Audio Engineering Society

Convention Paper 10558

Presented at the 152nd Convention
2022 May, In-Person and Online

MP3 compression classification through audio analysis statistics

Jamie McFarlane and Bharathi Raja Chakravarthi

National University of Ireland

Correspondence should be addressed to Jamie McFarlane (`jamf@mailbox.org`)

ABSTRACT

MP3 audio compression can be undesirable in circumstances where high-quality music presentation is required and there is a lack of automated, evidenced, and open-source methods to determine this. This study introduced a new and accessible approach to discriminate between compression levels and identify lossy audio transcoding. Machine learning classifiers were trained on feature sets of audio analysis statistics, derived from multiple step-wise re-encodings of compressed audio samples. Two classifiers, a stacked model and a XGBoost-based model, had comparable accuracies to previous examples in the literature and marketplace (Stacked: 0.947, XGBoost: 0.970, Literature reference: 0.965, Commercial reference: 0.980). For transcoded samples, which hide compression levels with post-processing, the new classifiers were less accurate than existing methods. However, all methods were inaccurate in identifying transcodes where artificial noise was added via the μ-law encoder. A command-line implementation is available at gitlab.com/jammcfar/kbps_detect_proto.

1 Introduction

MPEG Layer III (MP3) encoding is a widespread method of lossy audio compression which balances file size and audio quality. A bitrate of 128 kilobits per second (kBps) is considered sufficient for many presentation modes. This will have data strategically removed throughout the audio sample, with a strong emphasis on removing very high frequencies above 16kHz. However, there is a demand for less compressed audio with higher bitrates, especially in circumstances where presentation on high-end equipment is valued [1, 2, 3].

For any given MP3 file, the MP3 encoding bitrate is indicated on the file metadata. However, if the MP3 encoding is repeated at a higher bitrate than the current bitrate of the sample, this will increase the bitrate on the metadata while retaining the prior compression artifacts of the lower bitrate (e.g. a 128kBps MP3 transcoded to 320kBps). This is an example of transcoding, and this can not only further reduce sample quality though generation loss, but also disguise the actual quality to the consumer. This problem has created a demand for software which can detect lossy audio transcoding. The challenge here has also been complicated by the possible application of methods which could reduce the accuracy of transcoding detection. For instance, μ-law transcoding can be used to introduce noise into the upper regions of the frequency spectrum which would have been previously cleared by MP3 compression. Bandwidth extension methods, found in some more modern encoders, can also replicate elements of lower frequencies into the upper audible range and confuse detection methods [4].

1.1 Prior work

An important step in performing audio compression discrimination has been through applying mathematical transformations such as the Fast Fourier transformation (FFT) [5]. Useful products of this approach are spectrograms and frequency spectrums, that map time, frequency and signal intensity on to separate dimensions. These data types are suitable for training machine learning classifiers which can discriminate between compression levels and identify transcoding. For example, an approach using a Polynomial Support Vectors Machines (PSVM) model has been used to classify MP3 compression using frequency spectrum data above 16kHz [6]. Another approach trained a Convolutional Neural Network (CNN) on MP3 spectrograms and was also reportedly successful [7]. However, a challenge in following up an approach employing neural networks is the requirement for a large dataset of lossless audio. Despite previous examples in the literature and on the market, consumers are still limited in their ability to automatically verify compression levels in formats such as MP3. Available methods are either closed-source, unevidenced or are simply unavailable [6, 7, 8, 9]. Challenging these detection methods by modifying the audio signal to evade detection has also not been explored.

In this study, a new approach for determining compression levels was explored which was not dependent on acquiring a large training dataset. A novel feature generation process was used to generate training data for machine learning models. Here, MP3 samples were subject to transcoding and sound and image analysis statistics, such as entropy and image degradation, were derived from these copies. Different methods of preprocessing were also evaluated in a data mining style approach.

2 Methods

2.1 Proposed technique

At a high level, this new approach incorporates an additional round of lossy transcoding and then examines its effects on an audio sample of interest. The data removal by compression algorithms has a much smaller impact on samples that are already highly compressed (e.g. a 320kBps MP3 sample transcoded to 128kBps will be subject to much greater data removal than a 160kBps sample transcoded to 128kBps). This phenomenon was exploited to create training data for machine learning classifiers which could then be applied to identify compression levels in unseen audio samples.

