Artificial Intelligence Capstone Project1

Music Genre Classification

110550101-陳威達

Report Outline

- Introduction
  - About MGC problem
  - About this project

- Data acquisition
  - Dataset

- Feature Extract and Preprocessing
  - VGGish
  - Preprocessing

- Model
  - Model selection and parameter setting
  - Evaluating method

- Result
- Reference

Introduction

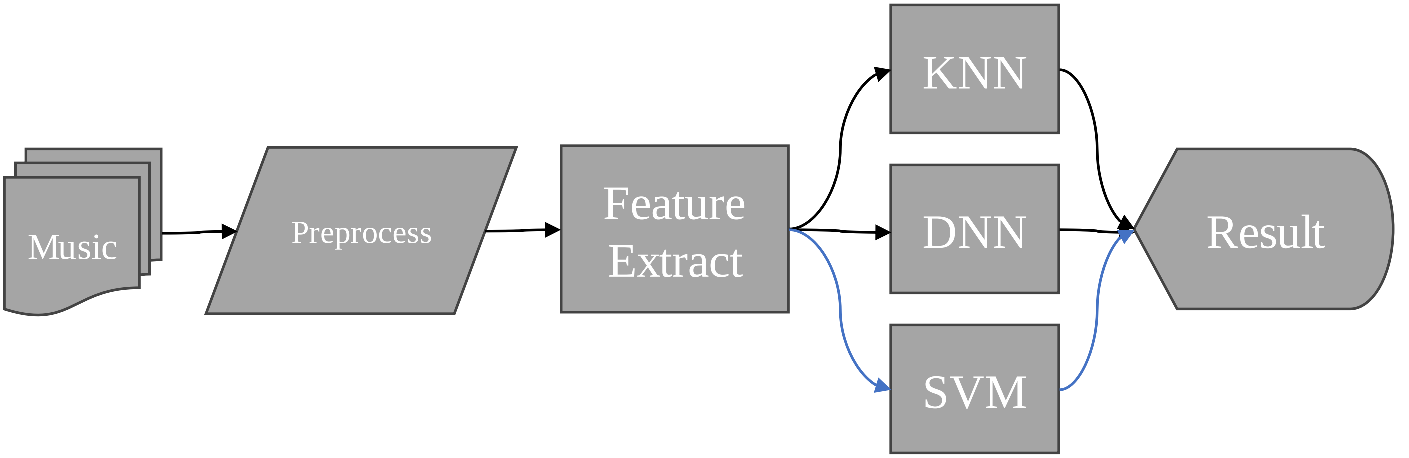
- About MGC

The MGC (Music Genre Classification) problem revolves around the automated categorization of musical compositions using computational methods, a critical task within the realm of music information retrieval. The process involves developing algorithms and techniques dedicated to organizing, searching, and retrieving music data based on distinct musical and cultural characteristics. Music genres serve as pivotal categories utilized by music streaming services, radio stations, festivals, and record labels to group similar music and offer personalized recommendations to users.

The difficult part in MGC problem is that music category that human defines is subjective, which makes the kind of problem inherently challenging. Human perceptions of music genres often carry subjective nuances, influenced by cultural, emotional, and contextual factors. These subjective interpretations introduce complexity as there may be considerable variation in how different individuals categorize and perceive music genres.

- About this project

In this work, I get the inspired by the paper[1] and try to utilize the open source music on YouTube to construct a model to identify the 6 different genres of music. The abstract of model is like:



Data Acquisition

• Dataset

The dataset acquisition method is referred from GTZAN Dataset[2]

The music source is from YouTube studio music library, which is free and generally be allowed to be utilized. Moreover, the library has labeled music genre so I use these labels as the target and download each type of music in this work.

I select six types of music; they are <u>classic, electronic dance music, hip hop , jazz and blue, pop, rock</u>. In each type, I download about 45~50 files and every music file is more than 30 seconds

```
# Get the # of the file in each folder
for music_genre in os.listdir("input"):
    print(music_genre, "contains", len(os.listdir(f"input/{music_genre}")), "files")
[90]  ✓ 0.0s

classic contains 45 files
edm contains 47 files
hip hop contains 50 files
jazz and blue contains 50 files
pop contains 48 files
rock contains 49 files
```

The amount of music files in each genre

The appearance of the audio files in the folder *The interface in YouTube Studio music library*

Feature Extract and Preprocessing

• VGGish [3]

VGGish is an audio feature extraction model developed by Google that is specifically designed for the task of audio classification, including tasks like environmental sound recognition and music genre classification. It is pre-trained on a large dataset of audio samples and extracts fixed-size embeddings or feature vectors from input audio clips.

