WILL DRAKE’S NEXT SONG MAKE IT TO SPOTIFY’S TOP 200

Introduction and Motivation

In today’s generation, streaming platforms have too much control over how artist distribute their work and how fans listen to music. Spotify is one of the leading streaming services that has become a significant platform for a song’s success. Here, the presence of a song by an artist directly contributes to the artist's popularity. So understanding the characteristics that contribute to a song's ability to make it onto the chart has become vital information for artists, record labels, and music directors within the music industry.

In this project, we aim to correlate the elements associated with a song that make it a commercial success, particularly in the context of songs charting on Spotify. Given the vast data of music content available on Spotify, it becomes important to analyze not only the intricate qualities of the song but also how these qualities resonate with the audience, which contributes to a song's performance on the charts.

The reason for choosing Drake as our primary artist has multiple purposes. First, Drake is a globally recognized artist with a significant presence on Spotify and his songs are often in top positions among various charts. This dominance provides us with a rich dataset due to the variability in the chart performance of his songs. Second, Drake's diverse work provides us with an opportunity to analyze how different sets of musical features within a single artist's portfolio. Lastly, studying a high-profile artist such as Drake may give us insights that are scalable or applicable to emerging artists seeking to replicate such success.

As mentioned before, the project has a large-scale commercial use but it also contributes to one of the areas where we can use Science to enhance performance arts. By analyzing the musical attributes and their impact on popularity, this project provides us with a better understanding of current trends in the music industry.

Problem definition and brief data description

The main problem we want to address through this project is the correlation of a song's chart success based on its diverse range of musical attributes. As we all know there are over 11 million artists and creators on Spotify (source link). Provided the total number of creators on Spotify and the volume at which songs are released on Spotify it becomes challenging and critical to understand how different factors influence a song's popularity and its chart performance. We aim to determine whether there is a correlation between specific attributes of a song that can serve as indicators to rank on music charts on Spotify, specifically using Drake's songs as an object variable.

The datasets used in this study provide comprehensive data on Drake's presence on Spotify. The **Drake_Spotify_Data.csv** dataset, obtained from Kaggle, contained 36 columns of information about the album and track. From there, we decided to select only the most important features related to a track. These features included danceability, energy, loudness, valence, tempo, acoustics, speechiness, and instrumentalness. These features are directly derived from Spotify's algorithms, which quantify the aspects of music heard by listeners.

The **New_Spotify_chart_song_ranks.csv** dataset was collected directly from Spotify's top 200 weekly USA chart and includes information on songs that have charted on Spotify, listing artists, track names, and URI codes for each track.

We then filtered the datasets to include songs only from the year 2020 onwards, as this was the limitation of the Spotify chart data we had acquired. For the **New_Spotify_chart_song_ranks.csv** dataset, we further filtered to include both songs where Drake was the main artist and where he was a featured artist.

Combining these two datasets helped us identify Drake's songs that have charted on Spotify and their attributes, allowing for an intricate analysis against those that have not charted on Spotify. We had 65 songs, 22 of which charted on the chart, and 43 of which did not. This pairing of datasets helps us formulate a binary classification problem: determining the likelihood of a song charting based on its quantified musical features

Further ahead in the report will talk about the methodologies used for data analysis, how the exploratory data analysis was conducted, and a detailed presentation of the findings and their implications for the music industry.

Methods

To compare the predictive performance of each model, we compute the estimated test MSE. This is done by 80:20 random splits of the dataset, where 80% of the data are used for training models, and 20% of the data are used for validation.

We start our analysis with logistic regression. Logistic regression is a parametric, discriminative model that calculates the probability of an observation belonging to a certain group. The nature of logistic regression makes it suitable for modeling probability as the computed probability is bounded by [0,1]. We could then choose the threshold to assign the observation into groups. For our project, we decided to use a threshold of 0.5 to remain neutral.

