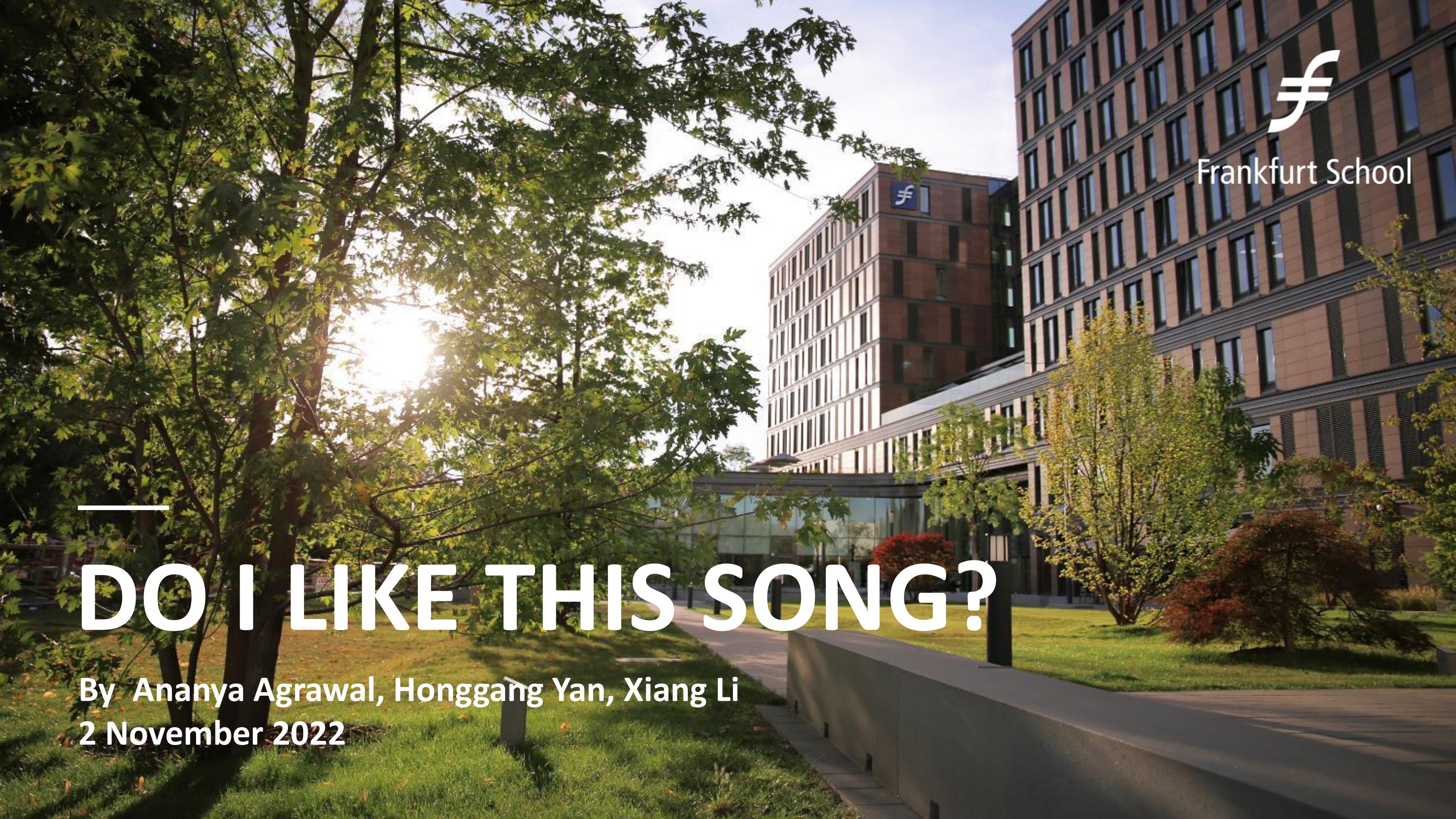


Frankfurt School

DO I LIKE THIS SONG?

By Ananya Agrawal, Honggang Yan, Xiang Li

2 November 2022

TABLE OF CONTENTS

INTRODUCTION

PROJECT OVERVIEW

Spotify Introduction

- One of the most popular music streaming platform in the world with 195 million paying subscribers
- Provides an API service which allows users to retrieve millions of songs' audio features in its library
- Examples of audio features (attribute): danceability, valence, energy, loudness, liveness etc.

