Popularity Classification on Spotify

Professor:

Zixuan (Maggie) Meng

The University of Texas at Dallas

December 12, 2022

I. Introduction

Spotify is a digital music streaming service that offers users access to millions of songs and videos from artists all over the world. Nowadays, the application is becoming more and more popular among everyone since you can access the most updated music trends by only a few simple registration steps. The dataset Spotify Dataset 1921-2020 containing 169k thousand tracks, collected by using Spotify Public API will dive deep into the analysis of how popularity is calculated based on different features using various supervised techniques.

II. Data

1.1. Data Description

Dataset: Spotify Dataset 1921-2020

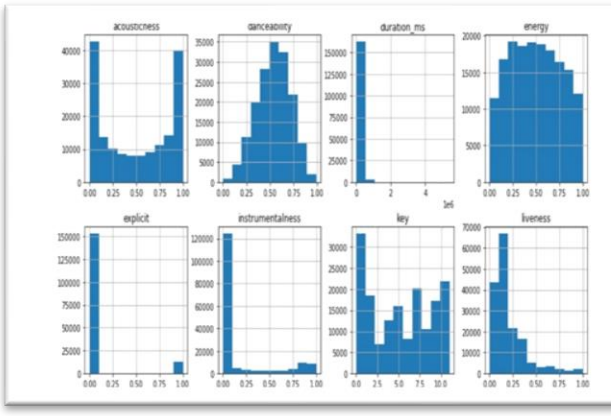
	mean	median	sd	variance	min	max	count	miss.val
acousticness	4.929812e-01	4.93000e-01	3.769843e-01	1.421172e-01	0	0.996	165711	0
danceability	5.366523e-01	5.46000e-01	1.756846e-01	3.086508e-02	0	0.988	165711	0
duration_ms	2.330355e+05	2.09760e+05	1.216101e+05	1.478901e+10	5108	5403500.000	165711	0
energy	4.936804e-01	4.88000e-01	2.670035e-01	7.129084e-02	0	1.000	165711	0
explicit	7.443079e-02	0.00000e+00	2.624714e-01	6.889126e-02	0	1.000	165711	0
instrumentalness	1.643654e-01	2.32000e-04	3.110348e-01	9.674263e-02	0	1.000	165711	0
key	5.201405e+00	5.00000e+00	3.508201e+00	1.230747e+01	0	11.000	165711	0
liveness	2.056760e-01	1.34000e-01	1.769469e-01	3.131022e-02	0	1.000	165711	0
loudness	-1.126281e+01	-1.03760e+01	5.629660e+00	3.169308e+01	-60	3.855	165711	0
mode	7.097960e-01	1.00000e+00	4.538578e-01	2.059869e-01	0	1.000	165711	0
popularity	3.205869e+01	3.40000e+01	2.140476e+01	4.581639e+02	0	100.000	165711	0
speechiness	8.447519e-02	4.48000e-02	1.207579e-01	1.458246e-02	0	0.968	165711	0
tempo	1.170550e+02	1.14908e+02	3.071705e+01	9.435372e+02	0	244.091	165711	0
valence	5.316040e-01	5.43000e-01	2.637594e-01	6.956902e-02	0	1.000	165711	0
year	1.977841e+03	1.97900e+03	2.532606e+01	6.414092e+02	1921	2020.000	165711	0

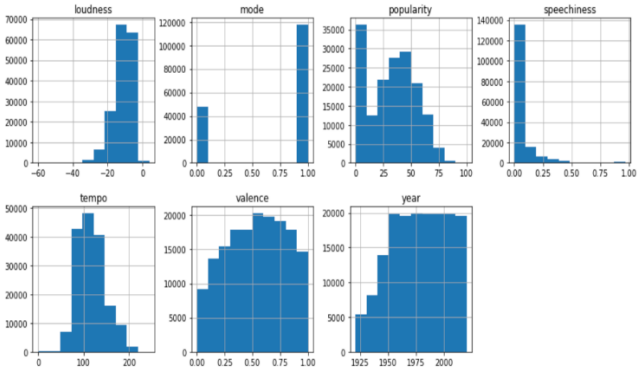
- **Primary**: IDs
- **Numerical**:
  - **Range from 0 to 100**
    - Popularity (our main interest)