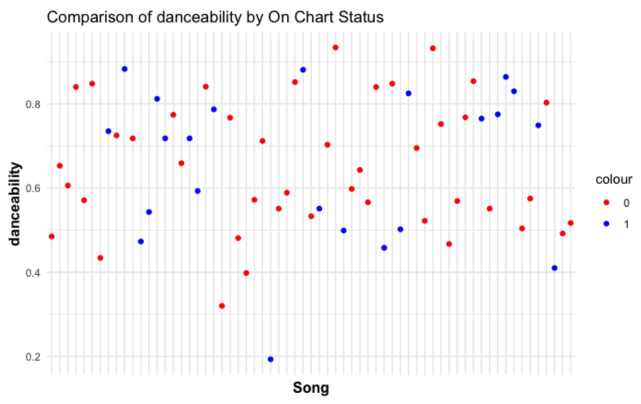
Moving on, we employ Random Forest(RF) for modeling. RF is a tree-based, discriminative, non-parametric model; it is also considered an ensemble method, where multiple simple weak decision trees are combined to form a model that has high predictive power. In the classification task, each decision tree splits the data so that it results in the most pure node, as measured by the Gini index or Entropy. This allows the model to decorrelate and thus results in a reduction of variance compared to using one big, complex decision tree. The advantage of using RF is that only a subset of the total predictors is allowed to be considered when constructing simple decision trees; for this model, we settle to use 3 variables that are chosen randomly. Compared to using logistic regression, RF allows non-linear decision boundaries, which could help to capture more complex relationships.

As our analysis matures, we consider the LASSO model. LASSO regression is a discriminative, parametric model. Since this is in the classification setting, LASSO regression is equivalent to logistic regression with regularization terms. Different from logistic regression, the LASSO regression is penalized for having complex models, which improves the model generalizations. This will ideally help the prediction model to have better out-of-sample performance. However, we acknowledge that better test MSE can only be guaranteed with carefully chosen lambda through cross-validation. Furthermore, the LASSO algorithm promotes sparse solution, that is LASSO allows coefficients to be shrunken completely to 0 with sufficient lambda, acting also as a model selection algorithm.

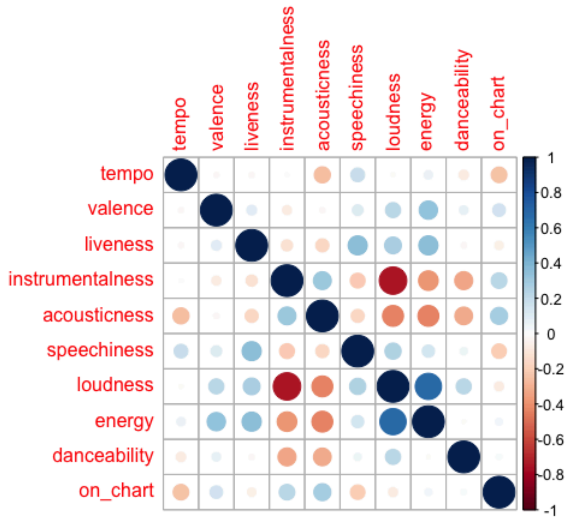
Finally, we escape the discriminative world and model using QDA, a parametric, generative model. From the beginning, the QDA model assumes that the underlying distribution of observation from each class is drawn from Gaussian distribution, and different groups have different covariance. The QDA model relies on Bayes' Theorem to compute the probabilities that observations belong to a certain group, assigning them to groups that have a higher posterior probability. Due to the ability to compute quadratic decision boundaries, the QDA model can capture complex relationships

Exploratory Data Analysis

Looking at the data it becomes evident that it is going to be hard to find a pattern between the variables and the predictor. The artist Drake is known for pop-rap music, this type of music tends to be more upbeat and danceable so one expects variables like tempo and danceability would have a high impact on whether or not these songs make it onto the charts. Looking at the chart shows we see the danceability score of the songs split into whether the song made it in blue and did not make it on the chart in red.

