Machine Learning Engineer Nanodegree

Capstone Project :
Music Genre Classifier

Rahul Mishra

January 7th, 2018

I. Definition

Project Overview

Music has some really powerful effect on our emotions, and Human ear is quite incredible at predicting the genre accurately, although music genre is a relatively complex concept that sometimes the music industry itself feels confused in assigning genre to some of the songs. Classification of genre by Machine is one of such technical approach which has a major drawback of predicting genre accurately. Previously many attempts has been made to design system to categorize music but the prediction result was not overwhelming. So, the problem continued that can machine predict genres of songs with better accuracy results?

The Project discusses various Machine Learning concepts of classification and improving the accuracy of those classification techniques to be used for Genre based Music Categorization is the goal of our Project. In this project Million Song Dataset, made available by LabROSA, containing 30 summary features for each music files one of 10 genre, being used as training and testing data.

The project focuses on implication of Classification algorithm like, Random Forest, Gaussian Naive Bayes, Support Vector Machine, Decision tree, K-Nearest Neighbor.

Lyrical modelling concept has been the second approach improvising Bag of Words technique over the dataset.

We aimed to study and apply various machine learning concepts to solve a real world problem like Music Categorization based on Genre and come out with an improved result of 62% F1-score while using Multiclass classification technique and 47% F1-score accuracy while applying Lyrical Modelling concept. For the fact that many songs are unlabeled and much more songs are being released daily, the classification technique predicting genre with an accuracy of 62% F1-score is quite of help.

Problem Statement

The problem we are trying to address here is, Can machine Learn Genre of a song by analysing the various feature of songs. Can we match and/or break the break benchmark set by the benchmark model.

Based on the research paper

From Classical to Hip-hop: Can Machine Learn Genre? A student publication by Aaron kravitz,Eliza Iupone, Ryan Diaz

We will start with the simplest classification algorithms evaluating their performances and comparing and approach the next better algorithm to beat the benchmark score.

Apart from this, I have added an experimental approach of guessing genre of song with its lyrics.

Metrics

F1-Score is the evaluation metric used by us to measure the performance of our models.F1 score conveys the balance between the precision and the recall.If we were looking to select a model based on a balance between precision and recall.

It is multiclass classification problem and accuracy won't be a great metric as the distribution of labels may or may not be uniform.

II. Analysis

Data Exploration

For this project the Million Song Genre Dataset has been used, that is been made available by LabROSA at Columbia university , for educational study purpose, Although we will be working with the subset that will be preprocessed by us.This dataset consists of about million songs of various genres, to be precise of 10 genres, namely classic pop and rock , classical , dance and electronica, folk, pop, hip-hop, jazz and blues, punk, metal & lastly soul and reggae .The distribution of songs into various genres can be recognized in Dataset Composition. This dataset is a well-organized classification dataset as it consist the features of each of the song in the dataset too. The Million Song Dataset has about million songs with related metadata and audio analysis features, In this dataset, we have each song as a training example containing 34 summary features along with the genre of the track , which would be of use during training as well as testing the classifiers for predicting the genre of particular combination of features.
The dataset contains various features that needs little description.

genre : is the music category of the particular track in the dataset.
track_id : is the musicmatch track id of the track.
artist_name : is the song's artist name.
title : title of the song.
loudness : is the property of a sound that is primarily a psychological correlation of physical strength (amplitude).
tempo : The tempo is the speed of the underlying beat for a piece of music. Tempo is measured in BPM, or Beats/Minute. The top number represents how many beats there are in a measure, and the bottom number represents the note value which makes one beat.
key : The key of a piece is a group of pitches, or scale upon which a music composition is created.
mode : Refers to a type of scale, coupled with a set of characteristic melodic behaviors.
duration : song duration (in Secs).
avg_timber1 - 12 : Timbre is then a general term for the distinguishable characteristics of a tone.
var_timber1 – 12 : Timbre is mainly determined by the harmonic content of a sound and the dynamic characteristics.