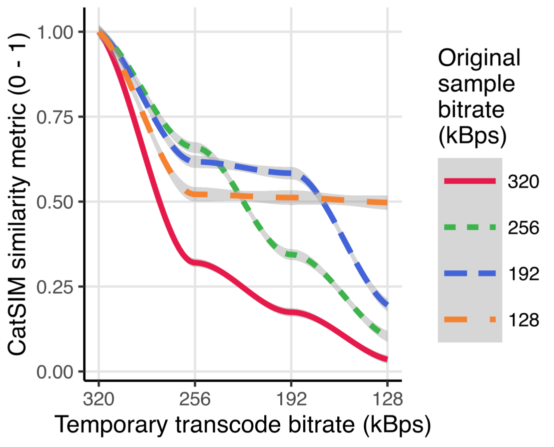

Fig. 1: An example of transcoding curves, using the CatSIM statistic. CatSIM is a β index, which compares the structural similarity between matrices. In this study, this measure used spectrograms as its input. The results shown are from applying this feature engineering method on the entire MP3 dataset ($n = 508$). The lines are Locally Estimated Scatterplot Smoothing (LOESS) curves with 95% confidence intervals.

In this feature engineering process, a short MP3 sample is temporarily transcoded into four commonly seen bitrates (i.e. 320kBps, 256kBps, 192kBps, 128kBps). Then, a statistical measure (or *index*) was taken for each of these temporary transcodes (e.g. mean, kurtosis, entropy). These measurements were then arranged in sequence, ordered by bitrate, and this data acted as a set of features for training machine learning classifiers. This set of data can be visualised as a curve, whose shape would vary depending on the compression level of the original sample [Fig. 1]. This process was repeated for a large number of files and different statistical measures to create the dataset for this study.

This approach to generating data is more complex and

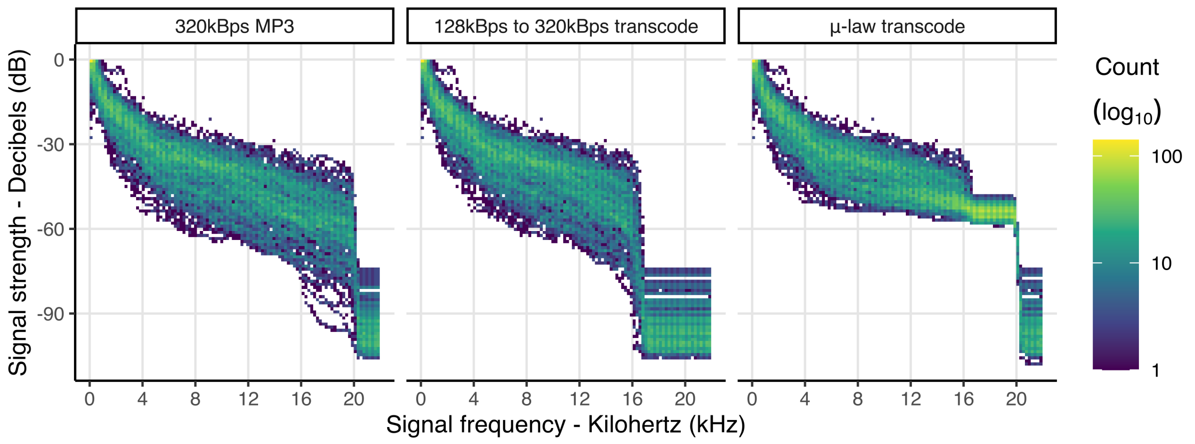

Fig. 2: Combined frequency spectrums of the 320kBps, 128kBps to 320kBps and μ-law transcoded samples used in this study, visualised as 2D-histograms. For each set, n = 127. 256, 192 and 128kBps samples are not shown. The 320kBps MP3 and 128kBps to 320kBps transcode spectrums show frequency cutoffs around 20kHz and 16kHz respectively. The μ-law transcodes have noise added up to 20kHz.

computationally intensive than by simply applying statistical measures directly to audio samples. However, this method provides two advantages. Firstly, it solves the problem of how to normalise the variance between different audio samples which vary greatly in their audio characteristics. For example, rock music would generally have a higher average signal than a quiet acoustic music, which would prevent using measures such the mean to discriminate a subtle characteristic such as compression level. The second advantage is that it allows us to explore the use of statistic measures which compare audio samples, by using two temporary transcodes at a time.

