ST4248 Project

Chong Wan Fei, Eldora Boo, Salman Yusuf, Teo Ming Jun

Contents

Chapter 1

Introduction

Music Popularity Prediction Using Machine Learning Techniques

1.1 Project Description

In this project, we aim to predict the popularity of a song based on its features using various machine learning models, such as Random Forest, Linear Regression, Support Vector Machines (SVM), and XGBoost. We will preprocess the data, perform Principal Component Analysis (PCA) for dimensionality reduction, and apply hyperparameter tuning to find the optimal parameters for each model. The performance of each model will be evaluated using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared, and accuracy. By comparing the results of the different models, we will determine the most effective approach for predicting a song's popularity. This analysis can be used to better understand the factors that contribute to a song's popularity and could potentially be employed by music producers, record labels, or streaming platforms to predict the success of new releases.

1.2 Authors

- Chong Wan Fei
- Eldora Boo
- Salman Yusuf
- Teo Ming Jun

1.3 Motivation

In the highly competitive music industry, understanding what makes a song or artist valuable is crucial for marketing strategies, album releases, and aligning efforts with public interest. Spotify, one of the largest streaming services, reaches over 191 million active users and publishes several charts, including a daily “Top 50” ranked by the number of streams. For artists, knowing when their work will become popular on the platform is essential for boosting marketing efforts and steering efforts towards more promising cases. Therefore, predicting a song’s popularity using musical characteristics can be of great interest to artists and the music industry.

Chapter 2

Data Collection and Preprocessing

2.1 Data Collection

Our dataset is obtained from Kaggle, where the user sourced the dataset from a research paper written by researchers in Brazil and published in Mendeley Data. Echo Nest and Lyrics Genius APIs were used to gather music data from Spotify's database and lyrics of over 28,000 songs published between 1950 and 2019. As this dataset lacks the information on the popularity of the songs, we extracted the popularity data from another dataset that used the Spotify API which calculates popularity based on the total number of plays the songs has and its recency.

2.2 Data Description

The dataset we are using after merging have a total of 17 variables including the target variable popularity, with a total of 13836 instances. The relevant variables are

variables	descriptions
POPULARITY	The popularity of a song calculated based on streams and its recency, rang
ACOUSTICNESS	A confidence range should be from 0.0 to 1.0 on the scale of whether the tr
DANCEABILITY	Suitability of a track, scaling from 0.0 to 1.0, possibly for dancing based or
DURATION (MS)	Duration of the track must be very less i.e., in milliseconds.
ENERGY	A perceptual measure of intensity and activity. ranging within 0.0 to 1.0.
INSTRUMENTALNESS	The measure of the vocal content in the track with a scale of 0 to 1.
KEY	The estimated overall key of the track transformed to the format of a num
LIVELINESS	Probability of the track being performed live in the presence of an audienc
LOUDNESS (dB)	The overall loudness of a track must be in average decibels.
MODE	Mode indicates the modality (major or minor) characteristic of a track dep
SPEECHINESS	The more exclusively speeches-like the recording (e.g., talkshow, audiobook
TEMPO	The overall estimated tempo of any track is always in beats per minute (B
VALENCE (float)	Track positiveness is described from its valence measures (i.e., high or low)

2.3 Preprocessing

As the dataset was published by researchers, the data was pre-processed and clean - with no rows having missing values. However, we noticed certain songs having more than one popularity score. This was due to duplicate songs e.g. the same track from a single and an album being ranked independently - having different popularity scores as explained by Spotify. We took the average of the scores in those cases as we believe this gives a more accurate representation of the song's popularity without introducing bias by favouring one release format.

Features such as name, artist, genre are removed from the dataset as these are irrelevant features. We also normalised numerical variables in the dataset to help standardise the features and reduce computation time. However, variables such as explicit, release_date, key, are excluded as these variables are either categorical or binary variables. Variables that are already ranging from a scale of 0 to 1 are also excluded.

The dataset was also split into training and test sets since this can help to prevent overfitting and give an estimate of the model performance.

Chapter 3

Exploratory Data Analysis

3.1 Distribution of Popularity

With popularity set as the response variable, we first did an analysis of the distribution of songs popularity using a density histogram (Fig.1). We discovered 2 findings. Firstly, there were many songs with 0 popularity, but we decided to leave this data in as songs on Spotify. Secondly, the majority of the songs have a popularity score of around 27.

