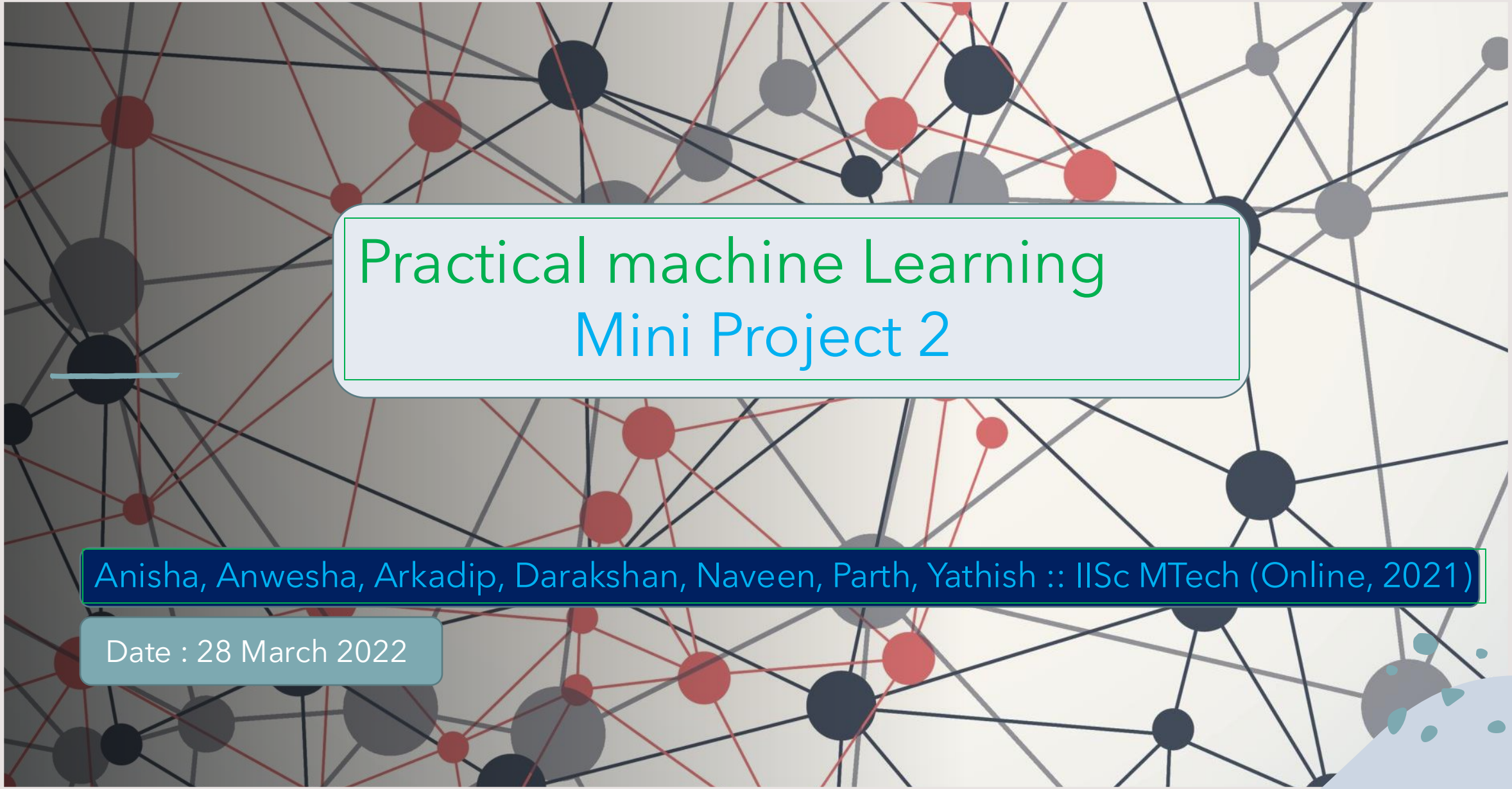


Practical machine Learning
Mini Project 2

Anisha, Anwesha, Arkadip, Darakshan, Naveen, Parth, Yathish :: IISc MTech (Online, 2021)

