Learning from Data Coursework

Name - Rishav Bhattacharyya
Email - rb957@exeter.ac.uk

Table of Contents

Research Question

What specific audio features contribute significantly to the classification of songs as highly popular on Spotify, and to what extent can machine learning models accurately identify and predict this elevated level of popularity?

Rationale:

This study is motivated by the exploration of distinct audio characteristics that play a crucial role in determining whether a song attains the distinction of high popularity on Spotify. Unlike conventional models that treat popularity as a continuous variable, our focus centers on discerning features associated with songs achieving the status of high popularity.

Employing advanced machine learning classification models, this research aims to unveil intricate patterns within audio features that serve as key determinants for high popularity classification. The objective is to identify the nuanced traits that set apart songs with widespread appeal, shedding light on the factors contributing to their classification as highly popular on the Spotify platform.

Unraveling the determinants of high popularity holds significant implications for artists, producers, and the music industry at large. By leveraging machine learning, this research seeks to provide a comprehensive analysis of the audio features influencing a song's elevation to high popularity, thereby contributing to a deeper understanding of the elements that resonate most strongly with audiences.

Dataset Properties : Introduction

Content: A collection of 114K Spotify tracks with audio features across 125 different genres

Format: CSV (Tabular Data)

```
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   track_id          114000 non-null  object
 1   artists           113999 non-null  object
 2   album_name        113999 non-null  object
 3   track_name        113999 non-null  object
 4   popularity        114000 non-null  int64
 5   duration_ms       114000 non-null  int64
 6   explicit          114000 non-null  bool
 7   danceability      114000 non-null  float64
 8   energy            114000 non-null  float64
 9   key               114000 non-null  int64
 10  loudness          114000 non-null  float64
 11  mode              114000 non-null  int64
 12  speechiness       114000 non-null  float64
 13  acousticness      114000 non-null  float64
 14  instrumentalness  114000 non-null  float64
 15  liveness          114000 non-null  float64
 16  valence           114000 non-null  float64
 17  tempo             114000 non-null  float64
 18  time_signature    114000 non-null  int64
 19  track_genre       114000 non-null  object
```

track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness
5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430
4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763
1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557
6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363
5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526
01MVOl9KtVTNfFiBU9I7dc	Tyrone Wells	Days I Will Remember	Days I Will Remember	58	214240	False	0.688	0.4810	6	-8.807	1	0.1050

Dataset Properties : Metadata

Column	Description	Column	Description
track_id	Spotify ID for the track	**loudness**	Overall loudness of a track in decibels (dB)
artists	Names of the artists who performed the track (multiple artists separated by ';')	**mode**	Mode indicates the modality (major or minor) of a track (1 - Major, 0 - Minor)
album_name	Album name in which the track appears	**speechiness**	Detects the presence of spoken words in a track (values: 0.0 to 1.0)
track_name	Name of the track	**acousticness**	Confidence measure of whether the track is acoustic (0.0 - not acoustic, 1.0 - high confidence acoustic)
popularity	A value between 0 and 100, indicating track popularity based on plays and recency	**instrumentalness**	Predicts whether a track contains no vocals (values: 0.0 - vocal, 1.0 - high likelihood instrumental)
duration_ms	Track length in milliseconds	**liveness**	Detects the presence of an audience in the recording (values: 0.0 - low liveness, 1.0 - high likelihood live)
explicit	Indicates if the track has explicit lyrics (true/false/unknown)	**valence**	Describes the musical positiveness conveyed by a track (0.0 - negative, 1.0 - positive)
danceability	Describes how suitable a track is for dancing (0.0 - least danceable, 1.0 - most danceable)	**tempo**	Overall estimated tempo of a track in beats per minute (BPM)
energy	Perceptual measure of intensity and activity (0.0 - low energy, 1.0 - high energy)	**time_signature**	Estimated time signature indicating how many beats are in each bar
key	The key the track is in, mapped to pitches using Pitch Class notation	**track_genre**	The genre in which the track belongs

Data Preprocessing: Data Cleaning

1. **Removing Duplicates**

- After inspecting I found there were 450 tracks(0.39% of all tracks) which are duplicates.

```
spotify_df = spotify_df.drop_duplicates()
```

2. **Handling Missing values**

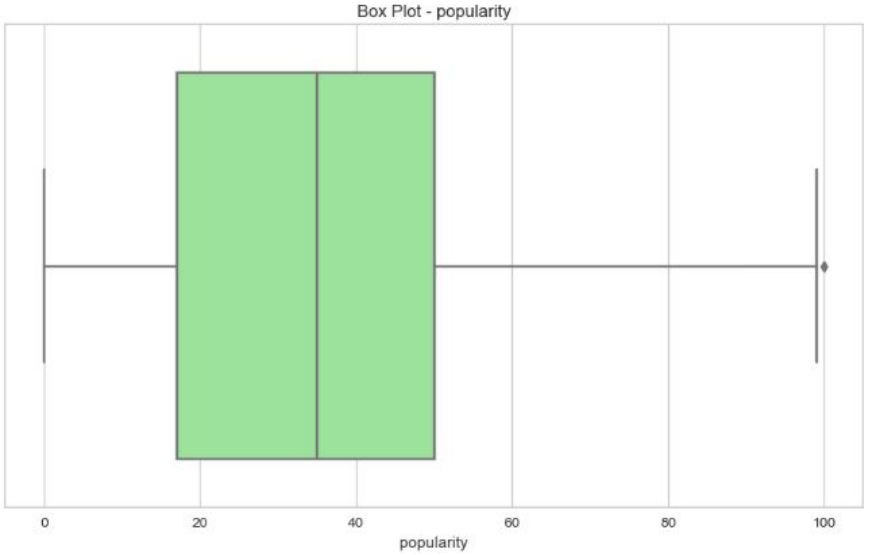
- After removing duplicates, I found 0.0008% of tracks,album and artist names were missing, I imputed them with ‘missing’ as a category as I didn’t want to lose any information for Exploratory Data Analysis

```
missing_columns = ['album_name','track_name','artists']
for column in missing_columns:
    spotify_df[column] = spotify_df[column].fillna('missing')
```

Data Preprocessing : Defining target variable

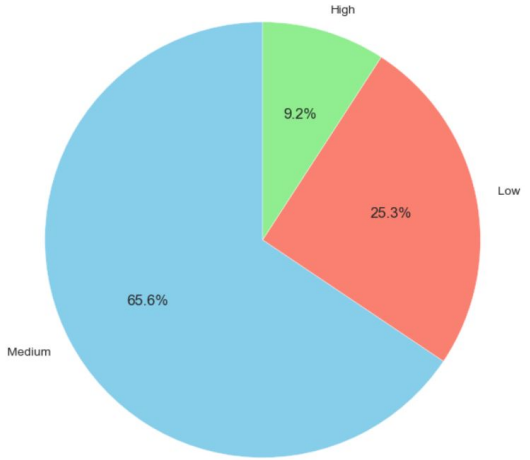
Popularity variable distribution

- According to the definition this variable mainly depends on recency and streams of the song

