The Effect of Text Data Augmentation Methods and Strategies in Classification Tasks of Unstructured Medical Notes

Hongxia Lu
Chapman University
Cyril Rakovski (rakovski@chapman.edu)
Chapman University

Research Article

Keywords: Data Augmentation, NLP, Deep Learning, Text Classification, Unstructured Medical Notes

Posted Date: September 15th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2039417/v1

Abstract

Background

Text classification tasks of unstructured medical notes are often challenged with the issues of highly imbalanced classes and/or small sample sizes. Data augmentation is a common approach to mitigate the impact of these issues and enhance model performance. However, not all augmentation methods improve model performance, and an uninformed and arbitrary choice of augmentation methods may hurt model performance instead. In addition, the widely used strategy of augmenting until balanced may not always work the best.

Methods

In this paper, we investigated the effect of 20 different augmentation methods and several different augmentation strategies in 16 classification tasks. The 16 classification tasks were divided into 4 groups based on their disease prevalence, and different augmentation strategies and the 20 augmentation methods were applied to different groups. The Transformer Encoder model was run in all tasks for each of the 20 augmentation methods and the strategies, and then their model performance was compared against each other and against that without augmentation.

Results

Our results show that in addition to being a fast augmenter, the Splitting Augmenter consistently improved the model performance in terms of AUC-ROC and F1 Score in all strategies for most tasks. For highly imbalanced tasks, the strategy that augments the minority class until balanced, improved model performance by the largest margin. For other tasks, the best performing strategy was the one that augments the minority class until balanced and then augments both classes by an additional 10%. The largest improvement was 0.13 in F1 score and an impressive 0.34 in AUC-ROC, and both were produced by the Splitting Augmenter in the strategy that augments the minority class until balanced.

Conclusions

Different text data augmentation methods have different effects on the model performance. Some enhance model performance, and others yield no improvement or even have an adverse impact. With the right choice of augmentation methods, the model performance can be substantially improved. For the highly imbalanced tasks, the strategy that augments the minority class until balanced yielded the largest improvement. For other tasks, the strategy that keeps augmenting both classes by an additional 10% after reaching balance enhanced model performance further.

Introduction

Unstructured medical data contains rich information about patients' health condition that is often not available in structured data fields where the type and amount of information is strictly limited. The novel,

sophisticated and powerful NLP deep learning models such as the Transformer and BERT have become effective tools for extracting information from unstructured textual data to make predictions in classification tasks [1–4]. Many studies have been carried out in the medical research community to make predictions about mortality or the presence of a certain disease condition [5–8]. However, due to the cost of the collection or annotation of the data, labeled unstructured medical notes often come in small sample sizes. In addition, the classes to be predicted in the medical field are often imbalanced. In our previous work which compared the behavior of seven deep learning algorithms for classification tasks of unstructured medical notes with different class imbalance levels, we showed that when the classes were highly imbalanced (especially when the sample size is small), all algorithms performed poorly largely due to the fact that there were too few data in the minority class for the algorithms to learn from [9].

A common approach to tackle the imbalance issue is to use data augmentation methods to increase the size of the minority class. Theoretically speaking, in addition to mitigating the impact of the class imbalance issue by increasing the size of the minority group, data augmentation methods may also help make the model more robust against noise if the kind of noise the augmentation brings in is similar to that of the data. For example, data augmentation in the Computer Vison field is a popular or even standard approach to overcome these issues and to enhance model performance [10,11]. Image data augmentation techniques include scaling, cropping, flipping, shifting, rotating, as well as changing color features (brightness, contrast, saturation, and hue), or injecting noise. They are intended to mimic the variation caused by different photographers with different devices in different lightings and environments. In text mining, common data augmentation techniques include deleting, inserting, swapping alphabets or words randomly to simulate the typing errors humans make when creating the texts [12–14].

This study aims to investigate the effect of 20 different text data augmentation methods on the classification efficacy in various class imbalance scenarios as well as with different strategies of data augmentation. All literature that we found on text data augmentation is focused on augmentation methods using a single augmentation strategy which is generally augmenting the minority class until balanced. However, different strategies may also have an impact on model performance. In addition, for classification tasks that are roughly balanced, it may help enhance model performance even more if additional data augmentation is applied to both classes after they become balanced. Therefore, depending on the imbalance level of the classes, we designed different augmentation strategies to investigate their effects on model performance.