Let's look how our Genre Song Dataset is formatted.
Here's how the data set looks inside dataset/msd_genre_dataset.txt

First 10 lines consist of metadata related to dataset and features available in the dataset.
After that each row in the file represents a data point i.e. song and its features. All the features and labels for a song are provided in a single line separated by a comma.

Ordering in as
%genre,track_id,artist_name,title,loudness,tempo,time_signature,key,mode,duration,avg_timbre1,avg_timbre2,avg_timbre3,avg_timbre4,avg_timbre5,avg_timbre6,avg_timbre7,avg_timbre8,avg_timbre9,avg_timbre10,avg_timbre11,avg_timbre12,var_timbre1,var_timbre2,var_timbre3,var_timbre4,var_timbre5,var_timbre6,var_timbre7,var_timbre8,var_timbre9,var_timbre10,var_timbre11,var_timbre12
Where each value is a feature and first element is our target label i.e. feature.

```
# MILLION SONG GENRE DATASET
# created by T. Bertin-Mahieux (2011) Columbia University
# tb2332@columbia.edu
# http://labrosa.ee.columbia.edu
# GOAL: easy to use genre-like dataset extracted from the MSD
#       Should not be used as a proper MIR task! Educational purposes only
# FORMAT: # - denotes a comment
          % - one line after comments, column names
            - rest is data, comma-separated, one line per song
%genre,track_id,artist_name,title,loudness,tempo,time_signature,key,mode,duration,avg_timbre1,avg_timbre2,avg_timbre3,avg_timbre4,avg_timbre5,avg_timbre6,avg_timbre7,avg_timbre8,avg_timbre9,avg_timbre10,avg_timbre11,avg_timbre12,var_timbre1,var_timbre2,var_timbre3,var_timbre4,var_timbre5,var_timbre6,var_timbre7,var_timbre8,var_timbre9,var_timbre10,var_timbre11,var_timbre12
classic pop and rock,TRFCOOU128F427AEC0,Blue Oyster Cult,Mes Dames Sarat,-8.697,155.007,1,9,1,246.33424,46.6730673684,14.6136842105,14.6642147368,0.176561052632,-9.34637684211,-12.3416989474,11.1833821053,7.40528842105,9.31376526316,3.20116947368,-0.152733684211,5.80970947368,14.9308204523,802.205948403,1255.51456863,580
.030471741,598.485222763,575.337671354,322.068602573,321.726029279,232.700608628,186.805302819,181.938688381,151.50801133
classic pop and rock,TRNJTPB128F427AE9F,Blue Oyster Cult,Screams,-10.659,148.462,1,4,0,189.80526,43.645377305,-87.3371503546,41.0515815603,7.81477021277,-12.989848227,-14.2535985816,6.12604539007,-2.44866241135,22.6917134752,-2.87270638298,1.42772482
27,-6.71073049645,22.7048426349,1561.30707153,2007.65306963,1043.47407325,585.694981147,564.013735701,510.17702214,400.200186
221,365.119588173,238.099707901,197.93375698,251.577525425
```

Apart from this, we will be using **Lyrical Modelling concept** , i.e. to classify genre from the lyrics of the song. We will be collecting available lyrics from the internet and implementing bag of words approach on them.

For this we will be scraping lyrics from **Lyrics Wikia**

Initially, the dataset we got was very unbalanced composition wise means to say one kind of labels were much more than the other kind. We corrected that by choosing 2000 samples from each available category and began working with them.

This is essentially Random Under-Sampling. Random Undersampling aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out.

