CS6364 MACHINE LEARNING HW 1

Sagar Sheth

Question 1: Machine Learning from Scratch: Kaggle Most Streamed Spotify Songs 2023

Objective:

The objective of this homework is to implement a machine learning algorithm from scratch using the Most Streamed Spotify Songs 2023". You are required to use first principle functions and are allowed to use tools such as numpy and pandas for data manipulation, but not any machine learning packages such as sklearn, pytorch, tensorflow, etc.

Data Preprocessing & Univariate Analysis

- Importing the data from .csv to a dataframe using the pandas Library.
- Using the pd.dtypes function to understand what data type has been assigned to columns.

```
print('Column Names:')
raw_data.dtypes
```

```
Column Names:

track_name               object
artist(s)_name           object
artist_count              int64
released_year             int64
released_month            int64
released_day              int64
in_spotify_playlists      int64
in_spotify_charts         int64
streams                  object
in_apple_playlists        int64
in_apple_charts           int64
in_deezer_playlists      object
in_deezer_charts          int64
in_shazam_charts         object
bpm                       int64
key                      object
mode                     object
danceability_%            int64
valence_%                 int64
energy_%                  int64
acousticness_%            int64
instrumentalness_%        int64
liveness_%                int64
speechiness_%             int64
dtype: object
```

Figure 1 Output of pd.dtypes

- Some the columns have been assigned object instead of int64, such as streams, in_deezer_playlists & in_shazam_charts. Therefore there mush have been some garbage data/Null Values/Different Number formats.
- I went from columns to column, performed Univariate Analysis on each of them to assess if they have null value or garbage values.
- For Univariate Analysis I have done the following:
    - If the data is a continuous feature:
        - Get the mean, median and mode of the data.
        - Plot a Boxplot
        - Plot a histogram
        - For certain features such as in_x_chart I treated it as if were categorical variables and checked the spread of charted songs v/s songs that did not.
    - If the data is a categorical variable:
        - Plot a Histogram.

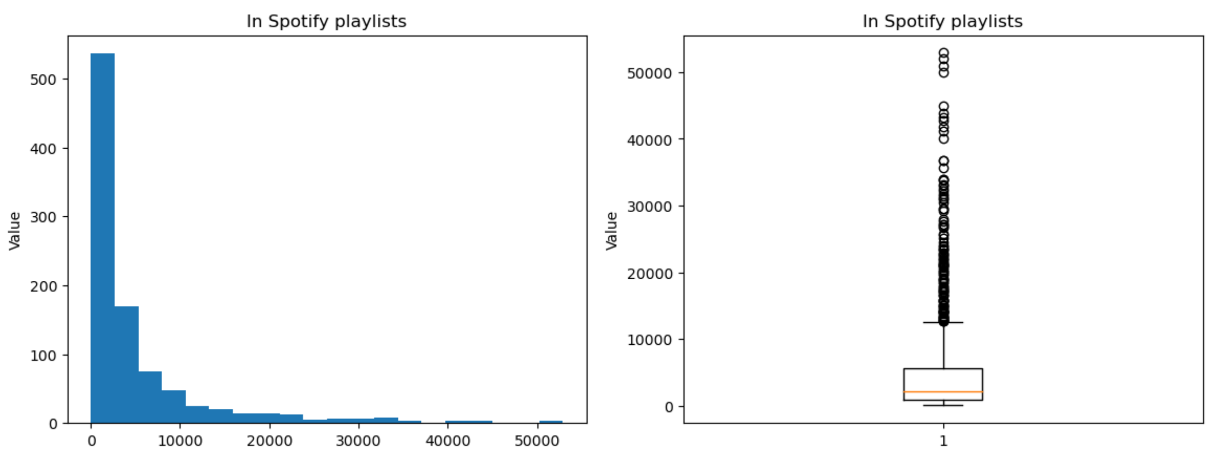


Figure 2 Univariate Analysis of in_spotify_playlists

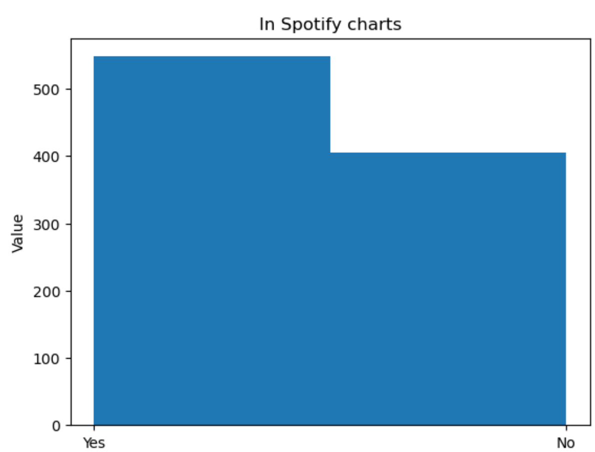


Figure 3 Univariate Analysis of in_spotify_charts

- For Streams & in_shazam_charts we convert it to numeric using pandas.

```
raw_data['streams'] = pd.to_numeric(raw_data['streams'], errors='coerce')
raw_data['in_shazam_charts'] = pd.to_numeric(raw_data['in_shazam_charts'], errors='coerce')
```

- For in_deezer_charts, we first remove the **','** and then convert it to numeric.

```
raw_data['in_deezer_playlists'] = raw_data['in_deezer_playlists'].str.replace(',', '')
raw_data['in_deezer_playlists'] = pd.to_numeric(raw_data['in_deezer_playlists'], errors='coerce')
```

- We create Hot Vectors for the categorical variables.

```
raw_data_2 = pd.get_dummies(raw_data, columns=['key', 'mode'], drop_first=True)
raw_data_2 = raw_data_2.drop(['artist(s)_name', 'track_name', 'released_year_cat'], axis=1)
```

- We drop in_shazam_charts as it has 50 null values and its correlation(calc below) with streams is very low.

