USE CASE STUDY REPORT

Group No.: Group 6

Student Names: Sahar Tariq and Haixia Bian

Executive Summary

This report studies different data mining techniques for predicting popularity of songs based on musical features. Spotify records musical features and popularity based on listeners' streams for all their songs. The data for 10,000 songs released in 2018 is acquired from Spotify's music library using Python and Spotify for Developers. Data processing steps (using R) include cleaning based on domain knowledge, zooming to limit scope of project to songs only, and standardizing and factorizing variables. Dimension reduction to pick relevant variables is done by multicollinearity, domain knowledge, and correlation analysis. The numerical popularity is categorized using median as cutoff into categorical popular or not popular. Data mining prediction techniques used for numerical popularity are linear regression, KNN, SVM, neural network and random forest. Data mining classification techniques used for categorical popularity are KNN and Discriminant Analysis. Analysis of error rates and lift charts show that best model is Simple Vector Machine. SVM is able to handle both categorical (with several categories) and numerical inputs.

We found that amongst today's listeners, songs' popularity is dependant on danceability, speechiness, happiness, acousticness, tempo, duration, sound quality, if recorded in a studio or of live, and if sung in minor or major key.

Table of Contents

I. Introduction

I.i) Background

Spotify is one of the biggest music streaming platforms, with 36% of the market share (as of 2018)[1]. Since it is a subscription-based service, Spotify's ability to gain and keep customers is dependent on appeasing to customers' music tastes. Data mining for analysis of technical features of music can find what type of music people like most, and predict if a newly released song will be popular or not. This information can help streaming companies decide whether to pursue acquiring exclusive streaming rights and whether to advertise the music on their frontpage. Spotify does not pay all artists the same rate per stream, rights holders receive an average per-play payout between $.006 and $.0084 [2]; data mining can facilitate the payment negotiation process by predicting the expected popularity of a song. This topic was chosen because it is interesting to see the effect the interaction of art, technology and business analytics.

I.ii) Introduction

This paper studies what type of music is most popular amongst Spotify listeners in 2019, and creates models to predict popularity from musical variables. The objective is to use data mining methods to classify popularity classification and to predict numerical popularity rating, based on the musical features of songs on Spotify.

The scope is limited to all songs released on the same year (2018). Tastes change with time, but this report does not include time-series evaluation, it only focuses on current music taste. Included in this report are the data exploration techniques, detailed description of the modelling processes, methods to compare models to choose the best.

II. Data Exploration and Visualization

Spotify has made available musical features information for all their songs. By using Python API and Spotify for Developers, the data for a random sample of 10,000 songs, all released in 2018, was downloaded. Python code for data collection is in Appendix A.

II.i) Data Description

The table below gives a brief explanation of each variable in the dataset.

Variable	**Explanation of Variable**
Song Name	*Self explanatory*
Artist Name	*Self explanatory*
Song ID	Unique Spotify identification code for songs
Popularity Rating	Rating of 0 – 100, based on average listens per unit time
Accousticness	Rating 0 – 1, if song is accoustic
Dancability	Rating 0 – 1, how danceable a song is
Energy	Rating 0 – 1, perceptual measure of intensity and activity
Speechiness	Rating 0 – 1, presence of spoken words instead of sung words

Happiness	Rating 0 – 1, musical positiveness conveyed in a track
Instrumentalness	Rating 0 – 1, presence/absence of vocal content
Liveness	Rating 0 – 1, if song was performed live or in a studio
Time stamp	Overall time signature of a track
Duration	Duration of track in milliseconds
Loudness	Average sound quality in decibels
Tempo	Estimated tempo of a song in beats per minute
Mode	If song is in major key (1) or minor key (0)
Key	Integers to represent 11 different musical pitch classes

II.ii) Data Quality

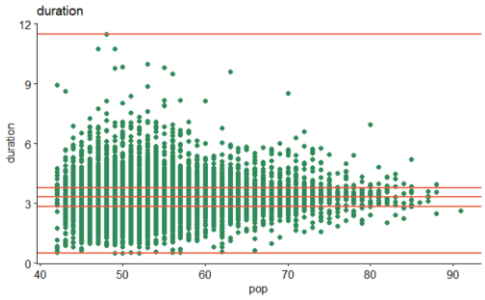
The data was imported to R for analysis. Appendix B shows the R code.
Using *which(is.na(sp))* R-code, it was found that the data had no missing values.
Using the unique Song ID, 380 rows were removed to avoid duplicate rows.

II.iii) Data Cleaning

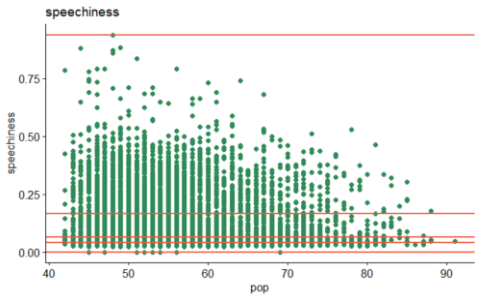
The goal of this project is to study songs. However, Spotify also contains podcasts, speeches, short clips, and other forms of audio files. The following data cleaning steps filtered the dataset to only contain songs. The following steps were taken by using domain knowledge of the variables, and basic charts:

- Limit Duration to 2 to 10 minutes to get rid of very short clips or long speeches.

The scatterplot with horizontal lines for the interquartile range helped decide the acceptable duration range for a song.

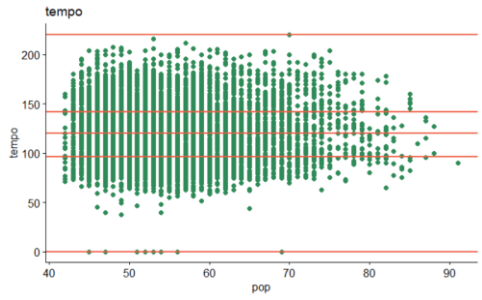


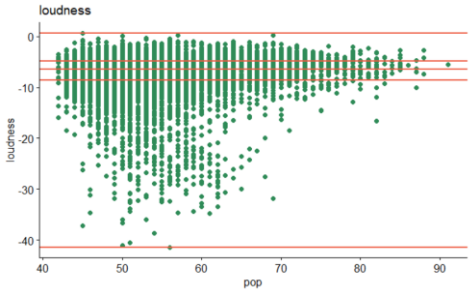
- Limit Speechiness to under 0.6 to remove speeches/podcasts. Data description specifies values over 0.6 have strong likelihood of not being songs. The scatterplot shows the data has a small percent of records over 0.6.



- Filter Tempo to delete records with 0 value; bpm cannot have 0 values for a song. The scatterplot show there are some song records with 0 tempo.

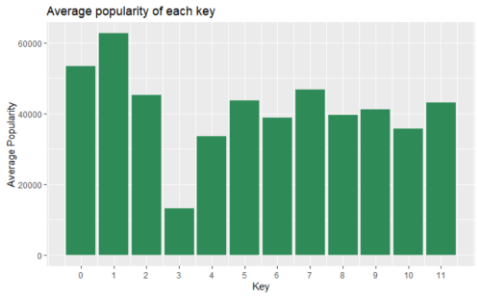
- Filter Loudness to delete records with more than 0 Loudness; sound quality is a range of 0 to negative 60, so the positive points seen on the scatterplot are incorrect.





- <u>Dance, Energy, Speech, Acoustic, Instrumental, Live, Happiness</u> ratings cannot be outside 0-1 range

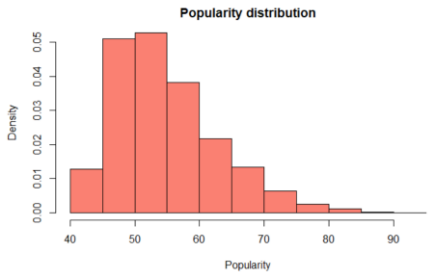
- <u>Filter Key</u> such that there are only integers 0 – 11, as each integer corresponds to a musical notation. The barchart assures that there are no values outside of 0-11 integer range.

