**Modeling Technique 1: Linear Regression**

**Brief Background of Linear Regression**

Linear regression modeling is a machine learning tool. This modeling technique utilizes multiple independent inputs to model our dependent y-value, which is our target output values. Linear regression modeling can find relationships between a set of features to predict our y-output datapoints based on Python’s Scikit-learn packages. The process of how the linear regression modeling technique works is to establish a straight line that fits a wide-spread datasets. The linear regression model will use the optimized parameters from the Gaussian distribution of the dataset. Thus, linear regression is a simplistic model and is a fundamental method of machine learning.

**Linear Regression Predictive Analysis Method**

In our project, we utilized the linear regression technique to predict how popular a song would be based on a select number of features. After cleaning and normalizing the dataset, we created a correlation table set between our target y-output feature, popularity, and each respective independent input feature. After careful analysis, the input features that had little-to-no correlation were dropped to create our design matrix, Φ, and our y-output array. In our Φ matrix with our selected features and the y-output array, we then splitted our dataset into training sets (X\_train, y\_train) and test sets (X\_test y\_test). Our test set consisted of 20% of the data. We then fitted our linear regression model with the training set. With our trained model, we used our X\_test set to output y-pred, our prediction array.

**Histogram Analysis of our y-Datasets**

Moreover, in Figure 1, we displayed the histogram of our y\_train, y\_test, and y\_pred arrays. In subplot 1, we showcased the distribution of the y\_train set, which appeared to be a normal distribution, but had a peak in the 0-10 bin. And in subplot 2, we noticed a similar distribution in y\_test as our y\_train set. Interestingly, we did not have the same behavior in our y\_pred array as it appeared to behave as a Gaussian distribution due to our simplistic linear regression model. Thus, although our y\_pred distribution showed that we can predict how popular a song will be within the Gaussian distribution of the y-test sets, we were not able to accurately predict the low-popularity songs as we see in subplot 2 in the 0-10 bin. Since we cannot control real-life scenarios of outliers or how unpopular a song would be, our linear regression model did not accurately perform well for low-popularity songs as seen in Figure 1, subplot 2.

Chart, histogram

Description automatically generated

Figure 1: Histogram for our y-datapoints, which included our y\_train, y\_test, and y\_pred sets. In the plot, it is showcased that our y\_train and y\_test sets behave similarly to a Gaussian distribution with another peak in the 0-10 bin. However, due to the simplistic manner of linear regression modeling, y\_pred aligns with the Gaussian distribution, but does not accurately predict low-popularity songs as seen in subplot 2.

**Conclusion and Results**

In Figure 2 subplot 1, we noticed that the y\_test and y\_pred outputs showed a good prediction as the points are closely condensed to one another. In addition, in Figure 2 subplot 2, we showcased the residual plot. It displayed how shaded the region of the y-axis, our residual values, specifically in the range between -20 to 20. This plot showed that the y\_test and y\_pred are close to one another. Specifically in our error analysis, using the root-mean-square error (RMSE), we calculated the training and testing RMSE as 14.33 and 14.44. If our training RMSE is very small, the model will overfit. In retrospect, if our testing RMSE is very large, then our model is inaccurate. Our goal was to find our respective training and test RMSE to be as close to one another. In addition, our coefficient of determination, R2 score, was approximately close to 0.4. Although it appears to have a low-correlation score, our linear regression model is still a good model because it is more difficult to predict human’s music behavior than physical processes.

Graphical user interface, chart

Description automatically generated

Figure 2: In subplot 1, we noticed that the y\_test and y\_pred arrays showed a good prediction as the points are closely aligned to one another. In subplot 2, we displayed the residual plot, in which the region between -20 to 20 of the residual values are shaded, meaning that the y\_test and y\_pred output arrays are close to each other.

Moreover, in Figure 3, we observed that the predicted and observed values followed a straight line. This model roughly followed a linear relationship and showed promising results on predicting how popular a song will be. However, we noticed that there was a large amount of datapoints that follow the horizontal line (y\_test = 0) of the observed values. This observation showed that our linear regression model did not accurately predict the unpopular songs due to the machine learning’s simplistic modeling technique.

Chart, scatter chart

Description automatically generated

Figure 3: In the plot, we noticed that the actual values vs. predicted values followed a linear relationship. This observation showed that our linear regression model did a good job on predicting how popular a song will be. However, we also noticed that the model did not predict how unpopular a song will be.

**Additional Discussion: Chebyshev’s Inequality**

In a probability distribution, Chebyshev’s Inequality is an idea that provides an understanding on the upper and lower limits of our dataset, especially if a dataset does not have a normal distribution. In utilizing Chebyshev’s Inequality, there is only a specific percentage of values, 1/k2 , where k is the standard deviation, that the datapoints will not exceed the distance from the mean. In other words, no matter the shape of the distribution, the majority of the datapoints will lie within a few standard deviations of the expectation. Using Chebyshev’s Inequality, we can remove outliers to prevent skewness of our results and improve our linear regression model to perform better.

As shown in Figure 4, the kernel density estimate has been plotted for y\_train, y\_test, and y\_pred, as it visualizes the continuous probability density curve while displaying multiple distributions. Specifically for y\_train and y\_test, the figures did not follow the normal Gaussian curve as it has two peaks. Our linear regression model did not accurately predict unpopular songs, as shown in the first peak. Thus, we used Chebyshev’s Inequality for estimation to provide insights about which outliers in our training data distribution can be removed for a better predictive model. In Figure 5, we tuned the k-value or the standard deviation, which had a range between [0, 3]. For each k-value, we found our upper and lower limit for which our training values should exist in our updated dataset. For each iteration of k-value, we trained our model with the updated dataset and outputted our predictive values using the test set. Then, we calculated the training and testing RMSE for each iteration and plotted it below in Figure 5 subplot 1. We observed that when k-value was the smallest, our model performed the worst due to overfitting the dataset. As k-value increases, our training and testing RMSE performs better. Specifically, as the k-value increases to 2.2, we notice that the training and testing RMSE values have their turning points approximately at 14.245 and 14.445, respectively. Compared to the linear regression model in our previous RMSE analysis, our training RMSE increased by the amount of 0.085 and our testing RMSE decreased by 0.004 using our optimized k-value as indicated in Figure 5 subplot 2. Using Chebyshev’s Inequality, our model showed less overfitting behavior and more accuracy. Even with Chebyshev’s Inequality our linear regression model had a slight improvement, however, the RMSE values are still relatively around 14. Thus, we seek more complicated modeling techniques in our final project.

Chart

Description automatically generatedChart

Description automatically generated

Figure 4: In the plot, we see that the KDE distribution of y\_train and y\_test has two peaks, in which we can apply Chebyshev’s Inequality to remove outliers depending on the standard deviation. However, although we improved on our training data, it perform worse for our testing data as our model overfits closer to the training dataset.

Chart, line chart

Description automatically generated

Figure 5: In the first subplot, we observed the Sigma vs. RMSE for each training and testing dataset. We observed that at the turning points for each training and testing RMSE, we had our optimized sigma value of 2.2. In the second subplot, we observed that at the value of 2.2, we had the smallest difference between the training and testing RMSE.