In this project, the output of each file is (1,128), which represents the feature of the specific file.

Data preprocessing and feature extraction

```
# Feature extraction
vggish = hub.load('https://tfhub.dev/google/vggish/1')

def vggish_extract(audiofile):
    y, sr = librosa.load(audiofile, sr = 44100)
    window = 20000
    stride = 5000
    total_time = librosa.get_duration(y = y, sr = sr)
    start = 0
    end = total_time * 1000
    return_list = []
    for i in range(start, int(end), stride):
        if i + window > end:
            break
        y_temp = y[i:i+window]
        feature = vggish(y_temp).numpy()

        if feature.shape[0] == 0:
            continue
        return_list.append(feature)
    return return_list
```

[3] ✓ 0.6s

The code for data preprocessing and feature extraction

• Preprocessing

According to the previous study[4], regardless of a smaller number but longer period as the input, the larger quantity of splits with shorter period of audio duration period can extract more accurate features. Hence, I select a 20s window with a 5s stride as range and review each audio file to extract feature. Since each genre has different numbers of files and each file has different period, the numbers of output in each class of music are different.

Model

• Model selection and parameter setting

The requirement in this project is two supervised machine learning model and one unsupervised learning model. Due to the well work of feature selection by VGGish, I don't select complex or pretrained model. In this project, I use a SVM, a fully connected neural network(DNN) and KNN as the algorithm to train my model

In the part of hyper parameters in SVM and KNN, I consider the setting of the study of [4]. As for the DNN, I use relatively shallower layers to avoid weight vanishing

```
models = {
    'knn': KNeighborsClassifier(n_neighbors = 1, algorithm= 'brute'),
    'svm': SVC(kernel= 'poly', degree= 6,tol= 0.001, coef0= 0.1 ,gamma= 'scale')
}
```
✓ 0.0s

```
NN = Sequential()
NN.add(Dense(128, input_dim=x.shape[1], activation='relu'))
NN.add(Dense(64, activation='relu'))
NN.add(Dense(class_num, activation='softmax'))
NN.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```
✓ 0.0s

```
models["DNN"] = NN
print(NN.summary())
```
✓ 0.0s

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 128)               16512

 dense_4 (Dense)             (None, 64)                8256

 dense_5 (Dense)             (None, 6)                 390

=================================================================
Total params: 25,158
Trainable params: 25,158
Non-trainable params: 0
_________________________________________________________________
```

The code for model setting

Model

• Evaluating method

Due to relatively small dataset , I select 10-fold cross validation method to measure the score of accuracy, f1, precision, recall and mcc to get the aggregate performance of each algorithm.

```python
# model evaluation
def evaluate_model(predictions, y_test):

    accuracy = accuracy_score(y_test, predictions)
    f1 = f1_score(y_test, predictions, average='weighted')
    precision = precision_score(y_test, predictions, zero_division=1, average='weighted')
    recall = recall_score(y_test, predictions, average='weighted')
    mcc = matthews_corrcoef(y_test, predictions)
    # auroc = roc_auc_score(y_test, predictions, multi_class= 'ovo')


    return {'accuracy': accuracy, 'f1': f1, 'precision': precision, 'recall': recall, 'mcc': mcc}
```

