COMP90049 Assignment 02

Anonymous

1 Introduction

Music is everywhere in our daily life. A music genre is a textual label created by human to identify and describe various music. A music genre classification (MGC) task aims to correctly identify the genre given an instance of music. The pioneered and innovative work in MGC field is carried out by (Dannenberg et al., 1997) [1], and (George et al., 2002) [2]. (Dannenberg et al., 1997) have shown that effective machine learning classifier can be built for MGC task. (George et al., 2002) have demonstrated that features extracted from music segment such as timbrel, rhythmic and pitch can improve the music genre classifier's performance. Further approaches like (Mckay et al., 2004) [3] built on top of these with extraordinary improvements or exciting new methods. (Mckay et al., 2004) have built music genre classifier based on instrumentation, texture, rhythm, dynamics, pitch statistics, melody and chords features. Recent work such as (Zhang et al., 2016) [4] and (Vishnupriya et al., 2018) [5] have shown that neural network can be used to classifying music genre based on audio features extracted by *MFCC* method without considering feature selection. In this assignment, we are going to apply machine learning techniques on the MGC problem and explore new knowledge in the MGC field.

2 Research question

1. Does using a combination of textual, meta, audio features perform best?
2. Is Gaussian Naive Bayes the most accurate model on the given MGC data-set in terms of accuracy and macro-f1?
3. Is Gaussian Naive Bayes the most general model on the given MGC data-set?

3 Data-set

The dataset we have chosen for the MGC problem is called the Million Song Data-set [6]. As summarized in the table 1, it has three feature categories: *audio*, *metadata* and *textual*. The textual feature is given as a list of comma-separated *tags*. The meta-features have textual feature like *title*, continuous feature like *loudness*, integer-valued feature like *time_signature* and binary feature like *mode*. The audio features are turned into 148 not interpretable float features by the *Temporal Echonest Features* extraction technique presented by (Schindler et al., 2014) [7]. The data-set has 8 classes [1]. We are given both training and valid data-set with labels by hold-out strategy and a test features data-set with no labels for contest submission.

4 Exploratory data analysis

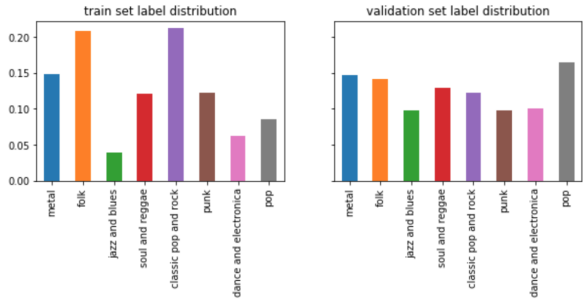


Figure 1: Label distribution

The given training and valid data-set have different and imbalanced label distribution as in figure 1. This will influence the model such as naive bayes and knn. We use **macro evaluation metric and cross-validation on training data-set to have an accurate evaluation to avoid such influence**.

[1] Soul and Reggae, Pop, Punk, Jazz and Blues, Dance and Electronica, Folk, Classic Pop and Rock, Metal

Category	Feature name	Feature type	Used	Transformation	Note
meta	title	String	N		
textual	tags	Comma separated word string	Y	TD-IDF	Standardized if model is magnitude sensitive
meta	loudness	Float continuous	Y		
meta	tempo	Float continuous	Y		
meta	time_signature	Integer	Y	Treated as continous	
meta	key	Integer categorical	Y	One-hot chage to continous	
meta	mode	Integer binary	Y	Treated as continous	
meta	duration	Float continuous	Y		
audio	vect_1 ... vect_148	Float continuous	Y		

Table 1: Feature engineering

5 Feature prepossessing & selection

As textual feature *tags* is a list of comma-separated words, unlike audio features, this cannot be directly applied to continuous feature-value-based machine model. We use term frequency-inverse document frequency (TF-IDF) [8] vectorizer to change it to word-wise feature with continuous values.