Creating Derived Metrics

- We generate a correlation matrix/heat map to get features which have a correlation with streams.

```
corr_matrix = raw_data_2.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr_matrix, annot=True)
plt.show()

streams_corr = corr_matrix['streams']
high_corr_vars = streams_corr[streams_corr.abs() >0.1]
high_corr_vars = high_corr_vars[high_corr_vars.index != 'streams']
```

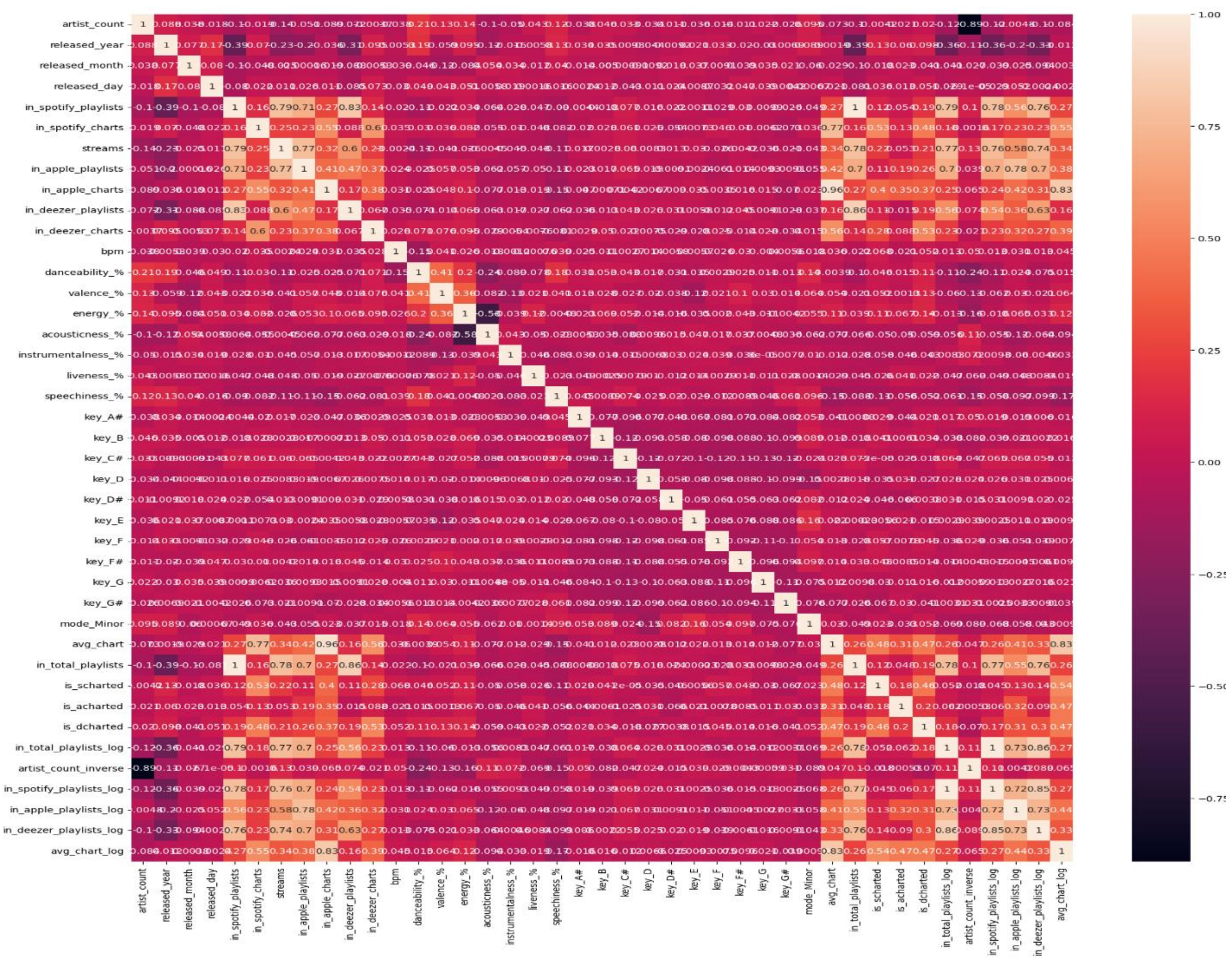


Figure 4 Heat Map/Correlation Matrix

- List of Features which absolute co-relation greater than 0.1 with streams
    - artist_count -0.136463
    - released_year -0.230803
    - in_spotify_playlists 0.789822
    - in_spotify_charts 0.245821
    - in_apple_playlists 0.772063
    - in_apple_charts 0.320234
    - in_deezer_playlists 0.598131
    - in_deezer_charts 0.228598
    - danceability_% -0.105457
    - speechiness_% -0.112333
    - in_total_playlists 0.783039
    - is_scharted 0.220106
    - is_dcharted 0.209187
- Creating the following derived metrics
    - avg_chart: takes the avg chart position of the song from spotify, apple and deezer.
    - in_total_playlists: takes the total of playlists the song is in from spotify, apple and deezer.
    - is_scharted: checks if the song is charted in spotify.
    - is_acharted: checks if the song is charted in apple.
    - is_dcharted: checks if the song is charted in deezer.

```
raw_data_2['avg_chart']=(raw_data_2['in_spotify_charts']+raw_data_2['in_apple_charts']+raw_data_2['in_deezer_charts'])/3
raw_data_2['in_total_playlists']=raw_data_2['in_spotify_playlists']+raw_data_2['in_apple_playlists']+raw_data_2['in_deezer_playlists']
raw_data_2['is_scharted'] = np.where(raw_data['in_spotify_charts'] != 0, 1, 0)