Data

The data used in this study are de-identified hospital discharge summaries that were made publicly available by Harvard University for a challenge in 2008 to classify these disease conditions [15]. The dataset contains 1,237 unique hospital discharge summaries that were annotated with intuitive and textual judgements of the presence or absence of 16 disease conditions. The literature on classifying these disease conditions using this dataset employed rule-based methods which involved domain experts

and heavy text preprocessing that are tailored for these specific discharge summaries in association with these 16 diseases [16–19].

In order to better demonstrate the effect of different text data augmentation methods as well as different augmentation strategies on the classification efficacy, we simplified the multi-class task into a binary-class problem for the intuitive labels only. The 1,237 discharge summaries were matched with 16 binary labels (presence or absence of each of the 16 diseases). Table 1 is a list of the disease prevalence of the 16 disease conditions with hypertriglyceridemia being the least prevalent (5%) and hypertension the most prevalent (73%). In cases when the disease prevalence is greater than 50%, the minority class becomes the negative group (disease absence group) and the classes are imbalanced in the reversed order.

Table 1. Disease Prevalence (N =1,237)

Disease	Disease Prevalence	Disease	Disease Prevalence
Hypertriglyceridemia	5%	GERD*	20%
Venous Insufficiency	7%	Depression	20%
Asthma	13%	Obesity	40%
Gout	13%	CHF*	43%
OSA*	14%	Hypercholesterolemia	47%
PVD*	15%	CAD*	55%
Gallstones	15%	Diabetes	66%
OA*	18%	Hypertension	73%

OSA* = obstructive sleep apnea; PVD* = peripheral vascular disease; OA* = osteo arthritis;

GERD* = gastroesophageal reflux disease; CHF* = congestive heart failure; CAD* = coronary artery disease

These discharge summaries mainly contain contents such as the description of symptoms, medical history, and information about physical examination and laboratory results, treatment or services provided if applicable. General text preprocessing (i.e. converting to lower cases, removing standard English stop words, digits, and punctuations) was applied to the summary notes before tokenizing, truncating, and padding. When data augmentation is applied, the preprocessing takes place after the data is augmented. However, in order to determine the length of the notes to be fed into the model, we preprocessed the data temporarily to find the distribution of the lengths (number of words) of the notes. The descriptive statistics of the lengths of the notes before and after preprocessing are shown in **Table 2**.

Table 2. Descriptive Statistics

Descriptive Statistics	Number of Words	
	Before Cleaning	After Cleaning
Minimum	146	50
25% Percentile	819	391
Median	1084	517
Mean	1170	557
75% Percentile	1425	687
Maximum	4280	2098
Standard Deviation	506	242

Methods

As shown in our previous work, the compact Transformer Encoder performs consistently better than the other six deep learning algorithms that were investigated for these classification tasks on this dataset [9]. Therefore, we chose to use the Transformer Encoder as the classifier in this study. The Transformer algorithm, designed for translation tasks by Vaswani et al, is a novel network architecture that is based solely on attention mechanisms[20]. The Transformer is composed of 6 encoders and 6 decoders with 8 heads in each encoder and decoder. The Transformer Encoder algorithm used for the classification task in this study is a simplified version of the Transformer with a single encoder (without decoders) with 2 heads.

We ran the Transformer Encoder model for the 16 classification tasks with and without data augmentation. The 16 tasks were divided into 4 groups based on their disease prevalence levels as shown in Table 3. There is a gap between Group 2 and Group 3 in terms of disease prevalence because we do not have any tasks whose disease prevalence falls between 20% and 40%. We do not have any tasks with disease prevalence greater than 80% either. However, if switching the 0 and 1 values in the outcome variable, those with prevalence between 60% and 80% can be considered as a group with prevalence between 20% and 40%.

Table 3
Task Groups

Group	Tasks	Description
1	Hypertriglyceridemia, Venous Insufficiency	Disease Prevalence < 10%
2	Asthma, Gout, OSA, PVD, Gallstones, OA, GERD, Depression	Disease Prevalence between 10% and 20%
3	Obesity, CHF, Hypercholesterolemia, CAD	Disease Prevalence between 40% and 60%
4	Diabetes, Hypertension	Disease Prevalence between 60% and 80%

For each group, we used different augmentation strategies as listed in Table 4. The strategies were named in such a way that they reflect two pieces of information. The first two characters represent which group this strategy is applied to, and the characters after the underscore represent how much data are augmented. For example, G1_20 represents the strategy that was applied to Group 1 where the minority class was augmented until the prevalence becomes 20%. G4_R10 means that this strategy was applied to Group 4 and the minority class was augmented until the prevalence is 50% and then raise both classes by an additional 10%.