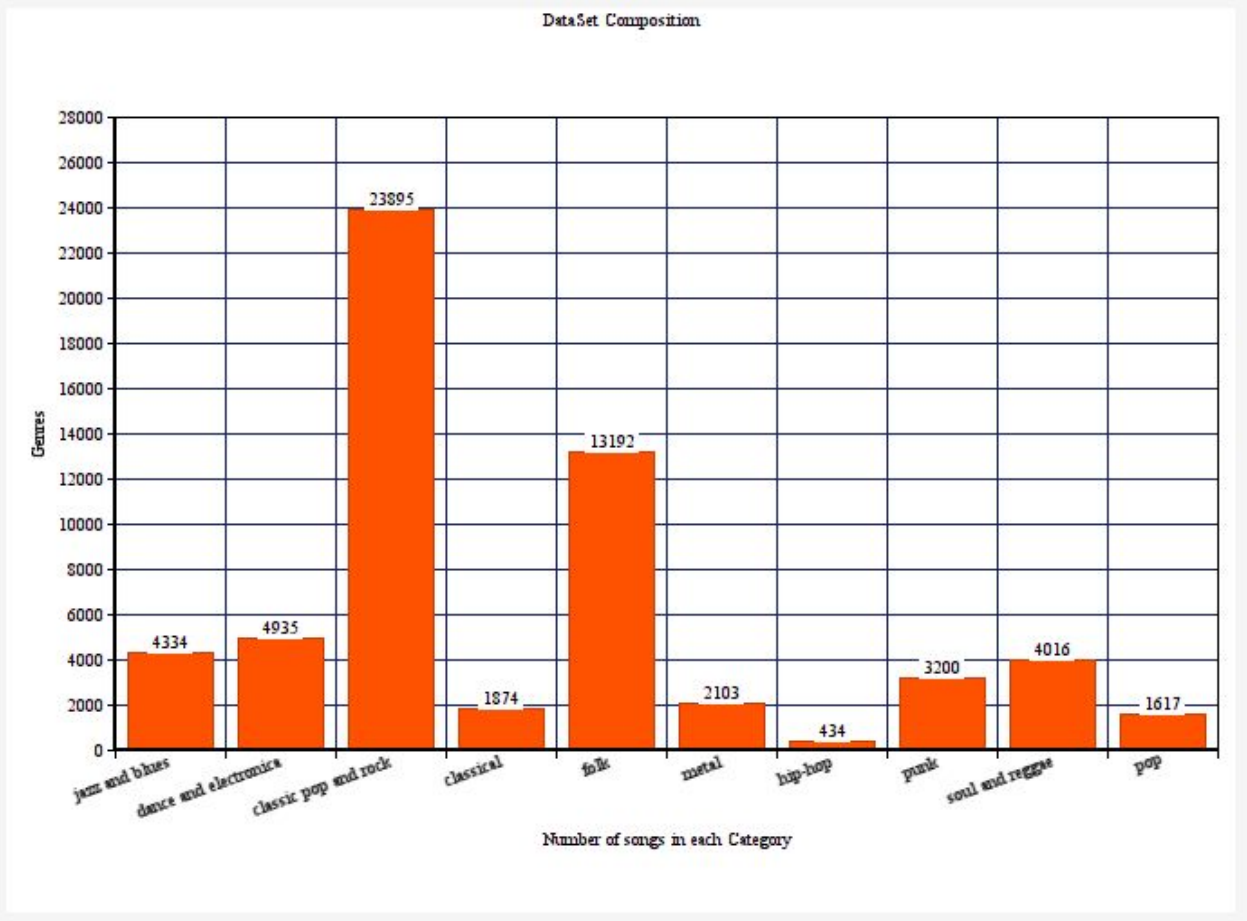
Then , after that we did some preprocessing over that data.

- Splitting into feature and labels.
- Feature Scaling on the data.
- Hot encoding the labels as this is a multiclass classification problem.