raw_data_2['is_acharted'] = np.where(raw_data['in_apple_charts'] != 0, 1, 0)
raw_data_2['is_dcharted'] = np.where(raw_data['in_deezer_charts'] != 0, 1, 0)
```

- Perform bivariate analysis using these features v/s streams.
- If the data follows a curved pattern, we perform the inverse transformation to make the pattern more linear.

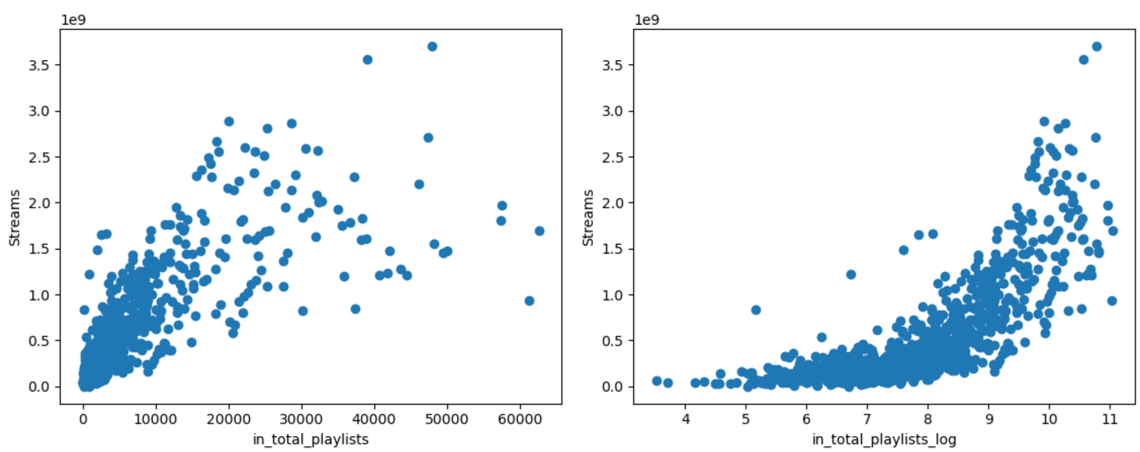


Figure 5 Transforming in_total_playlists using log

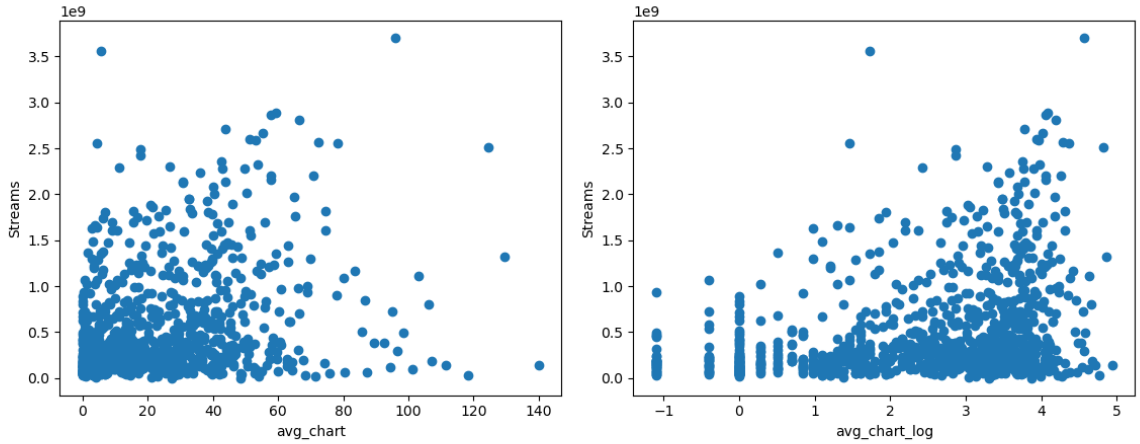


Figure 6 Transforming avg using log

- Correlation of the derived metrics v/s Streams.
  - avg_chart 0.335744
  - in_total_playlists 0.783039
  - is_scharted 0.220106
  - is_dcharted 0.209187
  - in_total_playlists_log 0.765984
  - artist_count_inverse 0.132196
  - in_spotify_playlists_log 0.758756
  - in_apple_playlists_log 0.584593
  - in_deezer_playlists_log 0.739196
  - avg_chart_log 0.340677

Data Processing Continued

- Using zscore from scipy.stats Library we check if there are any data points that outliers over the entire dataset. The output is 0 rows.

```
from scipy.stats import zscore

z_scores = zscore(raw_data_2)
abs_z_scores = np.abs(z_scores)

outliers = (abs_z_scores > 3).all(axis=1)
raw_data_2[outliers]
```

- Using the isna() function we identify rows which have null values.
  - We get one raw with null values in for the streams column. We drop this row.
  - We get no other rows that have null values.
- Normalizing the data for modelling.
  - Using the below snippet of code we normalize the data.

```
normalized_df = (df_cleaned - df_cleaned.min()) / (df_cleaned.max() -
df_cleaned.min())
```

  - It follows the following formula.

$$\frac{\boldsymbol{curr_{value} - min_{value}}}{\boldsymbol{max_{value} - min_{value}}}$$

- We create a new column theta0.