Table 4
Augmentation Strategies

Group	Strategies	Description
Group 1	G1_20	Augment the minority class until the prevalence becomes 20%
	G1_30	Augment the minority class until the prevalence becomes 30%
	G1_40	Augment the minority class until the prevalence becomes 40%
	G1_50	Augment the minority class until the prevalence becomes 50%
Group 2	G2_30	Augment the minority class until the prevalence becomes 30%
	G2_40	Augment the minority class until the prevalence becomes 40%
	G2_50	Augment the minority class until the prevalence becomes 50%
Group 3	G3_50	Augment the minority class until the prevalence becomes 50%
	G3_R10	Augment the minority class to 50%, then raise both classes by 10%
	G3_R20	Augment the minority class to 50%, then raise both classes by 20%
Group 4	G4_50	Augment the minority class until the prevalence becomes 50%
	G4_R05	Augment the minority class to 50%, then raise both classes by 5%
	G3_R10	Augment the minority class to 50%, then raise both classes by 10%

The purpose of using different augmentation strategies is to investigate the effect of different augmentation strategies and to see if there is a “sweet spot” for the amount of augmentation. Theoretically speaking, the more augmentation is applied the more noise is added to the data. Our hypothesis is that there may be a trade-off between increasing the sample size of the minority class and introducing unwanted noise to the data. In addition, for data with different class imbalance levels, the desired amount of augmentation may be different. The design of these strategies is based on this hypothetical trade-off. For Groups 1 and 2 which were highly imbalanced, we stopped augmenting after reaching 50% prevalence to avoid introducing additional unnecessary noise. For Groups 3 and 4, we kept augmenting (both classes) even after reaching 50% prevalence. For Group 3 which was roughly balanced, it did not take much augmentation to reach 50% prevalence and thus the effect of data augmentation may not be showing. Therefore, we kept augmenting and raised both classes by 10% and 20%. For Group 4 which was reversely imbalanced, we raised both classes by 5% and 10% because the number of samples to be augmented to reach 50% prevalence was more than that needed for Group 3.

For each classification task and each strategy, we used 20 data augmentation methods from the Python NLPAug library on 10 randomly split training sets (with stratification to ensure that the disease prevalence is the same in each training and test set for the same task). The descriptions of the 20 augmentation methods are listed in Table 5. A randomly selected sentence from the discharge summary notes was used as an example to show what each augmenter does.

Table 5
Data Augmentation Methods (located on Page 7)

Augmenter	NLPAug Function, Description, and Example
	Original Text: He has also undergone recent eye surgery of unknown origin and had some unknown pancreatic surgery in 1968 most likely for phlegmon or some sort of pancreatic condition.
Aug_0	**Function**: nac.KeyboardAug() **Description**: Simulates typing errors by replacing a random character with an adjacent alphabet on the keyboard **Augmented**: He has also knderbone recent eye surgery of unknown krigin and had some unknown pxndreatix eurgeru in 1968 most lijely for phlegmon or some skrt of panxdeqtic condition.
Aug_1	**Function**: nac.RandomCharAug(action="insert") **Description**: Simulates typing errors by randomly inserting a character in a word **Augmented**: He has also kundcergone grecent eye surgery of unknown origin and had szome junkn%own pancreatic surgery in 1968 most hlikely for phrlegmion or some soqrt of panclreat&i^c condition.
Aug_2	**Function**: nac.RandomCharAug(action="substitute") **Description**: Simulates typing errors by randomly substituting a character in a word **Augmented**: He has also undergone recens eye subgbry of unknown origin and had pome unknown ea)creatic surgery in w968 most likelx for p%legaon or some soat of pancreatic condition.
Aug_3	**Function**: nac.RandomCharAug(action="swap") **Description**: Simulates typing errors by randomly swapping two characters in a word **Augmented**: He has also nudergoen recetn eye ursgery of unknown origin and had some nunkown apncraetci surgery in 1968 most likely for phelgomn or some sort of apnrcetaic contdiion.
Aug_4	**Function**: nac.RandomCharAug(action="delete") **Description**: Simulates typing errors by randomly deleting a character in a word **Augmented**: He has aso undrone recnt eye surgery of unknown rigin and had some unknown ancratc surgery in 1968 most liely for phlegmon or some sort of pacreai cdition.
Aug_5	**Function**: naw.RandomWordAug(action="swap") **Description**: Simulates typing errors by randomly deleting a character in a word **Augmented**: He has also undergone recent eye of surgery unknown origin and had unknown some surgery in pancreatic most 1968 likely phlegmon for or sort some pancreatic of condition.