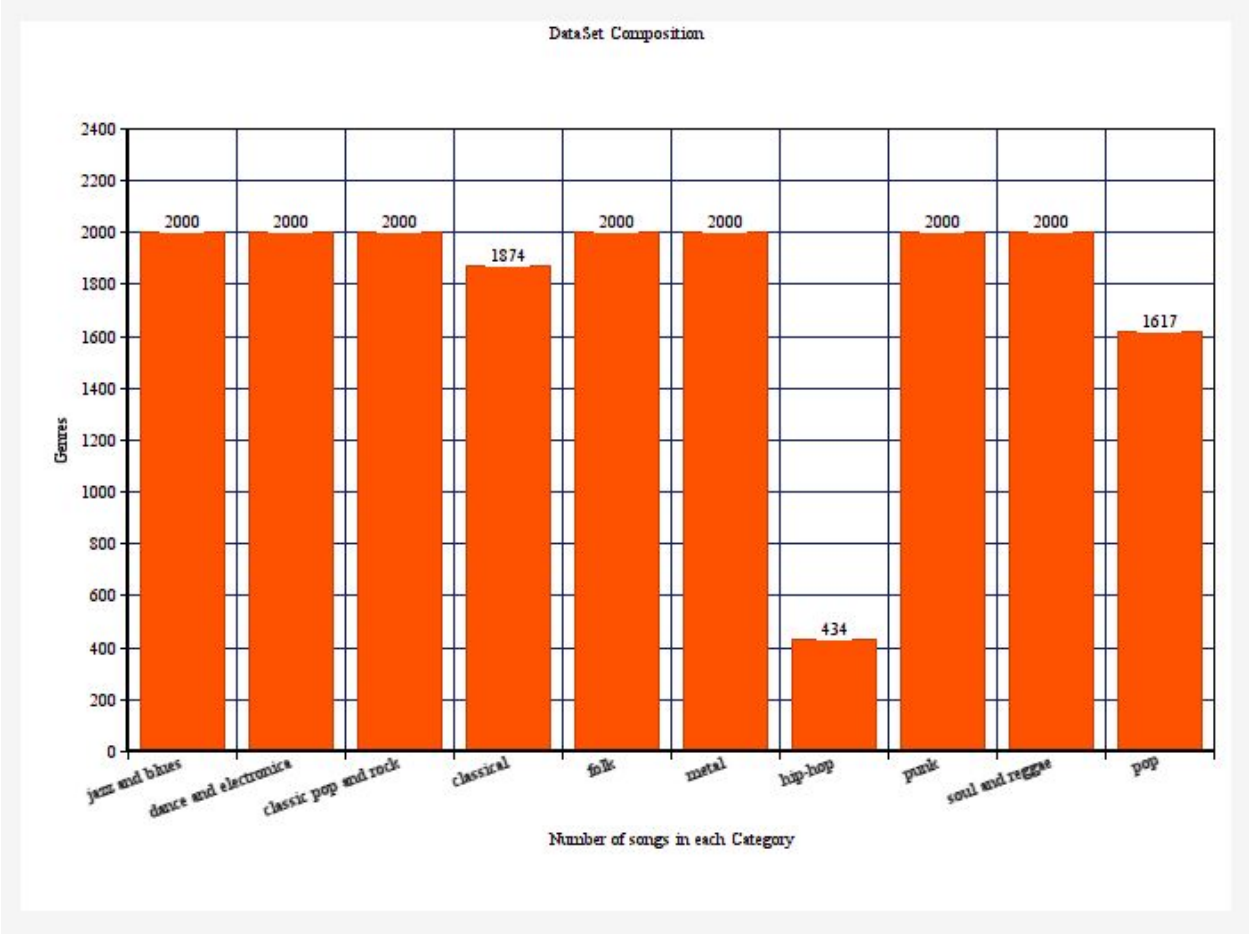
Then finally, we split the data into training and testing set and start working with it.

Exploratory Visualization

This was the actual composition of the million song database, as we can see around 40% of songs belong to classic pop and rock. This dataset would have favoured classic pop and rock much more than the other genre songs available in the dataset.



After our recomposition of the dataset and preprocessing discussed above. Our dataset looks like this.



To compensate for lesser number of hip-hop data points we used SMOTE

(Synthetic Minority Over-sampling Technique).

We imported **from imblearn.over_sampling import SMOTE** to do oversampling on our training data. So, that we can compensate the imbalance for Hip-Hop.

Algorithms and Techniques

Following are the Algorithms and Techniques were used during the process of project completion.

- We started with Naive Bayes Algorithm ,as it is the easiest to understand and implement. F1-Score achieved was set as the lower bound for F1-Score of 38%. Unlike some classifiers, multi-class labeling is trivial with Naive Bayes.