3.2 Analysing Individual Variables

Our group then subset the data to a size of 500 and analysed the individual variables using scatterplots to determine each feature's relationship to the response variable.

Based on the scatterplots (Fig.2), no one feature has a particularly strong relationship with the response variable popularity. Next, we computed the correlation matrix and the plot below to find out if any collinearity existed within our variables.

3.3 Correlation between Variables

Based on the correlation plot (Fig. 3), we observe a strong positive correlation between loudness and energy, and a strong negative correlation between energy and acousticness. As high correlations between variables might cause multi-collinearity issues in model training, model coefficients and predictions might be unstable and unreliable. To correct this, we considered either removing the highly correlated variable 'energy' or using Principal Component Analysis(PCA) to reduce the dimensions.

3.4 Principal Component Analysis

We will first test the effectiveness of PCA in potentially reducing the multicollinearity issues caused by the strong correlation of energy with loudness and acousticness.

Based on the results above, our group will keep the first 4 principal components as they have a combined PVE of 0.7037, explaining 70.37% of the variation in the data.

From the figure above, our group analysed the contribution of each original variable to each principal component, with higher loadings indicating a stronger relationship. The loading matrix can help identify which variables are most important in explaining the variation in the data, and how they are related to each other.

- 1st Principal Component: Mainly influenced by popularity, loudness, energy, and danceability, with positive loadings for these variables. This suggests that these attributes are strongly correlated with each other, and that songs that are popular and have high energy and danceability tend to be loud.
- 2nd Principal Component: Mainly influenced by duration_ms and instrumentalness, with positive loadings for these variables. This suggests that songs with longer duration and higher instrumentalness tend to be grouped together.
- 3rd Principal Component: Mainly influenced by explicitness, speechiness, and valence, with positive loadings for explicitness and speechiness, and negative loadings for valence. This suggests that songs with explicit language and high levels of speech are associated with lower valence (i.e. less positive emotional expression).
- 4th Principal Component: Mainly influenced by liveness and key, with positive loadings for liveness and key. This suggests that songs in certain keys and with high levels of audience participation (i.e. liveness) tend to be grouped together.

3.5 Scree Plot & Biplot

Further analysis done from the scree plot (Fig.6) agrees that 4 principal components best explain the variance in the data, and that energy has a strong positive correlation with loudness, as well as a strong negative correlation with acousticness(Fig.7).

Chapter 4

Statistical Analysis

To investigate the relationship between the musical characteristics and popularity of a song, we decided to explore using the following predictive statistical learning models: XGBoost, Random Forest, Support Vector Machines.

4.1 Model Selection

4.1.1 XGBoost

XGBoost is an approach that combines many decision tree models to create a single predictive model. Decision Trees are iteratively added to the model, where each new tree is trained to correct the errors of the previous trees.

XGBoost was chosen as it will be able to handle non-linear relationships between the music characteristics and the popularity score as it learns by optimising decision trees, effectively capturing non-linear relationships. Since the features might not have a linear relationship, we believe XGBoost would be suitable. Furthermore, it has regularisation, helping to prevent problems like overfitting such that our model would be able to generalise and predict for new songs well. As we have many features in our dataset, XGBoost which is able to identify feature importance and hence do feature selection is suitable for our dataset. This aligns with the findings from a study by Tian H. and Wen J. (2019) where XGBoost outperformed Logistic Regression in the music recommendation prediction based on the song's metadata.

```r
xgb_model <- xgboost(data=data.matrix(train.X), label=train.Y, nround=25)
xgb_model_predictions <- predict(xgb_model, data.matrix(test.X))
```

We defined the search grid for the hyperparameters we wanted to tune. Some common hyperparameters for XGBoost include nrounds, max_depth, eta, min_child_weight, subsample, colsample_bytree, and gamma. Then we used train() from the caret package, to train the XGBoost model on the training data using different combinations of hyperparameters from the search grid.

```
model <- train(avg_popularity~., data=train, method="xgbTree", trControl=train_control
```

4.1.2 Random Forest

Random forest is an approach that combines multiple decision trees to create a single model. Each tree is trained on a random subset of the data, to reduce overfitting. To get a final prediction, the predictions of all the trees are aggregated.