Augmenter	NLPAug Function, Description, and Example
Aug_6	**Function**: naw.RandomWordAug('delete') **Description**: Simulates typing errors by randomly deleting a word **Augmented**: Has also undergone recent surgery of unknown had some unknown pancreatic surgery in 1968 likely for phlegmon or some of condition.
Aug_7	**Function**: naw.RandomWordAug(action='crop') **Description**: Simulates typing errors by randomly deleting a set of consecutive words **Augmented**: Has unknown origin and had some unknown pancreatic surgery in 1968 most likely for phlegmon or some sort of pancreatic condition.
Aug_8	**Function**: naw.SynonymAug(aug_src='wordnet') **Description**: Generates new text by replacing words with their WordNet's synonyms. Another synonym source is 'ppdb'. **Augmented**: He has besides undergone recent eye or of unnamed origin and get some unknown pancreatic surgery in 1968 most probable for phlegmon or some variety of pancreatic condition.
Aug_9	**Function**: naw.AntonymAug() **Description**: Generates new text by replacing words with their antonyms **Augmented**: He lack also undergone recent eye surgery of known origin and refuse some known pancreatic surgery in 1968 least improbable for phlegmon or some sort of pancreatic condition.
Aug_10	**Function**: naw.SplitAug() **Description**: Generates new text by splitting a random word into two parts **Augmented**: He has also undergone recent eye surgery of unkno wn or igin and had so me unknown panc reatic surgery in 19 68 m ost likely for ph legmon or some sort of pancre atic condition.
Aug_11	**Function**: naw.SpellingAug() **Description**: Simulates spelling errors using pre-defined spelling mistake dictionary **Augmented**: He has also undergone recent eye cirurgery of unknown origin abd had some uknown pancreatic surgery in 1968 mora likely fur phlegmon of somg sort of pancreatic condithion.
Aug_12	**Function**: naw.ContextualWordEmbsAug(model_path='bert-base-uncased', action="insert") **Description**: Generates new text by inserting words based on the word embeddings from the 'bert-base-uncased' model **Augmented**: he has apparently also undergone recent eye surgery of almost unknown origin and previously had had some unknown pancreatic bypass surgery in january 1968 most likely for phlegmon or perhaps some such sort because of pancreatic condition.

Augmenter	NLPAug Function, Description, and Example
Aug_13	**Function**: naw.ContextualWordEmbsAug(model_path='bert-base-uncased', action="substitute") **Description**: Generates new text by replacing words based on the word embeddings from the 'bert-base-uncased' model **Augmented**: he has also had an eye surgery of recent origin or has some unknown pancreatic surgery from 1968 most likely for eye or some muscle or skin condition.
Aug_14	**Function**: naw.ContextualWordEmbsAug(model_path='distilbert-base-uncased', action="insert") **Description**: Generates new text by inserting words based on the word embeddings from the 'distilbert-base-uncased' model **Augmented**: he has also undergone recent surgical eye patch surgery of unknown cosmetic origin and had some relatively unknown cosmetic pancreatic surgery in late 1968 most likely likely for phlegmon or some unidentified sort indicative of chronic pancreatic condition.
Aug_15	**Function**: naw.ContextualWordEmbsAug(model_path='distilbert-base-uncased', action="substitute") **Description**: Generates new text by replacing words based on the word embeddings from the 'distilbert-base-uncased' model **Augmented**: he has also received recent eye surgery for unknown causes but have already undergone pancreatic surgery in 1968 most likely for phlegmon or some kind of liver condition.
Aug_16	**Function**: naw.ContextualWordEmbsAug(model_path='roberta-base', action="insert") **Description**: Generates new text by inserting words based on the word embeddings from the 'roberta-base' model **Augmented**: He also has apparently also undergone recent eye surgery of an unknown origin recently and had some previously unknown pancreatic surgery performed in 1968 most likely looking for phlegmon or some similar sort form of pancreatic condition.
Aug_17	**Function**: naw.ContextualWordEmbsAug(model_path='roberta-base', action="substitute") **Description**: Generates new text by replacing words based on the word embeddings from the 'roberta-base' model **Augmented**: He has since undergone multiple eye surgery of unknown origin and had a unknown pancreatic surgery in 2013 most likely involving phlegmon or some sort of pancreatic condition.