Although *title* is also a string-valued feature the same as *tag*, we think this feature has limited information. Because any genre music can have the same title if the author wants. And after TF-IDF transformation, majority of its features equals to 0 as the title has a limited number of words (not indicative for identifying the class). So, we decided not to use it.

The integer-valued feature *time_signature* is given without mentioning whether it is nominal or continuous. As it is an estimated number of beats per bar, we treat it as a continuous feature.

The integer-valued binary feature *mode* is treated as a continuous feature because it is 0-1 based.

The integer valued categorical feature *key* is transformed into continuous features by using the one-hot encoding [9].

The rest continuous features and theses features will be used in the later stage. **Using all three feature categories gives best macro-f1 on the benchmark** as shown in figure 2. This is because we use all possible available information in the data-set.

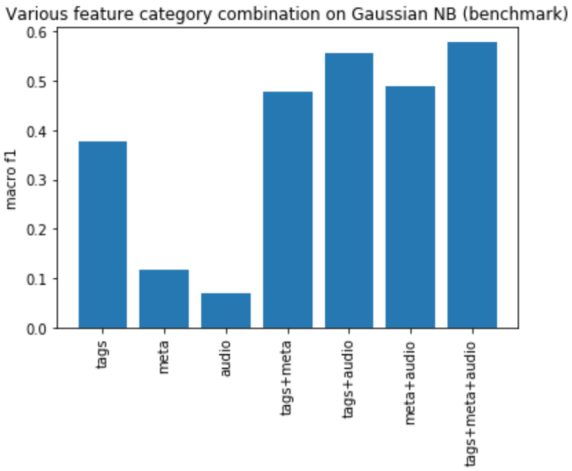


Figure 2: Feature Selection

6 Models and Performance analysis

6.1 Models considered

The model considered for this task are Gaussian Naive Bayes (**Gnb**) classifier [10], K nearest neighbor (**Knn**) classifier [11], Decision tree (**Dt**) classifier [12] (sklearn's Dt implementation only supports continuous features), Random forest (**Rf**) classifier [13] and Multi-layer perceptron (**Mlp**) classifier [14]. Although logistic regression can be changed to suit multi-class classification, it is actually a single neuron, Mlp is a more complex version of it. So we choose to use Mlp instead of it. All of these classifiers are implemented by Scikit-learn Python library [15]. Thus all require continuous data. Data are standardised for Knn and Mlp as they are magnitude sensitive. **Zero-R** is considered as baseline model. Because One-R is not suitable for high-arity attributes such as continuous attributes here. Gnb is selected as a benchmark against rest models because it is a naive model.

6.2 Evaluation metric

The model evaluation metric considered are **accuracy** and **macro-f1** (If a class has TP=0, treat f1 for that class = 0 to avoid 0 division problem.). [16] As there is huge and slight imbalanced class ditribution in train and valid data-set respectively, using **macro-f1 gives the best accuracy evaluation metric**. Error analysis including model bias and variance are also considered. The **learning curve** is a plot of learning performance of models over experience or time.

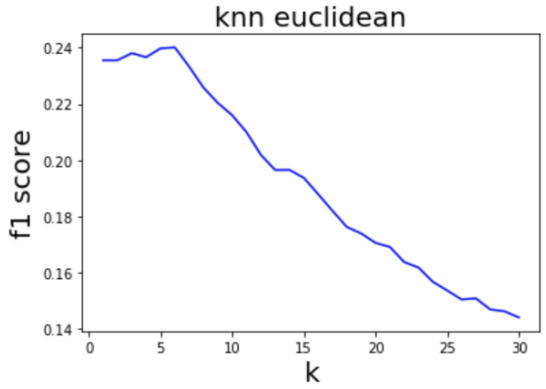
So, it can be used to investigate models' generalisation for error analysis.

6.3 Hyper-parameter tuning

Gnb has no hyper-parameter needed to be tuned. Hyper-parameters are tuned on the training set using **5-fold for faster and more accurate purpose**. All other hyper-parameters not mentioned below are using the default values in Scikit-learn.