For the statistical measures explored, 34 different indices were sourced from functions found in the open-source R packages *seewave*, *SPUTNIK* and *catsim* [10, 11, 12]. α-indices, requiring a single audio sample, were applied to each of the temporary transcodes to form a 4-point curve. β-indices, requiring pairs of samples, compared each temporary transcode back to the highest bitrate temporary transcode (e.g. 320kBps vs 256kBps). Many of these measures required inputs of frequency spectrums or spectrograms and these transformations were performed as needed.

This research also explored different combinations of pre- and post-processing which multiplied the size of the dataset. The first variation transformed the input audio signal into *ghost* audio [13]. This involves subtracting transcodes of lower bitrates from transcodes of higher bitrates, leaving only the difference in compression artifacts for analysis. In this study, each of the temporary transcodes were subtracted from the 320kBps version, which also reduced the size of each set of statistical measures by 1. The second variation was an attempt at feature reduction and used the slopes between the ordered statistical measures of the temporary transcodes as features. These variations produced four different processing possibilities (i.e. 2×2) and in total, 136 different types of feature sets were used.

2.2 Dataset and software requirements

Music audio WAV files from the MusDB and MedleyDB V.2.0 datasets were used as an uncompressed source (n = 127) [14, 15][1]. To form the MP3 dataset, each WAV file was encoded into 128, 192, 256 and 320kBps versions (n = 508). Note that this was a distinct step from creating the sets of temporary transcodes as part of the feature engineering process. Two sets of 127 transcoded MP3 files were then also created, originating from the 128kBps files [Fig. 2]. These were used to evaluate the ability of the classifiers at detecting transcoded samples. The 128kBps to 320kBps

[1]Data in the MusDB dataset which originated in the DSD100 dataset were manually removed as permission from this source could not be sought.

transcode set had the header data of 320kBps files, but the compression artifacts of 128kBps samples. The μ-law transcode set encoded the 128kBps files with the μ-law encoder which added noise up to 20kHz, followed by a final transcoding to 320kBps MP3. The main software requirements were the R language, for data processing and modelling, and ffmpeg for audio file encoding [16, 17].

2.3 Model training and evaluation

Two different predictive classifiers were evaluated. The first was a stacked model, which partitioned sets a features into submodels. In total, 136 unique submodels for each combination of statistical measure and preprocessing method were combined. The second model used the XGboost algorithm, which was presented with all available data at once and was used to baseline the effect of creating submodels using the individual feature sets. These classifiers were compared with two existing methods. The first was a PSVM classifier which was previously described in the literature and recreated using the studies original hyperparameters [6]. The second pre-existing method was the commercial product Fakin' The Funk (FTF), which identifies *faked* audio files [8].

To evaluate the stacked and XGboost models, nested five-fold Monte-Carlo cross-validation (MCCV) was performed with 75-25% splits in the data. This workflow was chosen to make better use of the small available dataset. The PSVM model did not require hyperparameter tuning and was trained and evaluated using unnested MCCV. The stacked model used radial basis function kernel SVMs (RBFSVM) for all of the submodels, and used least absolute shrinkage and selection operator (LASSO) regression to blend and prioritise the sub-model predictions. For computational efficiency, an additional feature selection step was performed at the start of each outer fold for the stacked model. This fitted untuned RBFSVM models for each submodel and selected the best 50, based on their training accuracy. Since this is an ordinal classification problem, both the stacked and XGboost models performed regressions against the file bitrate and the results were rounded to 128, 192, 256 or 320kBps. Further compromises were made to save on computational resources. Firstly, only the middle six seconds of each sample were used. Secondly, only the frequencies above 16kHz were used for indices which required FFT transformed data as inputs. This would likely also improve performance, as MP3 compression artifacts are most prevalent in this region.

There were two stages of evaluation. The first evaluated the different methods abilities to discriminate between MP3 bitrates. Here, the Stacked, XGBoost and PSVM results were calculated from the pooled MCCV holdout sets. Tests of model stability were performed to ensure the results from the different folds could be considered equivalent and combined for evaluation. The second stage evaluated their abilities to identify the original bitrate of transcoded samples, with the scope limited to evaluating 128kBps to 320kBps, and 128kBps to μ-law transcodes. This stage used *final* models for the Stacked, XGboost and PSVM classifiers that were trained on the entire dataset following the first stage. FTF, unlike the other methods, was an off-the-shelf product and for both evaluation stages simply analysed to all available full length MP3 samples using its default settings.