  For each test example , you want to find:
  Argmax P(i) (class(i)|data(i))
  Argmax P(i) (class(i)|data(i))
  In other words, you compute the probability of each class label in the usual way, then pick the class with the largest probability.

- Then After that we moved to SVM,as it can fit more complex dataset. It is suitable for regression and classification tasks.The benefit is that you can capture much more complex relationships between your data points without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive. we in our model build a two-class classifier over a feature vector $\Phi(\vec{x}, y)$ derived from the pair consisting of the input features and the class of the datum. At test time, the classifier chooses the class $y = \arg\max_{y'} \vec{w}^T \Phi(\vec{x}, y')$. The margin during training is the gap between this value for the correct class and for the nearest other class, and so the quadratic program formulation will require that $\forall i \; \forall y \neq y_i \; \vec{w}^T \Phi(\vec{x}_i, y_i) - \vec{w}^T \Phi(\vec{x}_i, y) \geq 1 - \xi_i$. This general method can be extended to give a multiclass formulation of various kinds of linear classifiers like we have used in our project. It is also a simple instance of a generalization of classification where the classes are not just a set of independent, categorical labels, but may be arbitrary structured objects with relationships defined between them.

- After Support Vector machine, we thought of try something different. kNN is slow when you have a lot of observations, since it does not generalize over data in advance, it scans historical database each time a prediction is needed.

- We finally came to Random Forest Algorithm, that has chosen as our benchmark model based on the research paper we are working with. The results in paper show that when value of n_estimators is set to/around 300 it gives the best F1-Score of 60%. Our random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement by default.The default values for the parameters controlling the size of the trees (e.g. max_depth, min_samples_leaf, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values.

  The features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data, max_features=n_features and bootstrap=False, if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behaviour during fitting, random_state has to be fixed.

- Try ensemble methods to further optimize this benchmark model with RandomForestClassifier and to tune hyperparameters using GridSearchCV.

- we thought of using this with OnevsRestClassifier. Also known as one-vs-all, this strategy consists in fitting one classifier per class. And this is how our model being trained,For each classifier, the class is fitted against all the other classes.In addition to its computational efficiency (only n_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly

used strategy for multiclass classification and is a fair default choice.

- In a different Approach we used Lyrical analysis with decision trees to predict genre of song using bag of words.

Benchmark

The benchmark model used and as suggest by the research paper under study is a random forest algorithm that gives a F1-score of 58%.

The parameters to achieve this is also mentioned in the paper.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

Implementation

Following preprocessing was done before using the dataset in both approaches

- Feature scaling for feature values.
    - We use MinMaxScaler to feature scale all the feature values,so that no feature dominates on other while training
    - Library Used: **from sklearn.preprocessing import MinMaxScaler**

- Hot Encoding the labels as this was multi class classification problem.
    - Used LabelEncoder to encode the multiple genres available to us, which can later inverse transformed back to actual genres.

    - Library Used: **from sklearn.preprocessing import LabelEncoder**

- Removing Stop Words and punctuation marks for Lyrical Analysis.
    - This was done using the nltk library in python.
    - Library Used:
    - **from nltk.corpus import stopwords**
    - **from nltk.stem import SnowballStemmer**

- After all this processing was done, we began splitting our dataset into training and testing set. But, the point to note is genre Hip-hop has very few data points in our processed dataset. For that we did used Synthetic Minority Over-sampling Technique to oversample the genre data points so that we get balanced number of data points. Following was imported in our code to use this from **imblearn.over_sampling import SMOTE.**

- We Started with Naive bayes classifier,as it is the simplest to implement and understand. We fitted our preprocessed data to it and got a **F1-Score of 38%**, which we used to set a lower bound benchmark for us.Imported following library for this **from sklearn.naive_bayes import GaussianNB.**

- After getting this benchmark, we thought of trying Support vector machines as it can perform better than naive bayes algorithm and better fit our complex multi-class data. After fitting we got a **F1-Score of 43%**. Imported following library for this **from sklearn.svm import SVC.**