Feature Selection & Finalizing Data for modelling

- We run the correlation matrix again on the entire dataset including the derived metrics and select the features with abs.correlation >0.1
    - artist_count -0.136463
    - released_year -0.230803
    - in_spotify_playlists 0.789822
    - in_spotify_charts 0.245821
    - in_apple_playlists 0.772063
    - in_apple_charts 0.320234
    - in_deezer_playlists 0.598131
    - in_deezer_charts 0.228598
    - danceability_% -0.105457
    - speechiness_% -0.112333
    - avg_chart 0.335744
    - in_total_playlists 0.783039
    - is_scharted 0.220106
    - is_dcharted 0.209187
    - in_total_playlists_log 0.765984
    - artist_count_inverse 0.132196
    - in_spotify_playlists_log 0.758756
    - in_apple_playlists_log 0.584593
    - in_deezer_playlists_log 0.739196
    - avg_chart_log 0.340677
- We now convert the data to a numpy array.
- The set the seed to 100 and randomly split the data;
    - Train 70% of the data.
    - Test 20% of the data.
    - Holdout 10% of the data.

Modelling & Methodology

- Model: Linear Regression
- Optimizer: Gradient Dissent
- Evaluation Metric: MSE
- Feature Selection: L1

1. We will first Model the Data using Linear Regression.
2. Then Use L1 with lamda=1. Based on the values of thetas, we will drop the feature.
3. Repeat Steps 1 & 2 with the new Data Step.
4. Continue until there is no significant change in change in MSE.

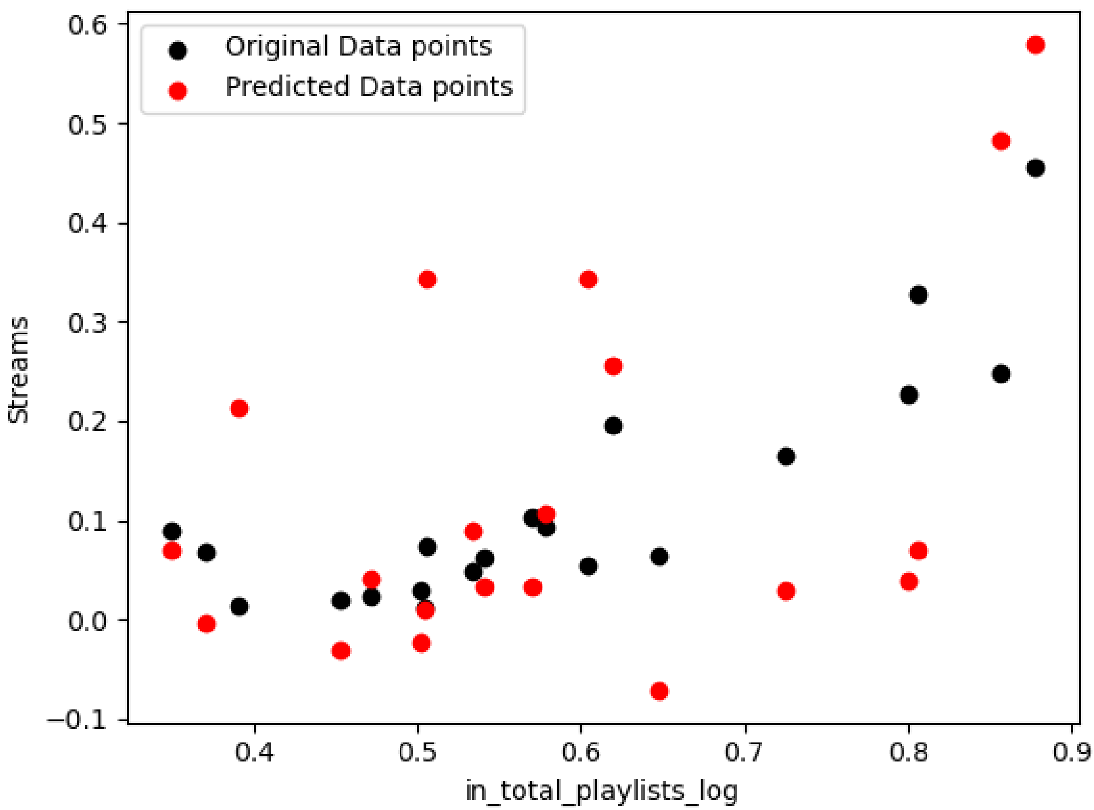


Figure 7 Output of 1st Model Visualized. For 20 Data points.