Range from 0 to 1	**Other ranges**	**Categorical:**
Acousticness	duration_ms (from 200k to 300k)	key
Danceability	tempo (from 50 to 150)	mode
Energy	loudness (from -60 to 0)	explicit
Instrumentalness	year (from 1921 to 2020)	artists

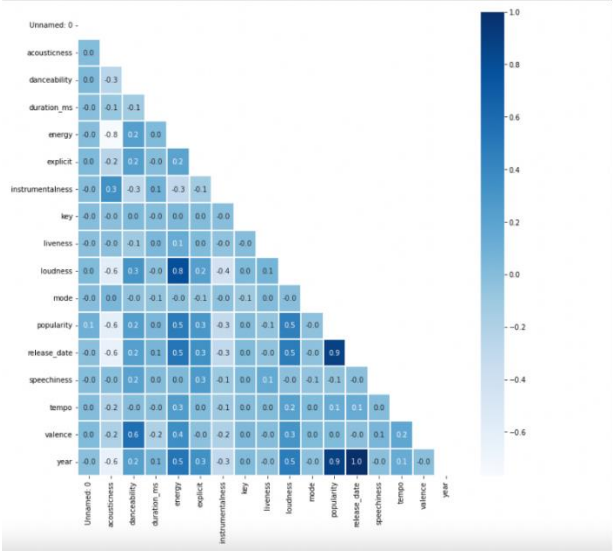
Valence	-	release_date
Liveness	-	name
Speechiness	-	-

1.2. Descriptive Statistics





1.3. Feature Selection



Based on the heatmap, we have combined correlated features and selected the important one.

{loudness, energy} -> **{loudness}**

{danceability, valence} -> **{danceability}**

{acousticness, instrumentalness} -> **{acousticness}**

{explicit, speechiness} -> **{explicit}**

Features
Year
Acousticness, Instrumentalness
Danceability, Valence
Explicit, Speechiness
Mode
Loudness, Energy
Tempo
Key
Liveness
Duration_ms

1.4. Data Preprocessing

- Remove duplicates
- Remove entries with N.A or invalid data in any column
- Remove redundant “Release Date” column
- Split data into train and validation dataset

1.5. Data Modeling

Popularity is our target dependent variable. It is a popular level of songs with wide appeal that is distributed to large audiences in the music industry. Our goal is to predict popularity based on a songs’ attributes and create a model which can predict popularity to the highest accuracy

2. Models

2.1. Linear Regression

- **Polynomial Regression**
- Best hyperparameter chosen: 5
- Overall accuracy: 79%
- **Ridge**
- Best hyperparameter chosen: 10
- Overall accuracy: 77%
- **Lasso**
- Best hyperparameter chosen: 0.0001
- Overall accuracy: 77%

2.2. Logistic Regression

- Logistic Regression is used when the dependent variable (target) is categorical. In our scenario, the target variable is popularity – 0 or 1 (categorical).
- Metrics obtained for Logistic Regression are -
    - Precision score: 70.7%
    - Recall score: 66.1%
    - Accuracy score: 85.9%
    - F1 score: 68%

- AUC: 0.90

- Best Hyperparameters chosen are -
  - C = 0.01
  - Penalty = L2 (Ridge - L2 penalty function uses the sum of the squares of the parameters and Ridge Regression encourages this sum to be small)

