COMP309

Assignment 3

Tingwei Dong

300525329

Exploring and understanding the Data

Important patterns 1

We have four large correlations:
energy - loudness: 0.743329
liveness - speechiness: 0.638230
acousticness - loudness: -0.612320
acousticness - energy: -0.601454

We need to consider the effect of multicollinearity, which can help in feature selection. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable.

Important patterns 2

Although the training.info() output shows no missing values in the dataset, I found out the missing values have different names instead of empty cells or NaN values.

- For the variable artist_name: the missing values are 'empty_field'.
- For the variable tempo: the missing values are '?'. That's why it is treated as an object in the dataset.
- For the variable duration_ms: Its minimum value is -1, and I'm assuming this is encoded as missing values because the length of a song can't be negative.
- For the variable time_signature: there is a value '0/4' in the test set, and it's not a reasonable value, so I'm assuming these are missing values. It also looks like the values have been encoded as date, but it's actually 4/4, 3/4, 1/4, and 5/5 for the time signature.

When I'm doing data pre-processing, I will take these into consideration.

Important patterns 3

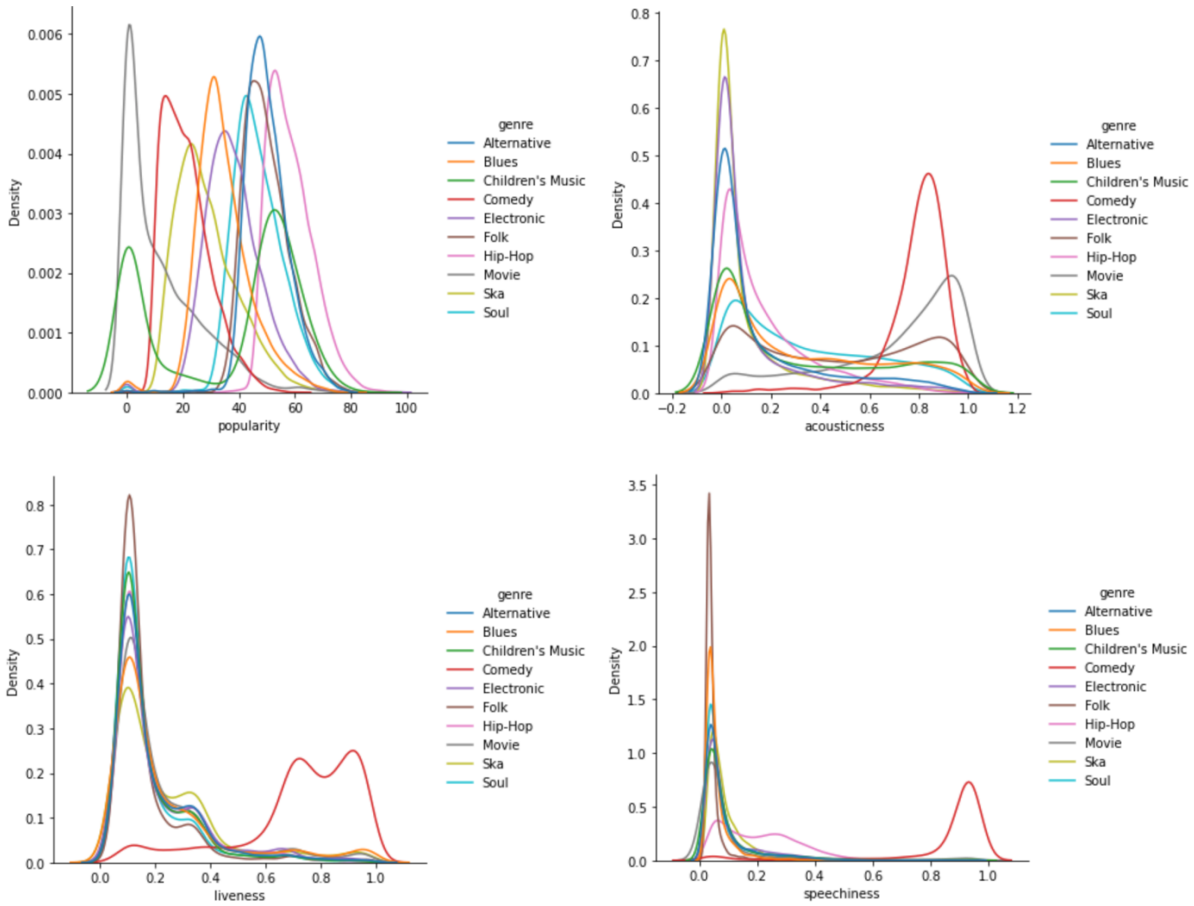
All the numeric variables have the range from 0 to 1, excluding popularity, loudness and tempo. For data pre-processing later on, I will take these into consideration when I'm dealing with missing values.