Augmenter	NLPAug Function, Description, and Example
Aug_18	**Function**: naw.TfIdfAug(action='insert') **Description**: Generates new text by inserting words based on TF-IDF statistics **Augmented**: The he has also undergone reathertage recent nerve eye surgery of laparoscopically unknown origin and had control some unknown pancreatic surgery in 1968 retention most treadmill likely hedges for phlegmon or some sort of pancreatic condition
Aug_19	**Function**: naw.TfIdfAug(action='substitute') **Description**: Generates new text by replacing words based on TF-IDF statistics **Augmented**: He has also undergone supplement eye surgery of donalson origin and jcorn cheairs unknown pancreatic surgery in 1968 waynetpidsran likely for phlegmon or montesonnixcape sort of hemianopsia dice

Table 5. **Data Augmentation Methods (at the end of the manuscript)**

For each strategy, all augmenters were run on the same 10 training sets and tested on the same 10 test sets except for the two TF-IDF augmenters (Augmenter 18 and Augmenter 19). The TF-IDF augmenters were run separately from the other 18 augmenters because an additional model had to be applied on the training set to calculate the TF-IDF scores for each word before applying the augmenters. These scores were then passed to the augmenter to determine which words to insert or replace.

For tasks where the number of original summary notes in the minority class was less than the number of augmented samples required, the notes were selected for augmentation with replacement, otherwise without replacement. We set the augmentation probability to be 0.3, and the minimum number of characters to be augmented as 200, and the maximum 2,000 for character level augmenters and a minimum of 100 and a maximum of 1,000 for word level augmenters. If the length of the note is between the minimum and the maximum, the augmentation probability (0.3) was applied, otherwise, the minimum (for short notes) or maximum (for long notes) were used instead.

Results

Figure 1 shows the average AUC-ROC of each augmenter in each strategy for Group 1 (Hypertriglyceridemia and Venous Insufficiency). These AUC-ROCs are compared against that of the same Transformer Encoder model without data augmentation which is represented by the straight black line in the figure. All augmenters except for Aug_6 and Aug_7 improved the model performance in terms of AUC-ROC in all four strategies for Hypertriglyceridemia. Strategy G1_50 improved the model performance for most augmenters than other strategies. For Venous Insufficiency, all augmenters except for Augmenters 0–2 and Augmenter 4 improved the model performance in most strategies. The improvement in AUC-ROC from strategy G1_20 was the smallest for most augmenters compared to other strategies.

Figure 2 shows the average F1 Score of each augmenter in each strategy for Group 1. For Hypertriglyceridemia, the average F1 scores from the first six augmenters and Augmenter 10 in all strategies were consistently higher than that of No Augmentation. With the rest of the augmenters, strategies G1_20 and G1_30 had higher F1 scores more often than the other two strategies. For Venous Insufficiency, all augmenters had higher F1 Scores than No Augmentation. Augmenters 3 and 10 in strategy G1_50 improved the model performance the most.

Since there were eight tasks in Group 2 which were too many to be reasonably plotted in the same figure, we divided them into two parts. Part 1 contains the first four tasks (Asthma, Gout, OSA, and PVD), and

Part 2 contains the rest (Gallstones, OA, GERD, Depression). Figures 3 and 4 show the Average AUC-ROC of each augmenter in each strategy for Part 1 and Part 2 of Group 2, respectively. The results show a similar pattern across all 8 tasks with peaks at Augmenters 3, 8, 10, and 11, and with Augmenter 10 in strategy G2_50 producing the largest improvement. Augmenters 9 and 11 did not form a peak but they substantially improved the AUC-ROC in most strategies for all but OSA and PVD. Augmenter 19 in strategy G2_50 noticeably improved the AUC-ROC for all but OSA. For all the peaking augmenters, strategy G2_50 improved the AUC-ROC the most. The largest improvement was 0.13 by Augmenter 10 in strategy G2_50 for Gallstones. For OSA, PVD, Gallstones and OA, nearly all non-peaking augmenters actually decreased the AUC-ROC.

Figures 5 and 6 show the Average F1 Score of each augmenter in each strategy for Part 1 and Part 2 of Group 2, respectively. The results for all eight tasks share a similar pattern that is also roughly similar to that of the AUC-ROC results. The peaks are also presented at Augmenters 3, 8, 10, and 11 in strategy G2_50. Strategy G2_50 was consistently outperforming the other strategies. The largest improvement was an impressive 0.34 which was produced by Augmenter 10 in strategy G2_50 for Gallstones. For Gout, OSA, PVD, and OA, the non-peaking augmenters decreased the F1 Score. For OA, even the outstanding Augmenter 10 decreased the F1 Score in strategy G2_20.

