Abstract

This is the project proposal for Cultural / Entertainment problem

24S1-C-NYP-ITI105 - MACHINE LEARNING PROJECT

Final Report

Lye Suh Jeng (7487427Y)  
Lee Li Neng (6203055B)

Lim Chan Boon (9704541M)

Date: 18 August 2024

Table of Contents

[1. Project 2](#_Toc174886529)

[2. ML Formulation 2](#_Toc174886530)

[3. High Level steps in Machine Learning 3](#_Toc174886531)

[5. Discover & Visualize Data to Gain Insights 4](#_Toc174886532)

[4.1 Data Visualization using Histogram 4](#_Toc174886533)

[4.2 Target’s Skewness and Kurtosis 4](#_Toc174886534)

[4.2 Data Visualization using Boxplots 5](#_Toc174886535)

[5.3 Data Visualization using Correlation Matrix 6](#_Toc174886536)

[5. Data Preparation & Feature Engineering 7](#_Toc174886537)

[5.1 Data Preparation & Feature engineering steps 7](#_Toc174886538)

[5.2 Lesson Learnt in Data Preparation & Feature engineering 9](#_Toc174886539)

[6. Modelling and Experiments 10](#_Toc174886540)

[6.1 Modeling and performance measurement 10](#_Toc174886541)

[6.2 Lesson Learnt in Modeling 12](#_Toc174886542)

[7. Model Deployment 13](#_Toc174886543)

[7.1 Leasson learnt in deployment 14](#_Toc174886544)

[8. Alternative to Machine Learning 14](#_Toc174886545)

[9. Code 14](#_Toc174886546)

[10. Experimental Log 14](#_Toc174886547)

# 1. Project

4. Cultural/Entertainment

a) The success of the song can often been measured by whether the song is on the Hit Chart such as Billboard Hot 100. It is important for music labels to know what makes a song successful so that they can focus their budget on making songs that has the highest chance of being successful.

# ML Formulation

We would like to create a machine learning solutions that can predict the song’s popularity score using dataset gotten from <https://www.kaggle.com/datasets/yasserh/song-popularity-dataset>.

This is a regression problem with target as song\_popularity, which represent the song popularity score.

The output of the solutions will be:

1. Song\_popularity score (range from 0 t o100). Score closer to 100 will be most popular and score near to 0 is less popular.
2. Interpretation of the score as follows:
   * ‘Very popular and has high potential to be top10’ if song\_popularity >= 75
   * ‘Moderate popularity’ if song\_popularity >= 50
   * ‘Low popularity’ if song\_polarity < 50

The input of the solutions will be all or some of the features in song\_data.csv depends on the outcome of the feature engineering as follows:

* + song\_duration\_ms
  + acousticness,
  + danceability,
  + energy,
  + instrumentalness,
  + key
  + liveness
  + loudness
  + audio\_mode
  + speechiness
  + tempo time\_signature
  + audio\_valence

# High Level steps in Machine Learning

The following steps are taken in the machine learning:

1. Load dataset
2. Discover & visualize data to gain insights
3. Split data into train and test datasets
4. Scale training data for Feature Selection
5. Feature scaling
6. Train, fine tune and evaluate models
7. Compare performance of models
8. Deployment

# Discover & Visualize Data to Gain Insights

## 4.1 Data Visualization using Histogram

After loading the dataset from song\_data.csv into pandas Dataframe, we ploted the folloing histograms:

A chart of different types of music

Description automatically generated with medium confidence

Observations:

* Target = Song\_popularity (Popularity score from 0 to 100 with 100 as the most popular song). The rest of the columns in the dataset are feratures.
* Instrumentalness has small data variance and may not be good feature to determine song\_popularity

## 4.2 Target’s Skewness and Kurtosis

* Song\_popularity has skewness of -0.5015, the tail of distribution is longer towards the left hand side of the curve
* Song\_popularity’s kurtosis is -0.17, which is less than 3. This is a platykurtic distriubtion.

## 4.2 Data Visualization using Boxplots

A group of diagrams with numbers

Description automatically generated with medium confidence

**Observations:**

* song\_duration\_ms, danceability, energy, liveness, loudness and speechiness and tempo seem to have outliers.
* instrumentalness can't be plot using boxplot, highly suspect that it's not a feature that is useful to determine song\_popularity.

## Data Visualization using Correlation Matrix

A screenshot of a computer screen

Description automatically generated

**Observations:**

There are multi-colinearity between the following features:

* danceability and audio\_valence (0.33)
* loudness and energy (0.76)

# 5. Data Preparation & Feature Engineering

## 5.1 Data Preparation & Feature engineering steps

The following are the steps taken in data preparation:

1. Identify target as song\_popularity (range from 0 to 100. Higher score means more popular)
2. Check & confirm no missing data and no non-numeric data.
3. Remove ‘song\_name’ (which represent name of songs) which doesn’t contribute to song\_popularity score.
4. Split training and test data using 80:20 ratio
5. Remove outliners from training data
   * As in box plots below, there are outliners in song\_durations, dancebility, liveness, loudness, speechiness and tempo

A group of diagrams with numbers

Description automatically generated with medium confidence

* In order not to remove too much data, we try to remove outliners in ‘tempo’, ‘danceability’ and ‘energy’ features. Data in these 3 featues which are outside of 50% of IQR for both Q1 and Q3 ends will be removed.
* A total of 279 records are removed. After removal, 14789 data remained.
* This approach improve the performance of the model trained using Identify 13 features as in other columns in the song\_data.csv.