As mentioned previously, the relationship between the features and popularity might not be linear and there are quite a number of features in the dataset. So, Random Forest model's ability to handle non-linear relationships and measure the importance of each feature like XGBoost makes it a suitable model. Furthermore, Random Forest is robust and is able to deal with missing values or noisy dataset. Despite our dataset being relatively clean, this would be useful for future implementations. This is also evident from research done by Pareek P. and Shankar P. (2022) on prediction of Spotify music tracks, where Random Forest outperformed Linear Support Vector Classifier and kNN for accuracy in prediction.

```
rf_model <- randomForest(avg_popularity~., data=train, mty=13, importance=TRUE)
rf_pred <- predict(rf_model, test)
```

For Random Forest, the most common hyperparameter to tune is the number of variables to consider at each split (mtry). We created a grid that covers a range of possible values for mtry and use train() from the caret package, to train the Random Forest model on the training data using different combinations of hyperparameters from the search grid.

```
rf_default <- train(avg_popularity~., data=reduced_train_data, method='rf', tuneGrid=tu
```

4.1.3 Support Vector Machine (SVM)

SVM is a type of model that can be used for regression tasks. SVM tries to find the optimal boundary that predicts the target variable. It maps the data to a high-dimensional feature space and then finds the hyperplane that maximises the margin between the predictions. It can also handle nonlinear relationships through the use of kernel functions.

Our dataset contains many features and the relationship between musical characteristics and the popularity might not be linear. Hence, SVM is an appropriate model since it can handle high-dimensional feature spaces and complex decision boundaries, through the use of kernels to transform data to be linearly separable.

```
svm_model <- svm(avg_popularity~., data=train, kernel="linear", scale=FALSE)
pred_test <- predict(svm_model, test)
```

To perform hyperparameter tuning for the SVM model, we will use the tune() function from the e1071 package to find the best cost and sigma parameters for the radial basis function (RBF) kernel. We will perform a grid search over a range of values for these two parameters to find the best combination.

```
tuned_svm <- train(avg_popularity~., data=train, method="svmRadial", trControl=train_control, pre
```

4.1.4 Linear Regression

Linear regression is an approach used to model the relationship between a dependent variable and independent variables by finding the best linear equation that describes the relationship. It can be used for prediction as well as understanding the relationship between variables.

We chose the linear regression model as this is a simple model that is easily interpretable, describing the relationship between the musical characteristics and popularity in a simple equation for prediction. In addition, our dataset contains both categorical and continuous variables which linear regression can handle and incorporate. Hence, we believe this is a suitable model for our dataset.

```
lasso <- cv.glmnet(as.matrix(X_train), as.numeric(y_train), alpha=1)
y_pred <- predict(lasso, as.matrix(X_test))
ridge <- cv.glmnet(as.matrix(X_train), as.numeric(y_train), alpha=0)
y_pred <- predict(ridge, as.matrix(X_test))
```

4.2 Model Evaluation

4.2.1 Evaluation Metrics

As the goal of our project is to predict the popularity score, which is a continuous variable, based on the features, this is a regression problem. Hence, it is most appropriate to use Mean Squared Error (MSE) as a metric to evaluate the performance of the model for the prediction of popularity scores. MSE is a measure of the average difference between the actual value and predicted value. In addition, we decided to use metrics such as R-squared to get a better understanding of the suitability of the model for the dataset. R-squared value explains the variance explained by the model.

4.2.2 Results

Model	RMSE_MSE	R_Squared
XGBoost without PCA	# MSE = 0.017 # RMSE = 0.13	# R-squared = 0.52
XGBoost with PCA	# MSE = 0.018 # RMSE = 0.13	# R-squared = 0.49
Random Forest without PCA	# MSE = 0.017 # RMSE = 0.131	# R-squared = 0.52
Random Forest with PCA	# MSE = 0.022 # RMSE = 0.149	# R-squared = 0.37
SVM without PCA	# MSE = 3.28 # RMSE = 1.81	# R-squared = 0.24
SVM with PCA	# MSE = 0.02 # RMSE = 0.15	# R-squared = 0.34
LASSO Regression without PCA	# MSE = 0.02 # RMSE = 0.14	# R-squared = 0.47
LASSO Regression with PCA	# MSE = 0.024 # RMSE = 0.15	# R-squared = 0.33
Ridge Regression without PCA	# MSE = 0.02 # RMSE: 0.14	# R-squared = 0.47
Ridge Regression with PCA	# MSE = 0.02 # RMSE = 0.16	# R-squared = 0.32

4.2.3 Evaluation of Model Performance

Based on the evaluation metrics, it can be seen that XGBoost without PCA and Random Forest without PCA perform similarly well with an MSE of 0.017 and an R-squared value of 0.52. LASSO Regression without PCA and Ridge Regression without PCA also perform relatively well with an MSE of 0.02 and an R-squared value of 0.47. SVM without PCA performs the worst with an MSE of 3.28 and an R-squared value of 0.24.