Project Goal

- We used a dataset from Kaggle with audio attributes of 2017 songs from Spotify to predict if a person would like a song. Therefore, we need to implement machine learning techniques
- Among all the techniques, we would like to verify which model suits our goal and we can train the model to obtain the best accuracy score
- Also, we aimed to eliminate variance and bias errors in our model

Reference: https://www.kaggle.com/datasets/geomack/spotifyclassification

PROJECT WORKFLOW

Data Acquisition & Preparation

- Collected data from Kaggle
- Performed EDA and data pre-processing

Model Selection

- Labeled data & categorical target variable - > classification models
- Used 3 models: Random Forest, Decision Trees, KNN

Model Training & Validation

- Used grid search to find the optimal hyperparameters for our models
- Used K-fold cross validation to reduce the possibility of overfitting

Evaluation

- Used accuracy test to evaluate our model

DATA

DATA OVERVIEW

1. Basic Information

- Songs: 2017 songs, represented by each row with song titles and artists listed in 2 columns

2. Song Attributes

- Attributes: 13 columns, including energy, danceability, liveness, tempo etc.
- Example: energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.

3. Song Preferences (Target Value)

- The contributor of this dataset labeled likes or dislikes for each song with “1” and “0”

	Unnamed: 0	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	...	target
count	2017.000000	2017.000000	2017.000000	2.017000e+03	2017.000000	2017.000000	2017.000000	2017.000000	2017.000000	2017.000000	2017.000000	...	2017.000000
mean	1008.000000	0.187590	0.618422	2.463062e+05	0.681577	0.133286	5.342588	0.190844	-7.085624	0.612295	0.092664		0.505702

DATA OVERVIEW

Attribute	Description
Acousticness	A measure from 0.0 to 1.0 of whether the track is acoustic
Danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable
Duration	The duration of the track in milliseconds.
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy
Instrumentalness	Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content
Key	The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
liveness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live
Loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track. Values typical range between -60 and 0 db
Mode	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
Speechiness	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value
Tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration
Time signature	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)
target	The contributor of this dataset labeled likes or dislikes for each song with “1” and “0”

Reference: https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

EDA | TEXT DATA

Top 20 Artists

	artist	count
0	Drake	13
1	Disclosure	12
2	FIDLAR	9
3	Crystal Castles	9
4	Kanye West	8
5	CHVRCHES	7
6	Young Thug	7
7	Beach House	6
8	ASTR	6
9	M83	6
10	Hot Chip	6
11	Future	6
12	Grimes	6
13	Tame Impala	5
14	The Partysquad	5
15	Purity Ring	5
16	Teams vs. Star Slinger	5
17	Duke Dumont	5
18	New Order	5
19	ILoveMakonnen	5

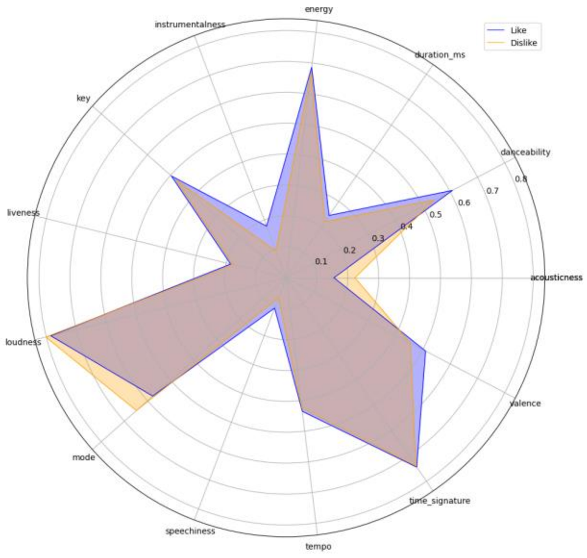
Word Cloud for Song Titles – Like / Dislike