Figures 7 and 8 show the Average AUC-ROC and Average F1 score of each augmenter in each strategy for Group 3 (Obesity, CHF, Hypercholesterolemia, and CAD), respectively. The AUC-ROC and F1 Score results for Obesity continue to show the same pattern as those of Group 2 except that the best performing strategy for most augmenters is G3_R10. For the other three datasets, the results show a very different pattern where there are no outstanding peaks or valleys. All augmenters in all strategies improved both the AUC-ROC and the F1 Score for CHF and CAD. For Hypercholesterolemia, most augmenters in strategies G3_R10 and G3_R20 improved both AUC-ROC and F1 Score, and most augmenters in strategy G3_50 decreased the AUC-ROC and F1 Score. Overall, strategies G3_R10 and G3_R20 performed better than G3_50. The largest improvement in AUC-ROC is 0.048, and the largest improvement in F1 Score is 0.065, both were from Augmenter 10 in strategy G3_R10.

Figures 9 and 10 show the Average AUC-ROC and Average F1 score of each augmenter in each strategy for Group 4 (Diabetes and Hypertension), respectively. The pattern of AUC-ROC for Diabetes was very similar to that of F1 Score of the same task. The AUC-ROC and F1 Score results for Hypertension also

share the same pattern but are different from that of Diabetes. For Diabetes, there were three groups of peaks. One was formed by Augmenters 2–4, one by Augmenters 8–11, and one by Augmenters 16 and 17. There are improvements from most of the augmenters in the first two peaking groups, but the improvement margin is not large. The largest improvement in AUC-ROC was only 0.007, and the largest improvement in F1 score was merely 0.009. The third group of augmenters decreased the performance in most strategies. For Hypertension, the most prominent peaking group was formed by Augmenters 8–11 for both AUC-ROC and F1 Score. The largest improvement was 0.034 in AUC-ROC and 0.012 in F1 Score. Both were produced by Augmenter 10 in strategy G4_50, although the improvements by the same augmenter in strategies G4_R05 and G4_R10 came very close.

Figure 11 shows the overall effect of each of the augmenters on the model performance in terms of AUC-ROC and F1 Score in all tasks combined. The light blue bars represent the difference in AUC-ROC between models with augmentation and without augmentation, and the dark blue bars represent the difference in F1 Score between models with augmentation and without augmentation. As expected, Augmenter 10 showed the highest improvement in both model performance measurements, followed by Augmenter 3, Augmenter 11, and then Augmenter 8. Augmenters 0, 1, 2, 4, 9, and 19 had a weak positive impact on the model performance measurements, and Augmenters 5 and 12–18 had an overall negative impact on the model performance measurements. The average improvement produced by Augmenter 10 was 0.04 in AUC-ROC, and 0.08 in F1 Score.

In addition to their effects on the model performance measurements, we also investigated the running time of each augmenter. Figure 12 shows the average running time per 1000 notes (specific to the discharge summary notes in this particular dataset) in ascending order. Augmenters 18, 19, 9, and 8 stood out as the top 4 slowest augmenters, with Augmenter 18 being the slowest (104.17 seconds per 1,000 notes). These four augmenters ran significantly slower than the others because Augmenters 18 and 19 are TF-IDF augmenters, and a model was run to calculate the TF-IDF scores for each word in the training set before the augmentation took place. Augmenters 8 and 9 are the Synonym and Antonym augmenters which needed to access predefined dictionaries to fetch the synonyms or antonyms for augmentation. The running time per 1000 notes for the rest of the augmenters are all less than 50 seconds. Our best performing augmenter Aug_10 only took 21.56 seconds on average to augment 1000 notes.

From the above figures, it is safe to say that Augmenter 10 (the Splitting Augmenter) is the best performing augmenter for these hospital discharge summaries. In order to investigate the effect of the different strategies on the model performance, we chose to control the variations from the different augmenters by using the AUC-ROC and F1 Score results from Augmenter 10 only. Since the strategies were different for each group, we plotted the graphs separately for each group as shown in Fig. 13. The light blue bars represent the difference in F1 Scores between Augmenter 10 and No Augmentation, and the dark blue bars represent the difference in AUC-ROC between Augmenter 10 and No Augmentation. For Group 2 and Group 3, the strategies G1_50 and G2_50 yielded the best performance. Strategies G1_50 and G2_50 were basically the same strategy since they both represented augmenting the minority class

until balanced (50% disease prevalence). The strategies that augment the least number of notes in each of these two groups improved the model performance the least. For Group 3, strategies G3_R10 improved the model performance slightly more than G3_R20, and both produced noticeably better performance than G3_50. Group 4 shows a similar pattern except that the improvement in F1 Score is much more than that in AUC-ROC, and that G4_R10 performed slightly better than G4_R05.