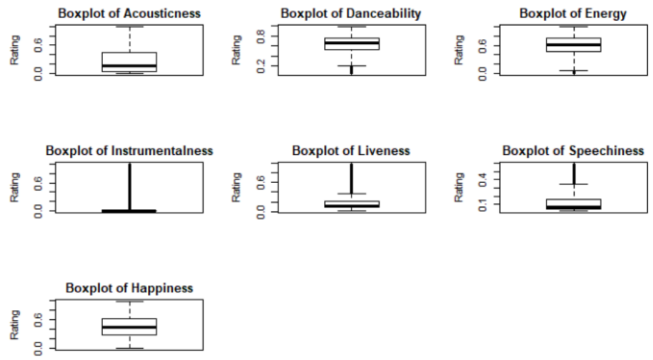
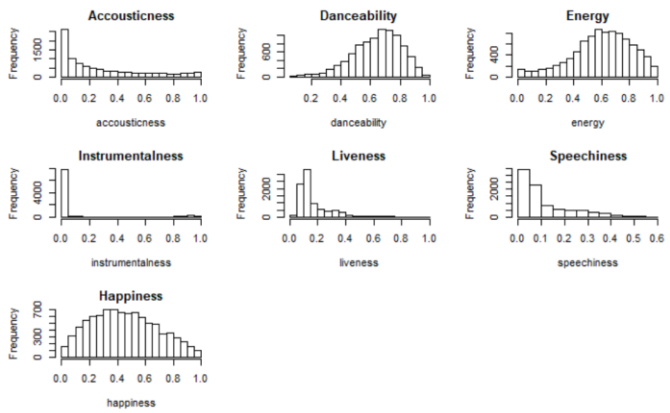
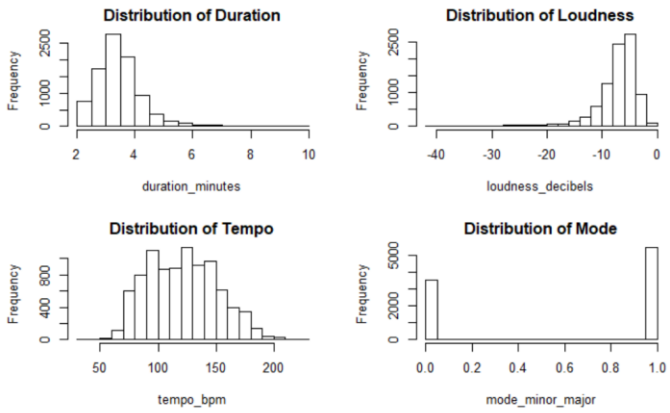


After filtering, the data has 8986 records.

II.iv) Data Distribution

Distribution plots help plan data preprocessing steps.

The plot for Popularity Rating distribution shows that it does not require any transformation since it is not overly skewed. Since it is the response variable, it will not be standardized.	

The boxplot of the ratings show that all their values fall between the same range 0 to 1, so they do not need to be normalized.	
The distributions of the the ratings show they are not all normally distributed, so they need to be standardized The distribution of Instrumentalness is very left skewed, meaning it requires logarithmic transformation. However, the data has many 0 values, so log(x) gives infinity values, and log(x+1) distributes the data same as just x. So we will not transform the data yet.	
The histogram of the numerical variables Duration, Loudness and Tempo show that they need to be standardized to make their scales comparable. Mode is a binary categorical variable and should not be standardized. Categorical variables need to be factorized.	

III. Data Preparation and Preprocessing

III.i) Dimension reduction

All the variables present in the dataset are not important for modeling.

- Domain Knowledge Method:
  - Remove the three unique identifiers as they are not modellable.
  - Remove Timestamp since song length is better represented by Duration.

- Remove Instrumentalness as most of its values are 0; no effect on popularity

- Multicollinearity: Variables in highly correlated pairs should be removed to avoid multicollinearity issues. This can be found by studying the Pearson correlation coefficients (shown below, values of interest in red boxes).
  - Drop Energy variable as it is highly correlated to 4 other variables; The data description from Spotify also stated that Energy was derived from those four variables. It is enough to just have those other variables instead of Energy as well.
  - Correlation: Drop Key as it has negligible effect on popularity and therefor not useful to this project.

	popularity	acousticness	danceability	duration_minutes	energy
popularity	1.000000000	-0.012172097	0.0932601889	-0.027047981	0.03456548
acousticness	-0.012172097	1.000000000	-0.2617404333	-0.050050082	-0.69292959
danceability	0.093260189	-0.261740433	1.0000000000	-0.214283736	0.14861259
duration_minutes	-0.027047981	-0.050050082	-0.2142837359	1.000000000	0.05367676
energy	0.034565478	-0.692929593	0.1486125877	0.053676763	1.00000000
instrumentalness	-0.080822283	0.404250005	-0.2712129686	-0.075330896	-0.42011984
key	-0.001903843	0.002983304	-0.0009003756	0.021900770	0.03940652
liveness	-0.020407363	-0.133317264	-0.0758335322	0.028621779	0.19985663
loudness_decibels	0.085266530	-0.615591327	0.3021021202	0.055331793	0.78528327
mode_minor_major	-0.022847266	0.062296957	-0.0733678934	0.008235333	-0.06498563
speechiness	-0.027945685	-0.139420938	0.2906565128	-0.128622060	0.08001876
tempo_bpm	0.007163889	-0.181463776	-0.0107690976	0.021240806	0.18474371
valence	0.041604828	-0.187383923	0.3738437084	-0.145345883	0.37193551

	instrumentalness	key	liveness	loudness_decibels	mode_minor_major
popularity	-0.080822283	-0.0019038431	-0.020407363	0.08526653	-0.0228472660
acousticness	0.404250005	0.0029833042	-0.133317264	-0.61559133	0.0622969574
danceability	-0.271212969	-0.0009003756	-0.075833532	0.30210212	-0.0733678934
duration_minutes	-0.075330896	0.0219007695	0.028621779	0.05533179	0.0082353329
energy	-0.420119839	0.0394065206	0.199856631	0.78528327	-0.0649856326
instrumentalness	1.000000000	-0.0149086939	-0.060585130	-0.62243102	-0.0064938561
key	-0.014908694	1.0000000000	0.006287476	0.02241834	-0.1796074503
liveness	-0.060585130	0.0062874758	1.000000000	0.12584930	-0.0013918395
loudness_decibels	-0.622431023	0.0224183352	0.125849298	1.00000000	-0.0372148246
mode_minor_major	-0.006493856	-0.1796074503	-0.001391839	-0.03721482	1.0000000000
speechiness	-0.149311914	0.0203622002	0.040653377	0.06556642	-0.0832239269
tempo_bpm	-0.126614989	0.0021540725	0.032889091	0.17688536	-0.0057812933
valence	-0.237219169	0.0405684671	0.032348170	0.35600189	0.0009240553