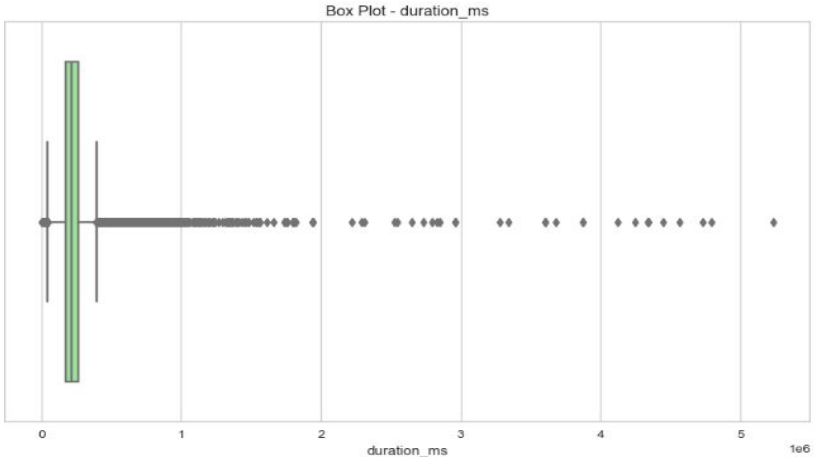


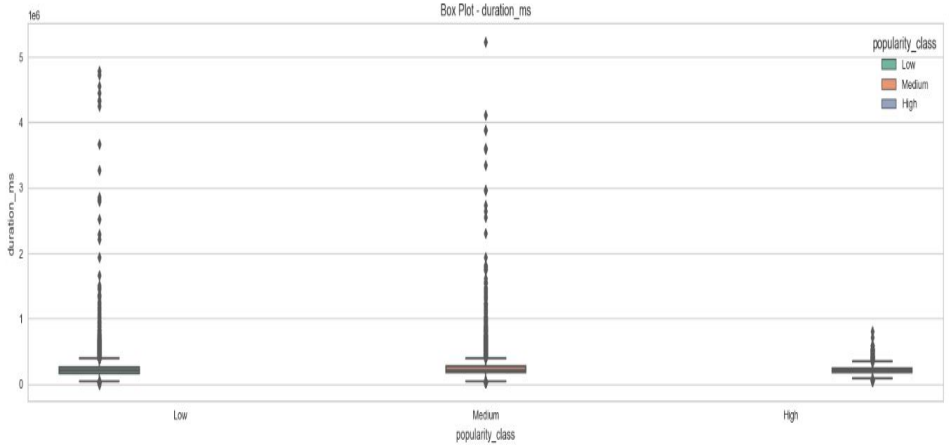
Popularity Class Definition

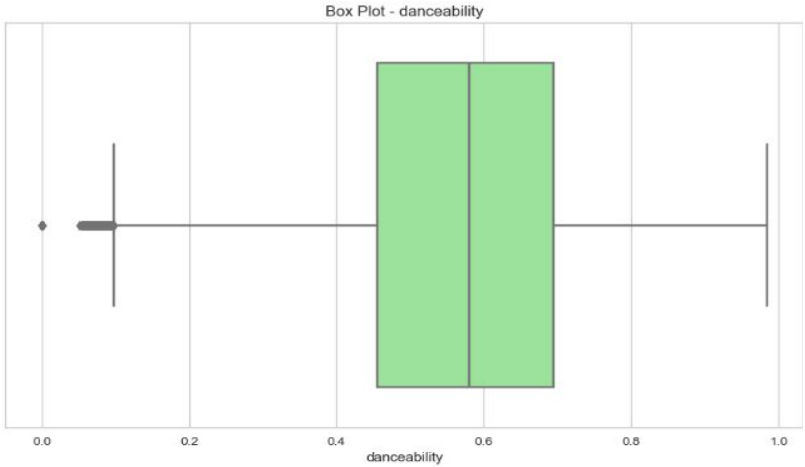
1. **Low Popularity:** Below 25th percentile(17)
2. **Medium Popularity:** Between 25th(17) and 90th(63) percentiles
3. **High Popularity:** Above 90th(63) percentile