Like Songs

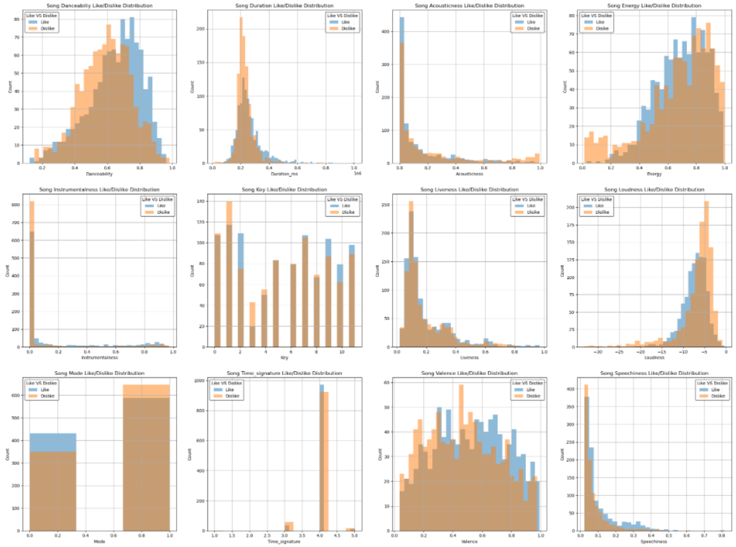
Dislike Songs

EDA | NUMERICAL DATA

Radar Chart – Like / Dislike

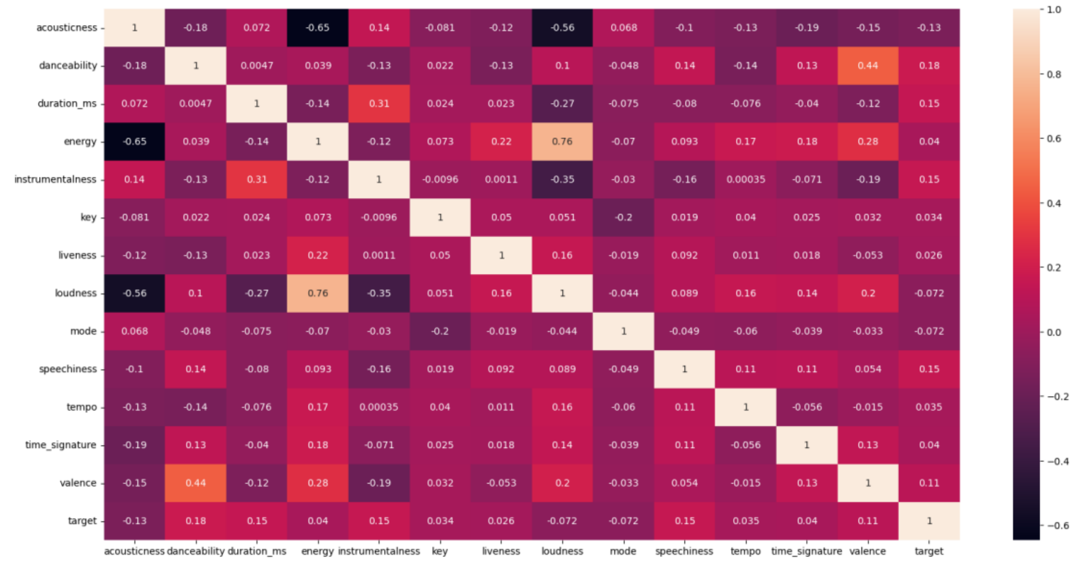


Histogram – Each Attribute



EDA | NUMERICAL DATA

Heatmap – Correlation among Attributes



- Energy and loudness have high positive correlation, and both of them are negatively correlated with acousticness
- Danceability, instrumentalness, speechiness and valence have relatively high positive correlation with like or dislike preference.
- Acousticness has reletively high negative correlation with like or dislike.

DATA PRE-PROCESSING

After collecting data from Kaggle, we dropped out text data and did minmax scaling for radar plot and KNN model

Drop Text Data

	acousticness	danceability	duration_ms	energy	instrumentalness	key
0	0.0102	0.833	204600	0.434	0.021900	2
1	0.1990	0.743	326933	0.359	0.006110	1
2	0.0344	0.838	185707	0.412	0.000234	2
3	0.6040	0.494	199413	0.338	0.510000	5
4	0.1800	0.678	392893	0.561	0.512000	5

liveness	loudness	mode	speechiness	tempo	time_signature	valence	target
0.1650	-8.795	1	0.4310	150.062	4.0	0.286	1
0.1370	-10.401	1	0.0794	160.083	4.0	0.588	1
0.1590	-7.148	1	0.2890	75.044	4.0	0.173	1
0.0922	-15.236	1	0.0261	86.468	4.0	0.230	1
0.4390	-11.648	0	0.0694	174.004	4.0	0.904	1

Minmax Scaling

Rescaling the range of features to scale the range in [0, 1]

	acousticness	danceability	duration_ms	energy	instrumentalness
0	0.010248	0.824826	0.190735	0.426363	0.022439
1	0.199998	0.720418	0.314481	0.350081	0.006260
2	0.034570	0.830626	0.171624	0.403987	0.000240
3	0.607034	0.431555	0.185488	0.328723	0.522541
4	0.180902	0.645012	0.381202	0.555533	0.524590
...	...	...	...	...	...
2012	0.001062	0.535963	0.261345	0.932872	0.002756
2013	0.088138	0.895592	0.168058	0.892189	0.001711
2014	0.008610	0.597448	0.193365	0.935924	0.004088
2015	0.001645	0.504640	0.171516	0.993897	0.693648
2016	0.002821	0.375870	0.190654	0.915582	0.000040