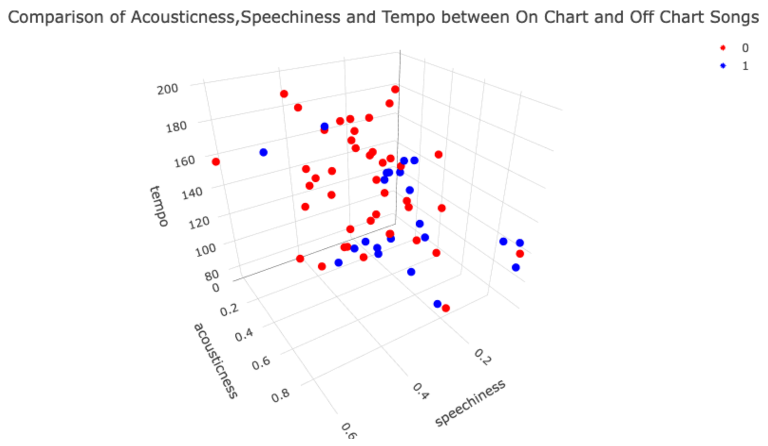


The scatter plot does not reveal a clear pattern that would indicate that the danceability score would affect the song being on the chart. The data points are scattered in different places and there is no dismissible pattern between the on the charts being the blue point and the red points off the charts.



The correlation matrix lets us see all the strong correlations between variables. More importantly, it allows us to identify strong correlations with "on_chart." The graph shows many variables have a strong correlation with one another, meaning there is some multicollinearity. We can see that almost every variable except danceability has some sort of correlation to the song being on the chart. The variables that seem to have the strongest correlation are tempo with a strong negative correlation, acousticness with a positive correlation, instrumentalness, and speechiness.

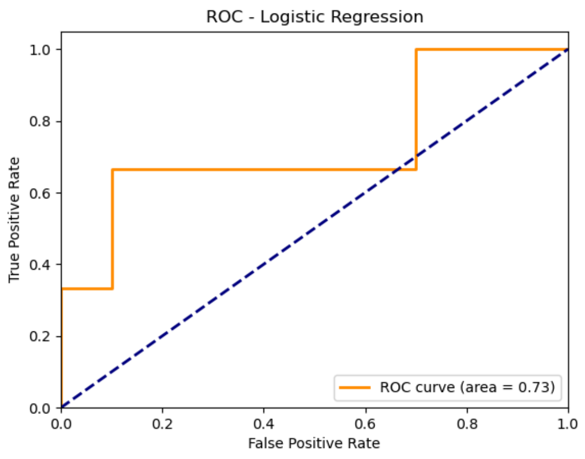
The variation between instrumentals and songs is too small to be included in the 3D graphs. In the graph, tempo, acousticness, and speechiness have been graphed to see if there is any visual linear relation. From the look of it, we can see that many songs that did not make it onto the charts are mostly gathered around having an acousticness level between 0 and 0.24, and a speechiness level between 0 and 0.4. Tempo doesn't seem to have a visual relationship as all the points are scattered along the z-axis.