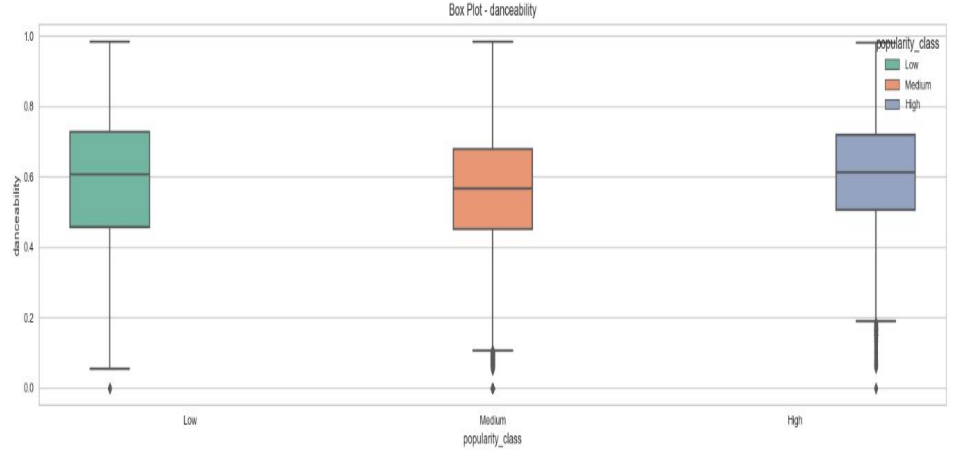


Exploratory Data Analysis: Numerical Features Part 1

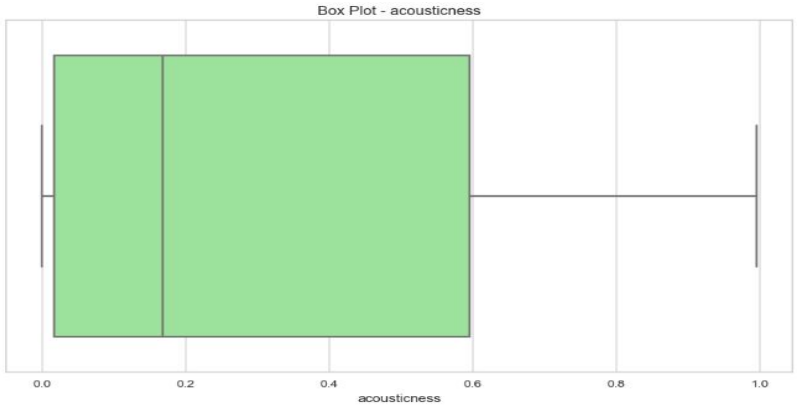


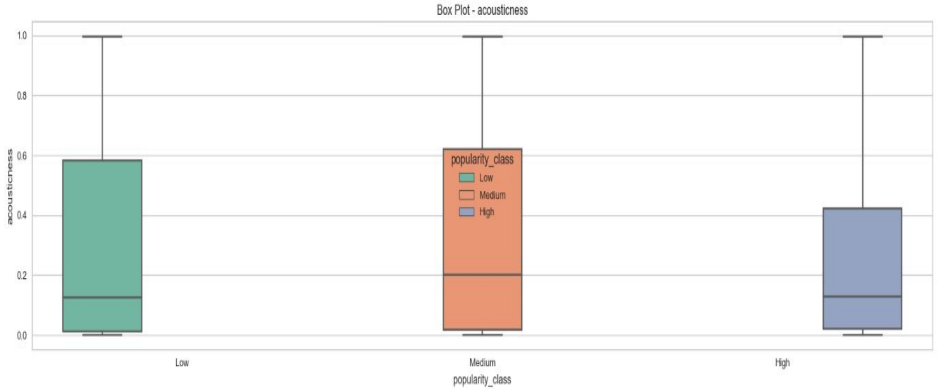


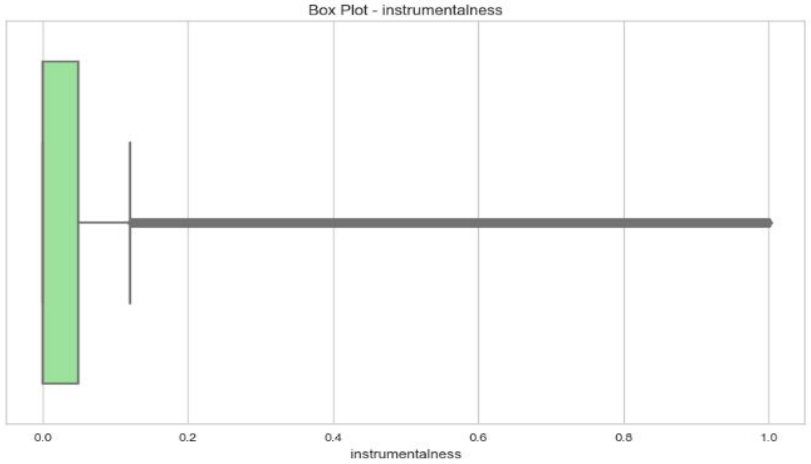


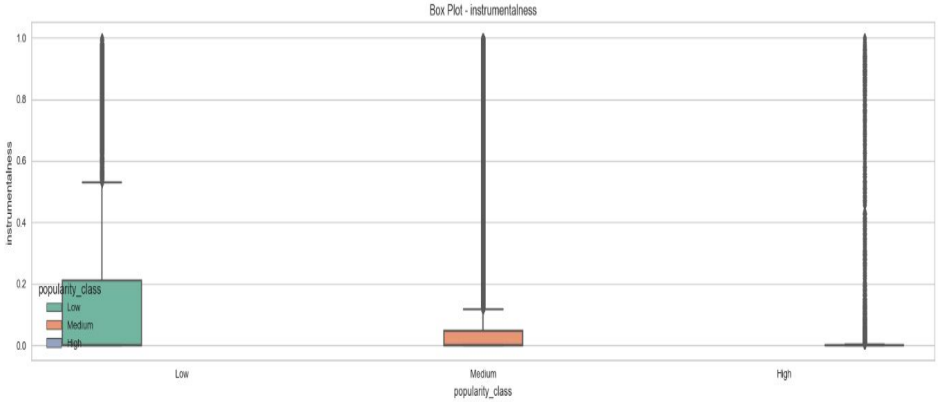