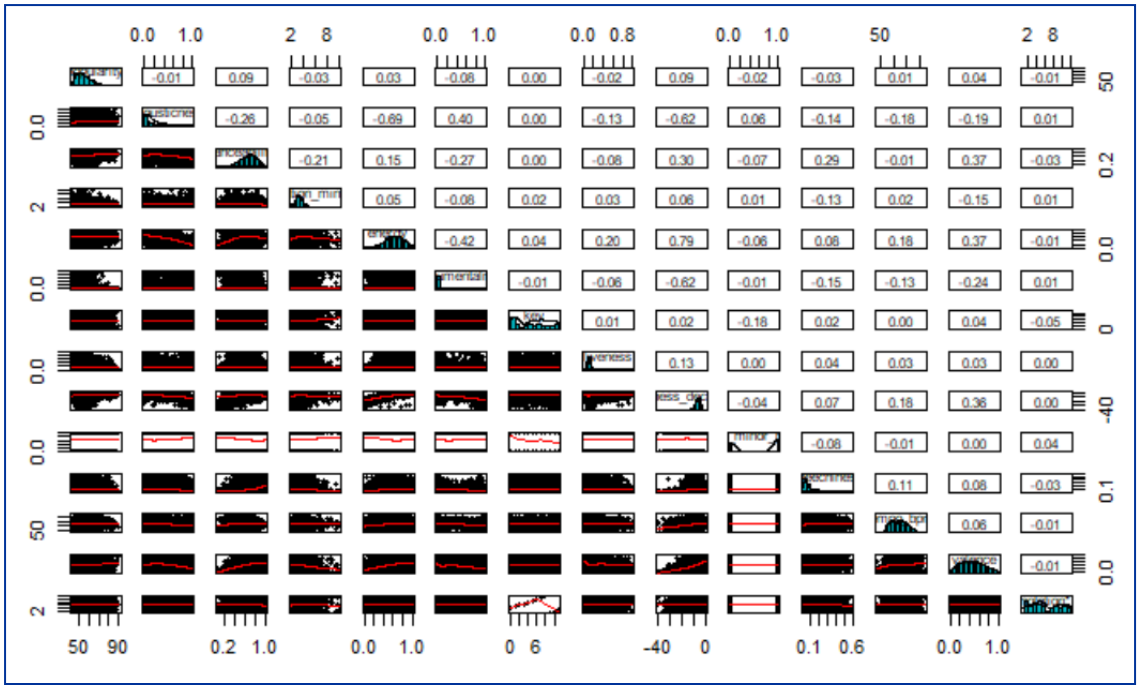
	speechiness	tempo_bpm	valence
popularity	-0.02794568	0.007163889	0.0416048280
acousticness	-0.13942094	-0.181463776	-0.1873839229
danceability	0.29065651	-0.010769098	0.3738437084
duration_minutes	-0.12862206	0.021240806	-0.1453458827
energy	0.08001876	0.184743710	0.3719355148
instrumentalness	-0.14931191	-0.126614989	-0.2372191695
key	0.02036220	0.002154073	0.0405684671
liveness	0.04065338	0.032889091	0.0323481703
loudness_decibels	0.06556642	0.176885356	0.3560018911
mode_minor_major	-0.08322393	-0.005781293	0.0009240553
speechiness	1.00000000	0.107193675	0.0750646061
tempo_bpm	0.10719368	1.000000000	0.0634810699
valence	0.07506461	0.063481070	1.0000000000

III. ii) Correlation Analysis

The Pearson correlations between variables need to be noted (shown above, values of interest in blue boxes), so their relationships can be included while modeling. Pearson coefficients above 0.3 show slight linear relationship, and above 0.7 show strong linear relationship. The following list are interrelated variables pairs:

- Loudness – Acousticness (inverse)
- Loudness-Valence
- Loudness-Danceability
- Danceability-Valence

The correlation plot for the variables is shown below:



III.iii) Preprocessing

As per the observations in Data Distribution section, the following list are the data preprocessing steps that were done on each variable:

- Numerical variables except for response variable were standardized
- Categorical variables Mode was factorized
- Other updates to data included changing Duration from milliseconds to seconds to make it easier to comprehend.
- For consistency and sensitivity, round all numerical values to 3 decimal places

III.iv) Variable Converting

For classification, Popularity Category variable was created based on the interquartile range of Popularity Rating. The split is at the median, with values below median classified as not popular (0), and equal to or above median as popular (1). Popularity Category is a binary variable and so is factorized.

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 42.00   49.00   54.00   55.27   60.00   91.00
```

```
   0    1
4392 4594
```

III.v) Variable selection

Finally, the variables used for modeling are:

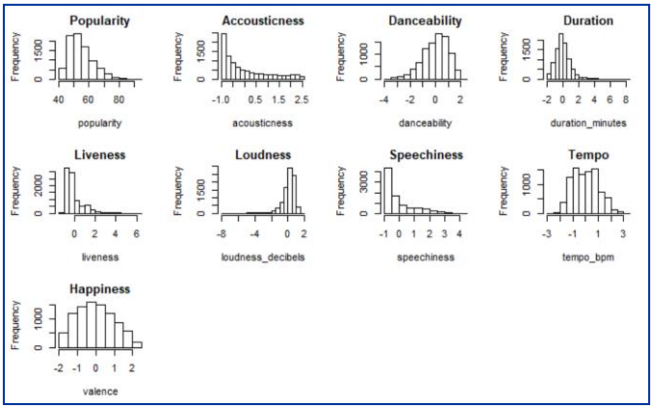
- Prediction response variable: Popularity Rating (Numerical values 0-100)

- Classification response variable: Popularity Category (0=Not popular, 1=Popular)
- Predictors: Danceability, Speechiness, Happiness, Acousticness, Tempo, Duration, Loudness, Mode, Liveness

The Data Summary and Distributions of the variables selected are as follows:

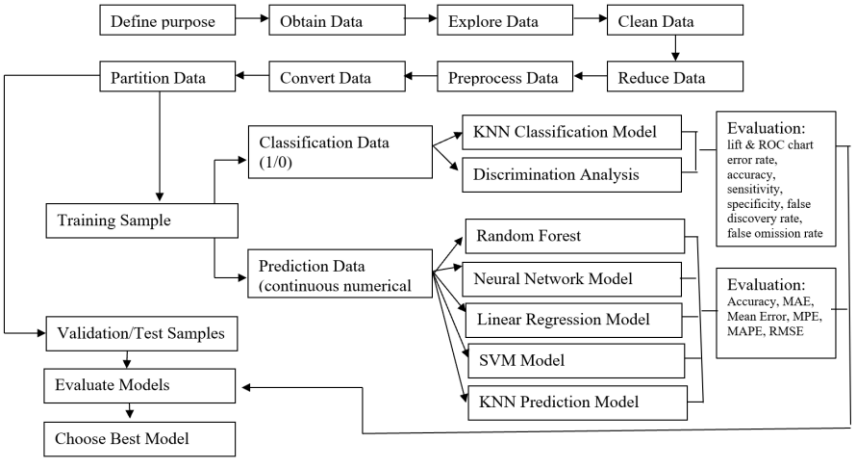
```
   popularity          key          mode_minor_major   notation          acousticness
 Min.   :42.00   2      :1133   0:3526           Length:8986        Min.   :-0.9490000
 1st Qu.:49.00   1      : 965   1:5460           Class :character   1st Qu.:-0.8210000
 Median :54.00   8      : 847                    Mode  :character   Median :-0.4070000
 Mean   :55.27   3      : 819                                       Mean   : 0.0000014
 3rd Qu.:60.00   6      : 788                                       3rd Qu.: 0.5942500
 Max.   :91.00   12     : 780                                       Max.   : 2.4460000
                 (Other):3654
  danceability       duration_minutes       liveness          loudness_decibels
 Min.   :-3.588000   Min.   :-1.811000   Min.   :-1.203000   Min.   :-7.725000
 1st Qu.:-0.634000   1st Qu.:-0.629000   1st Qu.:-0.574000   1st Qu.:-0.247000
 Median : 0.118000   Median :-0.121500   Median :-0.412000   Median : 0.229500
 Mean   : 0.000003   Mean   :-0.000008   Mean   : 0.000005   Mean   :-0.000003
 3rd Qu.: 0.735000   3rd Qu.: 0.454750   3rd Qu.: 0.219000   3rd Qu.: 0.591000
 Max.   : 2.092000   Max.   : 8.115000   Max.   : 6.058000   Max.   : 1.663000

  speechiness          tempo_bpm            valence          popularcategory
 Min.   :-0.858000   Min.   :-2.803000   Min.   :-1.9540000   0:4392
 1st Qu.:-0.703000   1st Qu.:-0.818500   1st Qu.:-0.7830000   1:4594
 Median :-0.479000   Median :-0.032000   Median :-0.0540000
 Mean   : 0.000012   Mean   : 0.000001   Mean   :-0.0000001
 3rd Qu.: 0.395000   3rd Qu.: 0.706000   3rd Qu.: 0.7360000
 Max.   : 4.259000   Max.   : 3.327000   Max.   : 2.3020000
```



IV. Data Mining Techniques and Implementation

Flowchart summarizes data processing and the types of data mining models



In the next sections, for each modeling method, brief modeling steps, each method's final model, and their performance and shortcomings are listed to help pick the best model.

IV.i) Multiple Linear Regression Prediction Model

Supervised learning for linear model of numerical response of Popularity Rating. This method works well on large datasets and can handle numerical variables.

Steps:

- Transform variables to be normally distributed – this took some iterations. Acousticness and Speechiness transformed to log so they are normally distributed.
- Iterate linear model to find model with best R^2 value: Add variable interrelations described in section III.B) Correlation Analysis. Add polynomial transformations. Use stepAIC method to remove unnecessary inputs.

Final model:

```
lm(formula = popularity ~ acousticness + poly(acousticness, 2) +
   danceability + loudness_decibels + poly(speechiness, 2) +
   tempo_bpm + valence + acousticness:loudness_decibels +
danceability:loudness_decibels,
   data = splr)
```

GVLMA validation

	Value	p-value	Decision
	<dbl>	<dbl>	<chr>
Global Stat	2117.3348345	0.0000000	Assumptions NOT satisfied!
Skewness	1081.4317820	0.0000000	Assumptions NOT satisfied!
Kurtosis	134.3120919	0.0000000	Assumptions NOT satisfied!
Link Function	0.9086969	0.3404604	Assumptions acceptable.
Heteroscedasticity	900.6822638	0.0000000	Assumptions NOT satisfied!

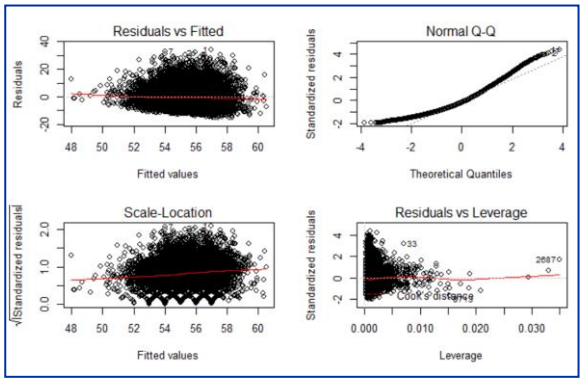
Model evaluation:

Model $R^2 = 0.034$

The gvlma validation and plots of typical approach regression diagnostics shows that the model does not meet normality, skewness, kurtosis and heteroscedasticity requirements for linearity. The R^2 value is also quite low.

Linear Regression is difficult since it requires many iterations for variable transformations. Simple models are preferred to complex ones with many transformations. The data does not fit linearity.

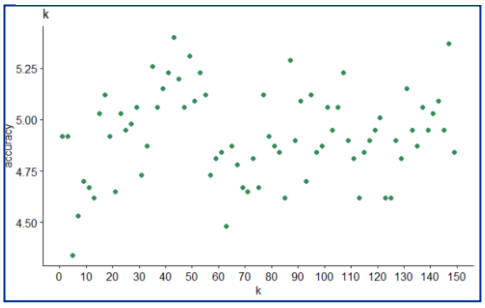
Typical Approach Regression Diagnostic Plots



IV.ii) KNN Prediction Model

Supervised learning model for predicting Popularity Rating.

Plot of k vs. accuracy of knn model



Steps:

- Choose best k: good starting point is k equals to the square root of number of rows in data, and then go up or down from there. We plot accuracy of k 1 to 150. K=61 has highest accuracy.

Final Model:

```
knn(train = sptraining, test = spvalid,cl = train_labels,
k=61)
```

Model evaluation:

- Model accuracy = 5.42%
- Prediction accuracy measures with validation set
    - MAE: 44.38
    - Mean Error: -44.38
    - MPE, MAPE: 4.79
    - RMSE 45.21

IV.iii) Neural Network Prediction

Supervised learning model for prediction of Popularity Rating

Final Model:

neuralnet(popularity~mode_minor_major+acousticness+danceability+duration_minutes+liveness+loudness_decibels+speechiness+tempo_bpm+valence,data = train.df, algorithm = 'rprop+')

Model Evaluation:

- Prediction accuracy measures with validation set
    - MAE: 6.15
    - Mean Error: -0.08
    - MPE, MAPE: 0.11
    - RMSE 7.78

IV.iv) SVM Prediction

Supervised learning model for prediction of Popularity Rating

Final Model:

mu.svm <- svm(predictor variables, response variable)

Model Evaluation:

- Prediction accuracy measures with validation set
    - MAE: 5.66
    - Mean Error: -1.31
    - MPE, MAPE: 0.104
    - RMSE: 7.58

IV.v) Random Forest Prediction

Supervised learning model for prediction of Popularity Rating

Final Model:

```
Call:
 randomForest(formula = popularity ~ ., data = MusicTrain)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 61.78227
                    % Var explained: 2.77
```

	IncNodePurity
mode_minor_major	3880.474
acousticness	32657.139
danceability	34132.929
duration_minutes	34004.503
liveness	29538.482
loudness_decibels	35735.406
speechiness	31722.746
tempo_bpm	32244.292
valence	30604.662

Model Evaluation:

- Prediction accuracy measures with training, validation and test sets.
- Training errors should be smaller than the validation errors

  - MAE:

Training set	Validation set	Test set
2.608	5.925	6.728

  - Mean Error:

Training set	Validation set	Test set
0.077	-1.433	2.661

  - MPE, MAPE:

Training set	Validation set	Test set
0.047	0.106	0.121

  - RMSE

Training set	Validation set	Test set
3.442	7.764	7.603

IV.vi) Discriminant Analysis Classification

Supervised learning model for classification of Popularity Category

Final Model:

lda(popularcategory~., training data's popularity rating)

Model Evaluation:

- Accuracy = 55%
- classification accuracy measures using validation and test sets. (Class of importance = 0)
  - sensitivity: 0.545
  - specificity: 0.557
  - false discovery rate: 0.545
  - false omission rate: 0.443

Lift Chart

Confusion Matrix

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  762  974
         1  636 1223

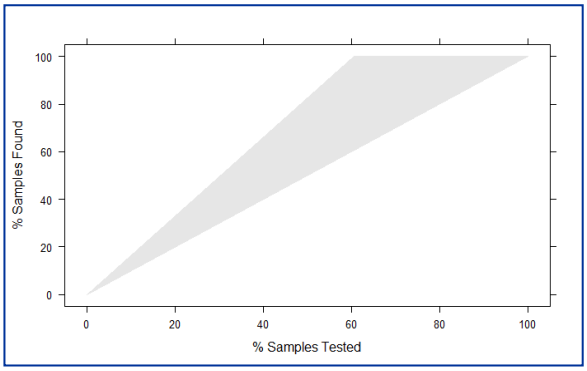
               Accuracy : 0.5522
                 95% CI : (0.5357, 0.5685)
    No Information Rate : 0.6111
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0974

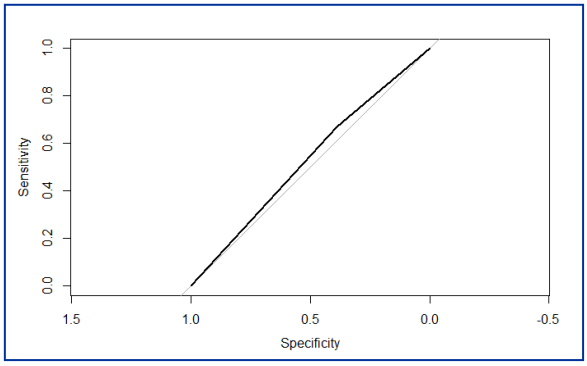
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5451
            Specificity : 0.5567
         Pos Pred Value : 0.4389
         Neg Pred Value : 0.6579
             Prevalence : 0.3889
         Detection Rate : 0.2120
   Detection Prevalence : 0.4829
      Balanced Accuracy : 0.5509

       'Positive' Class : 0
```



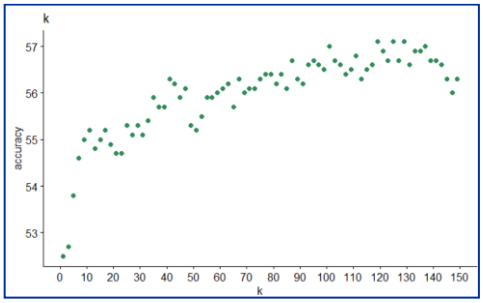
ROC chart



IV.vii) KNN Classification Model

Supervised learning model for classification of Popularity Category.

