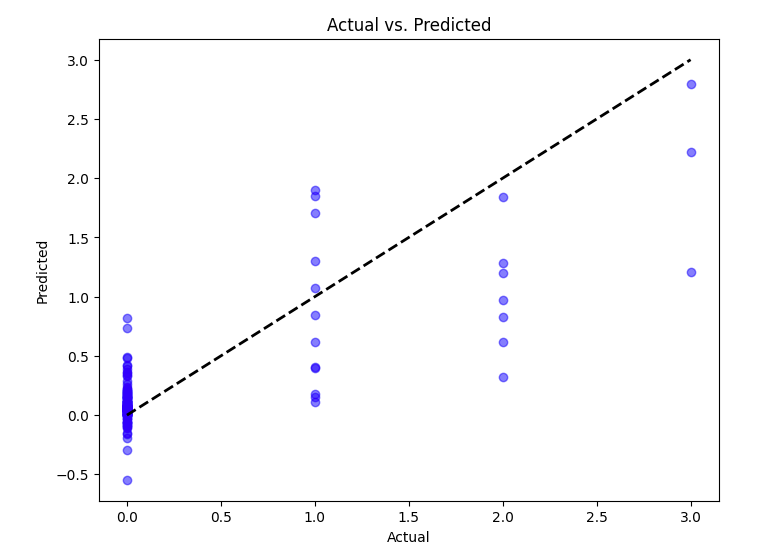


R-squared value of 0.62977 suggests that the model is moderately effective at predicting the stream counts of Spotify songs based on the features provided.

**Scatter plot for visualizing actual versus predicted values:**

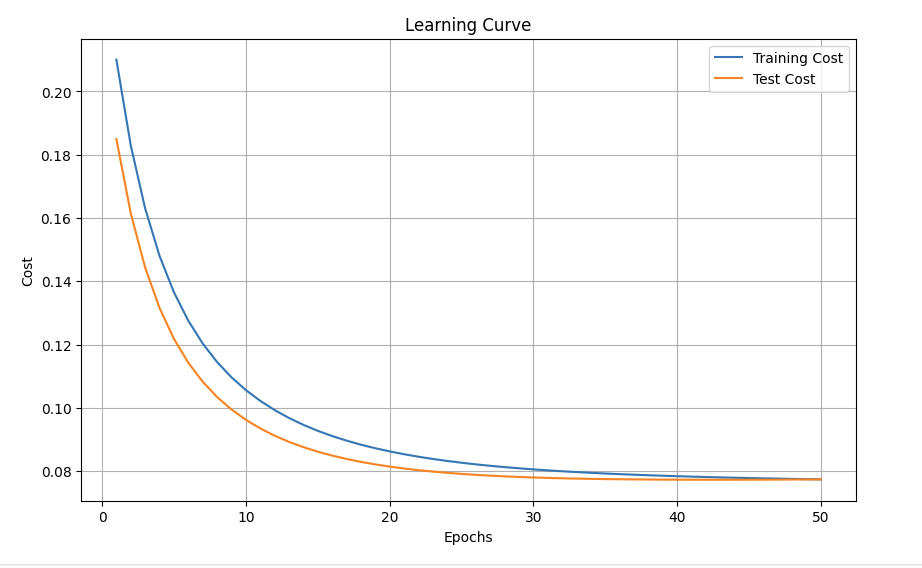


The diagonal line in the graph represents perfect predictions where the actual and predicted values are the same.



**Visualize Learning cure:**

We have plotted the model's performance on both the training set and the validation set as a function of the number of training samples or training iterations. This helps us understand how the model's performance improves as it is trained on more data or for more iterations.



## Performance Analysis

**Bias-Variance Trade-off:**

Bias: The training and test R-squared values (0.647 and 0.630, respectively) indicate that the model captures a significant portion of the variance in the data, suggesting low bias.

Variance: The slight decrease in R-squared from the training to the test set (0.017) suggests that the model might have slightly higher variance. However, the difference is not substantial, indicating reasonable variance control.