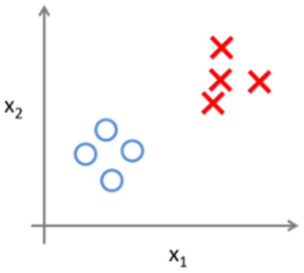
MODEL

MODEL SELECTION

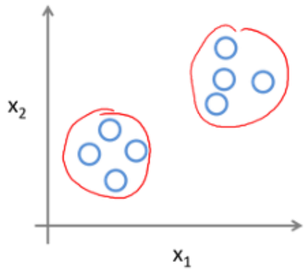
Supervised / Unsupervised

- Labeled data (like/dislike) - > supervised learning

Supervised Learning



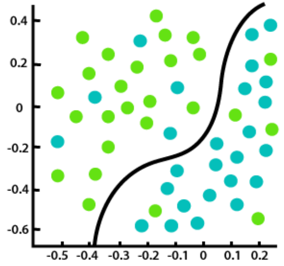
Unsupervised Learning



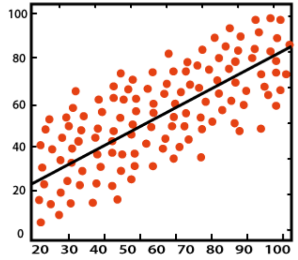
Classification / Regression

- Categorical target variable - > classification model

Classification



Regression



Model Selection

Decision Trees | **Random Forest** | **K-Nearest Neighbor (KNN)**

Reference: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

MODEL TRAINING & VALIDATION

To prevent overfitting and to find the optimal model for our project, we need to perform the following actions for our models

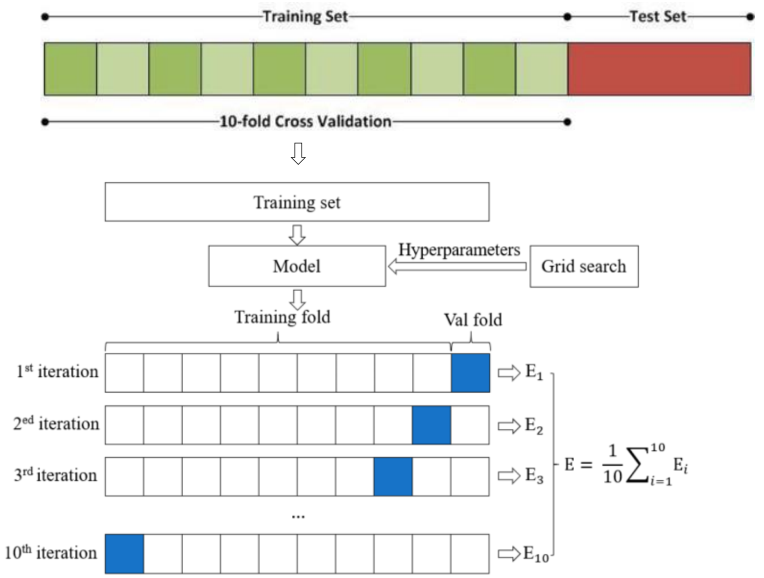
1. Split data into training set (75%) and testing set (25%)

2. Use Grid search to find the optimal hyperparameters

- Identify the models' hyperparameters
- Select the best hyperparameter combination

3. Use K-fold cross validation to reduce the possibility of overfitting

- Split the training set into training fold (k-1) and validation fold (the remainder)
- Train the model on training set and validate the validation set, repeat k times
- Take the average score of the validation results



Reference: https://www.researchgate.net/figure/K-fold-cross-validation-in-the-training-E-1-to-E-10-are-the-loss-functions-during-the_fig1_352231404

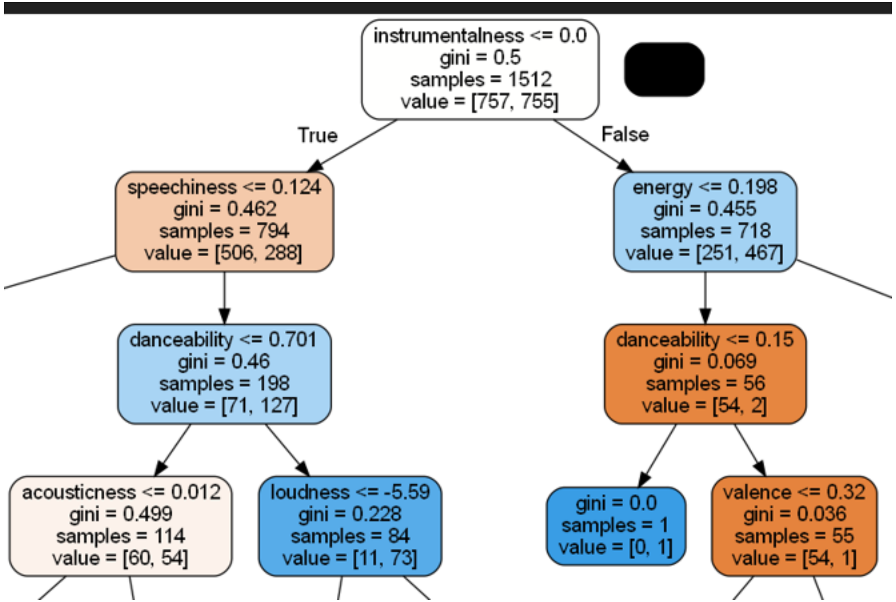
MODEL | DECISION TREE CLASSIFIER

Model Description

- A tree-structured classifier
- Its internal nodes represent the features of a dataset
- Its branches represent the decision rules
- Each leaf node represents the outcome