Exploratory Data Analysis: Numerical Features Part 2

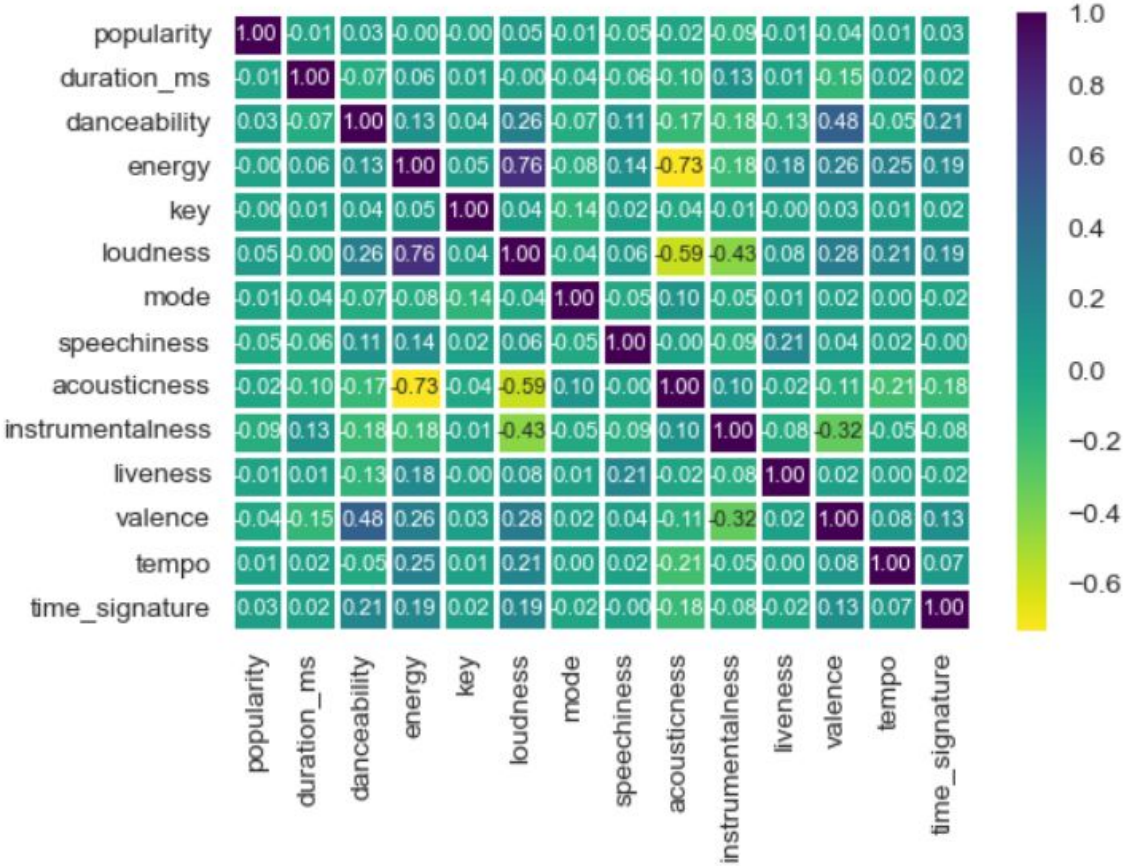




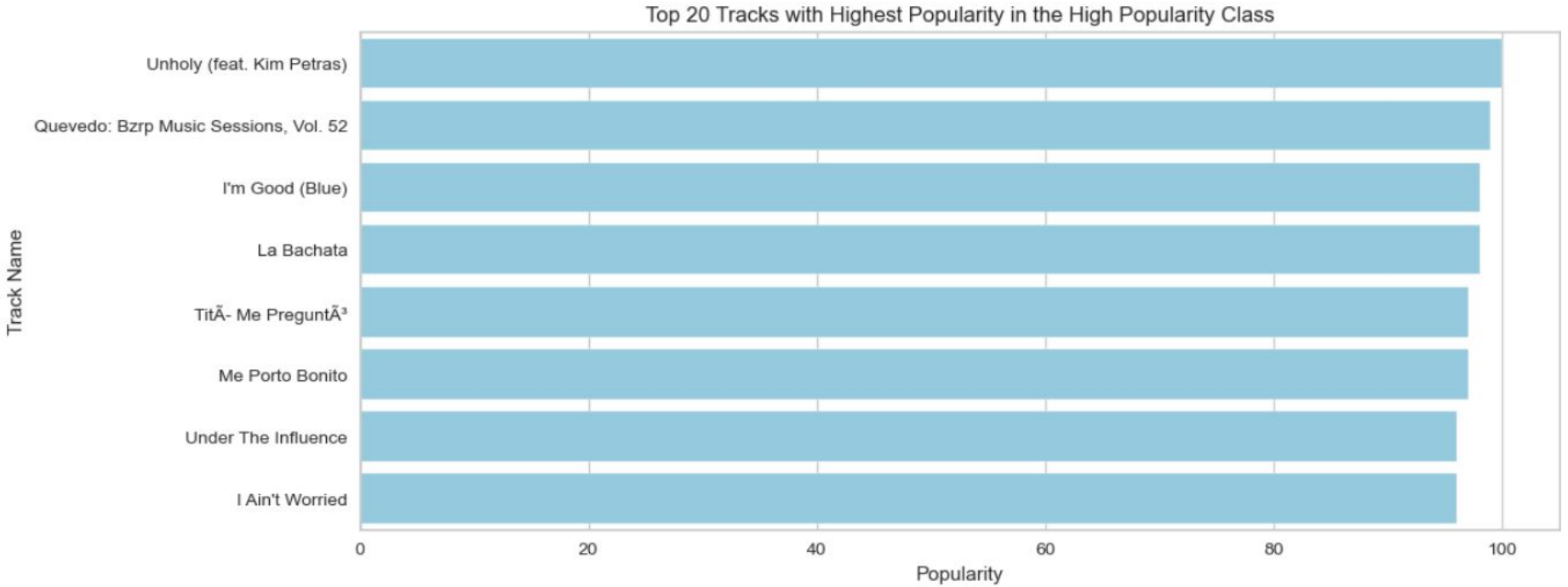




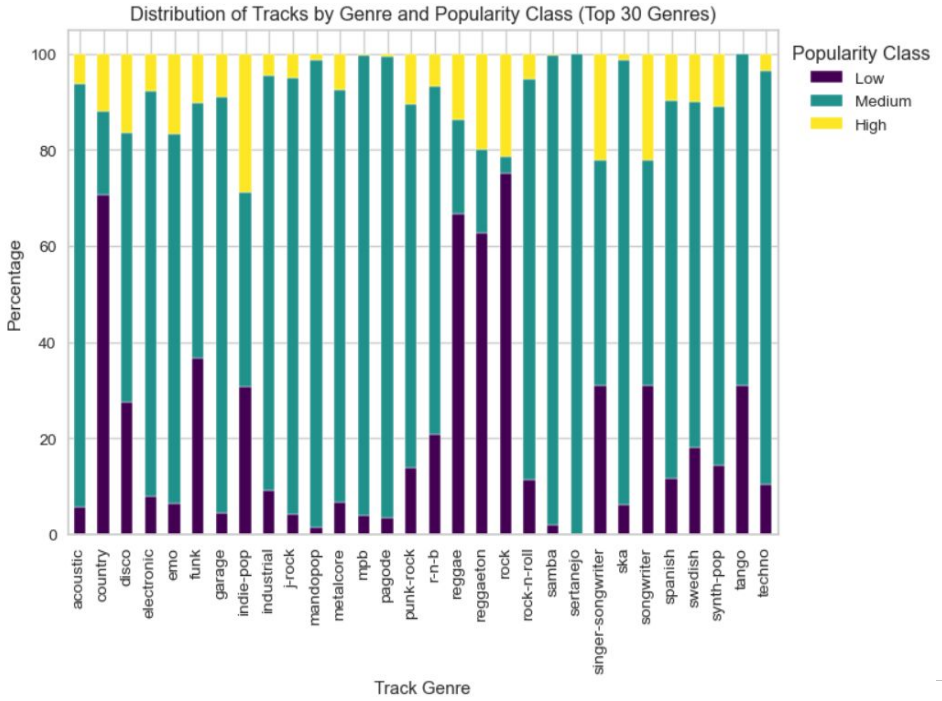
Exploratory Data Analysis: Numerical Feature Correlation