**Model Selection:**

The selected model is linear regression trained using gradient descent. This model choice is reasonable given the simplicity of the dataset and the linear relationship between features and the target variable.

Model complexity appears appropriate for the problem at hand, as evidenced by decent performance metrics.

**Performance Evaluation:**

Training Data: The training cost (0.077) indicates a relatively low error on the training set, suggesting that the model fits the training data reasonably well.

Test Data: The test cost (0.071) also indicates low error on unseen data, indicating that the model generalizes adequately to new observations.

**Analysis for Overfitting and Underfitting:**

Overfitting: There is no significant indication of overfitting as the performance metrics on the test data are close to those on the training data. This suggests that the model is not capturing noise in the training data.

Underfitting: The R-squared values on both training and test data are relatively high, indicating that the model is not overly simple and can capture the underlying relationships in the data.

Generalization:

The similar performance metrics on both training and test data suggest that the model generalizes well to unseen data.

Overall, the selected model demonstrates a good balance between bias and variance, indicating that it adequately captures the underlying patterns in the data without overfitting or underfitting. The analysis suggests that the model is suitable for generalization to new data, as evidenced by consistent performance on both training and test datasets.

## Challenges

1. No ML Libraries Allowed

**Context**: The restriction on using ML libraries forced me to rely solely on fundamental concepts.

**Impact**: Building everything from scratch required more effort and attention to detail.

**Solution**: I implemented gradient descent, cost functions, and model evaluation manually.

2. Feature Normalization

**Context**: The dataset contained features with varying scales (loudness, popularity, tempo).

**Challenge**: Not normalizing the data led to several issues:

**Magnitude Differences**: Features with large magnitudes dominated the model.

**Slow Convergence:** Gradient descent took longer due to varying scales.

**Interpretability:** Coefficients became hard to interpret.

**Regularization Sensitivity:** Regularization terms were affected by feature scales.

**Solution**: I applied Min-Max scaling (normalization) to mitigate these challenges.

## Learnings

Deepened Understanding: Implementing ML algorithms from scratch provided valuable insights into how models work under the hood.

Trade-offs: Choosing between Min-Max scaling and standardization required thoughtful consideration.

Appreciation for Libraries: While challenging, this experience made me appreciate the convenience of ML libraries.

# Question 2: Bias-Variance Tradeoff

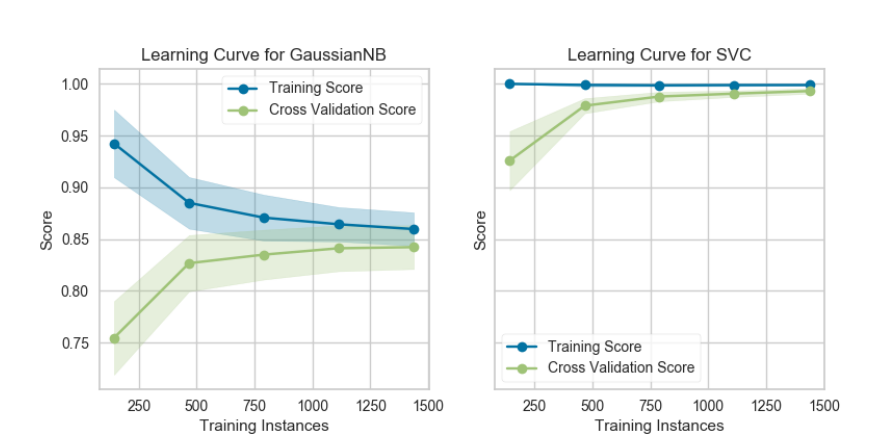
Consider the following scenario: You have trained two machine learning models

to perform a classification task. The performance of each model has been

evaluated on both the training dataset and an unseen test dataset (Cross

Validation Score). The following graph represents the model's performance as

the dataset size increases:



Based on the provided graph, please answer the following questions:

**A. At which dataset size (approximately) does each model seem to achieve**

**the optimal balance between bias and variance? Please justify your**

**answer.**

Ans.