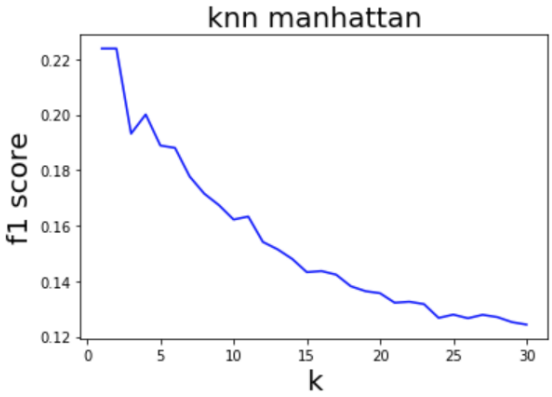
6.3.1 Knn

We use inverse distance weighting for Knn to eliminate the influence of imbalanced class. As shown in the figure 3 and 4, euclidean knn has best k = 6 and manhattan knn has best k = 1. As euclidean 6-nn has a higher macro-f1 score than manhattan 1-nn (0.24 v.s. 0.22), we choose **euclidean 6-nn** as Knn classifier.




```
k maximize f1-score : 6 with 0.24003692444554794
```

Figure 3: Tuning k for euclidean knn

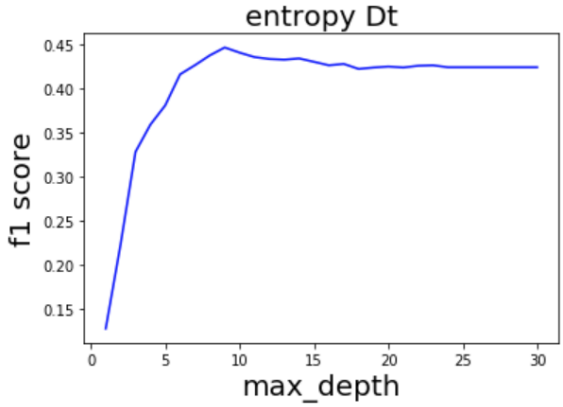



```
k maximize f1-score : 1 with 0.22380132378915638
```

Figure 4: Tuning k for manhattan knn

6.3.2 Dt

We use the entropy criterion (information gain) for Dt. As shown in the figure 5, the $max\_depth = 9$ maximize the macro-f1 score for Dt. So we choose entropy Dt with $max\_depth = 9$ as Dt classifier.

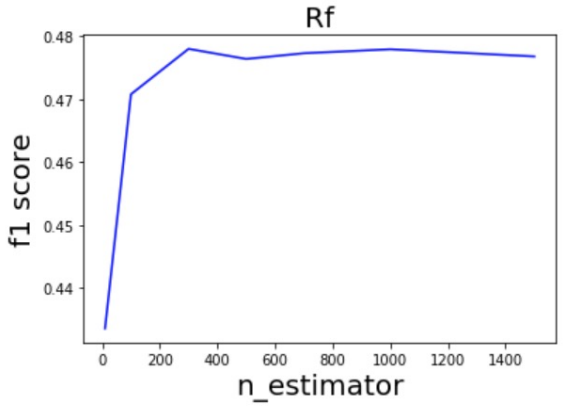



```
max_depth maximize f1-score : 9 with 0.446890710
```

Figure 5: Tuning max_depth for entropy Dt

6.3.3 Rf

Rf classifier is a bagging of multiple Dt classifier. We adapt the optimal $max\_depth = 9$ from section 6.3.2 for each Dt in Rf. As shown in the figure 6, $n\_estimator = 300$ (number of Dt in the Rf) gives the best macro-f1 score. So we choose Rf with $max\_depth = 9$ and $n\_estimator = 300$ as Rf classifier.