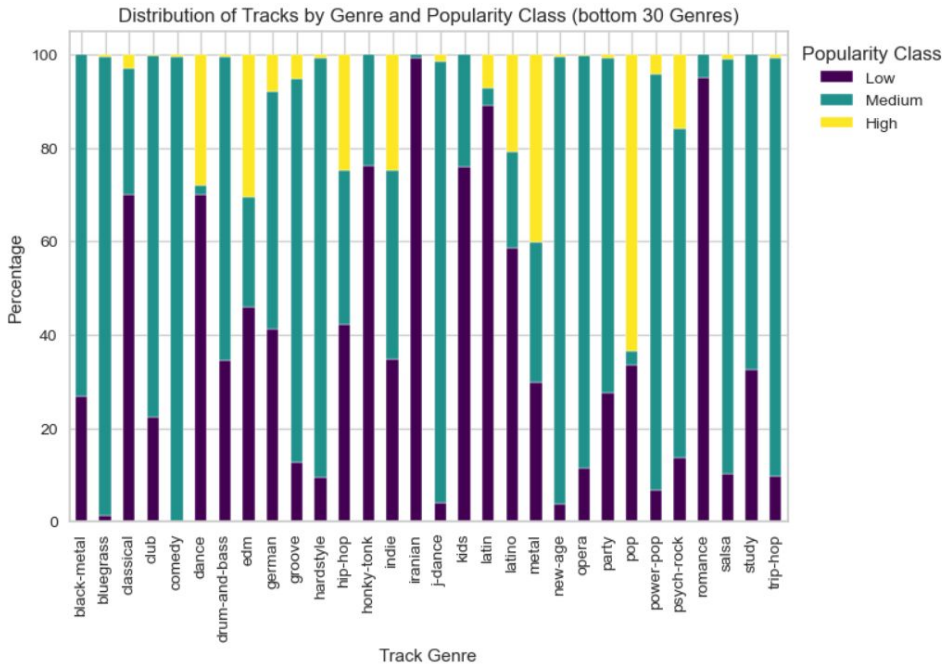


Exploratory Data Analysis: Top High popular tracks

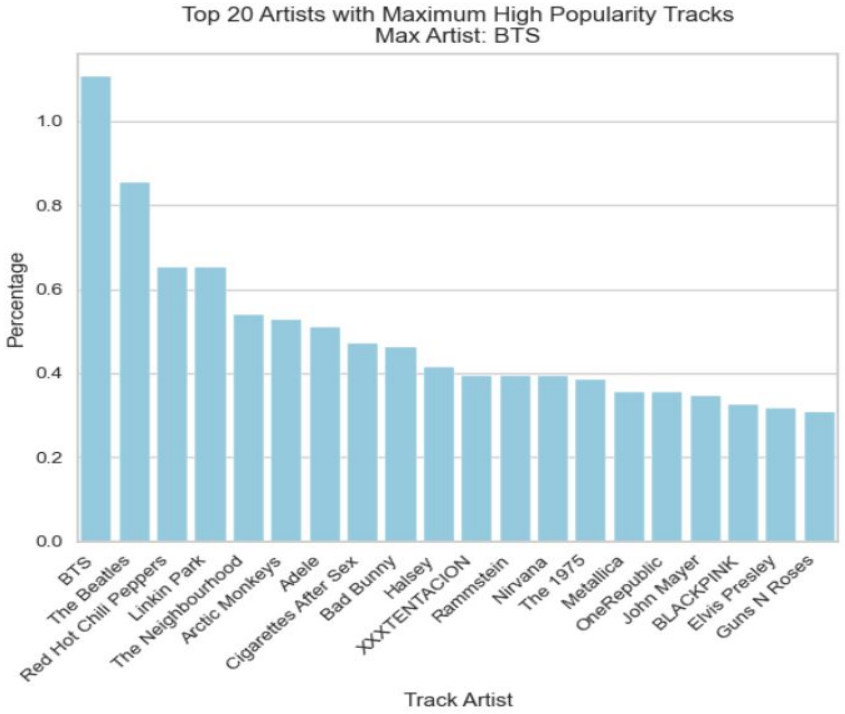


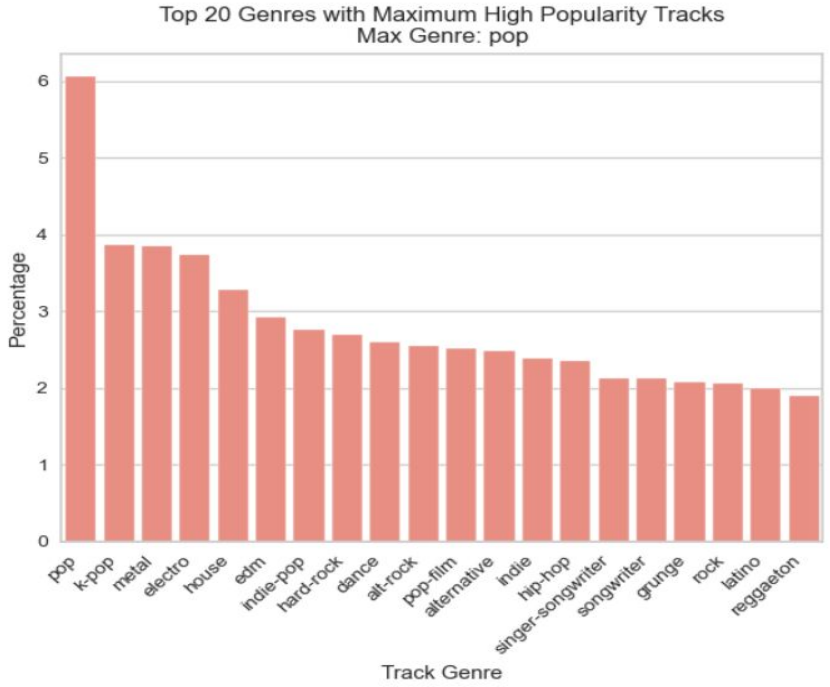
Exploratory Data Analysis: Popularity distribution of Track Genre





Exploratory Data Analysis: High popular Track genre and Artists





Exploratory Data Analysis: Summary

1. There is an observed trend where songs with higher popularity tend to have a shorter duration.
2. The danceability feature shows relatively consistent values across different popularity classes, suggesting that this attribute does not significantly differentiate between tracks of varying popularity.
3. High popular tracks consistently exhibit lower values for acousticness compared to tracks in other popularity categories.
4. Similarly, instrumentalness tends to be lower in high popular tracks compared to tracks in other popularity categories.
5. Features like loudness and energy show a comparatively higher positive correlation (0.73), while energy and acousticness demonstrate a notable negative correlation (-0.73).
6. Genres such as Pop, Kpop, Metal, EDM, Dance, and Indie Pop feature a higher prevalence of highly popular tracks.
7. Renowned artists like BTS, The Beatles, Red Hot Chilli Peppers, and Linkin Park have a notable presence among tracks with high popularity.

Model Training: Logistic Regression(Base)

- Data Preparation

1. Creating a dataframe for features(Removing popularity as a feature as it has direct correlation with the target variable)

```
features_df = spotify_df.drop(['track_id', 'artists', 'album_name', 'track_name', 'popularity','artists_cleaned','popularity_class'],axis=1)
```

2. Creating dummies for track_genre

```
features_encoded_df = pd.get_dummies(features_df, columns=['track_genre'], prefix='genre')
```

3. Scaling the features using standardisation as Logistic Regression is scale sensitive

```
features_scaled_encoded_df = pd.DataFrame(std_scaler.fit_transform(features_encoded_df), columns=features_encoded_df.columns)
features_scaled_encoded_df.head()
```

4. Splitting the dataset into train and test(70% training and 30% testing)

```
X_train_sc, X_test_sc, y_train_sc, y_test_sc = train_test_split(features_scaled_encoded_df, spotify_df['popularity_class'].values, test_size=0.3
```

5. Resampling the training data to handle class imbalance using oversampling

```
# Create the SMOTE object
smote = SMOTE(random_state=123)

# Use fit_resample to both fit the method and oversample the data
X_train_sc_resampled, y_train_sc_resampled = smote.fit_resample(X_train_sc, y_train_sc)
```

Model Training: Logistic Regression(Base)

- **Fitting the Model to Training data**

1. Getting the best parameters for Logistic Regression using GridSearch and Hyperparameter Tuning

```
#logistic regression model
lr = LogisticRegression()

#hyperparameter
lr_params = {'penalty': ['l1', 'l2'], 'solver': ['lbfgs', 'liblinear'], 'C': [1e-2, 0.1, 1]}

#gridsearch for hyperparameter tuning
#StratifiedKFold for model cross-validation (to handle overfitting)
grid_model_lr = GridSearchCV(estimator=lr, param_grid=lr_params, cv=StratifiedKFold(), verbose=4, n_jobs=2)

#fitting the model
grid_model_lr.fit(X_train_sc_resampled, y_train_sc_resampled)
```

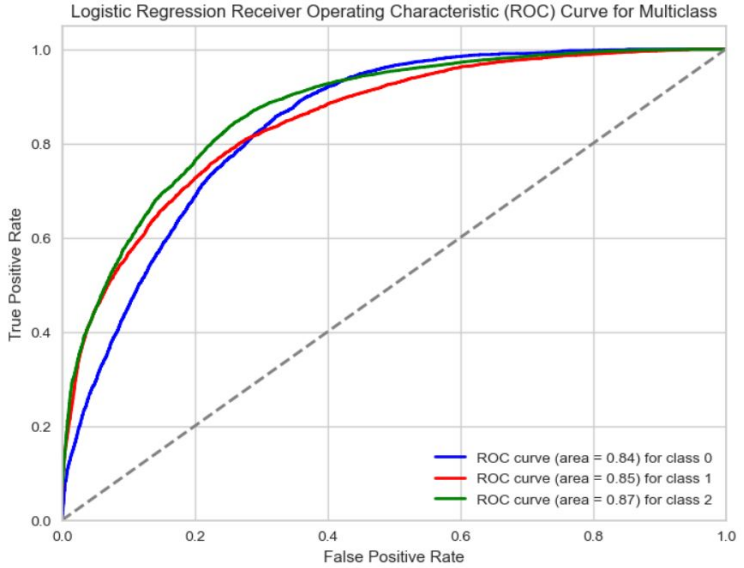
2. Best Parameter for Logistic Regression Model

```
# best logistic regression parameters
grid_model_lr.best_params_
```

```
{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
```