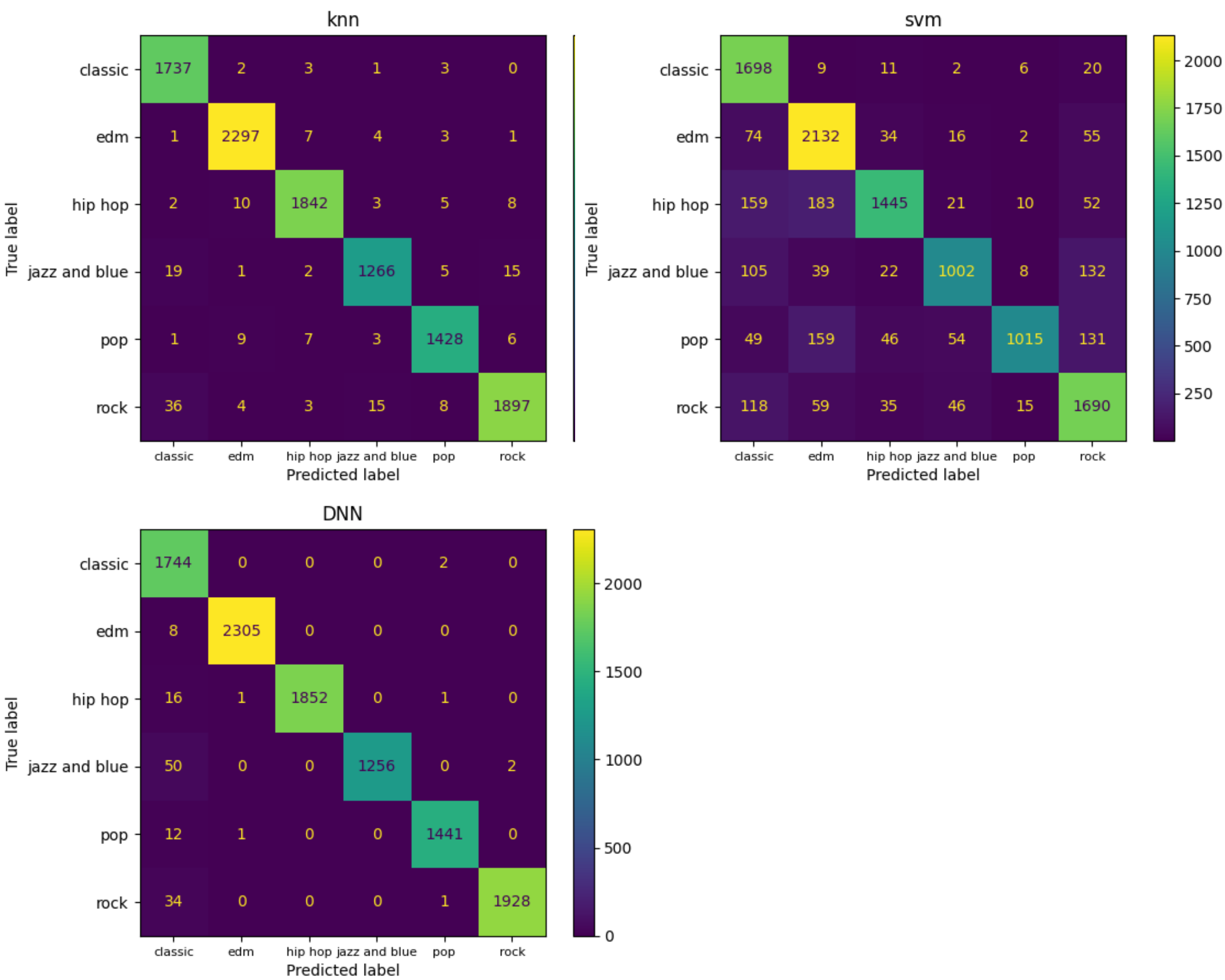
```python
kf = KFold(n_splits= 10, shuffle=True, random_state=42)
result = []
for kf_idx, (train_idx, val_idx) in enumerate(kf.split(x)):
    x_train, x_val = x.iloc[train_idx], x.iloc[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]
    for model_name, model in models.items():

        if model_name == 'DNN':
            model.fit(x_train, y_train, epochs=30, batch_size=32)
            predictions = np.argmax(model.predict(x_val), axis=1)
        else:
            model.fit(x_train, y_train)
            predictions = model.predict(x_val)
        fold_res = evaluate_model(predictions, y_val)
        fold_res['model'] = model_name
        fold_res['fold'] = kf_idx
        result.append(fold_res)
```

The code for estimating model performance

Result

In the result presentation, I additionally calculate the mean of previous 10-fold performance and draft the confusion matrix.




```
print("Cross-validation results: ")
print(all_result_df[-3:])
all_result_df.to_csv(output_folder + '/result.csv', index = False)
```

✓ 0.0s

```
Cross-validation results:
            model  accuracy        f1  precision    recall       mcc
30    DNN Average  0.940976  0.940789   0.942469  0.940976  0.928937
31    knn Average  0.861929  0.860928   0.861433  0.861929  0.833093
32    svm Average  0.760467  0.755032   0.766977  0.760467  0.711998
```

The result figures and statics

Reference

- A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computer Science [1]

- GTZAN Dataset - Music Genre Classification[2]

- Vggish[3]

- Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches [4]