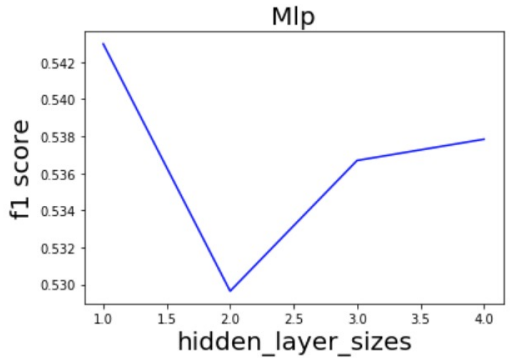



```
n_estimator maximize f1-score : 300 with 0.47797
```

Figure 6: Tuning n_estimator for Rf

6.3.4 Mlp

We use relu activation ($f(x) = max(0, x)$), default adam solver as it works well on large data-set in terms of speed and accuracy [15] and default $alpha = 0.0001$. Each hidden layer has 100 neurons but hidden_layer_sizes are tuned because the larger layer numbers mean the better ability for Mlp to learn complex mathematical relationship. As shown in figure 7, a single hidden layer with 100 neurons gives the best macro-f1 with 300 iterations.




```
hidden_layer_size maximize f1-score : 100 with 0.54298
```

Figure 7: Tuning hidden_layer_sizes for Mlp

6.4 Model Performance Comparison

We evaluate all the above models using all training instances with tuned hyper-parameters and compared against the baseline on the valid data-set.

As you can see from the table 2, all candidate models have **better performance than the baseline**. This means that the MGC problem is not difficult and improvements are achieved in our models.

Model	Accuracy	Macro f1
Zero-R	0.1222	0.0272
Gnb	0.6178	0.5792
Ada-boosted Gnb	**0.6245**	**0.5832**
Multinomial NB	0.4667	0.4466
Categorical NB	0.5667	0.5536
Dt	0.4533	0.4352
Bagged Dt	0.5778	0.5194
Rf	0.5978	0.5295
Mlp	0.6222	0.5755

Table 2: Comparison of various models on valid data-set

Knn has much worse performance than the benchmark. As shown by the figure 8, Knn is bad at predicting *jazz and blues* and *metal* music (0 evaluation metrics score). As shown by the figure 9, its majority prediction is *classic pop and rock*. As shown in the figure 10, outliers of *classic pop and rock* in the training data-set are influencing the classification of other music genres for Knn. The decision boundary is distorted by noise for Knn. The reasons can be that the training data-set is not representative enough for each music genre. Thus Knn cannot find correct neighbors.

	precision	recall	f1-score	support
classic pop and rock	0.20	0.84	0.32	55
dance and electronica	0.15	0.29	0.20	45
folk	0.30	0.20	0.24	64
jazz and blues	0.00	0.00	0.00	44
metal	0.00	0.00	0.00	66
pop	0.72	0.49	0.58	74
punk	0.33	0.14	0.19	44
soul and reggae	0.64	0.24	0.35	58
accuracy			0.28	450
macro avg	0.29	0.27	0.24	450
weighted avg	0.31	0.28	0.25	450

Figure 8: Knn classification report on valid data-set

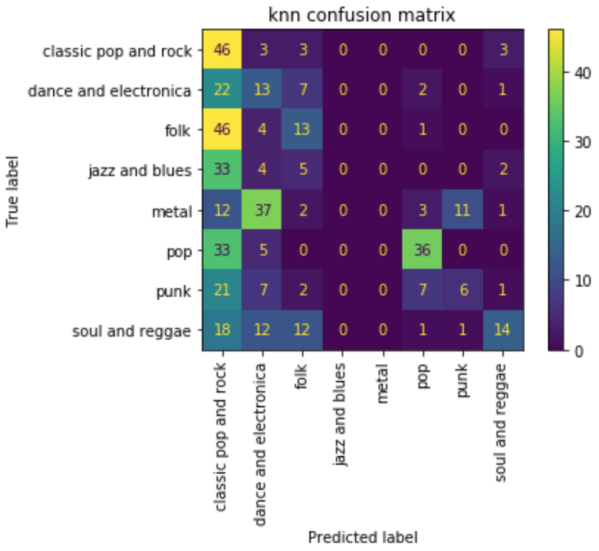


Figure 9: Knn confusion matrix on valid data-set report

```
classic pop and rock      107
dance and electronica      32
pop                        18
folk                       14
soul and reggae             8
punk                        4
jazz and blues              1
metal                       1
Name: closest genre, dtype: int64
```