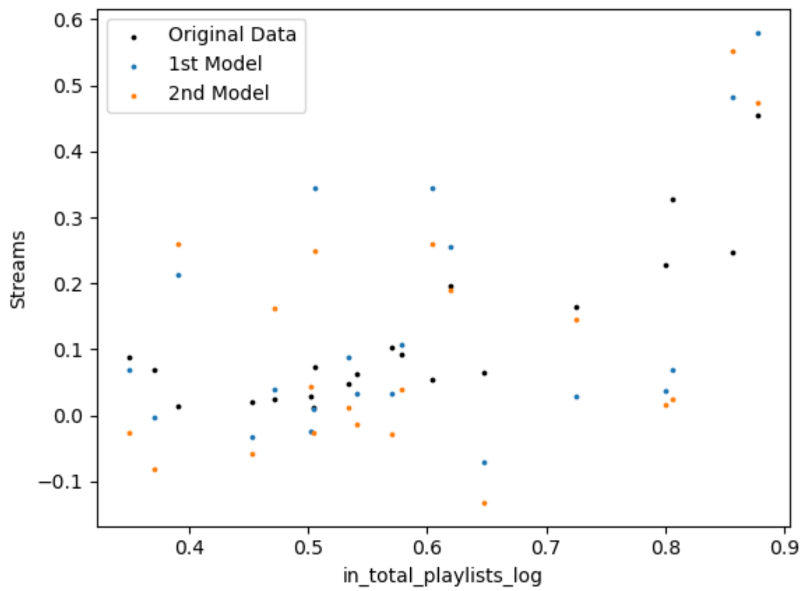


Figure 8 After Applying L1, dropping a feature and Modelling the new Data. For 20 data points

Model Evaluation

- Now we will evaluate the model on the Test Data set, if the $Test_{MSE} \gg Train_{MSE}$ then we have over fitted our Model and we will have start again.
    - Train MSE: 0.0026249082100613186
    - Test MSE: 0.003041144581239226

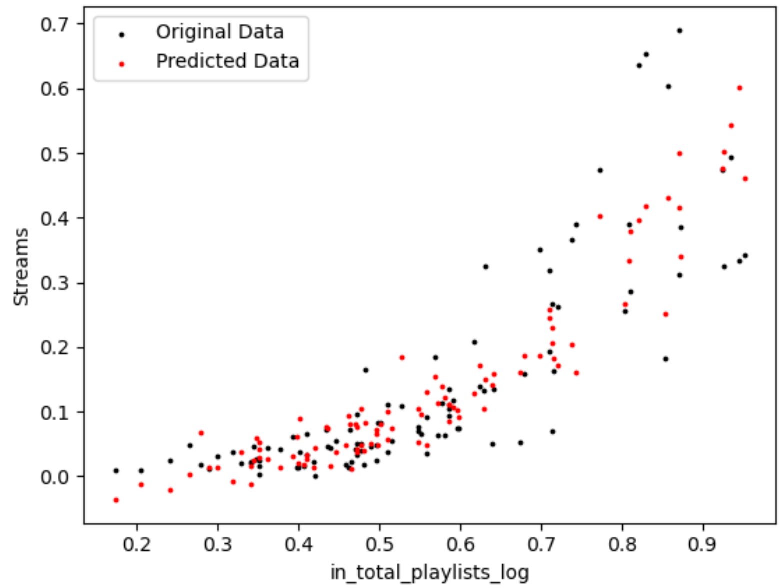


Figure 9 Original Data Test V/s Predicted Values

- The features used to generate the model are
    - Original Features
        - artist_count
        - released_year
        - in_spotify_charts
        - in_apple_playlists
        - in_deezer_playlists
        - in_deezer_charts
        - danceability_%
        - speechiness_%
    - Derived Metrics
        - avg_chart
        - in_total_playlists_log
        - in_spotify_playlists_log
        - in_apple_playlists_log
        - in_deezer_playlists_log
        - avg_chart_log

- We have therefore not overfitted our test data and therefore can Evaluate our Data against the Holdout Set
  - Train MSE: 0.0026249082100613186
  - Test MSE: 0.003041144581239226
  - Holdout MSE: 0.00295934243395952

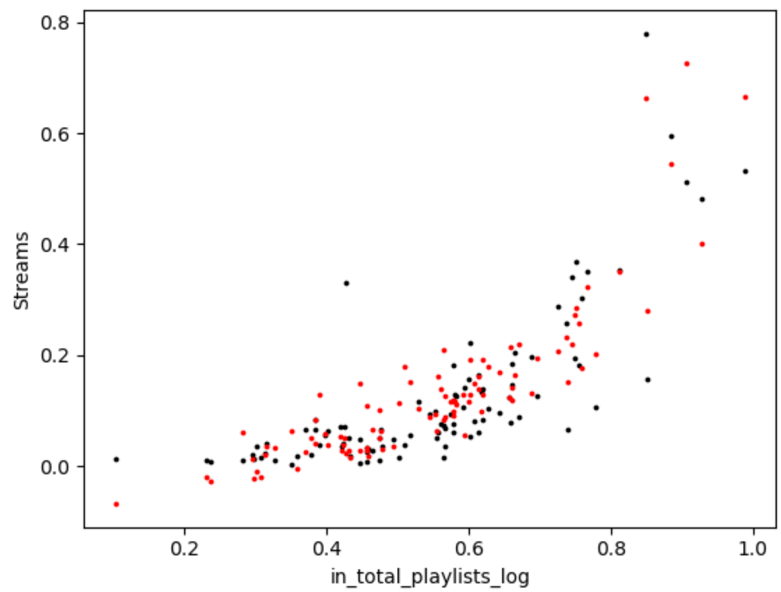


Figure 10 Original Data Test V/s Predicted Values

Bias Variaence Trade off

- Generate a model with none of the derived features and compare its performance to the model with the derived metrics. The features will be the following
  - artist_count
  - released_year
  - in_spotify_playlists
  - in_spotify_charts
  - in_apple_playlists
  - in_apple_charts
  - in_deezer_playlists
  - in_deezer_charts
  - danceability_%
  - speechiness_%
- New Model:
  - Train MSE: 0.0030310791318490194
  - Test MSE: 0.0035014876972878995
  - Holdout MSE: 0.0035674371408394227
- Original Model:
  - Train MSE: 0.0026249082100613186
  - Test MSE: 0.003041144581239226
  - Holdout MSE: 0.00295934243395952
- The Model with Derived metrics performs better.