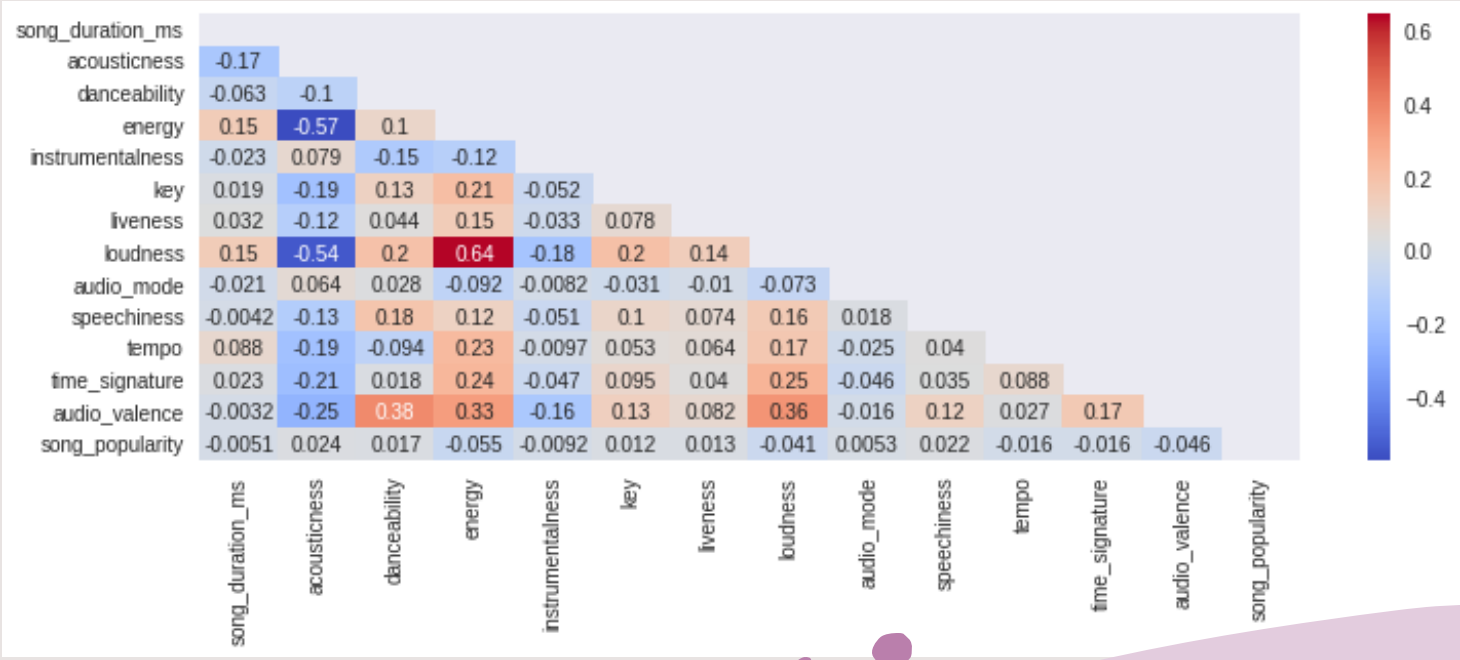
Date : 28 March 2022

Business Objective:

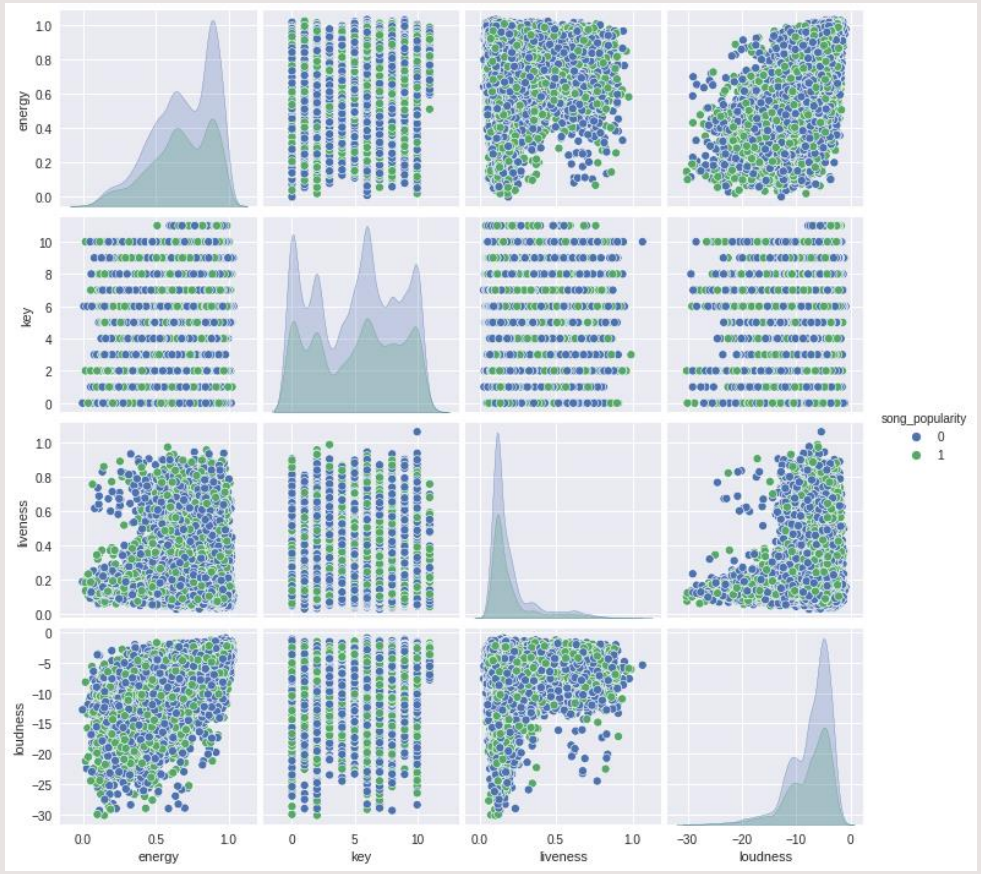
To predict, if a song is popular given various features of the song.