1. Handle Imbalanced datasets in training data:
   * Majority of the data are with song\_popularity in score range from 50 to 75 (third quartile, Q3)
   * The spread of song\_popularity of the total 14789 records are:
     + Q1 (song\_popularity score <= 25): 1897
     + Q2 (song\_popularity score > 25 & <=50): 4091
     + Q3 (song\_popularity score > 50 & <=75):6509
     + Q4 (song\_popularity score <= 25): 2382: 2036
   * Records in Q1, Q2 and Q4 are upsampled (duplicate some records using ‘resample’ function in sklern.utils) to the same amout of records as in Q3.
   * This approach prevent ML learning from being weaker in predicting song\_popularity in minitory quartiles, especially Q1.
   * This approach significantly improve the performance of the best trained model (Random Forrest) as showed in table below:

|  |  |  |
| --- | --- | --- |
|  | Without handling of imbalanced dataset | With handling of imbalanced dataset |
| Adjusted R2 (Train) | 0.263 | 0.975 |
| Adjusted R2 (Test) | 0.049 | 0.389 |

1. From the 13 features, identify 10 numeric and 3 categorical features.
   * Numeric features: 'song\_duration\_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'audio\_valence'
   * Categorical features: 'audio\_mode', 'time\_signature', 'key'
2. Encode categorical features using dummy enconding. Additional features introduced due to encording are:
   * 'audio\_mode\_0', 'audio\_mode\_1', 'time\_signature\_0', 'time\_signature\_1', 'time\_signature\_3', 'time\_signature\_4', 'time\_signature\_5', 'key\_0', 'key\_1', 'key\_2', 'key\_3', 'key\_4', 'key\_5', 'key\_6', 'key\_7', 'key\_8', 'key\_9', 'key\_10', 'key\_11'.
3. Scale numeric features using MinMaxScaler before feature selection
4. Feature reduction
   * 3 approaches are used:
     + Using Lasso Regression
       - GridSearch is used to find the best alpha parameter.
       - Using alpha = 0.008, it’s found that ‘time\_signature\_0’ and ‘time\_signature\_5 has Zero value (0) of coef\_ parameter ventor, which means tht these 2 features can be removed.
     + Using Variance Inflation Factor (VIF)
       - We use threshold of VIF value > 5 as the guidelines to remove a feature.
       - It’s found that 'time\_signature\_3', 'time\_signature\_4', 'key\_10', 'key\_9', 'key\_8', 'key\_7', 'key\_6', 'key\_5', 'key\_4', 'key\_3', 'key\_2', 'key\_1', 'key\_0', 'time\_signature\_5', 'key\_11', 'time\_signature\_1', 'time\_signature\_0', 'audio\_mode\_1', 'audio\_mode\_0' have VIF value of inf (Infinite) which is greater than 5.
   * We decided to remove features as recommeded by VIF. With this, 10 featues are remained, which are:
     + 'song\_duration\_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'audio\_valence'

## 5.2 Lesson Learnt in Data Preparation & Feature engineering

We learnt the following during the data prepration and feature engineering:

* We should only handle imbalanced data sets on training data only
* Handling Imbalanced data sets in regression problem is a must! Without it, we won’t train a good model at all despite doing any other feature engineering.
* Besides using Lasso Regression and VIF, we tried PCA (Principle Components Analysis) as well in feature or dimensionality reduction. (We tried before handle the imbalanced data sets). We found that PCA won’t be efficient for this case. As shown in the graph below, the need to added up many PC components in order to reach 95% of variance ratio, which end up didn’t reduce number of features.

A graph of a number of blue bars

Description automatically generated

* MinMaxScaler is better than StandardScaler in model trained using Random Forrest algorithms as showned in the table below. Model trained using MinMaxScaler has higher Adjusted R2 (test) compared to model trained using StandardScaler. We think it’s due to the dataset is not normally distributed and have outliers. Thus MinMaxScaler is a better choice than StandardScaler.
* Removing outliners help in improve the performance of model training. However, detecting outliners and the challenge of having too little data after removing outliners are challenge. Plotting box plot helps us to to see the distributing of data. However, some trial and error tests are required in order to confirm removing too many data.