3 Results

The results of this study are presented in three parts. The first part compares the accuracies and errors of the proposed classifiers with the two reference methods at discriminating between MP3 samples of different bitrates. This was followed by comparing the accuracies of the methods at identifying the original bitrate of transcoded samples. Finally, feature importance measures were explored to identify the most important statistical measures used by the classifiers.

3.1 MP3 compression classification

Tests of model stability for the Stacked, XGboost and PSVM models showed that the results from the MCCV folds were generated by equivalent models and could be pooled for evaluation (see Appendix). Balanced accuracy was evaluated, since *n* varied between the MCCV folds. The stacked and XGboost models returned accuracies of 0.947 and 0.970 respectively. The reference methods, FTF and PSVM, returned 0.983 and 0.965 respectively. Considering the confusion matrices of the methods, both FTF and the PSVM model were negatively biased and tended to overestimate the compression of the samples, while the stacked and XGboost models appeared to be more balanced in their misclassification [Fig. 3]. The stacked model produced the highest percentage of any error type, where 13.92%

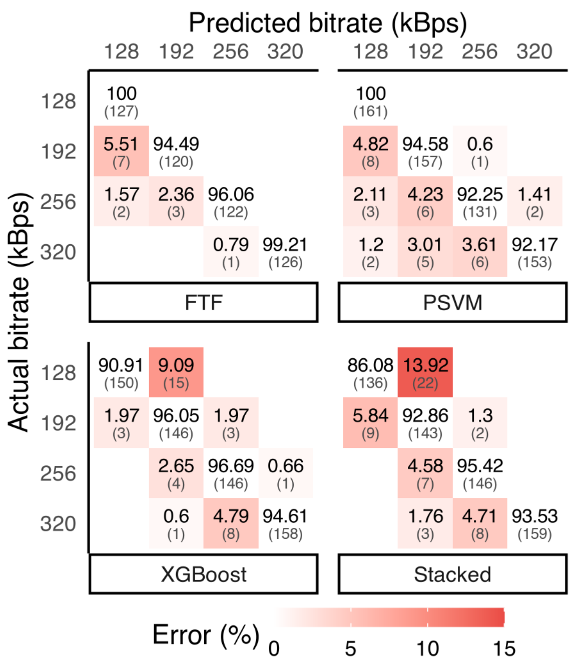

Fig. 3: Confusion matrices of MP3 bitrate prediction. Stacked, XGboost and PSVM models results were pooled from hold out sets. FTF analysed the MP3 dataset in a single pass. In each case, $n = 508$. Accuracies are percentages, with n in brackets.

of 128kBps samples were misclassified as 192kBps. The PSVM model made errors with the biggest distance, where two 320kBps samples were misclassified as 128kBps.

3.2 Transcode identification

The evaluation of the transcoded MP3 files considered accuracy as the evaluation metric, as the test data was only from one class (n=127). For the 128kBps to 320kBps transcodes, the new approach was less effective than the reference methods. The stacked model misclassified all samples as either 320kBps (95.3%) or 256kBps (4.7%), while the XGboost model misclassified 21.3% samples as 192kBps. FTF and the PSVM model had accuracies of 1.000 and 0.923 respectively, with the latter only giving misclassifications of 192kBps. For the μ-law transcodes, all methods performed poorly. FTF and XGboost misclassified all samples as 320kBps. The stacked model misclassified nearly all samples as 320kBps (95.28%), with a few exceptions being classifed as 256kBps (4.72%). The PSVM model had the best performance, but still only classified 19.69% correctly as 128kBps. The misclassifications were split between 320kBps (77.95%) and 256kBps (2.36%).

3.3 Feature importance

Feature importance scores were derived from submodel weighting for the stacked model, and internal gain scores for XGboost. In both cases, these were from the final models trained over the entire dataset. The most important statistics used by the primary and secondary models included CatSIM, cumulative frequency spectra difference and structural similarity [Tab. 1]. Combining the top five statistics from each model some patterns could be identified. The majority of these were β-indices (9/10). Different types of data processing methods were also favoured, with the majority being from slope transformed data rather than raw data (7/10). *Ghost* data as a pre-processing step did not feature in the top five features of either model.