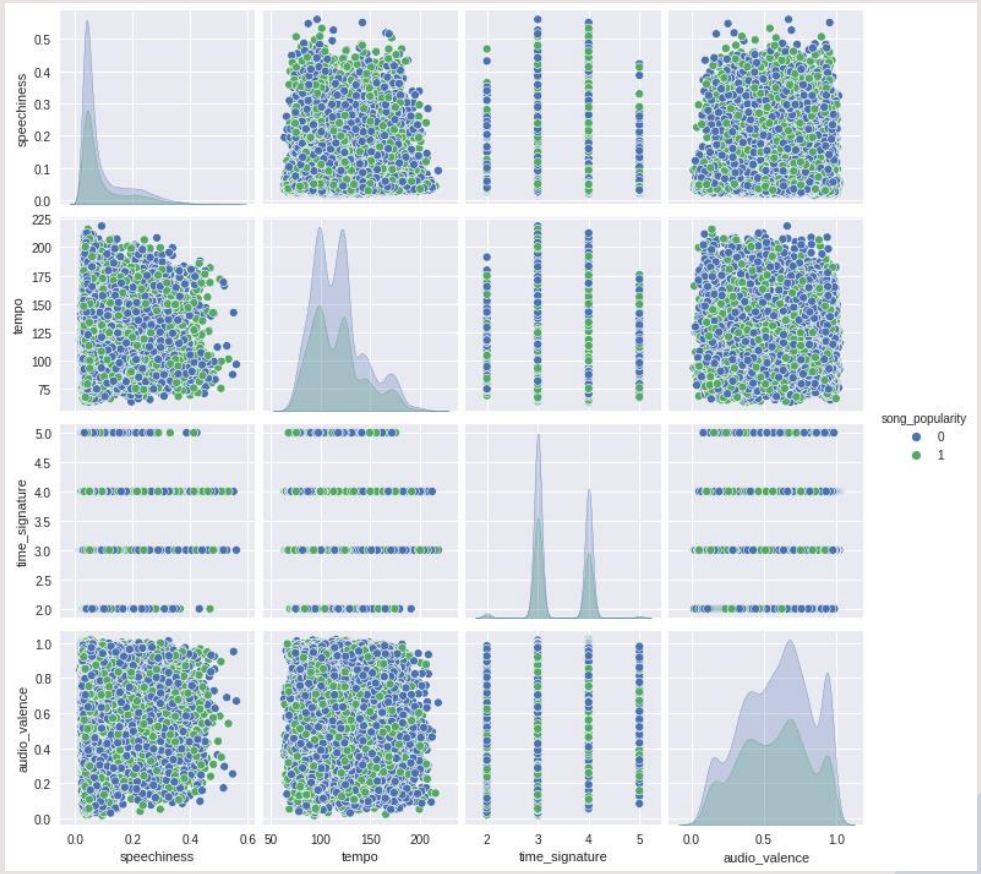
Correlation Heatmap

- Loudness and energy are positively correlated
- Acousticness is negatively correlated to loudness and energy
- Audio valence, energy, danceability and loudness have mild positive correlation



Paired Feature Scatterplots

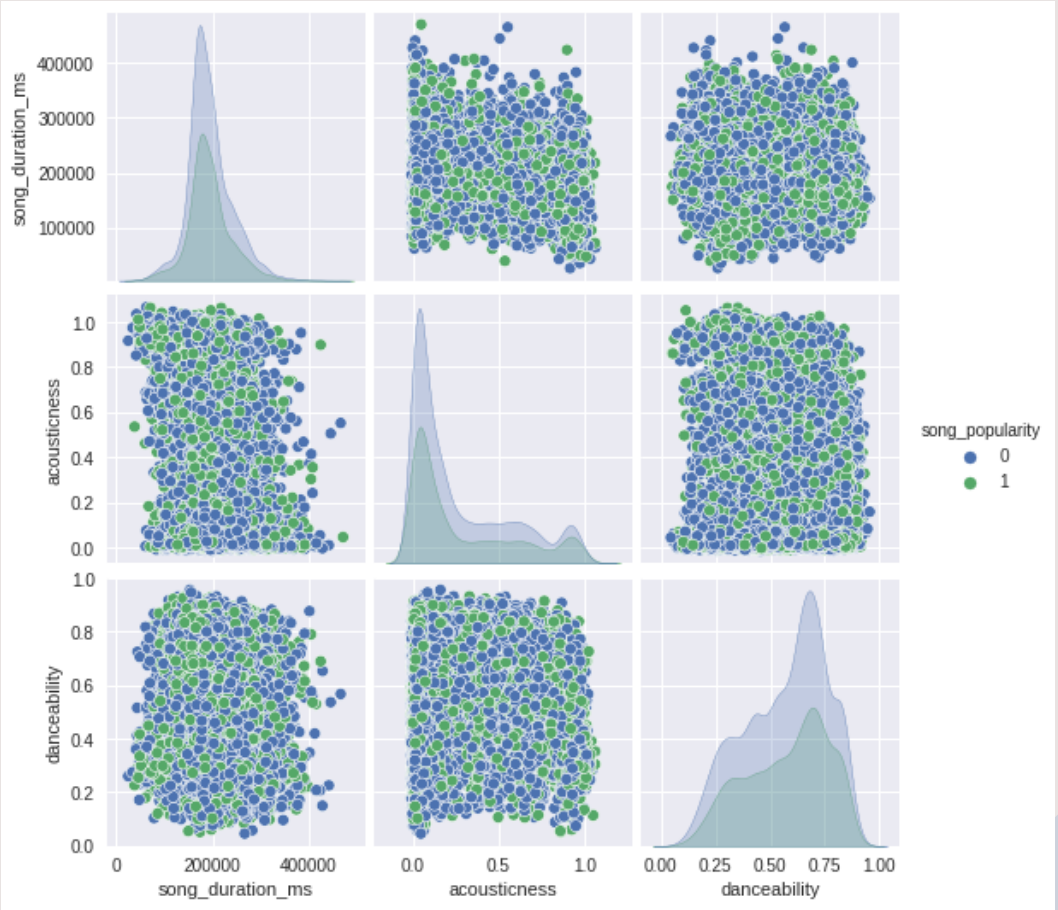




Paired Feature Scatterplots

- None of the pairs of features seem to be able to shatter the data based on song popularity in any significant capacity