**GaussianNB Model:**

The GaussianNB model’s Training Score starts high but decreases as the number of training instances increases.

The Cross Validation Score starts low but increases with more training instances until it stabilizes around a score of approximately 0.85.

The optimal balance between bias and variance occurs when the training and cross-validation scores are close, indicating that the model is neither overfitting nor underfitting.

Based on the graph, this balance seems to be achieved at around 750 training instances.

**SVC Model (Support Vector Classifier):**

For the SVC model, both the Training Score and Cross Validation Score start high and remain stable across different numbers of training instances.

The optimal balance between bias and variance appears to be achieved at around 1000 training instances for the SVC model.

**B. In which regime (high bias, high variance, or optimal) are each model**

**operating at the following dataset sizes:**

**a. Small dataset size (e.g., 250 data points)**

**b. Large dataset size (e.g., 1000+ data point**

Ans.

1. **Small Dataset Size (250 data points):**

For both models:

The training score is relatively high.

The cross-validation score is relatively low.

Regime: High Bias

Explanation: With a small dataset, the models are not able to learn complex patterns. They are overly simplified and generalize poorly to unseen data. This results in high bias (underfitting).

1. **Large Dataset Size (1000+ data points):**

For both models:

The training score remains stable.

The cross-validation score also remains stable and close to the training score.

Regime: Optimal

Explanation: As the dataset size increases, the models have more data to learn from. They achieve a good balance between bias and variance. The training and cross-validation scores are close, indicating that the models generalize well to unseen data.

**C . How would you modify the model's complexity to improve its performance,**

**if it is operating in the high bias regime? Conversely, what would you do if it**

**is operating in the high variance regime?**

**High Bias Regime:**

When dealing with high bias (underfitting), follow these steps:

1 Increase Model Complexity:

Switch to a more complex model (e.g., polynomial regression).

Add relevant features or perform feature engineering.

Adjust hyperparameters (e.g., polynomial degree).

Consider ensemble methods (e.g., random forests, gradient boosting).

2 Apply Regularization Techniques:

Use L1 (Lasso) or L2 (Ridge) regularization.

These techniques prevent overfitting by penalizing large coefficients.

3 Evaluate Bias-Variance Trade-off:

Monitor learning curves and cross-validation scores.

Aim for the right balance between bias and variance.

**High Variance Regime:**

To combat high variance (overfitting), follow these steps:

1 Decrease Model Complexity:

Switch to a simpler model (e.g., linear regression).

Remove irrelevant features.

Tune hyperparameters (e.g., decision tree depth).

Consider bagging (e.g., random forests) or dropout (for neural networks).

2 Cross-Validation and Regularization:

Use k-fold cross-validation.

Apply regularization techniques (L1, L2) to prevent overfitting.

3 Collect More Data:

If possible, gather additional data for better generalization.

**D Do you expect adding more data to improve the performance for each**

**model? Elaborate on your response.**

Ans.

**GaussianNB Model:**

Expectation: Adding more data is likely to improve the performance of the GaussianNB model.

Elaboration:

Let’s break down the essence of GaussianNB:

1 Probabilistic Model:

GaussianNB is based on Bayes’ theorem.

It calculates probabilities for different class labels given the observed features.

2 Conditional Independence Assumption:

It assumes that features are independent of each other, given the class label.

This simplifies the model but may not always hold in real-world scenarios.

3 Benefit of More Data:

With more data, GaussianNB can better estimate probabilities.

It learns underlying patterns more accurately.

4 Robustness and Generalization:

As the dataset size increases, the model becomes more robust.

It generalizes better to unseen examples.

5 Diminishing Returns:

However, beyond a certain point, additional data may not significantly improve performance.

Poor data quality or non-truly independent features can limit gains.

**SVC Model (Support Vector Classifier):**

Expectation: Adding more data may have limited impact on the performance of the SVC model.

Elaboration:

1 Optimal Hyperplane:

SVC aims to find the best hyperplane that separates different classes in the data.