XGBoost and Random Forest models are ensemble methods and are good at handling complex relationships and interactions among features. They can also handle missing data and outliers. However, these models can be prone to overfitting if the hyperparameters are not tuned properly. LASSO and Ridge Regression models are good at handling multicollinearity among features and can perform feature selection by shrinking the coefficients of less important features to zero. SVM models can handle non-linear relationships among features, but they can be sensitive to the choice of kernel function and can be computationally expensive for large datasets.

PCA can be used to reduce the dimensionality of the feature space by combining correlated features into new features called principal components. This can help in reducing overfitting and increasing the model's generalizability. However, in some cases, PCA may not improve the model's performance, as it may discard some useful information by combining features. This can be seen in the case of Random Forest with PCA and LASSO Regression with PCA, where the models perform worse than their counterparts without PCA.

In conclusion, based on the evaluation metrics and considering the strengths and weaknesses of each model, XGBoost without PCA and Random Forest without PCA can be considered the best models for predicting song popularity using musical characteristics. However, further tuning of hyperparameters and feature selection methods can be used to improve the performance of these models even further.

The LASSO Regression models have similar results for MSE and RMSE regardless of applying PCA. This could be due to LASSO regression models being able to do feature selection to identify relevant features, hence the reduction of features using PCA does not have any significance. The model without PCA is a better fit than the one with PCA - scoring 0.47 and 0.33 respectively - might be due to the penalisation of less important features, causing the model to be more robust to overfitting. Thus, it does better for unseen data.

The same applies to the Ridge Regression models which have similar results to LASSO Regression models.

Overall, one interesting finding was that with PCA, our models had seemed to perform worse than without. Our group had hypothesized that this could be due to the VIF for the factors that are collinearly related being low, hence dimension reduction not faring as well.

Chapter 5

Results and Discussions

5.1 Conclusion

In this report, we used different statistical learning models to predict a song's popularity using musical characteristics of the song. We collected the data from Kaggle where the dataset was originally extracted using Spotify's API and trained our models to predict the popularity of the song that was calculated by the number and recency of the streams. Comparing the performance of the different models on our test set, it can be seen that XGBoost without PCA performs the best with MSE of 0.017 and R-squared value of 0.52.

5.2 Limitations

Although our dataset is limited to English songs on one streaming platform, namely Spotify, we believe that the model proposed can be extended for use with other streaming platforms as well as songs in other languages as well. Expanding the datasets available would help the models be able to better predict the popularity of songs.

In addition, if there is access to the demographics of the users on the streaming platforms, this might be able to improve the models' performance in the prediction of song's popularity on certain platforms.

5.3 Future Work

Moreover, we can use more advanced techniques such as deep learning and ensemble models to improve the model's prediction accuracy. Additionally, it would be interesting to investigate how the popularity of a song changes over time and include time-series analysis in our models. Finally, it would be valuable to perform an interpretability analysis to understand which features contribute the most to a song's popularity and how they interact with each other.

Chapter 6

References

Tian, H., Cai, H., Wen, J., Li, S., & Li, Y. (2019). A Music Recommendation System Based on logistic regression and eXtreme Gradient Boosting. 2019 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2019.8852094

Pareek, P., Shankar, P., Pathak, M. P., & Sakariya, M. N. (2022). Predicting music popularity using machine learning algorithm and music metrics available in spotify. J. Dev. Econ. Manag. Res. Stud. JDMS, 9, 10-19. Web API reference: Spotify for developers. (n.d.). Retrieved March 26, 2023, from https://developer.spotify.com/documentation/web-api/reference/#/

Chapter 7

Appendix

7.1 Tuning SVM Hyperparameters: Without PCA

7.2 Tuning SVM Hyperparameters: With PCA

7.3 Tuning XGBoost Hyperparameters: Without PCA

7.4 Tuning XGBoost Hyperparameters: With PCA

7.5 Tuning Random Forest Hyperparameters: Without PCA

7.6 Tuning Random Forest Hyperparameters: With PCA