Important Parameters

- Criterion
- Max depth
- Min samples split
- Min sample leaf
- Max features



MODEL | DECISION TREE CLASSIFIER

Input

Step 1: Create grid for parameters

Code:

```
depth = np.arrange (1, 20 )
criterion = [ 'entropy’, 'gini’ ]
min_samples_split =  np.arange( 2, 20 )
param_grid = {'max_depth’: depth,
              'min_samples_split’:
               min_samples_split,
              'criterion': criterion}
```

Processing

Step 2: Implement Grid Search and Cross Validation

Code:

```
tree = GridSearchCV
(DecisionTreeClassifier(),param_grid, cv = 10,
verbose = 1, n_jobs = -1)
```

Step 3: Fitting 10 folds for each of 684 candidates, totaling 6840 fits

Code:

```
tree. fit (x_train, y_train)
```

Output

Step 4: Find the best estimator

Code:

```
tree. best_estimator_=
DecisionTreeClassifier(max_depth=4)
```

Step 5: Output the best accuracy score

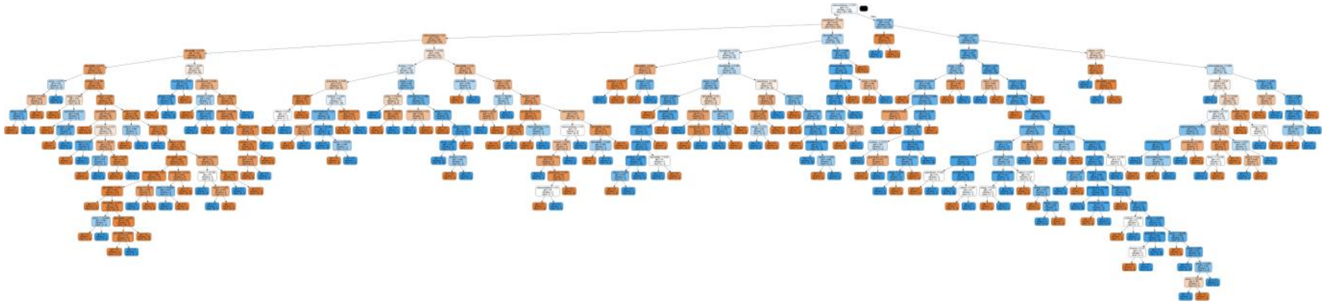
Code:

```
tree. best_score_ =  0.724
```

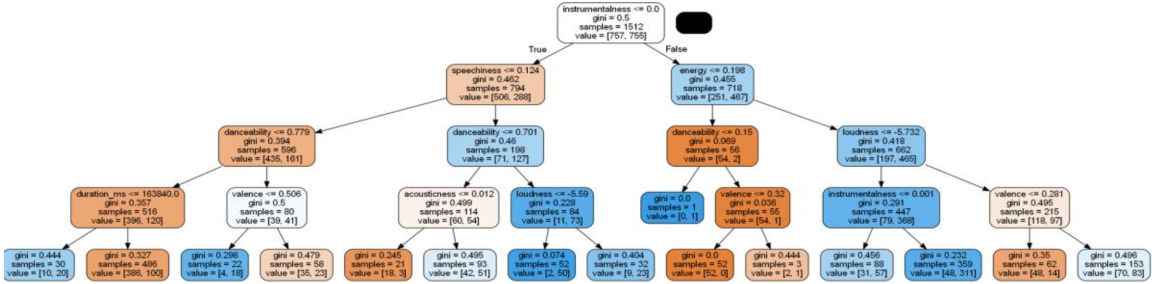
MODEL | DECISION TREE

Below is the image of decision tree before and after tuning the model

Before



After



MODEL | DECISION TREE

Advantages of Decision Tree

- It can be used for both Regression and Classification problems.
- Decision Trees are very easy to grasp as the rules of splitting is clearly mentioned.
- Complex decision tree models are very simple when visualized. It can be understood just by visualizing.
- Scaling and normalization are not needed.

Disadvantages of Decision Tree

- A small change in data can cause instability in the model because of the greedy approach.
- Probability of overfitting is very high for Decision Trees.