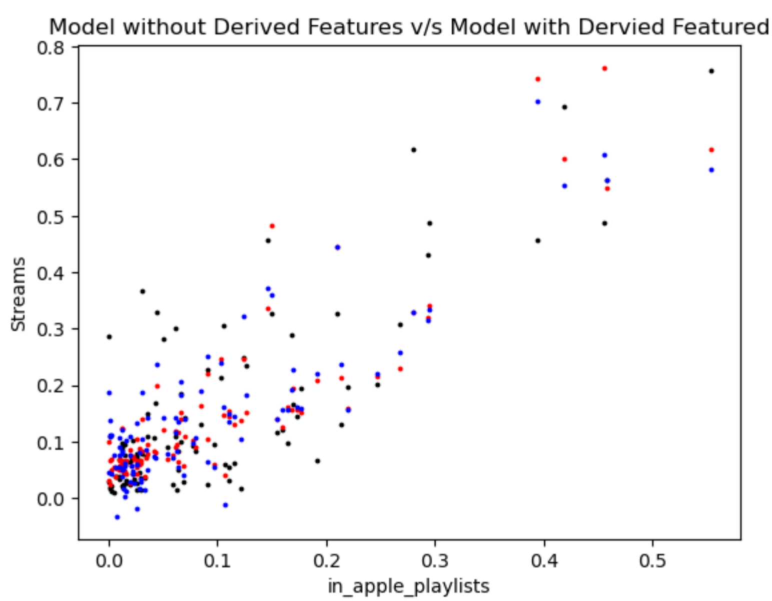


Figure 11 Model without Derived Features v/s Model with Derived Featured Train Data

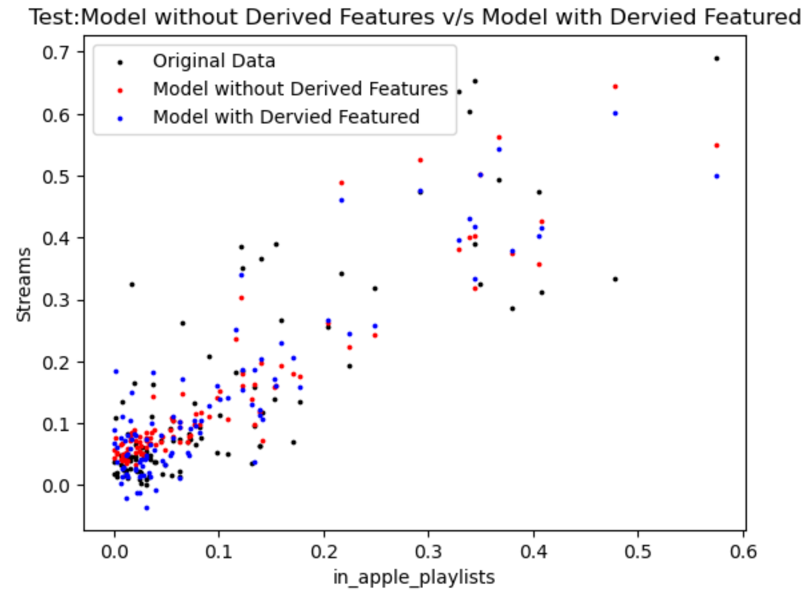


Figure 12 Model without Derived Features v/s Model with Derived Featured Test Data

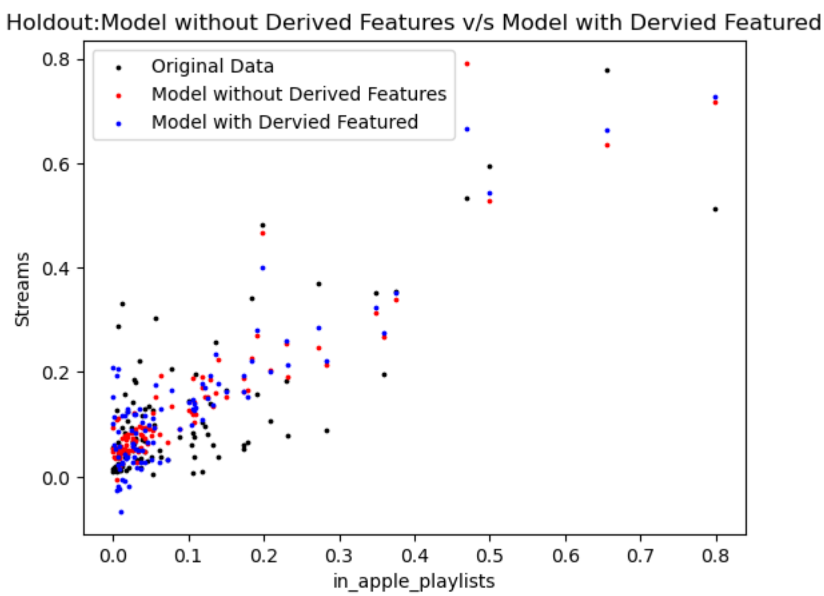


Figure 13 Model without Derived Features v/s Model with Derived Featured Test Data