Plot of k vs. accuracy of knn model



<u>Steps:</u>

- Find best k-value: Plotted k vs model accuracy. The first peak is at k=119, then accuracy starts decreasing.

<u>Final Model:</u>

knn(train = trainingdata, test = validation data,cl = training data's classification variable, k=119)

<u>Model evaluation:</u>

- Accuracy = 57%
- Classification accuracy measures using validation and test sets. (Class of importance = 0)
  - sensitivity: 0.570
  - specificity: 0.571
  - false discovery rate: 0.430
  - false omission rate: 0.429

- Performance matrix shows this model is wrongly classifying more 0's as 1's, than vice versa. It is worse to mislabel rejects as accepts than vice versa.

Confusion Matrix

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  781  955
         1  590 1269

               Accuracy : 0.5702
                 95% CI : (0.5539, 0.5865)
    No Information Rate : 0.6186
    P-Value [Acc > NIR] : 1

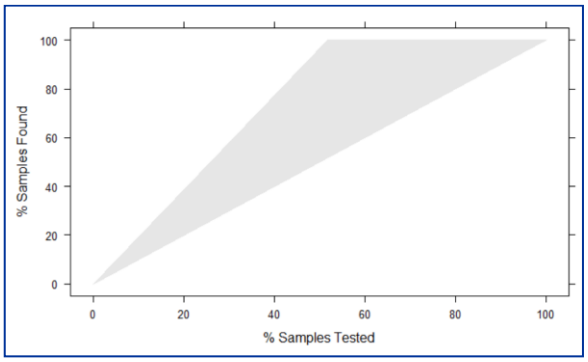
                  Kappa : 0.1334

 Mcnemar's Test P-Value : <2e-16

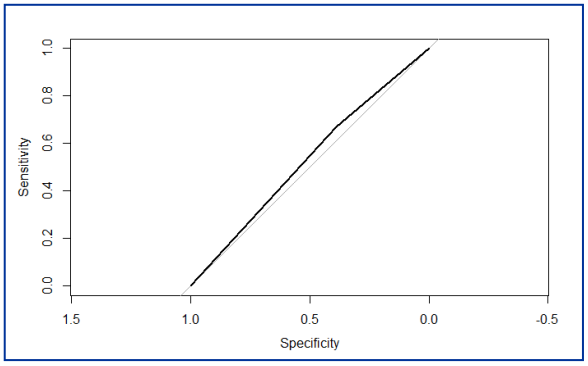
            Sensitivity : 0.5697
            Specificity : 0.5706
         Pos Pred Value : 0.4499
         Neg Pred Value : 0.6826
             Prevalence : 0.3814
         Detection Rate : 0.2172
   Detection Prevalence : 0.4829
      Balanced Accuracy : 0.5701

       'Positive' Class : 0
```

Lift Chart



ROC Chart



VI. Discussion and Recommendation

In the previous section, the models listed their evaluation metrics. Listed below is what each metric used means and how they are interpreted is:

- For Prediction
  - Mean Error: Average error. Less error is better.
  - MAE: Magnitude of the average absolute error. Lower is better.
  - MAPE: Gives a percentage score of how predictions deviate on average from the actual values. Lower is better.
  - RMSE: Similar to the standard error of estimate in linear regression. Lower is better.

- For Classification
  - Accuracy: Proportion of correctly classified records. Higher is better.

- Sensitivity: Percent of true Popular cases correctly classified. Higher is better.
- Specificity: Percent of true Not Popular cases correctly classified. Higher is better.
- False Discovery Rate: Percent of predicted Popular cases misclassified. Lower is better.
- False Omission Rate: Percent of predicted Not Popular cases misclassified. Lower is better.
- Lift Chart: Compares predictive performance to a baseline model without predictors. Better model is further away from baseline.
- ROC Chart: Area under ROC curve is the probability that a binary classifier gives a higher predictive score to a random positive example. Better model is closer to top left corner.

For linear regression model, R^2 measure is used to gauge goodness-of-fit. R^2 does not say much about the ability of the model to predict new cases. Since the linear regression model fails all assumptions of OLS regression, it will not be chosen as a good model.

In prediction models, KNN method has very low accuracy and very high errors, and so is immediately disqualified. Of the three remaining models, the highest error is with Neural Network. It is understood that the best hidden layers, threshold value and learning rate were not found for Neural Network, because that is a difficult and iterative process. The best models are simple and easier to replicate, therefor, due to the complexity of the process, Neural Network method is disqualified. Between Random Forest and SVM, the latter has the lowest errors, so SVM is chosen.

In classification models, KNN has 2% better accuracy than Discriminant Analysis. KNN also has higher sensitivity and specificity meaning more percent of true cases are being correctly classified, and has lower false discovery rate and false omission rate, meaning less percent of predicted cases are being misclassified. Its lift chart is higher (further from baseline). The area under curve in the ROC charts are quite close. Of the two classification models, KNN model has better performance, so KNN is chosen.

Finally, choosing between prediction and classification method for this problem. Is it better to approach this problem by using continuous popularity integers, or by categorical popularity factors? Prediction can show the infinitesimal changes in popularity by each metric and finding the equation relationship tying the variables together. That will be beneficial when deciding between songs that are quite similar. On the other hand, classification is simpler and can put songs into quick buckets. As a result, a recommendation will not be made between prediction and classification, instead we will realize that both are important.

Between the two final contenders of this research, SVM and KNN, we will choose SVM. This is because SVM works well with both predictive and categorical type problems. KNN is not the best for large datasets as for each case it repeatedly needs to find the

Euclidian distance, weighted average, the order, and then RMSE to find the best k. It takes a lot of computational time. Also, KNN can only handle if the response variable has two categories. Splitting the data into just two, popular or not popular with the median as cutoff, is too wide a range. It is better to have several categories, sliding from not popular, mediocre, normal, popular, to very popular. SVM method is able to handle when data has more than two categories for the response variable.

The model chosen is Support Vector Machine(SVM).
Recommendations for improvement is building both predictive and classification models using SVM, splitting the popularity category into categories using all the interquartile range instead of just median, and using both types together to evaluate each other.

VII. Summary

The chosen model of this research is Support Vector Machine,

This report discussed how Spotify music features data from Spotify API should be cleaned and how to use the variable's distributions for transformations. It also shows which musical features are important for finding popularity. Data was converted to predictive and categorical methods, and several types of models were built for each data type to find the best model. Discussion of the modelling process, and evaluation of the model's errors led to a shortlisting process. The final model was chosen due to its low errors, and ability to handle both categorical and predictive models. This model can help predict and classify if a song will be popular.

We found that amongst today's listeners, songs that are popular have high danceability, mid-level speechiness, high happiness, low acousticness, high tempo, duration of about three minutes, high sound quality, are recorded in a studio instead of live, and sung in minor key.

REFERENCES

https://variety.com/2018/biz/news/apple-music-global-market-share-streaming-spotify-1202942424/

https://wayback.archive-it.org/all/20141103193456/http://www.spotifyartists.com/spotify-explained/

APPENDIX A: DATA COLLECTION

Python code to collect data via Spotify for Developers and API

```
#ST

#CONNECT TO SPOTIFY

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

cid ="xx" #get these from spotify for developers website
secret = "xx"

client_credentials_manager = SpotifyClientCredentials(client_id=cid,
client_secret=secret)
sp =
spotipy.Spotify(client_credentials_manager=client_credentials_manager)
```

```
#GET SONG NAMES OF RANDOM SAMPLE

# create empty lists where the results are going to be stored
artist_name = []
track_name = []
popularity = []
track_id = []