Model Results: Logistic Regression(Base)

```
Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.69      0.37      3158
           1       0.65      0.56      0.61      8573
           2       0.89      0.71      0.79     22334

    accuracy                           0.67     34065
   macro avg       0.60      0.66      0.59     34065
weighted avg       0.77      0.67      0.71     34065

Confusion Matrix:
[[ 2180   438   540]
 [ 2317  4840  1416]
 [ 4231  2143 15960]]
Logloss:  0.7166003369747166
R-Squared -0.7280639450007478
auc:  0.8398059108683258
```



- Class 0 - High
- Class 1 - Low
- Class 2- Medium

My main focus is to increase the precision and recall for ‘High’ class

Model Training: Ensemble Models

- **Data Preparation**

1. Creating a dataframe for features(Removing popularity as a feature as it has direct correlation with the target variable)

```
features_df = spotify_df.drop(['track_id', 'artists', 'album_name', 'track_name', 'popularity','artists_cleaned','popularity_class'],axis=1)
```

2. Creating dummies for track_genre

```
features_encoded_df = pd.get_dummies(features_df, columns=['track_genre'], prefix='genre')
```

3. Splitting the dataset into train and test(70% training and 30% testing)

```
X_train, X_test, y_train, y_test = train_test_split(features_encoded_df, spotify_df['popularity_class'].values,
                                                    test_size=0.3, random_state=123)
```

- **Selection of Models**

1. Random Forest
2. Gradient Boosting
3. Extreme Gradient Boosting(XGB)
4. Extreme Gradient Boosting(XGB) with Resampling(SMOTE)

Model Training: Ensemble Models

- **Fitting the Models to Training data**

1. Getting the best parameters for Logistic Regression using GridSearch and Hyperparameter Tuning

```
# initializing ensemble models
rf_model = RandomForestClassifier(random_state = 123)
gboost_model = GradientBoostingClassifier(random_state = 123)
xgboost_model = XGBClassifier(random_state = 123)
```

```
# individual model hyperparameters
rf_params = {'n_estimators':[100, 200,300], 'max_depth':[2,3,4], 'min_samples_split':[4, 6,10], 'min_samples_leaf':[2,3,4]}
gbm_params = {'n_estimators':[100,200], 'learning_rate':[1e-2, 0.1, 1.0], 'max_depth':[2,3,4], 'min_samples_split':[2, 4, 6]}
xgb_params = {'n_estimators':[200,300],'learning_rate':[1e-2, 0.1, 1.0], 'max_depth':[4,6,10],
              'reg_alpha': [1e-2, 0.1], 'reg_lambda': [1e-2, 0.1],"gamma" :[0,10,30]}
```

2. Best Parameters for the Ensemble Models

Random Forest

n_estimators: 100, max_depth: 4, min_samples_split: 4, min_samples_leaf: 2

Gradient Boost

n_estimators: 200, learning_rate: 1.0, max_depth: 4, min_samples_leaf: 1

XGBoost

n_estimators: 300, learning_rate: 1.0, max_depth: 10, reg_alpha: 0.1, reg_lambda: 0.1, gamma: 0

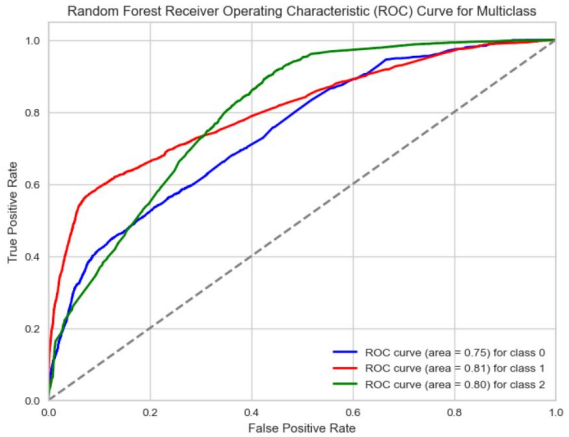
XGBoost with SMOTE

n_estimators: 300, learning_rate: 1.0, max_depth: 10, reg_alpha: 0.1, reg_lambda: 0.1, gamma: 0

Model Results: Ensemble Models

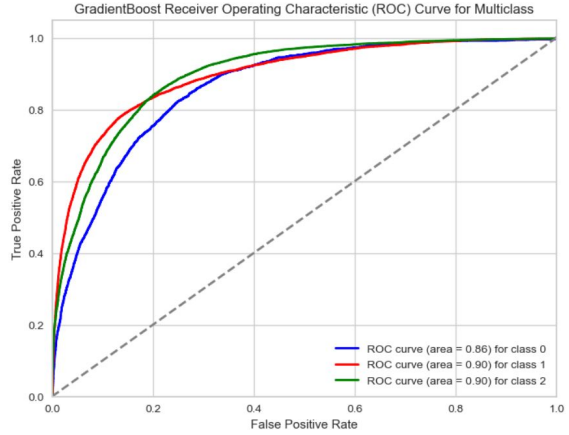
- **Random Forest**

```
Showing summary for random_forest model
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3158
           1       0.97      0.03      0.06      8573
           2       0.66      1.00      0.80     22334

    accuracy                           0.66     34065
   macro avg       0.54      0.34      0.29     34065
weighted avg       0.68      0.66      0.54     34065

Confusion Matrix:
[[    0     0  3158]
 [    0   274  8299]
 [    0     9 22325]]
Logloss:  0.7896336092437678
R-Squared -0.4247444290225868
auc:  0.7540658665953025
```



- **Gradient Boost**

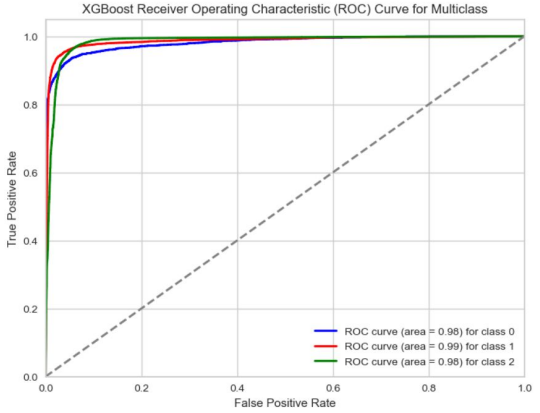
```
Showing summary for gradient_boost model
Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.22      0.30      3139
           1       0.72      0.66      0.69      8626
           2       0.83      0.92      0.88     22435

    accuracy                           0.79     34200
   macro avg       0.67      0.60      0.62     34200
weighted avg       0.77      0.79      0.78     34200

Confusion Matrix:
[[  689   908  1542]
 [  348  5691  2587]
 [  447  1315 20673]]
Logloss:  0.6172267447287477
R-Squared 0.10710113437997837
auc:  0.8419143253223279
```