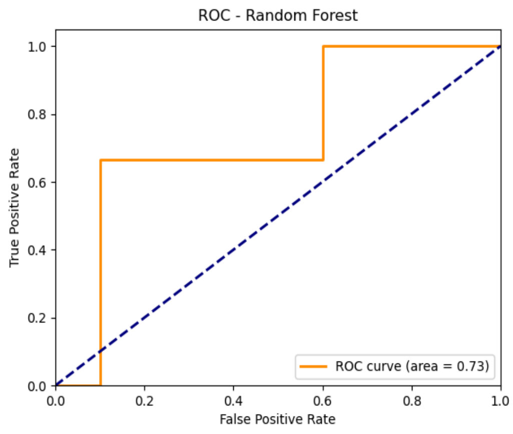


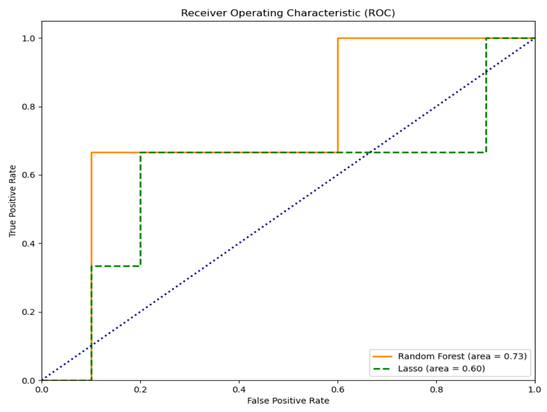
Result

To compare and choose the best models, we decide to use the estimated out-of-sample performance criteria. For logistic regression and Random Forest(RF), we obtained the test MSE of 0.23; for LASSO regression, we obtained the test MSE of 0.19 and for the QDA model we obtained test MSE of 0.38. Therefore, we can conclude that LASSO regression has the strongest predictive power out of the four models. Moreover, the LASSO regression gives a sparse solution, where all but 4 coefficients are shrunken to 0, basically performing model selections simultaneously. Not only that LASSO outperform others that we considered, but it also outperforms the ZeroR[1] classifiers, which we use as our baseline accuracy value of 0.23. A possible explanation for the poor performance of the QDA model compared to the other discriminative model can be due to its generative nature, where it assumes underlying Gaussian distribution. Perhaps this assumption is violated to a great extent.

Another metric for comparison is the area under the ROC curve (AUC) values, with AUC values closer 1 means greater classification power. In this case, the LASSO model turns out to be the worst model with AUC values of 0.60, whereas other 3 models have AUC values of 0.73.







[1] This classifier predicts all observations to be in the majority group and basically ignores all predictors variables.

Discussion and Outlook

Our project aims to provide insights into the factors that contribute to making a song successful. By using the machine learning techniques learned in class on the dataset from Spotify and Kaggle, we are eager to discover the patterns that could predict a song's chart success. In our project, we employed varieties of statistical models, from parametric to non-parametric, and discriminative to generative, such as, logistic regression, lasso regression, random forest, and quadratic discriminant analysis (QDA).

The strength of our project is the use of modern machine learning methods to gain insights into a popular topic with the potential of global scalability. In addition, we also included a variety of analytical methods (logistic, lasso, random forest, QDA) through which we demonstrated a thorough analysis that is useful in understanding the problem and comparing the predictive power of different modeling techniques.

Like any other project, we also have limitations that could affect the accuracy of our findings such as high test error rates, indicating the possibility of a model mismatch. The diversity of the data could also be a constraint and we aim to improve that. This limited dataset could affect our predictive accuracy. There are also outside factors that have a major influence on a song's performance, for instance, marketing efforts, external events, trends, and artists' social media presence.

Outlooks

Overall, our project set the stepping stones for a more complicated project in predicting song's popularity. Future research can address our limitation by expanding the dataset to include a broader range of songs and artists to yield more generalized insights. Furthermore, exploring more advanced machine learning techniques, such as deep learning methods, and integrating social variables like genre, social media influence, and market trends can provide a more generalized view of a song's potential for success. Implementing these changes, while it will take much more computational power and expertise, we're promised a clear improvement in predictive accuracy.

Conclusion

In conclusion, while we cannot definitively provide a model that predicts a song's success, our project stands as a testament that predicting a song's success on Spotify's top chart is a highly challenging yet fascinating endeavor. Music popularity is complex, many artists could be exceptionally talented and produce remarkable music but still won't make the Spotify top charts their entire career. Our research provides insights into this topic and we and the people after us must utilize our models and continue to research and refine a new model that can uncover the formula behind a hit song and revolutionize how artists and producers approach song production.

