Save & Load a trained ML model Research

Currently the ML models take in a sizable test file and predict genres after looking at the test file. We want to be able to input a single song and its statistics into the ML model and receive a genre. For this artifact, we will research and find out how to have an ML model that takes in a single song.

Current Functionality

Currently, our machine learning takes in an input csv of around 70 rows, each consisting of individual songs and their characteristics. The current functionality is using 2-fold cross validation on our test data. In other words, we split our data in half and train our model with the first half. We then test on the second half. Next, separately, we train the model with the second half of data, and test with the first. Then, we concatenate the test results to get predictions for every song in our test data.

When we run our file, we train and test with both halves of data. By doing it this way, we have less accurate predictions by only training with around 35 songs. We are also re-training the model every time we run it, which is wasting time at the end of the day. By saving our model and utilizing model persistence, we can ideally train with all of our songs, and test with a single song to produce a movie/genre suggestion for individual songs.

https://learn.microsoft.com/en-us/azure/databricks/mlflow/scikit-learn-model-deployment-on-azure-ml
https://learn.microsoft.com/en-us/azure/databricks/mlflow/models
https://www.mlflow.org/docs/latest/models.html#scikit-learn-sklearn
https://www.mlflow.org/docs/latest/python_api/mlflow.sklearn.html#module-mlflow.sklearn

Deployment

After we choose which ML model we want to save and reload from we should be able to deploy it using an API to AzureML. The azure documentation mentions a MLFlow model that will allow us to save a scikit-learn model that has been trained on our data to the Azure server. Once the model is in Azure ML we will be able to query the model and use the information from the query on our web app. The steps are to use the MLFlow to save our scikit-learn model, and then using the MLFlow API store the model on Azure to then query the model. MLFlow will also allow us to make updates and retrain our model if needed.

https://scikit-learn.org/stable/model_persistence.html
https://stackoverflow.com/questions/56107259/how-to-save-a-trained-model-by-scikit-learn
https://stackoverflow.com/questions/10592605/save-classifier-to-disk-in-scikit-learn
https://www.analyticsvidhya.com/blog/2021/08/quick-hacks-to-save-machine-learning-model-using-pickle-and-joblib/ - step by step article on how to save and load model

Model Persistence

By using either ‘pickle’ or ‘joblib’, we should be able to save our trained model, and then open it when we want to use it for predictions. If we have our model trained, saved, and deployed, then we won’t have to train it again unless we make a change to our training data. Then ideally, instead of using scikit’s ‘train_test_split’ to test & train on half of our dataset, we can train our model fully on the test dataset, and predict on a separate file that only contains one song and its characteristics.

https://stackoverflow.com/questions/67573767/can-training-dataset-and-testing-data-set-be-seperate-instead-of-split

Implementation

As stated above, the current functionality is using 2-fold cross validation on our test data. Ideally, we want to train the model on all of our test songs, and produce one genre from the user’s average listening statistics. Upon doing some googling and experimenting, it is in fact very possible to train with a large set of data, then test with only one input. By doing this, we would hopefully have a better-trained model, and produce one genre of movie to recommend based on the user’s average listening stats.

To achieve this, I will most likely want to create another test csv file containing one line of data. For example, I could extract and use my average spotify statistics, since we will eventually want to use a

user's average listening data. I would also want to refactor my machine learning code to remove any instances of 2-fold cross validation. I would simply fit the model to the 70-line test data, and then predict the genre using the one-line data as input.

Because we would be using the machine learning model for a recommendation, rather than predicting a right or wrong answer, having accuracy scores and a confusion matrix will no longer be useful. (Unless, of course, we assign a correct answer using our mapping document, similar to what we did in our test data, but this is not what we're aiming to do.) So, refactoring will be necessary to remove these calculations.

More refactoring will need to be done once we decide what song characteristics we are for-sure plugging into our model. Right now, I have 3 iterations running, each using different sets of characteristics. Since Spotify will return average characteristics for all of the user's listening data, these will most likely be our 8 inputs: tempo, acousticness, energy, loudness, danceability, happiness, speechiness, and liveness. Thus, once we finally decide what inputs to use, I can remove the unhelpful iterations so that we have one, working run that will produce our recommendation.