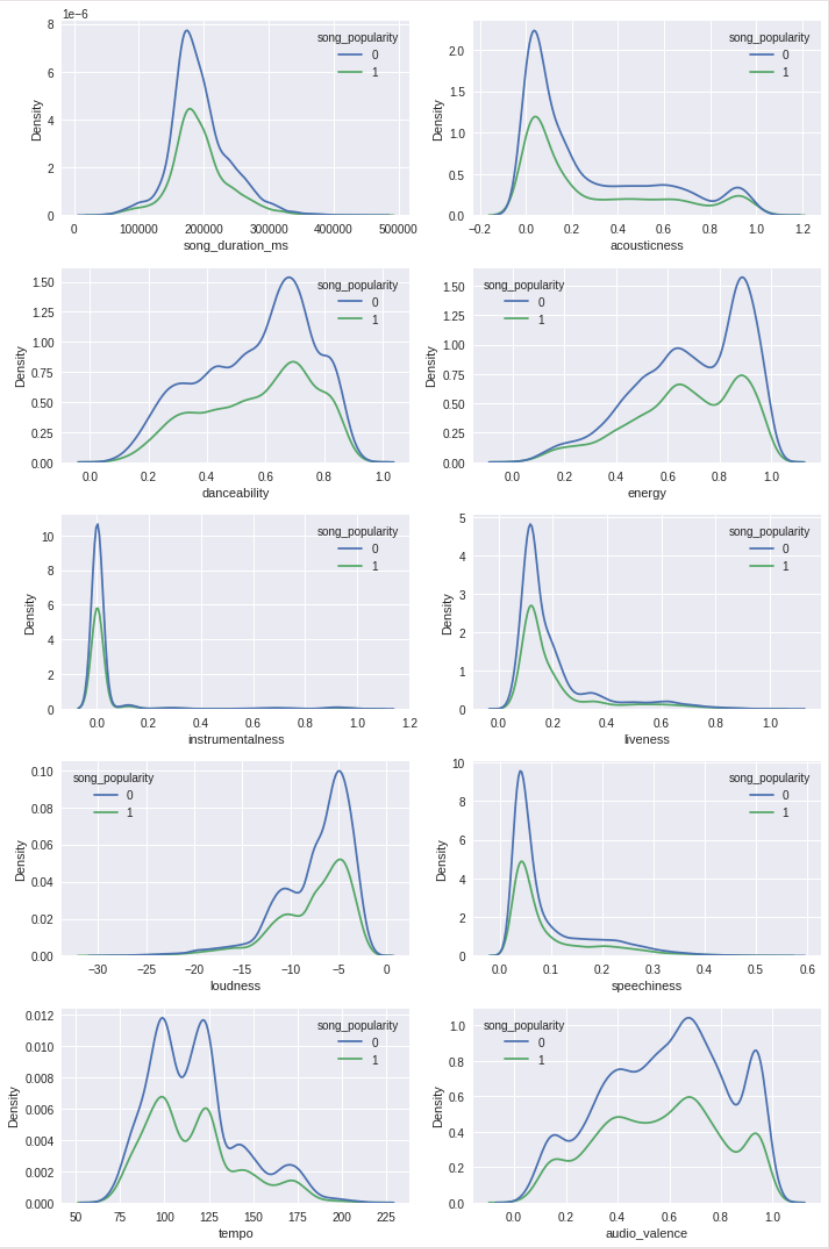
- The plots along the diagonals look interesting and are worth taking a closer look at