- Checking the impact of Training Data Size V/s MSE
- New Model

Data Points	Train MSE	Test MSE	Holdout MSE
50	0.003309	0.004195	0.003704
100	0.002893	0.004054	0.004297
150	0.002807	0.00355	0.003978
200	0.002599	0.003462	0.003901
250	0.002329	0.003476	0.003911
300	0.002312	0.003502	0.003816
350	0.002325	0.00351	0.003702
400	0.002544	0.003538	0.003632
450	0.002642	0.003528	0.003674
500	0.002636	0.003505	0.003627
550	0.002814	0.003501	0.003634
600	0.003194	0.003497	0.003592
650	0.003056	0.003495	0.003585

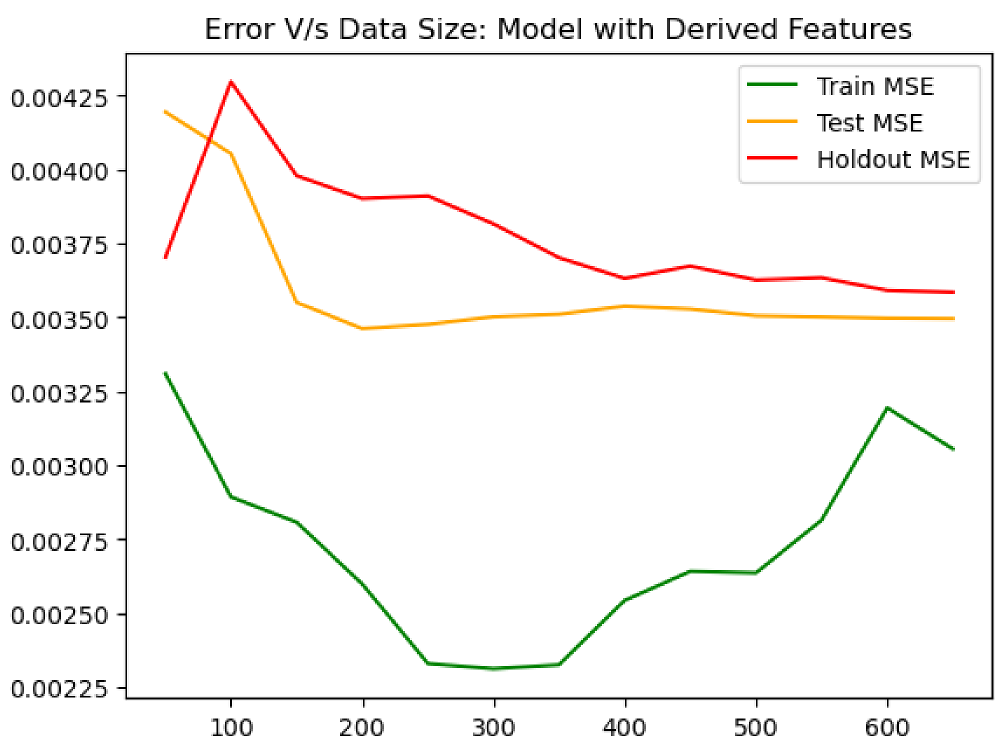


Figure 13 Model without Derived Features v/s Model no with Derived Featured Test Data

- Model With Derived Features

Data Points	Train MSE	Test MSE	Holdout MSE
50	0.002285	0.004369	0.003717
100	0.00225	0.003927	0.004103
150	0.002197	0.003153	0.003306
200	0.002253	0.002971	0.003287
250	0.00208	0.003	0.003256
300	0.002061	0.003015	0.003214
350	0.002108	0.003014	0.003143
400	0.00225	0.003112	0.003131
450	0.002265	0.003087	0.003144
500	0.002279	0.003081	0.003141
550	0.002452	0.003037	0.003045
600	0.00272	0.003054	0.002978
650	0.002634	0.003039	0.002982