2.3. Decision Tree

- Metrics obtained for decision tree (without pruning) are -
  - Test accuracy: 80.6%
  - Train accuracy: 99.7%
  - Inference: The model is overfitting, and the tree is very large. It is important to use pruning to prevent overfitting issue.
- Metrics obtained for decision tree (with pruning) are -
  - Test accuracy: 86.5%
  - Train accuracy: 86.1%
  - Precision score: 74.5%
  - Recall score: 62.2%
  - F1 score: 67.8%
  - Inference: After pruning, the model is performing better. The tree is easy to understand and small enough to visualize.
- **Ensemble Methods**
  - **Random Forest Classifier**
    - Random forest consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction.
    - It is computationally heavy to implement.
    - Random forest obtains the average of the several decision trees
    - Metrics obtained for Random Forest are -
      - Accuracy score: 86.8%
      - Precision score: 78.8%
      - Recall score: 57.9%
      - F1 score: 66.7%

- **Gradient Boost Classifier**
  - It is computationally faster than Random Forests as the number of trees is less (4 to 5).
  - Metrics obtained for Gradient Boost Classifier are -
    - Accuracy score: 86.7%
    - Precision score: 77.8%
    - Recall score: 59.1%
    - F1 score: 67.1%

2.4. SVM

- To apply the Support vector machine algorithm, firstly, we scaled the columns data using the Standard Scaler.
- One of the drawbacks of the SVM is that it is computationally very costly. Also, we have nearly 170000 observations, so it is impossible to use the actual data to train and test the models, especially the advanced kernel SVM Kernel models.
- So, we randomly created the six pairs of train–test data using different number of samples.

2.5. Linear SVM (Binary class)

- We applied the Linear SVM as our first SVM model with the default parameters
- We kept the random state 22 and dual = False
- The reason for Dual = False is to increase the model speed, as we do not have a number of columns more than the number of observations.
  - Accuracy score: 85.7%
  - Precision score: 70.1%
  - Recall score: 65.8%
  - F1 score: 67.

2.6. Kernel SVM (Binary class)

- Liner Kernel SVM
  - We applied the Liner kernel SVM with this parameter: {'C': [0.01, 0.1, 1]}
  - The best parament was at C = 0.01
  - Matrices values are the same as the model with default hyperparameter
- RBF SVM

- We applied the Kernel SVM with the default hyperparameter, which has RBF as default, and we received the following accuracy, precision, recall and F1 Score.
- Accuracy score: 86.7%
- Precision score: 78.8%
- Recall score: 58.1%
- F1 score: 66.9%

Confusion matrix for default Kernel SVM		
	P. Not Popular	P. Popular
A. Not Popular	12159 (73.3%) ↑	596 (3.5%) ↓
A. Popular	1598 (9.6%) ↑	2219 (13.3%) ↓

- After we added the GridSearchCV method to hyperparameter tuning with the parameter's 'C': [1100, 1200, 1400], 'gamma': [0.005, 0.01, 0.05]
- We found that the best parameters are at C = 1100 and degree = 0.005
- We received the accuracy, precision, recall and F1 score as below
- Precision score: 78.82%
- Recall score: 58.13%
- Accuracy score: 86.76%
- F1 score: 66.91%

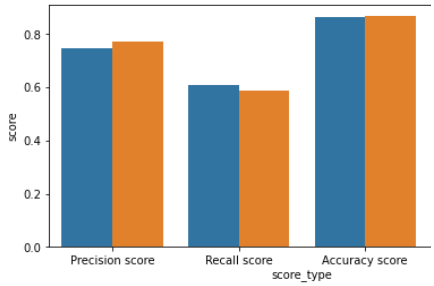
Confusion matrix for RBF SVM		
	P. Not Popular	P. Popular
A. Not Popular	6030 (72.7%) ↓	384 (4.6%) ↑
P. Popular	730 (8.8%) ↓	1142 (13.7%) ↑