Numerical Feature KDE plots

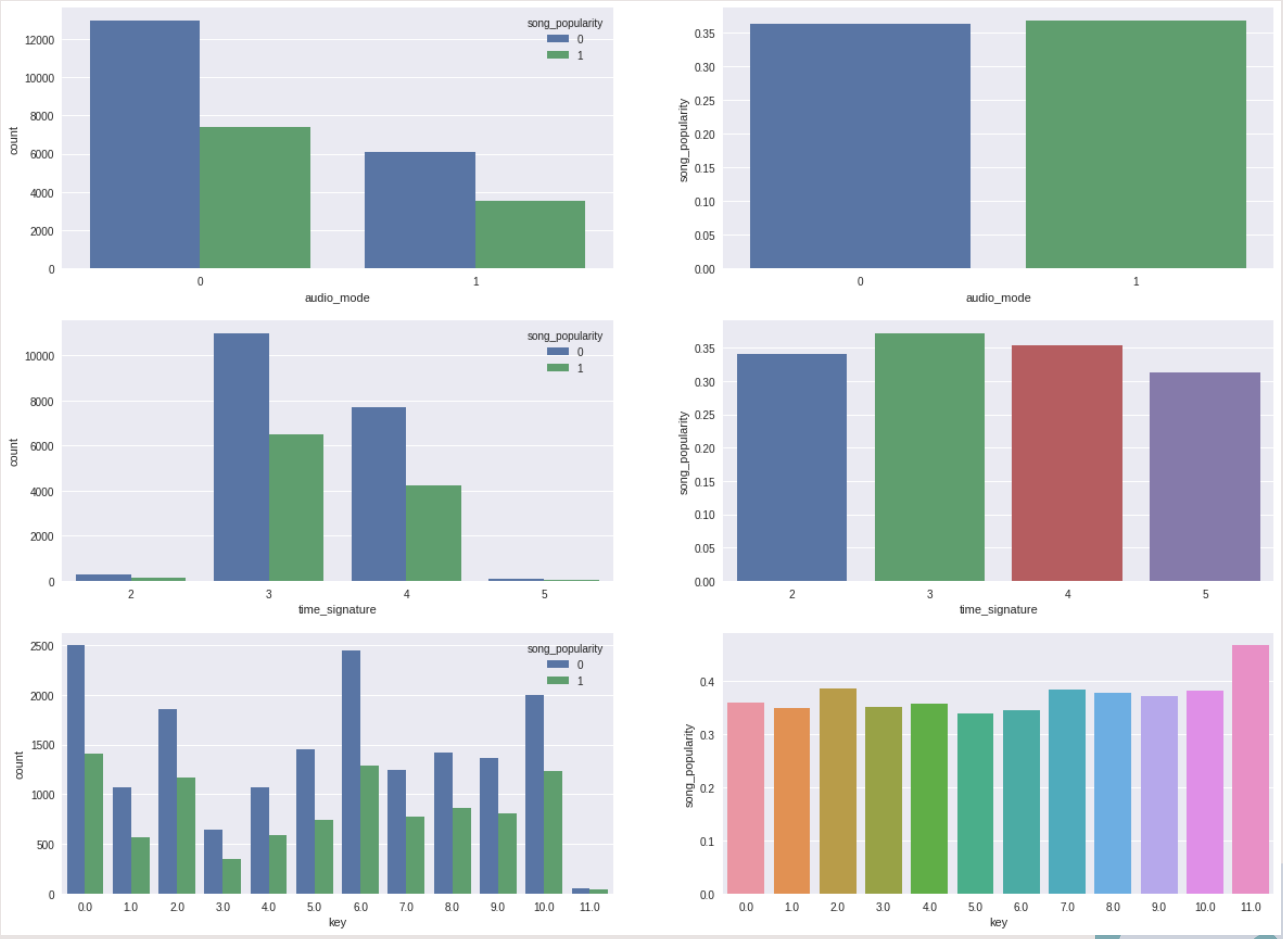
- Samples belonging to both labels seem to have identical kernel distribution estimates on each of the numerical features

- Popular song KDE looks like unpopular song KDE scaled to half



Categorical Feature Bar plots

- Percentage of popular song samples seems to be more or less consistent across the unique values of each of the categorical features
- The biggest deviation is observed for key 11, but even that is not a significant change and key 11 has significantly lower number of samples than other keys



Data Analysis and Imputation

	song_duration_ms	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	audio_mode	speechiness	tempo	time_signature	audio_valence	song_popularity
0	212990.0	0.642286	0.856520	0.707073	0.002001	10.0	NaN	-5.619088	0	0.082570	158.386236	4	0.734642	0
1	NaN	0.054866	0.733289	0.835545	0.000996	8.0	0.436428	-5.236965	1	0.127358	102.752988	3	0.711531	1
2	193213.0	NaN	0.188387	0.783524	-0.002694	5.0	0.170499	-4.951759	0	0.052282	178.685791	3	0.425536	0
3	249893.0	0.488660	0.585234	0.552685	0.000608	0.0	0.094805	-7.893694	0	0.035618	128.715630	3	0.453597	0
4	165969.0	0.493017	NaN	0.740982	0.002033	10.0	0.094891	-2.684095	0	0.050746	121.928157	4	0.741311	0

- **key**, **audio_mode**, **time_signature** and **song_popularity** have finite discrete values and are identified as Categorical columns.
- **song_duration_ms**, **acousticness**, **danceability**, **energy**, **instrumentalness** and **key** has missing data.
- We have used **KNN imputer** for Numerical data and **Simple Imputer** for categorical features.
- We tried Linear regression for Imputation but there was no significant improvement in overall scores.
- We also experimented with Mode, Median imputers and missing forest imputer but finally went with KNN and Simple imputer as those provided the best results for the given dataset.

Feature Engineering and Test-Train Split

- The class labels are not balanced in the dataset. Class 0 and Class 1 have 64% and 36% distribution in data respectively. To maintain the same ratio in train and test data, stratified split on the target variable **'song_popularity'** is applied.
- We also tried feature engineering on loudness, liveness and energy but there was no significant improvement in AUC or F1 score. This is also evident from the pair-plots and correlation matrix in EDA as there is not much correlation between the features.
- We have used **Standard Scaler** to scale the numerical features and **One Hot Encoder** for categorical features
- Since the class is mildly imbalanced, prediction is done using **Cost-Sensitive** models where every wrong prediction of minority class is heavily penalized. The weights used here are inverse distribution of classes.