#random sample. set sample size, sample rate max 100, and song year
for i in range(0,100,10):
    track_results = sp.search(q='year:2018', type='track',
limit=10,offset=i)
    for i, t in enumerate(track_results['tracks']['items']):
        artist_name.append(t['artists'][0]['name'])
        track_name.append(t['name'])
        track_id.append(t['id'])
        popularity.append(t['popularity'])


print('number of elements in the track_id list:', len(track_id))


#imports track names
import pandas as pd
df_tracks =
pd.DataFrame({'artist_name':artist_name,'track_name':track_name,'track_
id':track_id,'popularity':popularity})
print(df_tracks.shape)
df_tracks.head()
```

```
#SONG NAME DATA CLEANUP grouped =
df_tracks.groupby(['artist_name','track_name'], as_index=True).size()
grouped[grouped > 1].count()
df_tracks.drop_duplicates(subset=['artist_name','track_name'],
inplace=True) # doing the same grouping as before to verify the
solution grouped_after_dropping =
```

```
df_tracks.groupby(['artist_name','track_name'], as_index=True).size()
grouped_after_dropping[grouped_after_dropping > 1].count()
df_tracks[df_tracks.duplicated(subset=['artist_name','track_name'],keep
=False)].count() df_tracks.shape
```

```
#GET SONG METRICS OF THE RANDOM SAMPLE


# empty list, batchsize and the counter for None results
rows = []
batchsize = 50
None_counter = 0


for i in range(0,len(df_tracks['track_id']),batchsize):
    batch = df_tracks['track_id'][i:i+batchsize]
    feature_results = sp.audio_features(batch)
    for i, t in enumerate(feature_results):
        if t == None:
            None_counter = None_counter + 1
        else:
            rows.append(t)


print('Number of tracks where no audio features were
available:',None_counter)
print('number of elements in the track_id list:', len(rows))


df_audio_features = pd.DataFrame.from_dict(rows,orient='columns')
print("Shape of the dataset:", df_audio_features.shape)
df_audio_features.head()
df_audio_features.info()
```

```
#SONG METRICS DATA CLEANUP

columns_to_drop = ['analysis_url','track_href','type','uri']
df_audio_features.drop(columns_to_drop, axis=1,inplace=True)

df_audio_features.rename(columns={'id': 'track_id'}, inplace=True)

df_audio_features.shape
```

```
# MERGE BOTH DATAFRAMES

# the 'inner' method will make sure that we only keep track IDs present
in both datasets
df = pd.merge(df_tracks,df_audio_features,on='track_id',how='inner')
print("Shape of the dataset:", df_audio_features.shape)
df.head()
df.info()

df[df.duplicated(subset=['artist_name','track_name'],keep=False)]
```

```
#WRITE TO CSV
```

```
df.to_csv('try1.csv')
```

APPENDIX B: DATA PROCESSING

R codes to process and model data

DATA CLEANING CODE:

```
```{r}
#READ DATA

sp=read.csv("2018_massive.csv",
            stringsAsFactors=FALSE,na.strings = "")

#get rid of unique columns and time signature
sp=sp[,-c(1,2,3,4,17)]
```

```{r}
#CLEAN THE DATA : Basic Charts

#1.CHANGE DURATION TO MINUTES FROM MILLISECONDS, 3 decimel places
sp$duration_ms=round(((sp$duration_ms*0.001)/60),3)


#2.MISSING VALUES
which(is.na(sp))


#3.OUTLIERS:

#3a. duration
hist(sp$duration_ms,col="salmon",
     xlab="Duration (minutes)",
     freq = FALSE, #show densities instead of freq.
     main = "Duration distribution")
summary(sp$duration_ms)

library(ggpubr)
ggscatter(sp, x = "popularity", y = "duration_ms",
          xlab = "pop", ylab = "duration", title = "duration", color = "seagreen")+
    scale_x_continuous(breaks=seq(0,100,10))+
    geom_hline(yintercept=quantile(sp$duration_ms), color="tomato2", size=1)
```
```

```
quantile(sp$duration_ms, probs = c(0.1, 0.9))

library(dplyr)
  sp=dplyr::filter(sp, duration_ms >= c(2))
  sp=dplyr::filter(sp, duration_ms <= c(10))


#3b. loudness
hist(sp$loudness,col="salmon",
     xlab="loudness",
     freq = FALSE, #show densities instead of freq.
     main = "loudness distribution")

  ggscatter(sp, x = "popularity", y = "loudness",
          xlab = "pop", ylab = "loudness", title = "loudness", color = "seagreen")+
    scale_x_continuous(breaks=seq(0,100,10))+
    geom_hline(yintercept=quantile(sp$loudness), color="tomato2", size=1)

    sp=dplyr::filter(sp, loudness <= c(0))


#speechiness
 #scatter with quartiles

hist(sp$speechiness,col="salmon",
     xlab="speechiness",
     freq = FALSE, #show densities instead of freq.
     main = "Speechiness distribution")
quantile(sp$speechiness)


ggscatter(sp, x = "popularity", y = "speechiness",
          xlab = "pop", ylab = "speechiness", title = "speechiness", color = "seagreen")+
      geom_hline(yintercept=quantile(sp$speechiness), color="tomato2", size=1)


sp=dplyr::filter(sp, speechiness <= c(0.6))



#3c. tempo
  ggscatter(sp, x = "popularity", y = "tempo",
          xlab = "pop", ylab = "tempo", title = "tempo", color = "seagreen")+
      geom_hline(yintercept=quantile(sp$tempo), color="tomato2", size=1)

    sp=dplyr::filter(sp, tempo > c(0))
```

```
#3d. mode
mode1=dplyr::filter(sp, mode == c(1))
mode0=dplyr::filter(sp, mode == c(0))
mode=nrow(mode1)+nrow(mode0)


#3f. key
key=summarise(group_by(sp,key),
                        avgpop=mean(popularity, na.rm=TRUE))
#barchart

ggplot(data=sp, aes(x=sp$key, y=mean(sp$popularity))) +
  geom_bar(stat="identity", fill= "seagreen")+
  ggtitle("Average popularity of each key")+
     scale_x_continuous(breaks=seq(0,11,1))+
  xlab("Key")+ ylab("Average Popularity")


#4.ADD UNITS TO COLUMN NAMES
colnames(sp)=c("popularity", "acousticness", "danceability", "duration_minutes",
"energy", "instrumentalness", "key", "liveness", "loudness_decibels",
"mode_minor_major", "speechiness", "tempo_bpm", "valence")


#5. ADD KEY
sp$notation=ifelse(sp$key=="0", "C",
ifelse(sp$key=="1", "C#,Db",
ifelse(sp$key=="2", "D",
ifelse(sp$key=="3", "D#,Eb",
ifelse(sp$key=="4", "E",
ifelse(sp$key=="5", "F",
ifelse(sp$key=="6", "F#, Gb",
ifelse(sp$key=="7", "G",
ifelse(sp$key=="8", "G#,Ab",
ifelse(sp$key=="9", "A",
ifelse(sp$key=="10", "A#, Bb",
ifelse(sp$key=="11", "B",NA)))))))))))

```