Discussion

Our results show that not all augmenters enhanced model performance. In fact, some even had an adverse effect. This is reasonable since noise is often task- and domain-specific, and different augmenters introduce different noises. For example, if the data were obtained by scanning hard copy documents such as old books or newspapers using scanning devices, then the OCR (Optical Character Recognition) augmenter may be helpful since it mimics the errors produced while recognizing the characters (i.e. mistaken the number “0” as the alphabet “o”, etc.). With most medical notes that are typed in by doctors or nurses, typing errors are common noise. Typing errors can be caused by accidentally hitting the key adjacent to the correct one on the keyboard or a random wrong key, missing out a character or a word here and there, accidently swapping two characters or words, common spelling errors, or short handing, etc.. Augmenters 0–4 investigated in this study were designed to simulate typing errors at the character level by randomly inserting, deleting, replacing, or swapping characters. The rest of the augmenters were intended to either mimic typing errors at the word level by deleting, inserting, cropping, or splitting words randomly or create variations of the original texts by inserting words randomly or replacing words with synonyms or antonyms based on predefined synonym or antonym dictionaries. Another word level augmentation method was the TF-IDF method (Augmenters 18 and 19 in this study) where a model was run before augmentation to calculate the TF-IDF scores of the words in the training set. Words with high TF-IDF scores were considered important to the meaning of the texts. Low TF-IDF score words had a higher probability to be replaced (or inserted) by other low score words to retain the important information and create variations without changing the meaning of the original texts [21].

Most text augmentation methods were at the character or word level. Some techniques were at the sentence or the entire document level. For example, the Back Translation method translates the entire original text to a different language and then translates it back to the original language, which has the effect of saying the same thing but in a different way, or the same thing narrated by different people [22, 23]. We did an experiment with Back Translation on our data for one task, and then we had to leave it out of our study because it was too time consuming. On average, it took 6.8 minutes to augment a single summary note or 112 hours to augment 1000 notes. This was a major disadvantage of the Back Translation method for lengthy texts since it ran a translation model twice for each augmentation. In addition, it doesn’t handle text sequences that are longer than 1024 words. Since our notes are lengthy (with a mean of 1170 words and a maximum of 4280 words), many were longer than the maximum allowed by the Back Translation method, we augmented the notes sentence by sentence and then concatenated the augmented sentences back together which made it even slower.

Another augmenter that may be helpful with medical notes is the Reserved Word augmenter. This method requires a predefined reserved words list. Each list is a list of words with similar meaning. A word is randomly selected from the predefined reserved words list to replace a word of the similar meaning in the original text. For example, if we have observed that in our medical notes, the word patient often appears in four forms (patient, patients, pt, pts), then these four words can form a list and if any of the four words in the list is detected in the text, it will be randomly replaced with one of the other three. However, too few lists would not help because there won't be enough variations in the augmented text, while too many such lists would be time consuming to create and may require the help of a domain expert.

There are many other text data augmentation methods such as the Abstractive Summarization Augmenter (summarizes the original text), the Sentence Shuffling method (shuffles the order of sentences in a paragraph), and the Instance Crossover method (swap half of a text with half of another text of the same class) [24]. Each augmentation technique has its own merits and disadvantages and therefore may be suitable for some types of texts but not for others. Using the wrong augmentation methods can hurt model performance instead.

Just like different augmentation techniques can be mixed and applied to a single image in computer vision, text data augmentation methods can be used simultaneously on the same text. The NLPAug package provides a Sequential pipeline and a Sometimes pipeline that allow multiple augmenters to be applied to the same text. The Sequential pipeline applies all augmenters provided to the pipeline sequentially, and the Sometimes pipeline randomly selects a subset of the augmenters to apply. This method helps diversify the noise or variation of the texts. However, using too many augmenters simultaneously may result in texts that are too different from the original texts and thus may lead to decreased model performance.

There are plenty of publications on text data augmentation methods, but none, to our knowledge, on augmentation strategies in terms of the amount of augmentation in different class imbalance scenarios for text classification tasks. Our results show that augmenting the minority class until the two classes are balanced generated the best model performance for highly imbalanced tasks with prevalence less than 20%, and augmenting until balanced and then raising both sides by 10% produced the most improvement for tasks with prevalence between 40% and 80%. Although Group 1 suffers the most from the class imbalance issue, it is not the one that benefited the most from data augmentation. Instead, the eight tasks in Group 2 enjoyed the most improvement in both AUC-ROC and F1 Score. One likely explanation is that there are too few samples in the training set in the minority class (only 50 samples for Hypertriglyceridemia, and 62 for Venous Insufficiency), and even with the variations introduced by data augmentation methods, they were still very limited and not capable of representing the variations in the test set well. The results also show that, for roughly balanced tasks, augmenting the minority group until balanced and then raising both classes to 10% enhanced model performance further and by a larger margin than raising both classes to 20%. This suggests that balanced classification tasks may also benefit from data augmentation but the more is not necessarily the merrier.