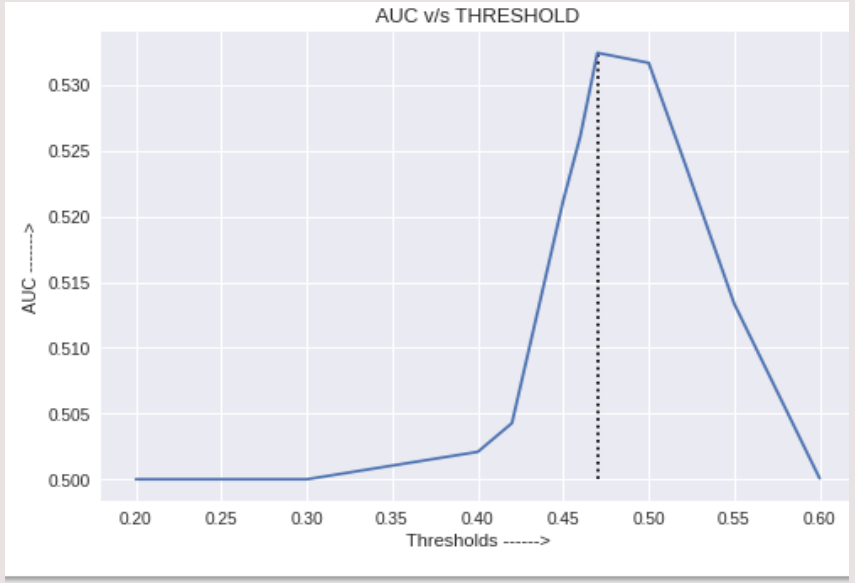
Logistic regression

	Accuracy Score	Confusion Matrix	AUC	F1 Score	Precision Score	Recall Score
Default	0.6348	4757 10 2729 4	0.4996	0.00291	0.2857	0.0014
Balanced	0.5393	2669 2098 1357 1376	0.5316	0.4433	0.3960	0.5034
Cutom weight w={0:36,1:64}	0.5318	2503 2264 1247 1486	0.5343	0.4584	0.3962	0.5437
THRESHOLD = 0.47	0.4717	1471 3296 666 2067	0.5324	0.5106	0.3854	0.7563

Logistic Regression

Using THRESHOLD of 0.47 performs well.

Note: We tried L1,L2 and elastic net regularization and there was no improvement in the score.




```
threshold_new[np.argmax(auc_list)]
```

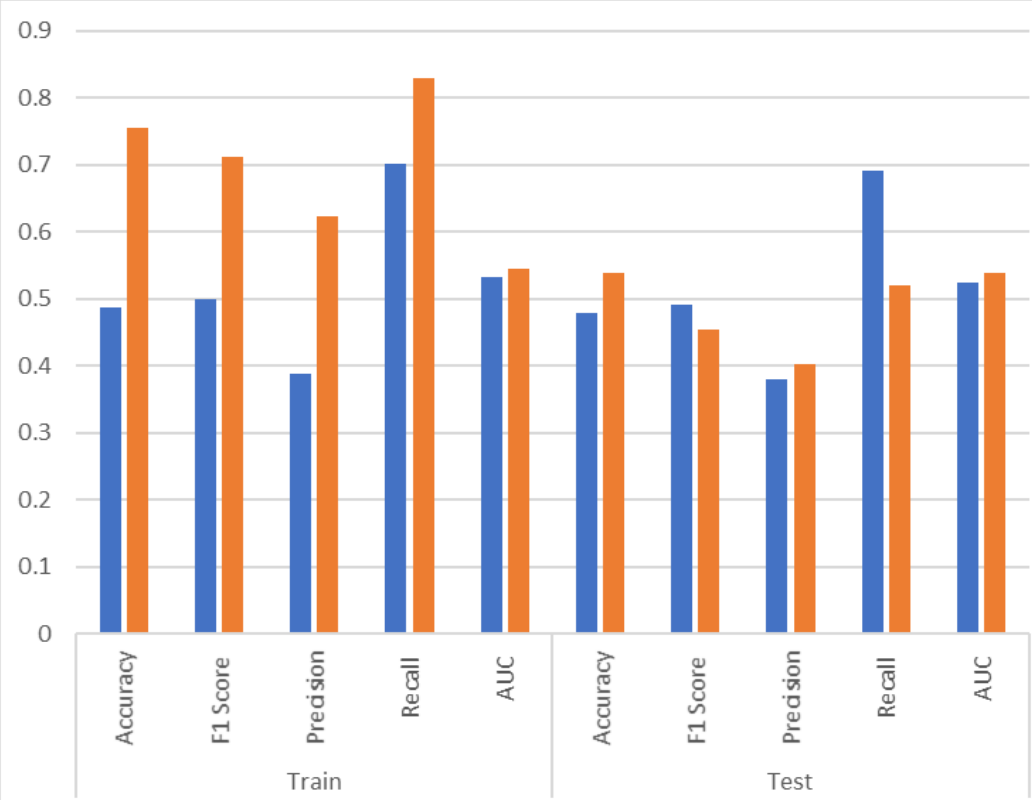
```
0.47
```

Support Vector Machine (SVM)

Gaussian SVM performs better than both Linear SVM and Polynomial SVM.