Table 1: Top five features for the stacked and XGboost models, ranked by internal scoring.

Model	Measure	Score
Stacked	CatSIM	0.24
	Cumulative freq. spectra dist	0.20
	Structural similarity	0.13
	Manhattan dist	0.12
	Kulback-Leiber dist	0.12
XGboost	Cumulative freq. spectra dist	0.45
	Kulback-Leiber dist	0.25
	Log spectral dist	0.11
	Total entropy	0.04
	Kolmogorov-Smirnov dist	0.01

4 Discussion

Overall, the evaluation suggests that the new approach can produce models which are similar in performance to existing methods at discriminating between MP3s. Though existing methods currently perform better at transcode detection, there may be the potential for this approach to improve with further development. The results of the stacked model at detecting 128kBps to 320kBps transcodes was surprisingly poor and it could be modeling some unknown aspect of this type of transcoding. When it comes to μ-law transcoding, all methods performed poorly. Though the prevalence of this in the wild is unknown, current methods to not appear to be robust to the introduction of signal noise and this is an area that should improved upon. Future work should also evaluate the effect of applying bandwidth extension methods if possible. Whether the proposed approach is built upon or not, future methods could incorporate some of the learnings of this study. A stacked model combining the PSVM model with a different model based on other statistical measures is one possibility. Another could build on the PSVM approach, that uses the slopes between points as a feature reduction method.

In terms of execution speed, the proposed method is currently slower than existing methods. Analysing transcoded samples took the PSVM model $\sim$3 seconds per sample, while the XGboost model too $\sim$26 seconds and the stacked model took $\sim$30 seconds. Removing unimportant features, reducing the number of encodings, and re-writing the code in a lower level language are among the possible optimisations that would all help significantly.

In terms of scope, this study was limited by a small sample size and the number of potential statistical measures for feature generation. Additionally, the 16kHz cutoff, which was enforced to reduce computation time and improve performance, could be challenged in future work and may work well when using samples with naturally little audio data above this threshold. Other popular *lossy* encoders such as Advanced Audio Coding (AAC) and Opus could also be included [1, 18].

5 Summary

In summation, this study developed a new method for determining the compression levels of MP3 samples. It can be used to create machine learning classifiers with accuracies comparable to existing methods at discriminating between MP3 bitrates, though it currently appears to be less effective at identifying transcoded samples. A reproduction of the PSVM method, an evaluation of a commercial offering, and challenging these with noisy samples were also accomplished in the process. The study was limited in the scope of its evaluation, but this approach can serve as a basis for future studies with a larger scope for evaluating more audio sample types and different compression codecs. The evaluated classifiers, with supplementary information, have been made available online as prototype command line interface at: https://gitlab.com/jammcfar/kbps_detect_proto.

References

[1] Brandenburg, K., "MP3 and AAC Explained," in *Proceedings of the 17th International Conference of High Quality Audio Coding*, AES, Florence, Italy, 1999, uRL: https://www.aes.org/e-lib/browse.cfm?elib=8079.

[2] Harris, A., "Do Young People Actually Care About the Quality of Their MP3s?" in *Proceedings of the 130th convention for the Audio Engineering Society*, pp. 9–12, AES, London, UK, 2011, uRL: https://www.aes.org/e-lib/browse.cfm?elib=16566.

[3] Hines, A., Gillen, E., Kelly, D., Skoglund, J., Kokaram, A., and Harte, N., "Perceived Audio Quality for Streaming Stereo Music," in *Proceedings of the 22nd ACM international conference on Multimedia*, MM '14, pp. 1173–1176, ACM, New York, NY, USA, 2014, ISBN 978-1-4503-3063-3, dOI: 10.1145/2647868.2655025.

[4] Dietz, M., Liljeryd, L., Kjorling, K., and Kunz, O., "Spectral Band Replication, a novel approach in audio coding," in *Proceedings of the the 112th Convention of the Audio Engineering Society (AES)*, AES, Munich, Germany, 2002, uRL: https://www.aes.org/e-lib/online/browse.cfm?elib=11328.