Figure 10: Knn Closest neighbor of wrongly classified as *classic pop and rock* valid instances

Both Dt and Rf are rule-based learning. Dt performs slightly worse than rf because it is a simpler model.

Gnb performs best in terms of macro-f1. Although we have imbalanced class distribution in the training instances, compare the figure 11 and 1, Gnb's prediction class distribution seems not affected by the training data-set class distribution which means the number of training instances is enough for Gnb. The reason that Gnb outperforms all other models because it can still perform well even if the independence assumption is violated. For example, *loudness* and *vec_1* can have a 0.945383 Pearson correlation [8] (highly dependent). The one-hot encoded features are not Gaussian distributed which should influence the Gnb's performance, however, Gnb still performs well. As shown by the figure 12, Gnb is good at predicting *pop*. This is an example for Gnb still works well even if features are not Gaussian distributed as other NB has a worse performance by the table 2.

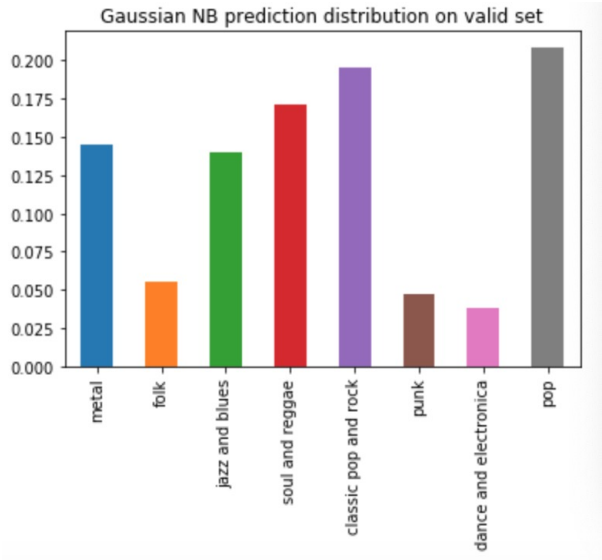


Figure 11: Gnb valid data-set prediction distribution

For Rf and Mlp, there is not so much to say because of their bad interpretability.

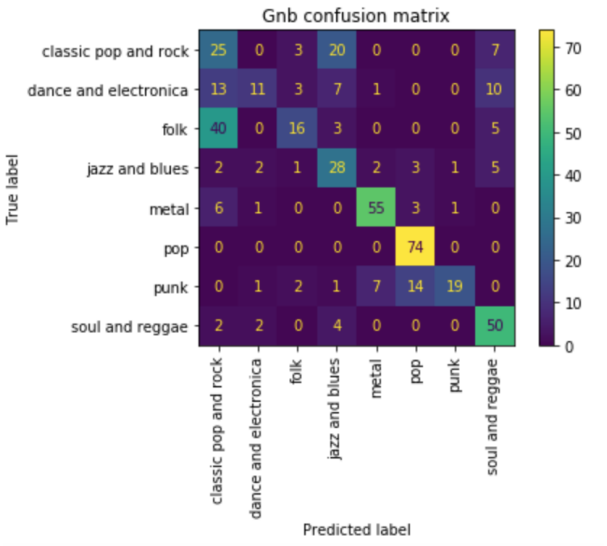


Figure 12: Gnb confusion matrix on valid data-set report

6.5 Error analysis

We use shuffled stratified 5-fold *learning_curve* from sklearn to do error analysis of models.