Important patterns 4

Most songs are not instrumentals or speechiness, so if I'm using these as my predictors, they won't contribute that much compared to other variables. However, we can tell by looking at the training.describe() table, and there is a big difference between values until the 75% quartile. We can also find out from the pair plot that the distribution looks almost like a straight line for these two variables.

Visualisation

After drawing the pair plot, I noticed some interesting trends. So let's take a look at these plots individually.



- Each genre has different mean values for the ‘popularity’ variable.
- ‘Comedy’ and ‘Movie’ variables have a different mean of acousticness compared to other genres, and their acoutsticness values tend to be larger.
- Comedy has a vastly different distribution of liveness and speechiness compared to other genres, and the means values are close to 1, while the rest are more toward 0.

Developing and testing your machine learning system

Pre-processing steps

- Drop ‘instance_id’, ‘track_name’ and ‘track_id’ variables since each instance has unique values, which are shown from the Kaggle Data Explorer page. However, they won’t help predict the genre of a given song.
- Replace missing values in 'duration_ms' and 'tempo' to np.NaN.
- Change the ‘tempo’ variable type to float.
- Impute the missing values with mean.
- Normalised the numeric values.

Categorical variables

- Replace missing values in 'artist_name' and 'time_signature' to np.NaN.
- Fill the missing values with a new category “Missing”.
- One hot encoding for 'artist_name', 'time_signature' and 'mode' variables.
- Ordinal encoding for ‘key’ variable.

- Note that I didn't do anything about the variables with high correlations because their correlation values are below 0.85, and we have limited features.

Training/testing/cross-validation set

The training and test data are from the Kaggle page. To prevent overfitting the test set, I'm using the 5-fold cross-validation to help me get an unbiased estimate of the model when comparing or selecting between final models.

Initial Model (Default Parameters)

As this is a classification problem, I have chosen 5 models that I've learned before from the sklearn package. They are Naive Bayes, Logistic Regression, Decision Tree, K Nearest Neighbour, and Random Forest.

For my first try, I used the default parameters for all my models to see the accuracies before feature selection and hyperparameters tuning. I have also set the random state as 0 to get the same results every time. Here are the accuracies for each model:
- Naive Bayes (66.4%)
- Logistic Regression (63.8%)
- Decision Tree (53.2%)
- K Nearest Neighbour (59.0%)
- Random Forest (65.8%)

I was going to use Support Vector Machines and Gradient Boosting, but they took forever to run, and I couldn't get the results, even though I've tried to set the maximum iterations, etc.

Second Model (Feature Selection)

For the second try, I've used PCA with the standardised data and SelectKBest(), which uses chi-square scores to select the best predictors for my model.

PCA	
Naive Bayes	49.8%
Logistic Regression	53.8%
Decision Tree	49.3%
K Nearest Neighbour	60.9%
Random Forest	61.9%

SelectKBest()	
Naive Bayes	43.6%
Logistic Regression	50.3%
Decision Tree	44.5%
K Nearest Neighbour	49.7%
Random Forest	54.4%

From the above accuracy tables, we can see that the prediction accuracies got worse with feature selection, so this won't be my final model.

Third Model (Hyperparameter Tuning)

I've used grid search and randomised search to tune my models.

Naive Bayes	66.5%
Logistic Regression	78.6%
Decision Tree	-
K Nearest Neighbour	-
Random Forest	50.8%

While choosing reasonable parameters for tuning, I've used this as a reference.
https://www.kaggle.com/code/kenjee/titanic-project-example/notebook

Although I wrote code for tuning all the models, some of them took way too long, and I couldn't get any results.

Final model

Leader Board	
Naive Bayes	68.0%
Logistic Regression	79.6%

My final model is logistic regression, which has the highest accuracy in cross-validation and leader board test sets. I will definitely try more models for future approaches and leave more time to tune the parameters.

Reflection

Personally, I would say my model is a bit difficult to interpret, as it starts with a simple logistic regression model. Still, after tuning the hyperparameters for better accuracy, I've chosen these best estimators, which makes the model more complicated.

The biggest problem will be whether my audience or even myself actually knows how my model makes predictions, especially for people without machine learning knowledge. However, my model does give a pretty decent accuracy overall.