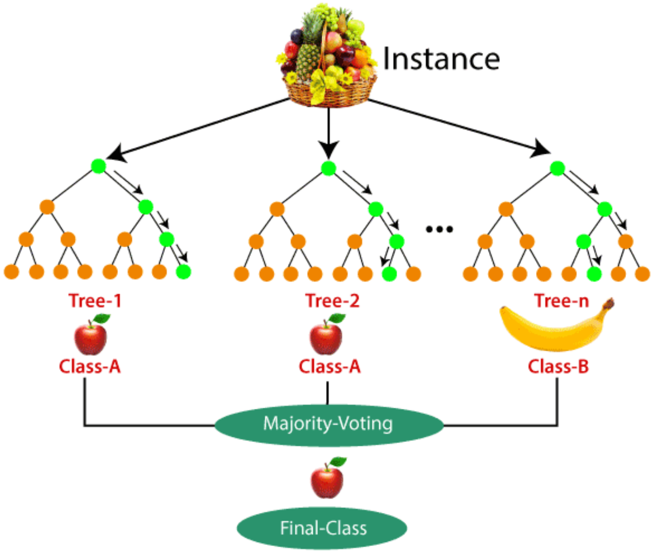
MODEL | RANDOM FOREST

Model Description

- A classifier that contains a number of decision trees on various subsets of the given dataset
- Takes the average to improve the predictive accuracy of that dataset

Ensemble Techniques

- Ensemble learning is a model that makes predictions based on a number of different models
- By combining individual models, the ensemble model tends to be more flexible (less bias) and less data-sensitive (less variance)



Reference:

- https://www.javatpoint.com/machine-learning-random-forest-algorithm
- https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

MODEL | RANDOM FOREST

Input

Step 1: Create Grid for Parameters

Code:

```
max_depth = np.arange(1,10)
max_samples = np.arange(100,1000,100)
min_samples_split = np.arange(2,10)
random_grid = {'max_depth': max_depth
,"max_samples":max_samples
,"min_samples_split":min_samples_split}
```

Processing

Step 2: Implement Grid Search and Cross Validation

Code:

```
forest =
GridSearchCV(RandomForestClassifier(),random_
grid, verbose=1, n_jobs = -1)
```

Step 3: Fitting 5 folds for each of 648 candidates, totaling 3240 fits

Code:

```
tree. fit (x_train, y_train)
```

Output

Step 4: Find the best estimator

Code:

```
forest.best_estimator_= RandomForestClassifier
(max_depth=9, max_samples=700,
min_samples_split=4)
```

Step 5: Output the best accuracy score

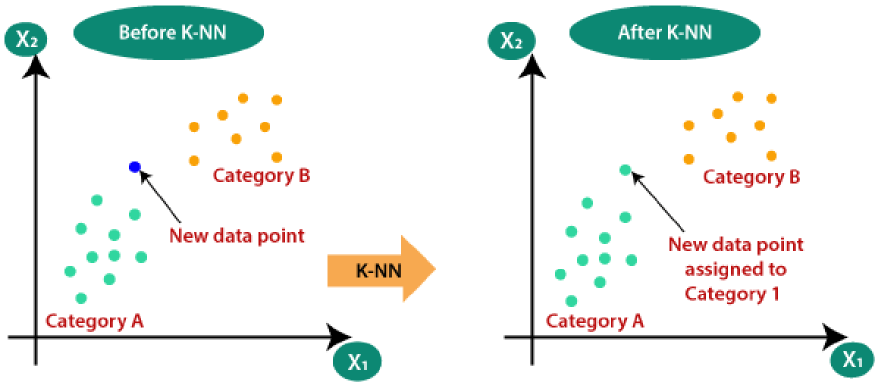
Code:

```
forest.best_score_ = 0.784
```

MODEL | KNN

Model Description

- Uses 'feature similarity' to predict the values of new datapoints
- the new data point will be assigned a value based on how closely it matches the points in the training set



Reference: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

MODEL | KNN

Input

Step 1: Create Grid for Parameters

Code:
neighbors = np.arange(1, 200)

Processing

Step 2: Implement Grid Search and Cross Validation

Code:
neigh = GridSearchCV (KNeighborsClassifier (), param_ grid, scoring = 'accuracy', cv =5)

Step 3: Fitting 5 folds for each of 200 candidates, totaling 1000 fits

Code:
neigh.fit (knn_x_train, knn_y_train)

Output

Step 4: Find the best estimator

Code:
neigh.best_estimator_ = KNeighborsClassifier (n_neighbors = 23)

Step 5: Output the best accuracy score

Code:
neigh.best_score_ = 0. 691

RESULT

MODEL TUNING

Decision Tree

Optimal Hyperparameter

- Criterion: Gini
- Max depth: 4
- Min samples split: 2

Random Forest

Optimal Hyperparameter:

- Max Samples: 700
- Max depth:9
- Min samples split: 4

KNN

Optimal Hyperparameter:

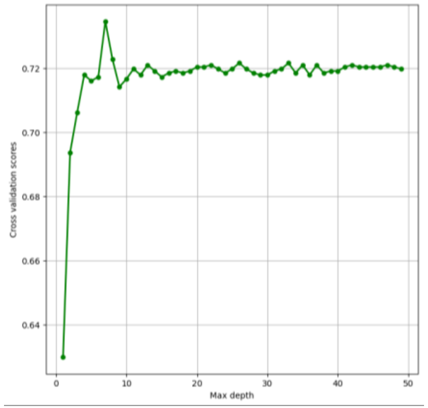
- Neighbors: 23

MODEL VALIDATION & EVALUATION

Decision Trees

- **Best Score for training: = 72.4%**
- **Score for testing: = 72.1%**

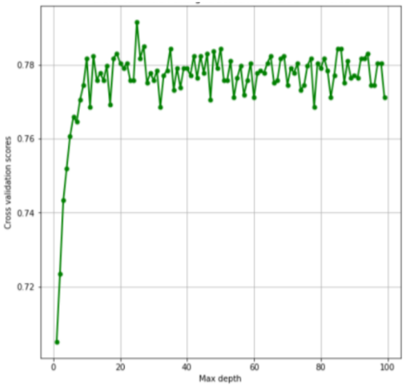
Cross Validation and Grid Search on Decision Trees
(visualization example: max depth)



Random Forest

- **Best Score for training: = 78.4%**
- **Score for testing: = 75.4%**

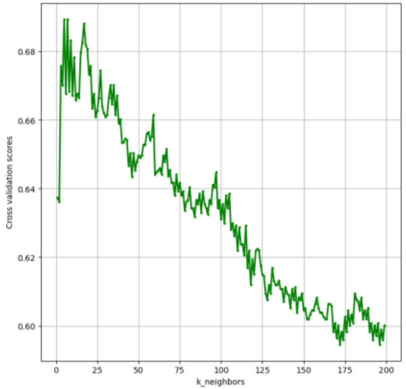
Cross Validation and Grid Search on Random Forest
(visualization example: max depth)



KNN

- **Best Score for training: = 69.1%**
- **Score for testing: = 66.1%**

Cross Validation and Grid Search on KNN
(visualization example: k neighbors)



CONCLUSION

CONCLUSION

Takeaways from our project

- Our project is a typical example of supervised learning projects with categorical dependent variable, which can be accomplished by classification models such as decision tree classifier, random forest classifier and KNN.
- Random forest classifier gave the best accuracy score among all three models which suggests ensemble techniques are more advantageous comparatively with single models.
- In our models, the difference between the scores for training and testing is low, hence we can conclude our models has low bias errors.

Limitations of our project

- KNN did not perform very well since there were 13 attributes(dimensions) in each song. This indicates that KNN might have relatively low accuracy dealing with multiple dimension problems.

Potential improvement

- Feature engineering: to improve the accuracy of our models
  - ➢ Remove the collinearity in the dataset
  - ➢ Remove the outliers