As shown by the figure 13, for Knn and Mlp, both of them have a large gap between training score and cross-validation score which suggests a high variance. The reason can be that Mlp is a complex model (needs a lot of data) for MGC problem with given data-set. A simpler model should be considered. Getting more training data or reducing features used is likely to solve this issue for both of them. [17] In the other hand, bagging [18] can be used to reduce variance. However, sklearn's implementation of Mlp classifier does not support the *sample_weight* in *fit()* and Knn requires too much training time in bagging. So using bagging for them is not suitable. What's more, both of them has 100% accuracy and macro-f1 but a lower cross-validation score which means they are overfitting on the training set. As a result, noise in the data is captured.

For Dt and Rf, there are still some gaps between training score and cross-validation score when train_size is large which means models have variance. The solution can be getting more training data or reducing features used. Bagging can be used for Dt as it is a simple model while Rf is a complex model which takes a long training time.

For Gnb, its accuracy and macro-f1 on valid data-set in table 2 is much higher than the score in the figure 13 when train_size is large. This is because we have the evaluation variance. The

solution can be using cross-validation rather than hold-out when splitting training and valid data-set. As Gnb has low variance when train_size is large, its evaluation score is not high which suggests it has a bias. Ada-boosting [18] can be used to reduce model bias.

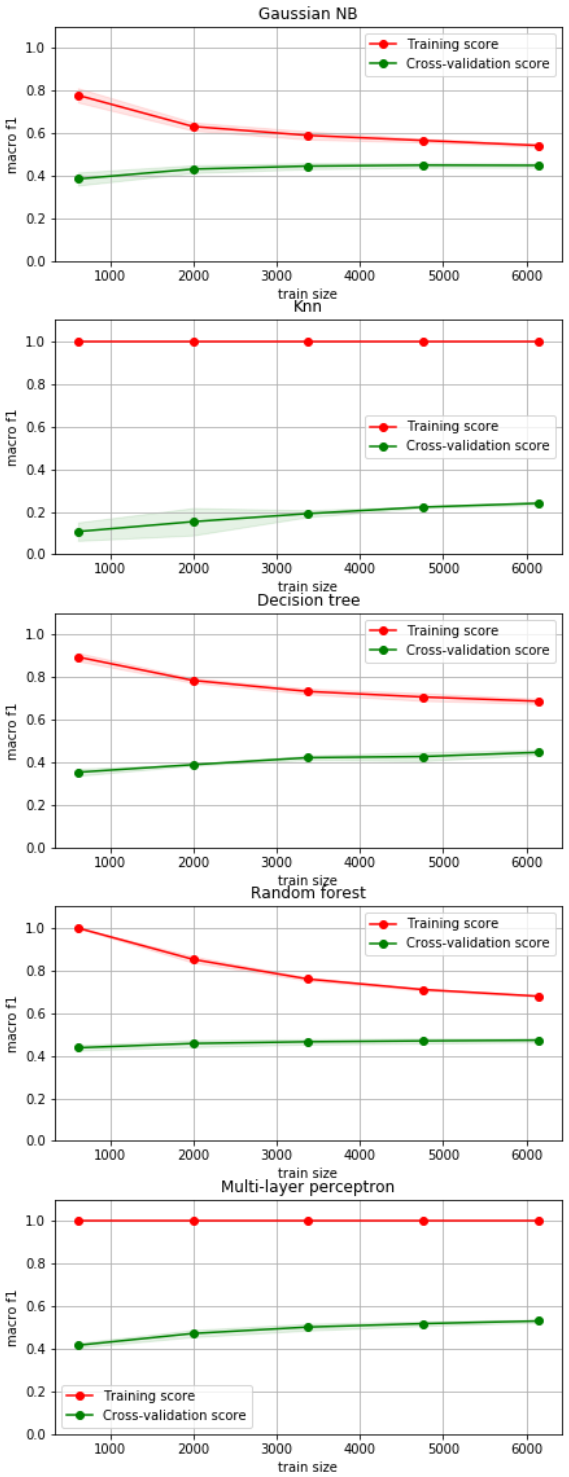
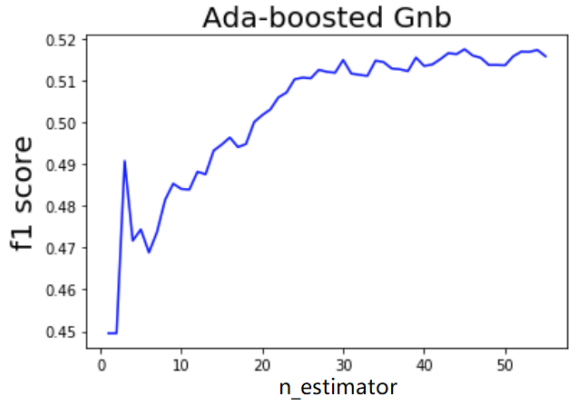