	Accuracy Score	Confusion Matrix	AUC	F1 Score	Precision Score	Recall Score
Linear(C=0.1)	0.5392	2669 2098 1358 1375	0.5315	0.4431	0.3959	0.5031
Polynomial (deg=2, C=0.1)	0.5356	2478 2289 1194 1539	0.5415	0.4691	0.4020	0.5631
Gaussian(C=0.1,gamma=0.1)	0.5395	2526 2241 1213 1520	0.5430	0.4681	0.4041	0.5561

Decision Trees/Random forest



Decision Tree Classifier criterion='gini'
class_weight={0:36, 1:64}
Regularization:
1. max_depth = 2
2. max_features=2
Cross Validation: RepeatedStratifiedKFold
i. Number of folds = 10
ii. Number of repetitions = 3

Random Forest Classifier
i. class_weight={0:36, 1:64}
ii. max_depth=10
iii. Max_features=7
iv. N_estimators=200

Cross Validation: RepeatedStratifiedKFold
i. Number of folds = 10
ii. Number of repetitions = 3

MODEL	ACCURACY	AUC	F1 SCORE	PRECISION	RECALL
Decision Tree	0.48	0.52	0.49	0.38	0.69
Random Forest	0.54	0.54	0.45	0.4	0.52

Random Forest performs much better than Decision Trees. Class weights was used as a parameter in both the models.

Hard Voting & Soft Voting Classifier

For Voting classifier **Logistic Regression , Gaussian RBF SVM , Decision Tree & Random Forest** has been combined to create an ensemble

Models	Accuracy	AUC	F1 Score	Precision	Recall
Logistic Regression	0.53	0.53	0.45	0.40	0.54
Gaussian	0.55	0.55	0.474	0.41	0.56
Decision Tree	0.48	0.52	0.49	0.38	0.69
Random Forest	0.54	0.54	0.45	0.41	0.55

Hard Voting Scores				
Accuracy	AUC	F1 Score	Precision	Recall
0.55	0.55	0.47	0.41	0.54

Soft Voting Scores (Default Threshold)				
Accuracy	AUC	F1 Score	Precision	Recall
0.61	0.52	0.26	0.42	0.19

Soft Voting Scores (Threshold = 0.47)				
Accuracy	AUC	F1 Score	Precision	Recall
0.56	0.55	0.46	0.42	0.52

Adaptive Boosting

- Ada Boost trains predictors sequentially by correcting its predecessor
- It assigns weights to misclassified training instances & trains second classifier using updated weights

Ada Boost scores without Hyperparameter tuning				
Accuracy	AUC	F1 Score	Precision	Recall
0.54	0.51	0.38	0.37	0.39

Confusion Matrix	Predicted	
Actual	2987(TN)	1780 (FP)
	1664(FN)	1069(TP)

Tuned Hyperparameters

Decision Tree (Base Estimator) Hyperparameters	Ada Boost Hyperparameters	
Max_depth	Learning rate	N_Estimators
2	0.1	140

Ada Boost scores with Hyperparameter tuning				
Accuracy	AUC	F1 Score	Precision	Recall
0.53	0.55	0.49	0.40	0.61

Confusion Matrix	Predicted	
Actual	2333 (TN)	2434 (FP)
	1061(FN)	1672(TP)

After Hyperparameter Tuning overall score of Ada boost model has improved

XG Boost –
***Inherited from G*radient boosting library.**

XG Boost scores				
Accuracy	AUC	F1 Score	Precision	Recall
0.6416	0.5099	0.0477	0.75	0.0246

Confusion-Matrix Predicted	
14234 (TN)	67 (FP)
7997 (FN)	202 (TP)

learning_rate=0.1, gamma=1, n_estimators=75

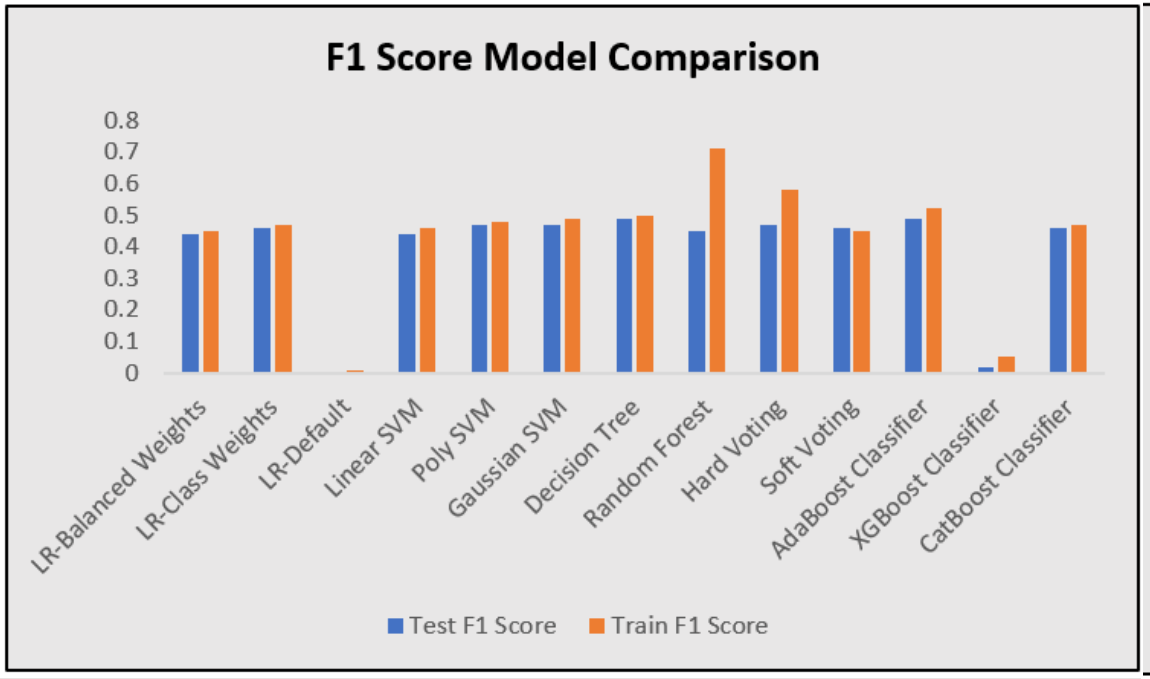
Categorical Boost –
***Inherited from G*radient boosting library.**
We can use CatBoost without any explicit pre-processing