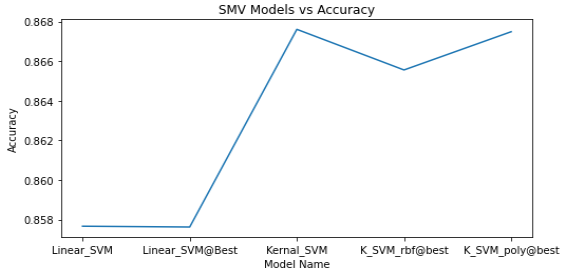
- **Polynomial SVM**
  - After we added the GridSearchCV method to hyperparameter tuning with the parameter: 'C': [10, 15, 20], 'degree': [2,3,4]
  - We found that the best parameters are at C = 15 and degree = 3
  - The best accuracy, precision, recall and F1 score are
  - Precision score: 77.21%
  - Recall score: 58.65%
  - Accuracy score: 86.74%
  - F1 score: 66.66%

Confusion matrix for Polynomial SVM		
	P. Not Popular	P. Popular
A. Not Popular	6090 (73.4%) ↑	324 (3.9%) ↑
A. Popular	774 (9.3%) ↓	1098 (13.2%) ↓

- Conclusion for binary SVM classification

- The precision for SVM with RBF is relatively low compared to polynomial SVM, but on the other hand, the recall for RBF SVM is high.
- As a result, the accuracy and F1 are almost same
- The best model is polynomial if we consider the accuracy.





2.7. Kernel SVM (Multi-class)

- After that, we decided to classify the song into three categories, so we applied the Liner kernel SVM with the following hyperparameter 'C': [0.1, 0.5, 1].
- We used the decision function shape = 'ovo' (one to one)
- We classify the song at the following cutoff
  - Class 2 >= 50, Class 1 >= 25, Class >= 0
- We received the best hyperparameter for Linear SVM at with C = 0.5

Confusion matrix			
	P. 0	P. 1	P. 2
A. 0	2544 (30%)	390 (4.7%)	11 (0.1%)
A. 1	309 (3.7%)	2790 (33%)	370 (4.4%)
A. 2	33 (0.3%)	686 (8.2%)	1153 (14%)

- Accuracy = 73.7%
- After that, we used the kernel SVM with the {'C': [50, 100, 1000] , 'gamma': [0.001, 0.005,0.01]}
- And received the best accuracy of 78.28% at 'C' = 50, 'gamma' = 0.01

3.KNN

The k-nearest neighbors' algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about

the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

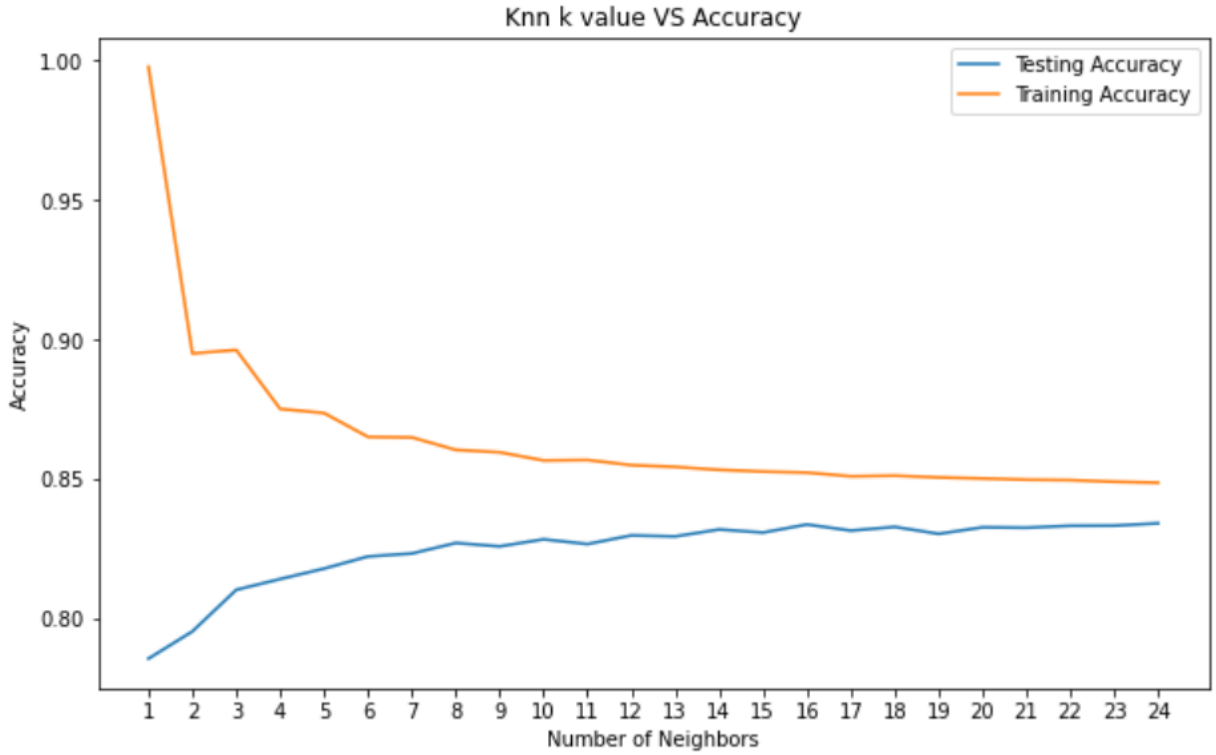
i. Pre-possessed the data using Standard Scalar and transformed some of the features
ii. Used popularity as an index to distinguish the data into two parts.
iii. Trained the data with random state = 42, and test_size = 0.25
iv. Used KNeighborsClassifier