Model Results: Ensemble Models

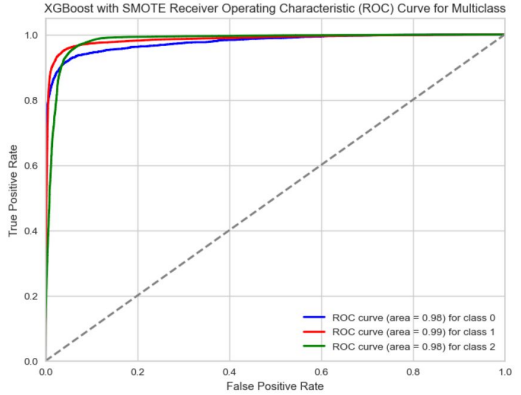
- **XGBoost**

```
Showing summary for xgboost model
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.83      0.88      3158
           1       0.94      0.93      0.93      8573
           2       0.96      0.98      0.97     22334

    accuracy                           0.95     34065
   macro avg       0.94      0.91      0.93     34065
weighted avg       0.95      0.95      0.95     34065

Confusion Matrix:
[[ 2614   183   361]
 [   71  7962   540]
 [  130   353 21851]]
Logloss:  0.19121568971192512
R-Squared 0.7883295167770168
auc:  0.9820184261110535
```

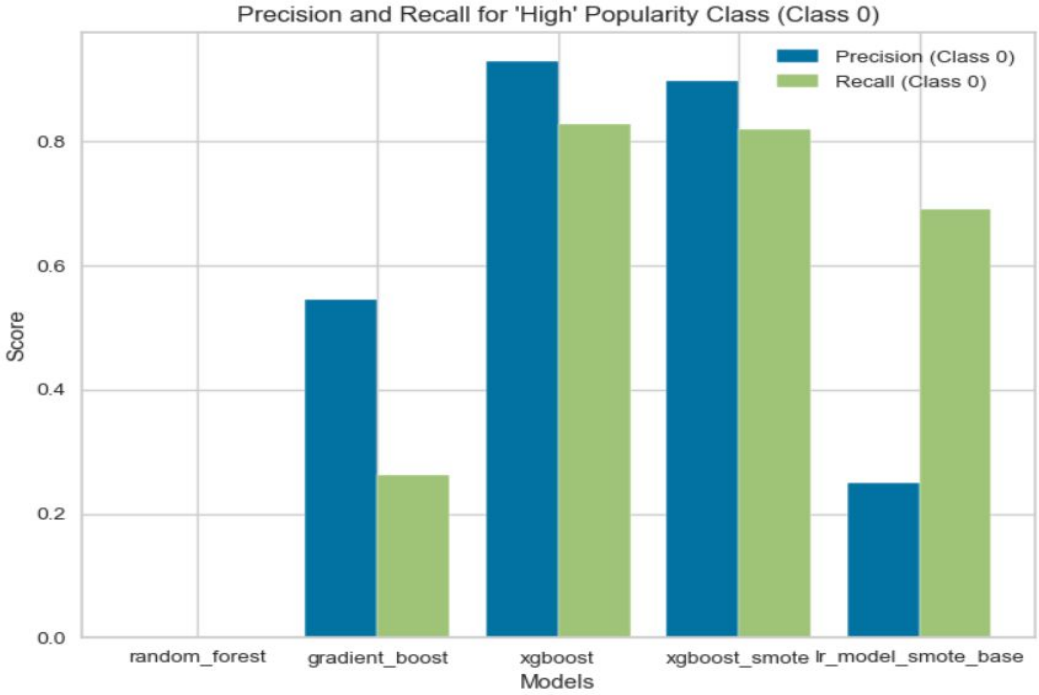


- **XGBoost with SMOTE**

```
Showing summary for xgboost_smote model
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.82      0.86      3158
           1       0.93      0.93      0.93      8573
           2       0.96      0.97      0.97     22334

    accuracy                           0.95     34065
   macro avg       0.93      0.91      0.92     34065
weighted avg       0.95      0.95      0.95     34065

Confusion Matrix:
[[ 2586   183   389]
 [   85  7965   523]
 [  210   436 21688]]
Logloss:  0.2024925176464667
R-Squared 0.7534933588181074
auc:  0.9794320064355961
```

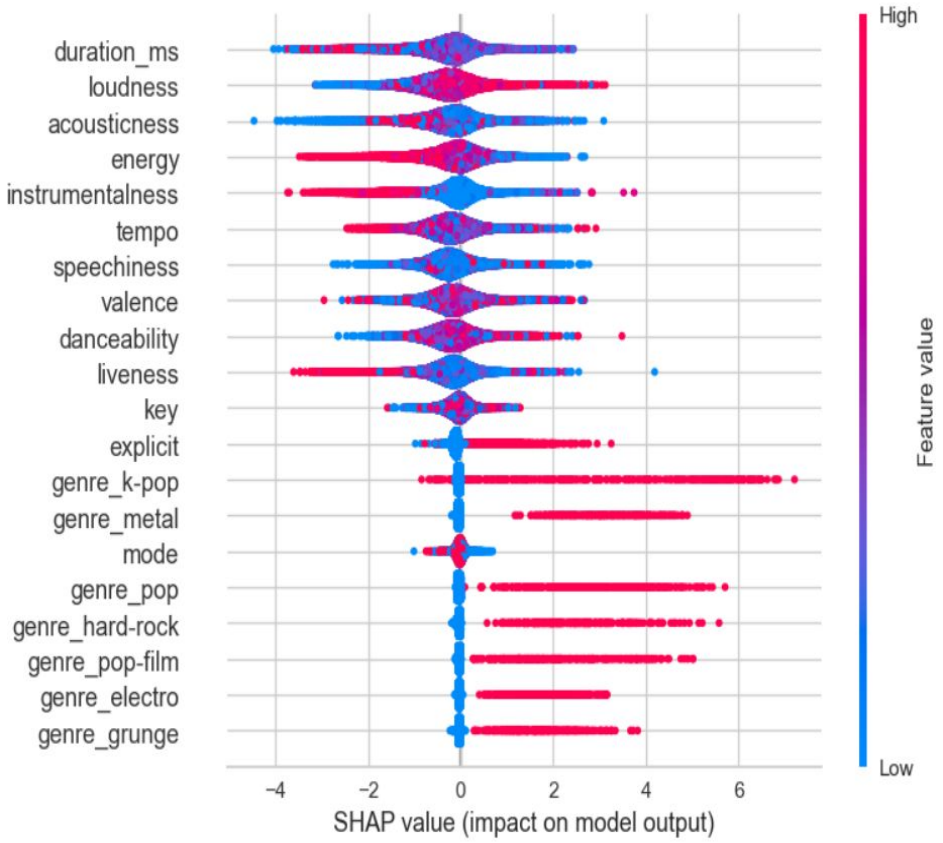


Model Comparison and Selection



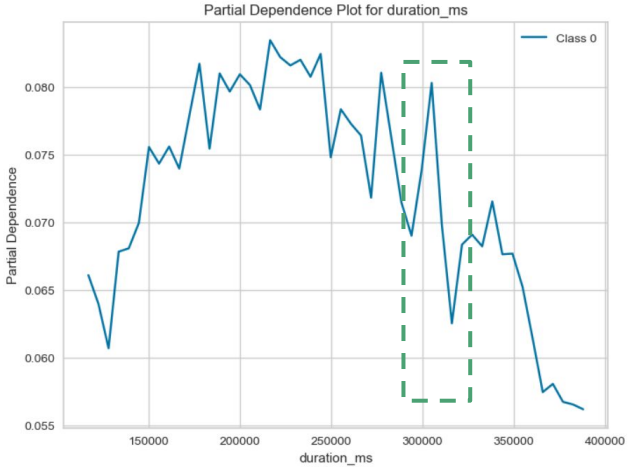
It is very clear from the model comparison in terms of predicting the High popularity class that **XGBoost Model is the best for our case**.