Code

GitHub Link: https://github.com/ktpanda2025/STA_141C_FINAL_PAPER

All the datasets can be obtained from the GitHub and one can recreate our code by downloading the dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, LassoCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, roc_curve, auc, mean_squared_error
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from statsmodels.stats.outliers_influence import variance_inflation_factor
import warnings


Spotify_data =
pd.read_csv("/Users/devansh/Downloads/New_Spspotify_chart_song_ranks.csv") #The
data we got from Spotify
kaggle_data = pd.read_csv("/Users/devansh/Downloads/Drake_Spotify_Data.csv") #The
data we got from Kaggle

#Upon looking at the data, we saw that there were a lot of duplicates. So, we
remove
#all the duplicate tracks based on 'uri' and 'artist_names', then count the number
of artists per song.

artist_per_song_count = Spotify_data.drop_duplicates(subset=['uri',
'artist_names']) \
    .groupby('uri') \
    .size() \
    .reset_index(name='count') \
    .sort_values(by='count', ascending=False)

#We now need to filter the Spotify data for tracks by Drake, and again remove
duplicates based on 'uri',
drake_chart_spotify = Spotify_data[Spotify_data['artist_names'] == "Drake"] \
    .drop_duplicates(subset='uri') \
    .sort_values(by='WeekDate')
```

```
#As the trends are always changing we wanted to make sure that we are focusing
#on the later trends only
#So, we filter the Kaggle data for tracks released in or after 2020 and remove
duplicates based on 'track_name'.

kaggle_data_2020 = kaggle_data[kaggle_data['album_release_year'] >= 2020] \
    .drop_duplicates(subset='track_name')

#We need to join the filtered Kaggle dataset and Drake's Spotify chart data
#on their respective track URIs. This merges data for tracks found in both
datasets.
inner_join_result = pd.merge(kaggle_data_2020, drake_chart_spotify,
left_on='track_uri', right_on='uri', how='inner')

left_join_data = pd.merge(kaggle_data_2020, drake_chart_spotify,
left_on='track_uri', right_on='uri', how='left')


#Now we need to select a subset of columns for analysis, focusing on musical
attributes, track names etc.
#This prepares the data for further analysis on the correlation between these
attributes and chart performance.
new_data = left_join_data[['tempo', 'valence', 'liveness', 'instrumentalness',
'acousticness',
                      'speechiness', 'loudness', 'energy', 'danceability',
'WeekDate',
                      'track_name_x']]



with warnings.catch_warnings():
    warnings.simplefilter("ignore", category=pd.errors.SettingWithCopyWarning)
    new_data['on_chart'] = np.where(new_data['WeekDate'].isna(), 0, 1)

new_data = new_data.sort_values(by='WeekDate')


new_data.drop(columns='WeekDate', inplace=True)
new_data.rename(columns={'track_name_x': 'track_name'}, inplace=True)


new_data.columns

np.random.seed(45)
#For an initial EDA we are checking the correlation between each col of the data
correlation_matrix_full = new_data.corr(numeric_only=True)
print(correlation_matrix_full)
```

```
#We are now focusing on a few features we want to check how they affect the success
rate
features_to_compare = ['danceability', 'energy', 'loudness', 'valence', 'tempo',
'acousticness', 'speechiness', 'instrumentalness','liveness']

X = new_data[features_to_compare]
y = new_data['on_chart'] #Target Binary Variable

#Splitting the data at 80% - 20% ratio split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
test_data_combined = X_test.copy()
test_data_combined['on_chart'] = y_test
mean_on_chart = test_data_combined['on_chart'].mean()
print("Mean of 'on_chart' in the test dataset:", mean_on_chart)


#We now scale the data using the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Performing Logistic Regression
logistic = LogisticRegression()
logistic.fit(X_train_scaled, y_train)
predict_test_logistic = logistic.predict(X_test_scaled)
prob_logistic = logistic.predict_proba(X_test_scaled)[:,1]
accuracy = accuracy_score(y_test, predict_test_logistic)
print(f"Logistic Regression Test Accuracy: {accuracy}")
print(classification_report(y_test, predict_test_logistic))
test_error_logistic = mean_squared_error(y_test, predict_test_logistic)
print("This is the MSE for Logistic" , test_error_logistic)