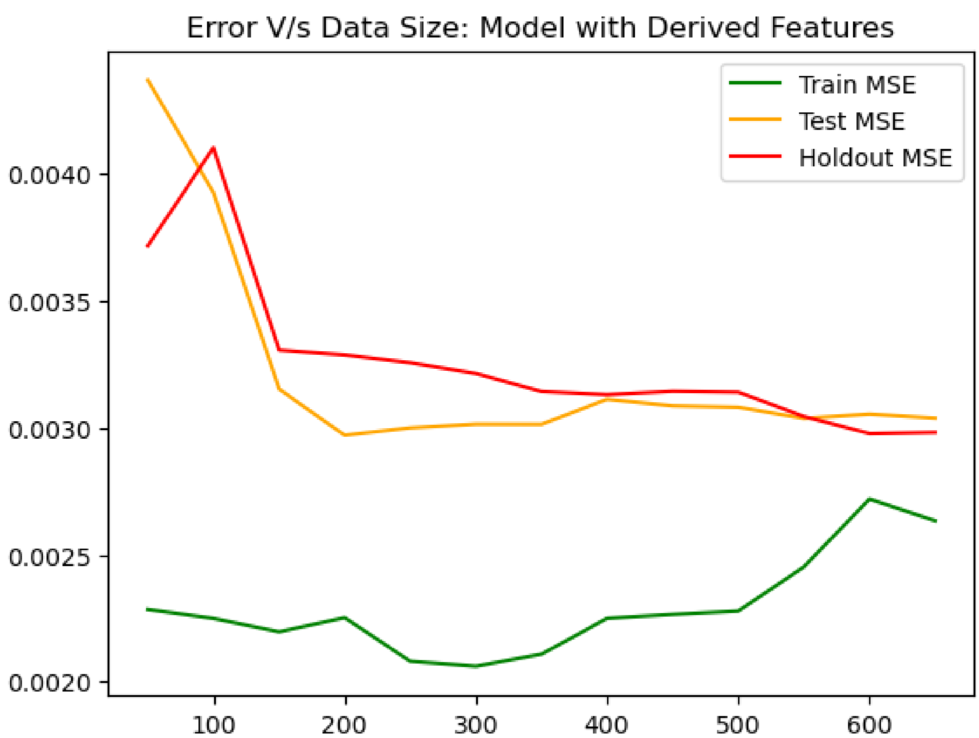


Figure 14 Model without Derived Features v/s Model no with Derived Featured Test Data

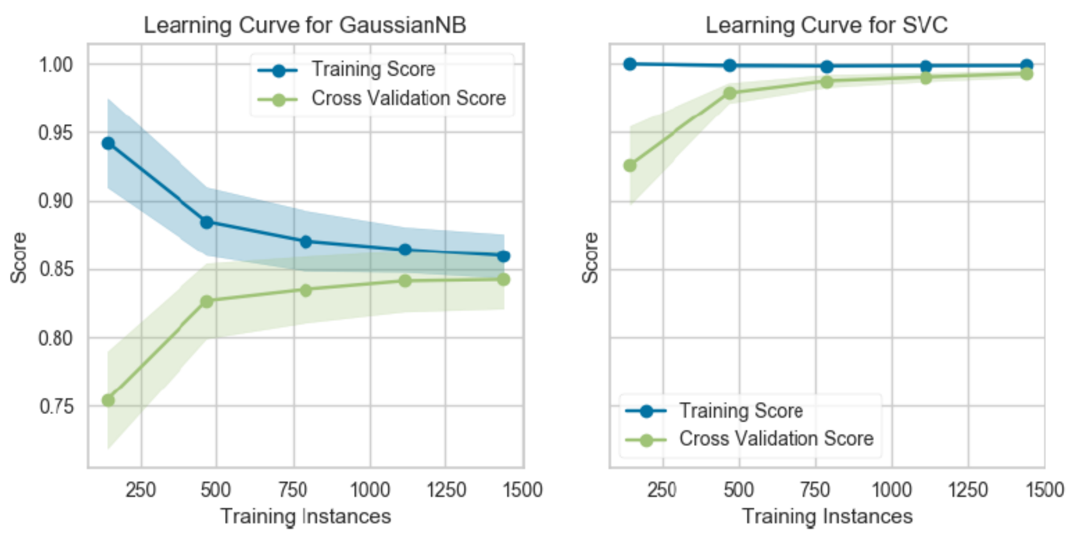
Learnings

- Garnered skills in
    - data preprocessing,
    - addressing missing values
    - dealing with outliers
    - rectifying data formats
    - converting string variables to numeric form
    - implementing data scaling.
- Deepened my comprehension of L1 Regularization, understanding its advantages over standard linear regression.
- Mastered the art of hyperparameter (λ) tuning to optimize model performance.
- Familiarized myself with metrics such as mean squared error to evaluate model efficacy.

Challenges Faced

- Navigating the complexities of converting textual data into numeric forms and imputing missing values using appropriate strategies like mean or mode substitution.
- Recognizing and incorporating only relevant features, ensuring both computational efficiency and model accuracy.
- Grasping the significance of the right λ value, as it profoundly impacts bias and variance.
- Striving to achieve the right equilibrium between bias and variance for optimal model generalization on unfamiliar data.
- Identifying the best-suited machine learning model that delivers precise predictions without excessive computational overhead.

Question 2



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.

The Data point ~750 is where the model achieves Balance between Variance and Bias as there is very minor change in the difference between the errors of Training V/s Cross Validation. From >250 to ~500 the difference between Errors is decreasing rapidly suggesting that the Training Data had high variance. From ~500 to ~750 the difference between Errors isn't decreasing as rapidly as before but isn't close to flattening but post ~750 the difference between is decreasing at a very low rate and coming very close to flattening.

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 points)

a) Both Models have High Variance and Low Bias at 250 Data points.
b) Both Models have Low Variance and Low Bias at 1000+ Data points.

Assignment 1

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?

c) If a model is operating at
    a. High Bias, we can try to do the following to help improve.
        i. Creating Derived Features by transforming the existing features for eg making higher polynomials, taking log etc.
        ii. Adding more features to the model to increase its complexity.
    b. High Variance, we can try to the following.
        i. If possible, one can try adding more training data.
        ii. Adding regularization parameters while training the model.
        iii. Decreasing the complexity by decreasing the no of feature we use while modeling.

D. Do you expect adding more data to improve the performance for each model? Elaborate on your response.

d) No, adding more data will not improve the performance significantly or at all as by analyzing the Training Score V/s Cross Validation Score post 1250(graph 1) ~750(graph 2) datapoints the Scores are almost flattening as compared to prior dataset sizes and the difference between them is also very less. After 1000+ datapoints in Graph A, ~750 we can infer the variance is very low and therefore adding more data will not help.

e) Fig