Figure 13: Error analysis of models in terms of macro-f1 (similar trend for accuracy)

6.6 Ensemble learning

As shown in the figure 14, $n\_estimator = 45$ gives the best Ada-boosted Gnb on the cross-validation of training. As shown by the table 2, boosting has some improvements on Gnb's performance by reducing bias and introducing variances as shown in figure 15.




```
n_estimators maximize f1-score : 45 with 0.5174
```

Figure 14: n_estimators for Ada-boosted Gnb

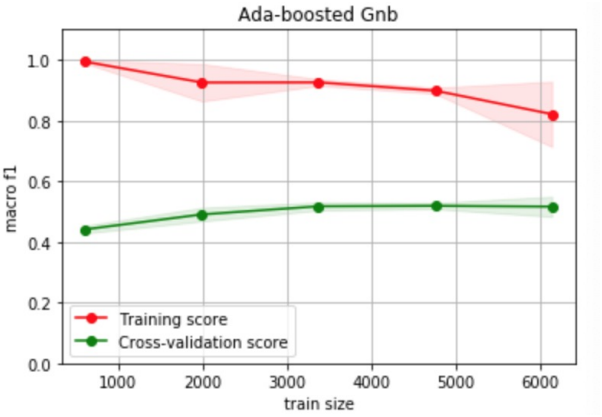


Figure 15: Ada-boosted Gnb's error analysis in terms of macro-f1 (similar trend for accuracy)

By comparing the figure 16 with 12, Gnb's performance on predicting *punk* and *folk* are dramatically improved by Ada-boosting while performance on other classess are maintained.

As shown in figure 17, $n\_estimator = 54$ gives the best bagging Dt on the cross-validation of training. As shown by the table 2, bagging has improved Dt's performance on the valid data-set by reducing variance in the model.

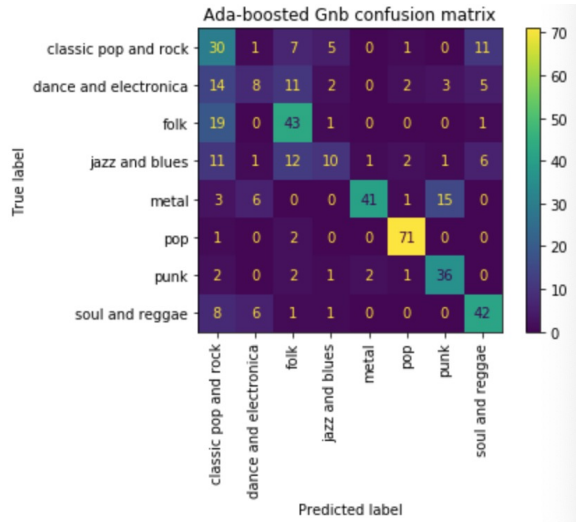
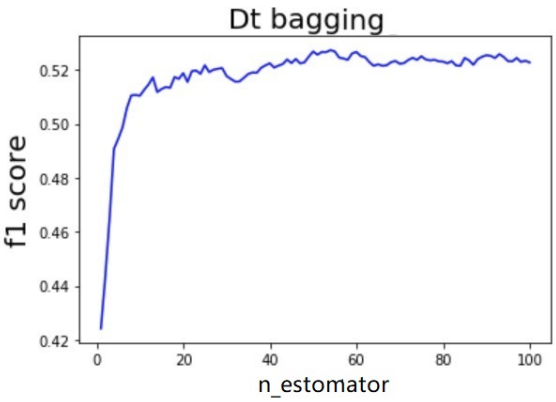