This hyperplane maximizes the margin between classes.

2 Support Vectors:

In high-dimensional spaces, SVC relies on support vectors.

These are data points near the decision boundary.

They play a crucial role in defining the hyperplane.

3 Impact of Data Size:

If the data is already well-separated and support vectors are well-defined, adding more data may not significantly change the decision boundary.

However, if the data is noisy or overlapping, additional data could refine the boundary.

4 Regularization and Kernel Choice:

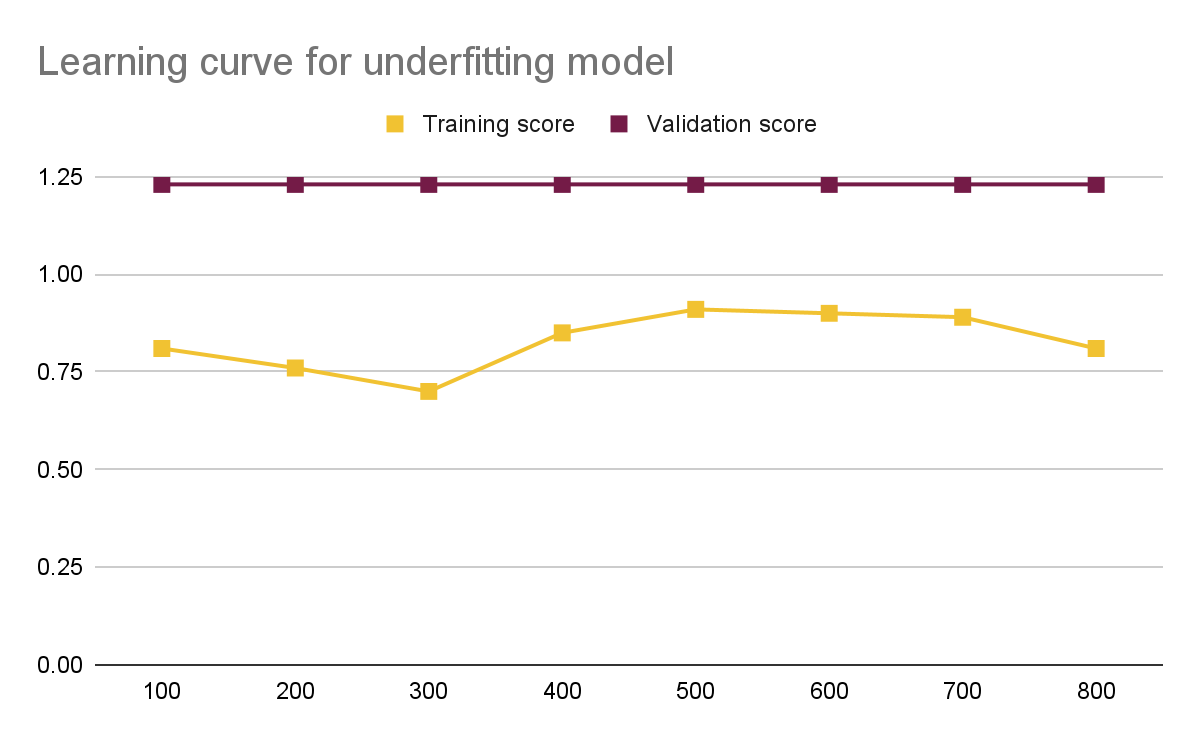
Regularization (C parameter) influences the trade-off between fitting the training data and generalizing to unseen data.

Kernel choice (linear, polynomial, radial basis function) affects the model’s sensitivity to data size.

**E Plot a similar plot for a hypothetical binary classification such as above**

**where the model underfits. Draw the curves for both training and validation**

**scores as a function of the Training Instances size**

****

Training Instances

The Training Score starts high but decreases as the number of training instances increases.

The Validation Score remains low and does not improve significantly with more training instances.

The model is overly simplistic and fails to capture the underlying patterns in the data, resulting in high bias (underfitting).