[5] Kandadai, S., Hardin, J., and Creusere, C. D., "Audio quality assessment using the mean structural similarity measure," in *Proceedings of the 16th IEEE International Conference on Acoustics, Speech and Signal Processing, (ICCSP)*, pp. 221–224, ICCSPA, Las Vegas, NV, USA, 2008, dOI: 10.1109/ICASSP.2008.4517586.

[6] D'Alessandro, B. and Shi, Y. Q., "MP3 bit rate quality detection through frequency spectrum analysis," in *Proceedings of the 11th ACM workshop on Multimedia and security*, pp. 57–62, Association for Computing Machinery, New York, USA, 2009, ISBN 978-1-60558-492-8, dOI: 10.1145/1597817.1597828.

[7] Hennequin, R., Royo-Letelier, J., and Moussallam, M., “Codec independent lossy audio compression detection,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 726–730, ICASSP, New Orleans, LA, USA, 2017, dOI: 10.1109/ICASSP.2017.7952251.

[8] Decker, U., “Fakin’ The Funk - Detect the true quality of your audio files in one batch (v.3.0.0.139),” URL: https://fakinthefunk.net/en/index.html, (accessed Feb. 22, 2021).

[9] GAR software, “Similarity App,” http://www.similarityapp.com, (accessed Feb. 05, 2021).

[10] Sueur, J., Aubin, T., and Simonis, C., “Seewave: a free modular tool for sound analysis and synthesis,” *Bioacoustics*, 18, pp. 213–226, 2008, doi: 10.1080/09524622.2008.9753600.

[11] Paolo, I., Gonçalo, C., Zoltan, T., K, N. J., and C, G. R., “SPUTNIK: an R package for filtering of spatially related peaks in mass spectrometry imaging data,” *Bioinformatics*, 36(1), pp. 178–180, 2018, dOI: 10.1093/bioinformatics/bty622.

[12] Thompson, G. Z. and Maitra, R., “CatSIM: A Categorical Image Similarity Metric,” *arXiv*, 2020, uRL: http://arxiv.org/abs/2004.09073.

[13] Maguire, R., “The Ghost in the MP3,” in *Proceedings of the 40th International Computer Music Conference (ICMC)*, pp. 14–20, International Computer Music Association, Athens, Greece, 2014, uRL: smc.afim-asso.org/smc-icmc-2014/papers.

[14] Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., and Bittner, R., “MUSDB18-HQ - an uncompressed version of MUSDB18,” DOI: 10.5281/zenodo.3338373, 2019.

[15] Bittner, R., Wilkins, J., Yip, H., and Bello, J., “MedleyDB 2.0: New Data and a System for Sustainable Data Collection,” in *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR-16)*, pp. 7–11, ISMIR, New York, NY, USA, 2016, uRL: wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/08/bittner-medleydb.pdf.

[16] FFmpeg contributors, “FFmpeg (v.4.3.1-4ubuntu1),” https://www.ffmpeg.org, (accessed Feb. 14, 2021).

[17] R Core Team, “R: A Language and Environment for Statistical Computing (v.4.0.2),” URL: https://www.R-project.org/, (accessed Nov. 28, 2020).

[18] Valin, J., “Definition of the Opus Audio Codec,” RFC 6716, Mozilla Corporation, 2012.

[19] Fleiss, J. L., Levin, B., and Paik, M. C., *Statistical Methods for Rates & Proportions*, Wiley-Interscience, New Jersey, NJ, USA, third edition, 2003.

6 Appendix

The MCCV trained models were tested for instability by conducting a comparison of matched samples (n = 625). Here, a Mantel-Haenszel Chi-squared statistic was calculated between the different folds for each model type [19, p. 387 - 394]. Non-significant results would indicate that the models in the folds would be roughly equivalent and allow the results of each fold to be pooled. When setting α at 0.05, for all tested classifiers, there were no statistically significant differences found in the ratings of individual samples, by the different models, between the folds [Tab. 2].

Table 2: Model stability testing for the trained models.

Model	Mean[a]	n	df	Q[b]	p
PSVM	0.964	635	4	0.212	0.995
XGBoost	0.970	635	4	0.313	0.989
Stacked	0.947	635	4	0.417	0.981

[a] Balanced accuracy mean. [b] Q: Mantel-Haenszel χ^2 statistic.