When k=1 we got the following accuracies:

With KNN (K=1) train accuracy is: 99.76%

With KNN (K=1) test accuracy is: 78.56%

v. KNN k value vs Accuracy




```
Best accuracy is 0.834073573428599 with K = 24
```

As we can see from the above graph, we have almost 99.76% accuracy at k=1 (as seen in the above calculations as well) and as we increase the number of k, the testing accuracy is also increasing and towards the end running parallel to the train accuracy. This also decreases our problem of overfitting the model as concluded in the graph.

vi. KNeighbor Classifier with GridSearchCV

Best k is: {'n_neighbors': 25}
Mean validation score is: 83.65%
Test accuracy: 83.36%

vii. KNN Confusion Matrix

Accuracy score: 83.36%

Precision score: 81.53%

Recall score: 76.36%

F1 score: 78.86%

Confusion matrix		
	A. Pos	**A. Neg**
P. Pos	21678	2912
P. Neg	3979	12859

As seen in the above confusion matrix, we get a good balance between the precision and recall at the same time, the accuracy is also good enough for our model for its prediction.

4. Conclusion

4.1. Performance Evaluation and Model Selection

Model	Sub-Category	Accuracy	Precision	Recall	F1
Linear Regression	Polynomial Regression, with hyperparameter = 5	79% (R Square)			
Logistic Regression	L2 Regularization used	85.9%	70.7%	66.1%	**68.3%**
Decision Tree	Decision Tree with pruning gives better performance	86.5%	74.5%	62.2%	**67.8%**
SVM	Kennel Polynomial degree 3	86.7%	77.2%	58.6%	**66.6%**
KNN	Knn with k=25	83.3%	81.5%	76.3%	**78.8%**

-> Best model chosen: KNN based on the F1 score, as for precision and recall both have equal importance. Also, we have chosen F1 over accuracy as accuracy considers the TN, while we have an unsymmetric class. Therefore, to assume the TN as part of the model performance matrix might be misleading.

4.2. Future Scope

- Our next plan is to explore Neural Networks models (ANN, CNN) to find non-linear dependency with complex relationships between the features. The model keeps learning

until it comes out with the best set of features to obtain a satisfying predictive performance.

- We also hope to make use of audio files and lyrics to further improve our model's performance in the future.
- With the advanced computational resources, we want to implement the computational costly model to come up with better model