NEXT STEP

UNSUPERVISED LEARNING

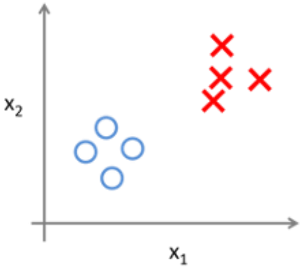
- K-MEANS CLUSTERING FOR RECOMMENDING SIMILAR SONGS

MODEL SELECTION

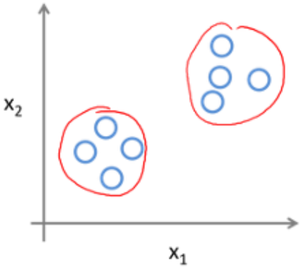
Supervised / Unsupervised

- Exclude labels - > Unsupervised Learning

Supervised Learning

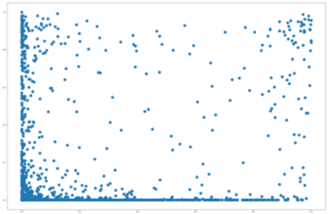


Unsupervised Learning

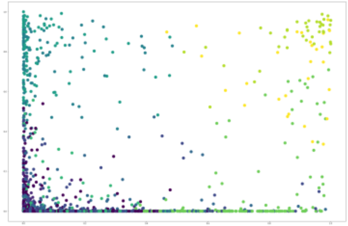


Clustering

Before Clustering



After Clustering



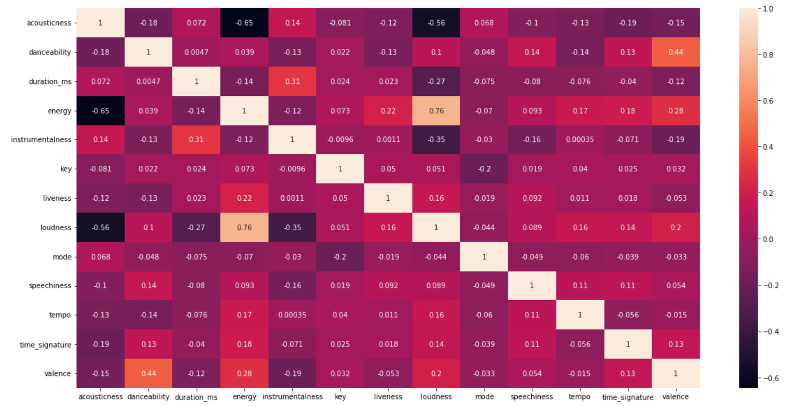
Model Selection

K-Means

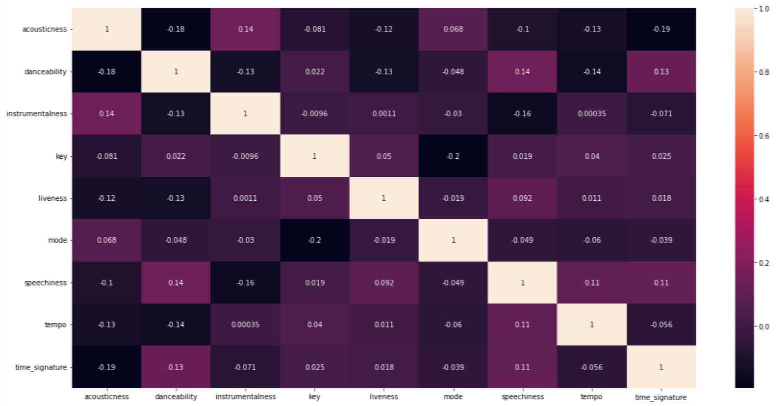
Reference: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

FEATURE SELECTION

Heatmap – Before Feature Engineering



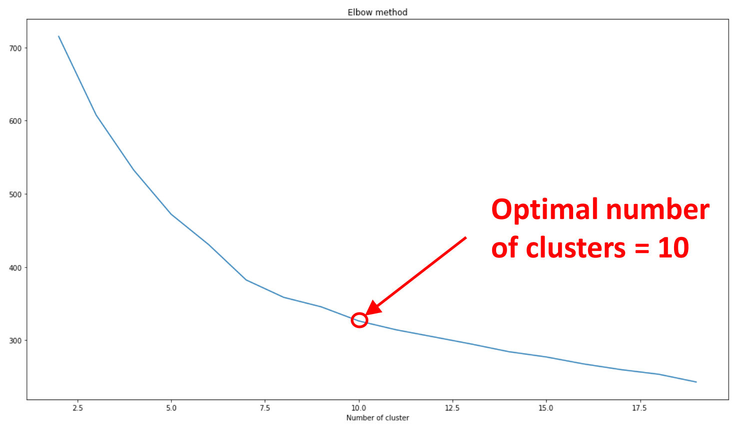
Heatmap – After Feature Engineering



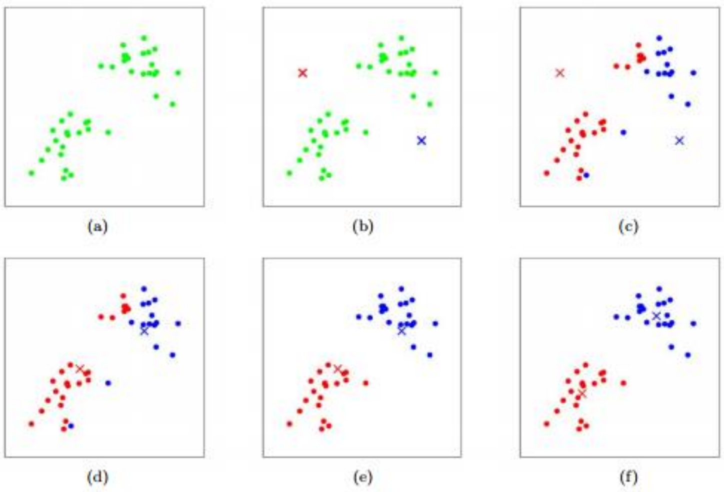
MODEL | K-MEANS

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.

Elbow Method



K-Means Algorithm



Reference:

- https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning
- https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

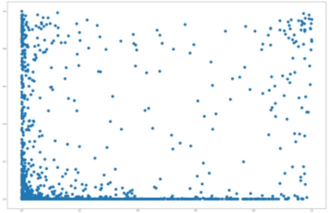
MODEL | RESULT

Input: 1 songs

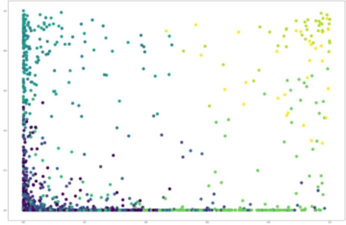
- **Song title: Candy**
- **Artist Name: Dillon Francis**

Clustering

Before Clustering



After Clustering



Output: 5 songs

- **Mask Off**
- **Redbone**
- **Xanny Family**
- **Childs Play**
- **Cemalim**

THANKS FOR YOUR ATTENTION