Figure 16: Ada-boosted Gnb confusion matrix on valid data-set report




```
n_estimators maximize f1-score : 54 with 0.52728
```

Figure 17: n_estimators for bagging Dt

7 Conclusion

We found that using all three feature categories gives the best result on the benchmark. Textual and audio features are the most and the least indicative feature category respectively. Ada-boosted Gnb gives the most accurate and general model for the given MGC data-set.

8 Future Direction

Association rule learning can be used for tag words to learn a pattern of words that relates to classes. More complex multi-layer stacking classifier can be tried if we have better computation power.

References

[1] R. B. Dannenberg, B. Thom, and D. Watson, "A machine learning approach to musical style recognition," *Proceedings of the 1997 International Computer Music Conference*, pp. 344–347, 1997. DOI: 10.1.1.46.527.

[2] T. George and C. Perry, "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002, ISSN: 22195491.

[3] C. Mckay and I. Fujinaga, "Automatic Genre Classification Using High-Level Musical Feature Sets," *Proceedings of the 5th International Conference on Music Information Retrieval*, pp. 525–530, 2004.

[4] W. Zhang, W. Lei, X. Xu, and X. Xing, "Improved music genre classification with convolutional neural networks," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 08-12-Sept, pp. 3304–3308, 2016, ISSN: 19909772. DOI: 10.21437/Interspeech.2016-1236.

[5] S. Vishnupriya and K. Meenakshi, "Automatic Music Genre Classification using Convolution Neural Network," *2018 International Conference on Computer Communication and Informatics, ICCCI 2018*, pp. 2018–2021, 2018. DOI: 10.1109/ICCCI.2018.8441340.

[6] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, pp. 591–596, 2011.

[7] A. Schindler and A. Rauber, "Capturing the temporal domain in echonest features for improved classification effectiveness," *Proceedings of the 10th InternationalWorkshop on Adaptive Multimedia Retrieval (AMR)*, vol. 8382, pp. 214–227, 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-12093-5{_}13.

[8] "Pearson's Correlation Coefficient," in *Encyclopedia of Public Health*, W. Kirch, Ed., Dordrecht: Springer Netherlands, 2008, pp. 1090–1091, ISBN: 978-1-4020-5614-7. DOI: 10.1007/978-1-4020-5614-7{_

}2569. [Online]. Available: https://doi.org/10.1007/978-1-4020-5614-7_2569.

[9] D. Harris and S. Harris, *Digital design and computer architecture*. Morgan Kaufmann, 2010, p. 129, ISBN: 978-0-12-394424-5.

[10] E. Jacob, "Natural Language Processing," in, MIT Press, 2019, ch. 2.2.

[11] H. Jiawei, K. Micheline, and K. Morgan, "Data Mining: Concepts and Techniques," in, 2nd ed., 2006, ch. 2.

[12] T. Mitchell, "Machine Learning," in, 1997, ch. 3.

[13] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001. DOI: https://doi.org/10.1023/A:1010933404324.

[14] E. Jacob, "Natural Language Processing," in, 2019, ch. 3.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] Sokolova and Lapalme, "A systematic analysis of performance measures for classification tasks.," *Information Processing and Management*, vol. 45, pp. 427–437, 2009.

[17] C. Sammut, G. Webb, and E. I, *Bias Variance Decomposition*. Springer, 2011, pp. 100–101.

[18] T. Pang-Ning, S. Michael, and K. Vipin, *Introduction to Data Mining*. Addison Wesley.