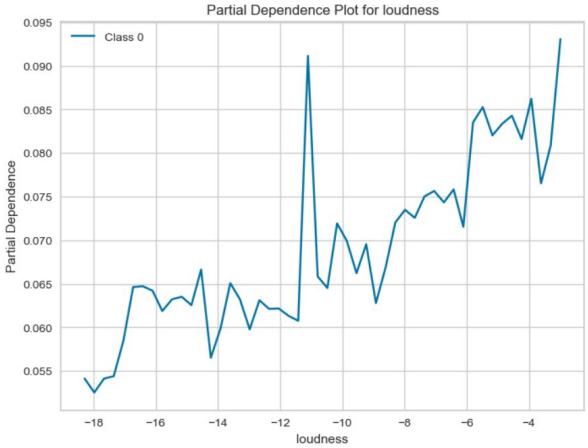
SHAP plot for High Popularity Class

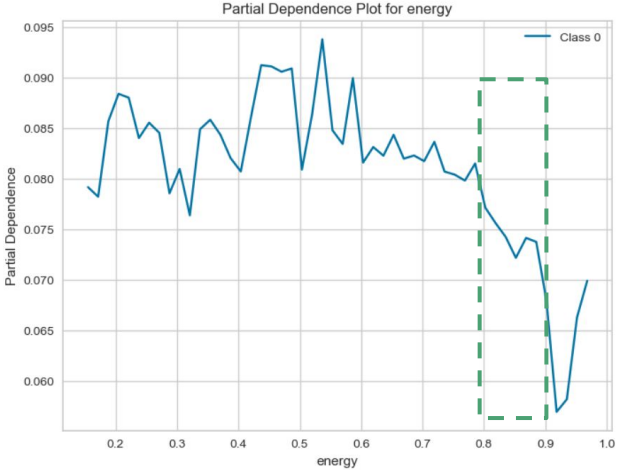


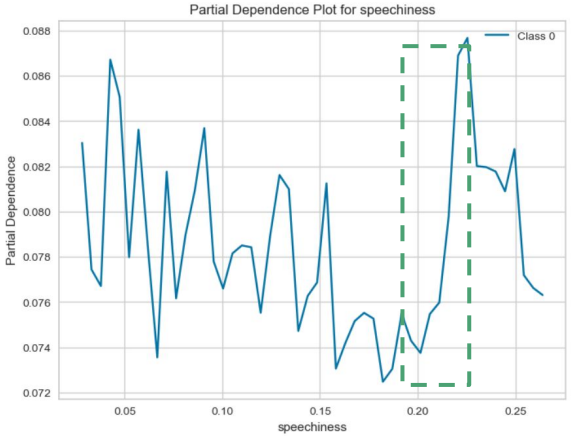
- **SHAP plots** provide a detailed view of feature importance, outlining the specific impact of each feature value on individual predictions.
- Unlike traditional methods, **SHAP plots reveal how individual feature values intricately influence predictions**, offering a more nuanced perspective in complex models.
- **Loudness** exhibits a positive impact on predicting high popularity, whereas **energy** shows a counterintuitive negative impact despite their high positive correlation.
- **Instrumentalness, tempo, and liveness** negatively affect model output, contributing to counterintuitive predictions.
- **Genres like Kpop, Pop, Metal, Hard Rock, Pop Film Electro, and Grunge** positively impact predicting the high popularity class.

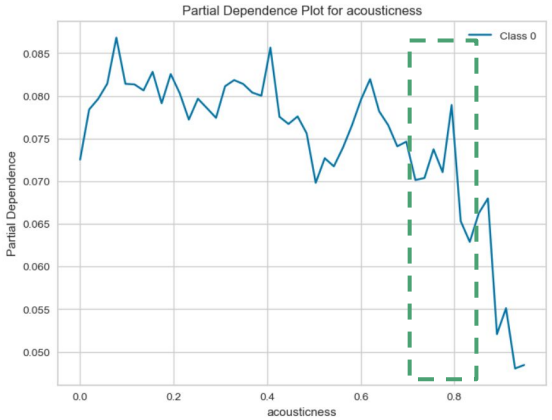
Partial Dependence : Plots for High Popularity Class(Cont. Var)

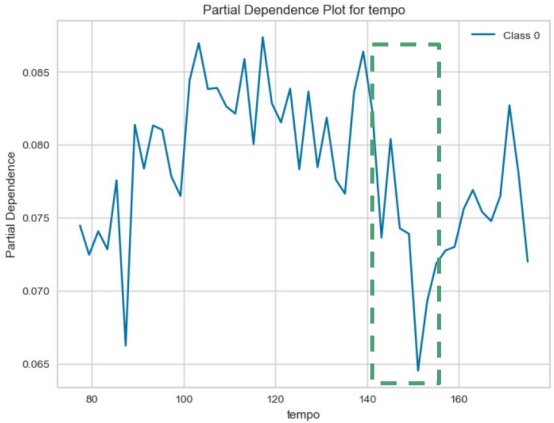












Partial Dependence : Observations

1. **A partial dependence plot** represents the predicted probability or the predicted class probability for a specific class. It shows how the probability of a particular outcome changes as the values of a specific feature vary while keeping other features constant.
2. **Duration** : As the duration of a song goes beyond 300K ms the probability of it being highly popular decreases.
3. **Loudness**: The probability of a song being high popular increase gradually with loudness.
4. **Energy**: If the energy in a song goes beyond 80% then the probability of it being highly popular decreases.
5. **Speechiness**: As the number of words in a song goes beyond 20% there is spike in probability of it being a highly popular song.
6. **Acousticness**: As the acousticness level in a song goes beyond 80% the probability of it being a highly popular song decreases drastically.
7. **Tempo :** If the tempo of a song is between 140 -160 the probability of it being a highly popular song decreases.

Summary and Conclusion

Conclusion:

This research presents a nuanced examination of the factors influencing a song's classification as highly popular on Spotify. Utilizing machine learning models and data analysis, we identified patterns, including the impact of song duration, danceability, and loudness, as well as the significance of acousticness and instrumentalness in highly popular tracks. The importance of track genre emerges as a key determinant, revealing diverse listener preferences. Employing advanced methodologies like SHAP plots and partial dependence, we delved into the intricate interplay of features, offering actionable insights for the music industry. These findings provide stakeholders with valuable tools for tailoring content, navigating genre preferences, and enhancing the likelihood of creating highly popular tracks. The precision-driven XGBoost model stands out as a strategic asset for industry players navigating the dynamic landscape of music popularity.

Limitations:

While the study provides valuable insights, several limitations must be acknowledged. The absence of specific features like recency, stream count, and artist popularity, which play pivotal roles in Spotify's popularity metric, poses challenges. The dataset's reliance on imputed data and the oversampling technique for class imbalance introduces potential biases. Additionally, the models' interpretability relies on feature importance tools, which may not capture all nuances. Continuous updates to the dataset and considerations for evolving user preferences could enhance the study's robustness over time.