```{r}
#UNDERSTAND THE DATA : Distribution Charts

sp$key=as.factor(sp$key)
```

```
hist(sp$popularity,col="salmon",
    xlab="Popularity",
    freq = FALSE, #show densities instead of freq.
    main = "Popularity distribution")
attach(sp)
opar <- par(no.readonly = TRUE)
par(mfrow=c(3,3))
boxplot(acousticness, main = "Boxplot of Acousticness", ylab = "Rating")
#axis(2, at = seq(0,1,0.1), labels = seq(0,1,0.1))
boxplot(danceability, main="Boxplot of Danceability", ylab="Rating")
boxplot(energy, main="Boxplot of Energy", ylab="Rating")
boxplot(instrumentalness, main="Boxplot of Instrumentalness", ylab="Rating")
boxplot(liveness, main="Boxplot of Liveness", ylab="Rating")
boxplot(speechiness, main="Boxplot of Speechiness", ylab="Rating")
boxplot(valence, main="Boxplot of Happiness", ylab="Rating")
par(opar)
detach(sp)

attach(sp)
opar <- par(no.readonly = TRUE)
par(mfrow=c(3,3))
hist(acousticness, main = "Accousticness", ylab = "Frequency")
#axis(2, at = seq(0,1,0.1), labels = seq(0,1,0.1))
hist(danceability, main="Danceability", ylab="Frequency")
hist(energy, main="Energy", ylab="Frequency")
hist(instrumentalness, main="Instrumentalness", ylab="Frequency")
hist(liveness, main="Liveness", ylab="Frequency")
hist(speechiness, main="Speechiness", ylab="Frequency")
hist(valence, main="Happiness", ylab="Frequency")
par(opar)
detach(sp)


attach(sp)
opar <- par(no.readonly = TRUE)
par(mfrow=c(2,2))
#axis(2, at = seq(0,1,0.1), labels = seq(0,1,0.1))
hist(duration_minutes, main = "Distribution of Duration", ylab = "Frequency")
hist(loudness_decibels, main = "Distribution of Loudness", ylab = "Frequency")
hist(tempo_bpm, main = "Distribution of Tempo", ylab = "Frequency")
hist(mode_minor_major, main = "Distribution of Mode", ylab = "Frequency")
par(opar)
detach(sp)
```

```{r}
#DATA REDUCTION + CORRELATION ANALYSIS
sp$key=as.numeric(sp$key)
cor(sp[,-c(14)]) #bivariate correlations

library(psych)
pairs.panels(sp,
             method = "pearson",
             hist.col = "#00AFBB",
             density = TRUE,
             ellipses = FALSE)


sp=sp[,-c(5,6)] #dimension reduction get rid of energy and instrumentalness

```

```{r}
#STANDARDIZE
sp2=sp[,-c(1,5,8,12)]
sp3=sp[,c(1,5,8,12)]

for (i in 1:8) {
  sp2[,i]=(sp2[,i]-mean(sp2[,i]))/(sd(sp2[,i]))

}

sp2=round(sp2, digits=3)

#arrange dataframe
spotify=cbind(sp3, sp2)

#FACTORIZE
spotify$mode_minor_major=as.factor(spotify$mode_minor_major)
spotify$key=as.factor(spotify$key)

```

```{r}

#ADD POPULARITY CATEGORIES
spotify$popularcategory=1

summary(spotify$popularity)

spotify$popularcategory=ifelse(spotify$popularity<median(spotify$popularity), "0", "1")
```

```
table(spotify$popularcategory)

spotify$popularcategory=as.factor(spotify$popularcategory)
```
```{r}
#VARIABLE SELECTION
summary(spotify)

attach(spotify)
opar <- par(no.readonly = TRUE)
par(mfrow=c(3,4))
hist(popularity, main = "Popularity", ylab = "Frequency")
hist(acousticness, main = "Accousticness", ylab = "Frequency")
hist(danceability, main="Danceability", ylab="Frequency")
hist(duration_minutes, main="Duration", ylab="Frequency")
hist(liveness, main="Liveness", ylab="Frequency")
hist(loudness_decibels, main = "Loudness", ylab = "Frequency")
hist(speechiness, main="Speechiness", ylab="Frequency")
hist(tempo_bpm, main = "Tempo", ylab = "Frequency")
hist(valence, main="Happiness", ylab="Frequency")
par(opar)
detach(spotify)

write.csv(spotify, file="spotifyclean.csv")
```
```

## LINEAR REGRESSION CODE:

```{r}
splr=read.csv("spotifyclean.csv",
              stringsAsFactors=FALSE,na.strings = "")

splr=splr[,-c(1,3,5, 14)]
```
```{r}
#VARIABLE TRANSFORMATION FOR LINEARITY
qplot(x=acousticness, bins=30, data = splr)#very skewed
qplot(kader:::cuberoot(splr$acousticness))
splr$acousticness <- -log((1/(kader:::cuberoot(splr$acousticness)))-1, base = exp(1))
qplot(log(splr$acousticness))#best tranformation
{qqnorm(splr$acousticness)
qqline(splr$acousticness)}
qqPlot(splr$acousticness)

```
```

```{r}
#MODELS

#all the variables
model1 <- lm( popularity
~mode_minor_major+acousticness+danceability+duration_minutes+liveness+loudness_d
ecibels+speechiness+tempo_bpm+valence, data = splr)

summary(model1)

#all variables plus all interactions
model2 <- lm( popularity
~mode_minor_major+acousticness+danceability+duration_minutes+liveness+loudness_d
ecibels+speechiness+tempo_bpm+valence+
loudness_decibels:acousticness+loudness_decibels:valence+loudness_decibels:danceabili
ty+danceability:valence, data = splr)
summary(model2)

#reduction by AIC
library(gvlma)
library(MASS)
stepAIC(model2, direction="backward")

#reduced model
model3=lm(popularity ~ mode_minor_major + acousticness + danceability +
   loudness_decibels + speechiness + tempo_bpm + valence +
acousticness:loudness_decibels +
   danceability:loudness_decibels, data = splr)
summary(model3)

#add polynomial acousticness and speechability from distribution graphs
model4=lm(popularity ~ mode_minor_major + acousticness + poly(acousticness,2)+
danceability +
   loudness_decibels + speechiness + poly(speechiness,2) + tempo_bpm + valence +
acousticness:loudness_decibels +
   danceability:loudness_decibels, data = splr)
summary(model4)

stepAIC(model4, direction="backward")

#reduced model
model5=lm(popularity ~ acousticness + poly(acousticness, 2) +
   danceability + loudness_decibels + poly(speechiness, 2) +
   tempo_bpm + valence + acousticness:loudness_decibels +
danceability:loudness_decibels,
   data = splr)
```

```
summary(model5)

stepAIC(model5, direction="backward")


model6=lm(formula = popularity ~ poly(acousticness, 2) +
    danceability + loudness_decibels + poly(speechiness, 2) +
    tempo_bpm + valence + acousticness:loudness_decibels +
danceability:loudness_decibels,
    data = splr)
summary(model6)
stepAIC(model6, direction="backward")
```

```{r}
#MODEL PERFORMANCE
confint(model6)

par(mfrow=c(2,2))
plot(model6)

library(gvlma)
gvm=gvlma(model6)
summary(gvm)

```

NEURAL NETWORK CODE:

```{r}
library(dummies)
library(ggplot2)
library(car)
library(kader)
library(MASS)
library(scales)
library(psych)
library(e1071)
library(outliers)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(mgcv)
library(ROCR)
library(gains)
library(Metrics)
```

```
library(caTools)
library(e1071)
library(gmodels)
library(corrplot)
library(hydroGOF)
library(psycho)
library(gridExtra)
library(grid)
library(pROC)
```

```{r}
sp=read.csv("spotifyclean.csv",
            stringsAsFactors=FALSE,na.strings = "")

sp=sp[,-c(1,5,14)]
```

```{r}
#data sampling

set.seed(111)
train.index = sample(row.names(sp), 0.60*dim(sp)[1])
valid.index = setdiff(row.names(sp), train.index)
train.df = sp[train.index,]
valid.df = sp[valid.index,]
```

```{r}
#PREDICTION
library(neuralnet)

#model
#model <- neuralnet(Output~Input,trainingdata, hidden=10, threshold=0.01)
model1 = neuralnet(popularity~
mode_minor_major+acousticness+danceability+duration_minutes+liveness+loudness_de
cibels+
                speechiness+tempo_bpm+valence,
data = train.df,
algorithm = 'rprop+')


#prediction using training and validation
results1 <- compute(model1, valid.df) #exclude response variable
predicted1=data.frame(as.numeric(results1$net.result))
```

```
#RMS error
library(Metrics)
RMSE(results1$net.result, valid.df$popularity) #7.78
MAE(results1$net.result, valid.df$popularity) #0.11
mape(results1$net.result, valid.df$popularity) #6.15
me(results1$net.result, valid.df$popularity)


me(predicted1$results1.net.result, valid.df$popularity) #-0.08
```

KNN CODE:

```{r}
spdf=read.csv("spotifyclean.csv",
              stringsAsFactors=FALSE,na.strings = "")