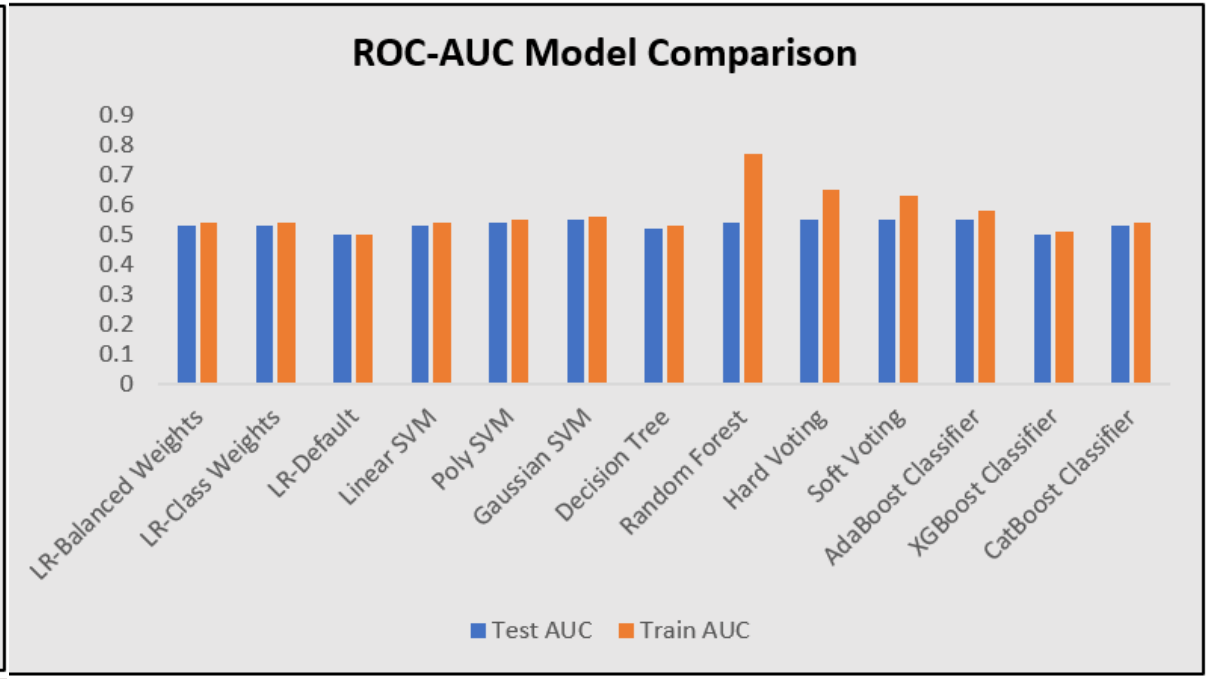
Cat Boost scores				
Accuracy	AUC	F1 Score	Precision	Recall
0.525	0.5376	0.472	0.397	0.582

Confusion-Matrix Predicted	
7055 (TN)	7246 (FP)
3427 (FN)	4772 (TP)

iterations=3, learning_rate=0.1, depth=4, class_weights=w ,random_state=42

Conclusion





Model	Accuracy	AUC	F1 Score	Precision	Recall
Logistic Regression-Balanced Weights	0.54	0.53	0.44	0.39	0.5
Logistic Regression-Class Weights	0.53	0.53	0.46	0.4	0.54
Logistic Regression-Default	0.63	0.5	0.003	0.29	0.001
Linear SVM	0.54	0.53	0.44	0.4	0.5
Polynomial SVM (degree=2)	0.54	0.54	0.47	0.4	0.56
Gaussian SVM	0.54	0.55	0.47	0.41	0.55
Decision Tree	0.48	0.52	0.49	0.38	0.69
Random Forest	0.54	0.54	0.45	0.4	0.52
Hard Voting	0.55	0.55	0.47	0.41	0.54
Soft Voting	0.56	0.55	0.46	0.42	0.52
AdaBoost Classifier	0.53	0.55	0.49	0.41	0.61
XGBoost Classifier	0.63	0.5	0.02	0.41	0.01
CatBoost Classifier	0.52	0.53	0.46	0.39	0.57

Thank you

Questions & Feedback