- After this we thought change our approach try fitting out our model with k-NN algorithm, as since it does not generalize over data in advance, it scans historical database each time a prediction is needed.But that didn't work out well and we

got a F1-score of 41% which was comparable to SVM. Imported following library for this **from sklearn.neighbors import KNeighborsClassifier.**

- Then we came back to our benchmark model, which was a randomforestClassifier will trained it at the tuned value, it give us a **F1-Score 60%** , which is better than as proposed by the research paper.For this we imported **from sklearn.ensemble import RandomForestClassifier.**

- To further improve this we thought of using OnevsRestClassifier because of its computational efficiency and trained this model over GridSearchCV to get tuned parameters, which we got as **min_samples_leaf=2, min_samples_split=2 and n_estimators=300.** For this we imported **from sklearn.grid_search import GridSearchCV.**

- After getting this parameters we actually fitted our OnevsRestClassifier with RandomforestClassifier as an estimator. We got a better **F1-Score of 61%**.

- After getting these results we thought of cross-validation using StratifiedKFold where each fold contains balanced distribution of Labels, for this we imported **from sklearn.model_selection import StratifiedKFold.**

- We got an average **F1-Score of 52%** on our cross validation , that seems to work out for us we have remove 1/10th of data for fitting and validating over it.

Refinement

Refinement done here began and input level itself. We first corrected the dataset composition. Then we feature scaled everything, encoded the labels.After that we improve the benchmark model by using ensemble methods and then tuning the hyperparameters using GridSearchCV

Following were the parameters that were tuned through GridSearchCV.

min_samples_split = 2

The minimum number of samples required to split an internal node.

min_samples_leaf = 2

The minimum number of samples required to be at a leaf node

n_estimators = 300

The number of trees in the forest.

We chose these parameter as they are related to over and under fitting our data , hence all of these are very critical parameters to tune.

Then after we got our tuned parameter, we fitted our model with that and achieved a higher **F1-Score of 61%** as compared to **58%** as mentioned in research paper.

In our other approach, we processed our lyrics using nltk library and removed components that don't give us information.

IV. Results

Model Evaluation and Validation

The final model performed well and achieve the required benchmark score of 58% and above. It passed with flying color with a F1-Score of 61%.

We reached to this model through benchmark model. It was based on it. We further improved it by OnevsRestClassifier

Along with using ensemble methods for further optimising our model using GridSearchCV, we thought of using this with OnevsRestClassifier. Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.
After running our model through GridSearchCV we get optimisation values for the parameters.

- min_samples_split = 2
- min_samples_leaf = 2
- n_estimators = 300

For validation of our model we used StratifiedKFold to cross validate our model's performance, we got an average F1-score of 52% of on the 10 folds we cross validate. Our model is doing well not overfitting or underfitting.

Justification

The final model used is by far the best model and it got such high F1-score fitting only on the 1% of dataset that is impressive.

V. Conclusion

Free-Form Visualization

As clear from the visualisation given above the Random Forest Classifier from ensemble method performs the best and give us the highest F1-Score of 61% and breaks the suggested F1-score of 58% as mentioned in the research paper.

Reflection

Following conclusions can be drawn from the above observations:

- Highest F1-score of 61% achieved using OnevsRestClassifier on RandomForestClassifier of ensemble methods.

- Second experimental approach of lyrical analysis Model showed decent score with F1-Score of 56% without hypertuning.

The most difficult part was to find hyperparameters to tune our final OnevsRestClassifier with Random Forest using GridSearchCV. And most interesting part was to get a good model from Lyrical Analysis ,it seemed cool to predict genre just from lyrics of song and explore further from their on.

If we were to summarize our entire machine learning pipeline,it started with processing our dataset for proper and balanced composition. Then we did some preprocessing of our inputs by feature scaling, label encoding and then fitted our model and calculated performance. After that we hypertuned parameters and then again fitted model with those parameters and cross validated our scores.

Improvement

There can be a lot of improvements to this project. Few improvements that come to my mind are:

- Running the model over more data, original million song dataset with much more features.

- Applying PCA on features and do dimensionality reduction to reduce computation time.

- Use NLP and CNNs together to understand natural language and produce much better F1-Scores.