#remove notation
spdf=spdf[,-c(1,3,5)]
```

```{r}
#SAMPLING 60-40
set.seed(111)
train.index = sample(row.names(spdf), 0.6*dim(spdf)[1])
valid.index = setdiff(row.names(spdf), train.index)
train.df = spdf[train.index,]
valid.df = spdf[valid.index,]
```

```{r}
#CREATE MODEL MATRIX (REMEMBER TO FACTOR KEY and MODE)

sptraining <- model.matrix(~factor(mode_minor_major)+ acousticness+danceability+
duration_minutes+liveness+ loudness_decibels+ speechiness+ tempo_bpm+
valence,data=train.df)

spvalid <- model.matrix(~factor(mode_minor_major)+ acousticness+danceability+
duration_minutes+ liveness+ loudness_decibels+ speechiness+ tempo_bpm+
valence,data=valid.df)

sptraining=data.frame(sptraining[,-c(1)])
spvalid=data.frame(spvalid[,-c(1)])
```

```
```

```{r}
#SEPARATE THE OUTPUT COLUMN FROM EACH

train_labels <- train.df$popularity
valid_labels <- valid.df$popularity


train_class <- train.df$popularcategory
valid_class <- valid.df$popularcategory

```
```{r}
#KNN OF POPULAR CATEGORY - CLASSIFICATION

library(class)

#PICK BEST k
#best way to pick k is sqrt of number of fields = 70 odd number

#initialize a data frame with two columns: K, and accuracy
accuracy.df <- data.frame(k=seq(1,150,1), accuracy=rep(0,150))


for (i in 1:150) {
  knn.class <- knn(train = sptraining, test = spvalid,cl = train_class, k=i)
  accuracy.df[i, 2] <-
    100*sum(valid_class==knn.class)/NROW(valid_class)
}

accuracy.df$accuracy=round(accuracy.df$accuracy, digits=1)
accuracy.df <- accuracy.df[accuracy.df$k %% 2 != 0, ] #get rid of even k values


library(ggpubr)
ggscatter(accuracy.df, x = "k", y = "accuracy",
          xlab = "k", ylab = "accuracy", title = "k", color = "seagreen")+
    scale_x_continuous(breaks=seq(0,150,10))


#MODEL
classification_knn <- knn(train = sptraining, test = spvalid,cl = train_class, k=119)
knnclass=data.frame(classification_knn)
```

```
library(gmodels)
confusionMatrix(as.factor(valid.df$popularcategory),classification_knn)

#plot of lift chart
library(CustomerScoringMetrics)
cumGainsChart(classification_knn, valid_class, resolution = 1/10)
lift.knn <- lift(relevel(as.factor(valid_class), ref="1") ~ knnclass$classification_knn)
xyplot(lift.knn, plot = "gain")

r <- roc(as.numeric(valid_class), as.numeric(knnclass$classification_knn))
plot.roc(r)
auc(r)

```

```{r}
#KNN OF POPULARITY RATING - PREDICTION

#PICK BEST k
#best way to pick k is sqrt of number of fields = 70 odd number

#initialize a data frame with two columns: K, and accuracy
accuracy.df2 <- data.frame(k=seq(1,150,1), accuracy=rep(0,150))


for (i in 1:150) {
  knn.pred <- knn(train = sptraining, test = spvalid,cl = train_labels, k=i)
  accuracy.df2[i, 2] <-
    100*sum(valid_labels==knn.pred)/NROW(valid_labels)
}

accuracy.df2$accuracy=round(accuracy.df2$accuracy, digits=2)
accuracy.df2=na.omit(accuracy.df2)
accuracy.df2 <- accuracy.df2[accuracy.df2$k %% 2 != 0, ] #get rid of even k values


ggscatter(accuracy.df2, x = "k", y = "accuracy",
          xlab = "k", ylab = "accuracy", title = "k", color = "seagreen")+
    scale_x_continuous(breaks=seq(0,150,10))



#MODEL
prediction_knn <- knn(train = sptraining, test = spvalid,cl = train_labels, k=61)
knnpred=data.frame(as.numeric(prediction_knn))
```

#Model Evaluation

RMSE(knnpred$as.numeric.prediction_knn., valid_labels)#45.21
me(knnpred$as.numeric.prediction_knn., valid_labels) #-44.38
MAE(knnpred$as.numeric.prediction_knn., valid_labels) #44.38
mape(knnpred$as.numeric.prediction_knn., valid_labels) #4.79

```

## SVM, RANDOM FOREST, LDA CODE:

```{r Import cleaned data}
Music_=read.csv("spotifyclean.csv",
                stringsAsFactors=FALSE,na.strings = "")

Music=Music_[,-c(1,3,5,14)]

```

```{r}
#SAMPLING 50-30-20

set.seed(111)
#SAMPLING 50-30-20
set.seed(111)

train.index2 <- sample(row.names(Music), 0.5*dim(Music)[1])
valid_Test.index <- setdiff(row.names(Music), train.index2)

MusicTrain <- Music[train.index2,]
valid_Test.df <- Music[valid_Test.index,]


valid.index2 <- sample(row.names(valid_Test.df),
                       0.6*dim(valid_Test.df)[1])
test.index2 <- setdiff(row.names(valid_Test.df),
                       valid.index2)

MusicValid <- valid_Test.df[valid.index2,]
MusicTest <- valid_Test.df[test.index2,]

```

```{r Random forest }

```
rf <- randomForest(popularity~., data = MusicTrain)
rf
importance(rf)

rf.pred.train <- predict(rf, MusicTrain)
rf.pred.valid <- predict(rf, MusicValid)
rf.pred.test <- predict(rf, MusicTest)

#use rmse to evaluate the performance
RMSE(rf.pred.train, MusicTrain$popularity)#3.400
RMSE(rf.pred.valid, MusicValid$popularity)#7.839
RMSE(rf.pred.test, MusicTest$popularity)#7.549

#use MAE
MAE(rf.pred.train, MusicTrain$popularity) #2.66
MAE(rf.pred.valid, MusicValid$popularity)#6.21
MAE(rf.pred.test, MusicTest$popularity) #6.00

#MAPE
mape(rf.pred.train, MusicTrain$popularity) #0.047
mape(rf.pred.valid, MusicValid$popularity)#0.112
mape(rf.pred.test, MusicTest$popularity) #.108

#Use Mean error
me(rf.pred.train, MusicTrain$popularity)#0.086
me(rf.pred.valid, MusicValid$popularity)#0.319
me(rf.pred.test, MusicTest$popularity)#0.198

```
```{r SVM}
mu.svm <- svm(Music[,-1], Music[,1])
data.frame(actual=Music[,1], predicted=mu.svm$fitted)


RMSE(mu.svm$fitted, Music$popularity)#7.58
me(mu.svm$fitted, Music$popularity) #-1.31
MAE(mu.svm$fitted, Music$popularity) #5.66
mape(mu.svm$fitted, Music$popularity) #0.104

```

```{r LDA}
Music2=Music_[,-c(1,3,5,2)]
```

```
set.seed(111)
train.indexm <- sample(row.names(Music2), 0.6*dim(Music2)[1])
valid_Test.indexm <- setdiff(row.names(Music2), train.indexm)

MusicTrain.cl <- Music2[train.indexm,]
MusicValid.cl <- Music2[valid_Test.indexm,]

mu.lda <- lda(popularcategory~., MusicTrain.cl)

mu.lda.pred <- predict(mu.lda, as.data.frame(MusicValid.cl[,-10]))
data.frame(actual=MusicValid.cl[,10], predicted=mu.lda.pred$class)

confusionMatrix(as.factor(MusicValid.cl$popularcategory), as.factor(mu.lda.pred$class))


lift.lda <- lift(relevel(as.factor(MusicValid.cl$popularcategory), ref="1") ~
mu.lda.pred$class)
xyplot(lift.lda, plot = "gain")

r <- roc(as.numeric(MusicValid.cl$popularcategory), as.numeric(mu.lda.pred$class))
plot.roc(r)

auc(r)
```