# 6. Modelling and Experiments

## 6.1 Modeling and performance measurement

The following algorithms has been tried to train the regression model:

* Polynomial Regression (PR)
* Multiple Linear Regression (MLR)
* Elastic Net Regression (ENR)
* Adaboost (ADA)
* Gradient Boosting Regressor (GBR)
* Support Vector Regression wth RBF kernel (RBF)
* Random Forest Regressor (RF)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PR** | **MLR** | **ENR** | **ADA** | **GBR** | **RBF** | **RF** |
| **r2\_train** | 0.183695 | 0.086228 | 0.085989 | 0.073046 | 0.976131 | 0.489282 | 0.975591 |
| **r2\_test** | -0.188035 | -0.079827 | -0.078239 | -0.067682 | 0.319368 | -0.058251 | 0.383862 |
| **adj\_r2\_train** | 0.183412 | 0.085912 | 0.085673 | 0.072725 | 0.976123 | 0.489105 | 0.975583 |
| **adj\_r2\_test** | -0.190881 | -0.082413 | -0.080822 | -0.07024 | 0.317737 | -0.060786 | 0.382386 |
| **mse\_train** | 628.217362 | 703.226505 | 703.409908 | 713.371148 | 18.369188 | 393.041651 | 18.784636 |
| **mse\_test** | 568.066675 | 516.326269 | 515.567315 | 510.519305 | 325.448947 | 506.009797 | 294.610463 |

Table below shows the performance comparision of various models

Legend of Performance Measurements:

* R2\_train: coefficient of determination, R2  of train data
* R2\_test: coefficient of determination, R2  of test data
* adj\_r2\_train: Adjusted R2  of train data
* adj\_r2\_test: Adjusted R2  of test data
* mse\_train: Mean squared error of train data
* mse\_test: Mean squared error of test data

Among all the performance measurement, we use adjusted R2  of test data (adj\_r2\_test) as bench mark. GBR and RF are the best models as both have high adj\_r2\_test(adj\_r2\_test) . RF has slighly higher adjusted\_R2\_test compared to GBR. Thus, we choose RF as the best model to predict the song\_popularity score.

Hyperparameter tuninig is performed using either Random search or Grid search to find out the best parameter in all models except PR models.

In order to visualise the performance of the trained model, we plot histograms and Residual plot of the best model trained using Random Forrest as shown below:

A comparison of a graph

Description automatically generated with medium confidence

A graph of blue dots

Description automatically generated

Both plots give us a rough idea how good is the trained models. We obeserve that:

* From histogram, we can see some discrepancy between predicted data and real test data. For instance, the model predicted less number of song\_popularity score for score range from 0 to 20 compared to test data
* The prediction work better for song\_popularity score in Q1 (<=25) and Q3 (>= 75).
* Many reconds in song\_popularity in range of Q3 and Q1 are wrongly predicted in range 40 to 60.

## 6.2 Lesson Learnt in Modeling

We observed that:

* Hyperparameter tunning and cross-validatation helps to improve the performance of the models.
* Non-parametric algorithms such as Random Forrest and Gradient Boosting Regressor can trained a better models compare to parametric algorithms such as Polynomial Regressor. However, non-parametric algorithms require more compute resource. Without GPU, it’s time-consuming to train a model using

non-parametric algorithm.

# 7. Model Deployment

We use Gradio to create an intuitive web-based user interface and use Hugging Face to host the machine learning model. The following is the snapshot of the GUI for music label to key in song’s feature and predict the Song’s Popularity score.

A screenshot of a computer

Description automatically generated

The url of the deployed application in Hugging Face space is: https://huggingface.co/spaces/suhjeng/Song-Popularity-Predictor

Here’re the high level procedures in our deployment:

* Use pickle framework to save MinMaxScaler to a .pkl file (as in <https://huggingface.co/spaces/suhjeng/Song-Popularity-Predictor/blob/main/mm_scaler1.pkl> and save the trained modle to another .pkl ( as in https://huggingface.co/spaces/suhjeng/Song-Popularity-Predictor/blob/main/best\_rf\_model.pkl) .
* Write a python code to use Gradio module to create a GUI interface
* Prepare ‘requirements.txt’ file that list down out the required modules and library required to install and run the model, as in <https://huggingface.co/spaces/suhjeng/Song_Popularity_Predictor/blob/main/requirements.txt>. We also need to find out the python module version used in model training and specify the exact version of respective modules in requirements.txt, other we hit error during deployment. Here’re the content of requirments.txt:
  + numpy==1.26.4
  + gradio==4.41.0
  + scipy
  + scikit-learn==1.3.2
  + joblib==1.4.2
  + pandas==2.1.4
  + pickle5

## 7.1 Leasson learnt in deployment

It’s important to have scale the inputs gotten from user interface before passing it to predictor.

# 8. Alternative to Machine Learning

Without this song popularity predictor, music labels may need to to recruit artists that are music expert to spent time listening and select new songs that are potentially to be popular.

This predictor helps music labels to filter out those song’s that are predicted to has low popularity score and have artists spend time to listen to song that has high popularity score.

# 9. Code

* The source code in colab format can be found in https://github.com/dy018/project105/blob/main/25\_105\_ML\_project\_song\_final.ipynb
* The model (model.pkl), scaler (mm\_scaler.pkl) and python code used to deploy the model (song\_popularity\_predictor.py) are in

https://huggingface.co/spaces/suhjeng/Song-Popularity-Predictor/tree/main

# 10. Experimental Log

The experimental log can be found in <https://github.com/dy018/project105/blob/main/105_Experiment_log.xlsx>