#Performing ROC for Logistic Regression
fpr_lr, tpr_lr, _ = roc_curve(y_test, prob_logistic)
roc_auc_lr = auc(fpr_lr, tpr_lr)
plt.figure()
lw = 2
plt.plot(fpr_lr, tpr_lr, color='darkorange', lw=lw, label='ROC curve (area =
%0.2f)' % roc_auc_lr)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Logistic Regression')
plt.legend(loc="lower right")
plt.show()

#Confusion Matrix for Logistic Regression
confusion = confusion_matrix(y_test, predict_test_logistic)
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix - Logistic Regression')
plt.show()

#Random Forest
rf = RandomForestClassifier()
rf.fit(X_train_scaled, y_train)
predict_test_rf = rf.predict(X_test_scaled)
prob_rf = rf.predict_proba(X_test_scaled)[:,1]
accuracy = accuracy_score(y_test, predict_test_rf)
print(f"Random Forest Test Accuracy: {accuracy}")
print(classification_report(y_test, predict_test_rf))
test_error_rf = mean_squared_error(y_test, predict_test_rf)
print("This is the MSE for Random Forest" , test_error_rf)


#ROC for Random Forest
fpr_rf, tpr_rf, _ = roc_curve(y_test, prob_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)
plt.figure()
plt.plot(fpr_rf, tpr_rf, color='darkorange', lw=lw, label='ROC curve (area =
%0.2f)' % roc_auc_rf)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Random Forest')
plt.legend(loc="lower right")
plt.show()

#Confusion Matrix for Random Forest
confusion = confusion_matrix(y_test, predict_test_rf)
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix - Random Forest')
plt.show()

#Perform Lasso regression to see whic hcef results in 0
lasso = LassoCV(cv=5).fit(X_train_scaled, y_train)
lasso_coefficients = pd.Series(lasso.coef_, index=features_to_compare)
lasso_predict = lasso.predict(X_test_scaled)
test_mse_lasso = mean_squared_error(y_test, lasso_predict)
```

```
print("Test MSE for Lasso:", test_mse_lasso)
print("Lasso Coefficients:")
print(lasso_coefficients)

#check for a non-linear relationship
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train_scaled, y_train)
pred_test_qda = qda.predict(X_test_scaled)
accuracy = accuracy_score(y_test, pred_test_qda)
print(f"QDA Test Accuracy: {accuracy}")
print(classification_report(y_test, pred_test_qda))
test_error_qda = mean_squared_error(y_test, pred_test_qda)
print("This is the MSE for QDA" , test_error_qda)



avg_features_by_chart_status =
new_data.groupby('on_chart')[features_to_compare].mean()
avg_features_by_chart_status.T.plot(kind='bar', figsize=(12, 6))
plt.title('Average Feature Values: Charted vs Not Charted')
plt.ylabel('Average Value')
plt.xticks(rotation=45)
plt.legend(title='Charted', labels=['Not Charted', 'Charted'])
plt.show()


fpr_rf, tpr_rf, _ = roc_curve(y_test, prob_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)
fpr_lasso, tpr_lasso, _ = roc_curve(y_test, lasso_predict)
roc_auc_lasso = auc(fpr_lasso, tpr_lasso)
plt.figure(figsize=(10, 8))
plt.plot(fpr_rf, tpr_rf, color='darkorange', lw=2, label='Random Forest (area =
%0.2f)' % roc_auc_rf)
plt.plot(fpr_lasso, tpr_lasso, color='green', lw=2, linestyle='--', label='Lasso
(area = %0.2f)' % roc_auc_lasso)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle=':')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
#Link to github("https://github.com/ktpanda2025/STA_141C_FINAL_PAPER")
```