Due to the time constraint and computational limitations, we cannot exhaust all augmentation methods available and all augmentation strategies possible. However, based on our results so far, applying the top four augmenters (Augmenters 10, 3, 11, and 8) using the Sometimes pipeline may enhance model performance further. In addition, experimenting strategies that keep augmenting both classes after reaching class balance for Groups 1 and 2, and raising both sides by 20% and 30% or more after balanced for Groups 3 and 4 is also worth exploring. In addition, there are many other factors that can affect model performance besides model choice, model hyperparameter settings, augmentation methods and strategy choice. For each augmentation method, there are also hyperparameters to tune. For example, the minimum and maximum number of characters (or of words depending on the method) allowed to be augmented in a single text sequence, the augmentation probability (percentage of characters or words allowed to be augmented), customized stop words list, and customized tokenizers may also have an influence on model performance.

Conclusion

As our results show, different text data augmentation methods have different effects on model performance. Some enhance model performance, and others yield no improvement or even have an adverse impact. For our hospital discharge summaries, the Splitting Augmenter (Augmenter 10), the Swapping Character Augmenter (Augmenter 3), the Spelling Augmenter (Augmenter 11), and the Synonym Augmenter (Augmenter 8) were the top four augmenters that improved model performance. For the tasks where the disease prevalence was between 10% and 20%, the strategy that augments the minority class until balanced improved model performance by the largest margin. For those where the disease prevalence was between 40% and 60%, the best performing strategy was the one that augments the minority class to balanced and then augments both classes by an additional 10%. The largest improvement was 0.13 in F1 score and an impressive 0.34 in AUC-ROC, and both were produced by the Splitting Augmenter in the strategy that augments the minority class until balanced for Gallstones.

In summary, with the right choice of augmentation methods which can be a single augmenter or a mix of different augmenters, the model performance in terms of AUC-ROC and F1 Score can be substantially improved. For the highly imbalanced tasks, the strategy that augments the minority class until balanced yielded the most improvement. For other tasks, the strategy that keeps augmenting both classes by an additional 10% after reaching balanced enhanced model performance further.

Abbreviations

NLP
Natural Language Processing
BERT
Bi-directional Encoder Representations from Transformers
AUC-ROC
Area Under the Curve of the Receiver Operating Characteristic

TF-IDF
Term Frequency – Inverse Document Frequency
OCR
Optical Character Recognition
OSA
Obstructive Sleep Apnea
PVD
Peripheral Vascular Disease
OA
Osteo Arthritis
GERD
Gastroesophageal Reflux Disease
CHF
Congestive Heart Failure
CAD
Coronary Artery Disease

Declarations

Ethics approval and consent to participate: This study used de-identified data from the Harvard Website and followed all ethical considerations instituted by the providers of the data. No participants are involved in this study.

Consent for Publication: Not applicable.

Availability of data and materials: The data that support the findings in this study are derived from datasets publicly available (after simple steps for approval) on https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/. The code for this study is available on https://github.com/Hanna520/Text_Data_Augmentation/.

Competing interest: The authors declare no competing interest regarding this article.

Funding: This study received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author contributions: HL aggregated, managed and quality checked the medical discharge summary notes; carried out the AI model fitting, summarized the results, and wrote sections of the manuscript. CR designed the study, interpreted the results, and provided guidance and insights. Both authors read and approved the final manuscript.

Acknowledgements: Not applicable.

References

1. Shaheen Z, Wohlgenannt G, Filtz E. Large scale legal text classification using transformer models. arXiv preprint arXiv:201012871. 2020.
2. Soyalp G, Alar A, Ozkanli K, Yildiz B. Improving Text Classification with Transformer. In: 2021 6th International Conference on Computer Science and Engineering (UBMK). 2021. p. 707–12.
3. Adhikari A, Ram A, Tang R, Lin J. Docbert. Bert for document classification. arXiv preprint arXiv:190408398. 2019.
4. González-Carvajal S, Garrido-Merchán EC. Comparing BERT against traditional machine learning text classification. arXiv preprint arXiv:200513012. 2020.
5. Miotto R, Percha BL, Glicksberg BS, Lee HC, Cruz L, Dudley JT, et al. Identifying acute low back pain episodes in primary care practice from clinical notes: Observational study. JMIR Med Inform. 2020;8(2):e16878.
6. Feder A, Vainstein D, Rosenfeld R, Hartman T, Hassidim A, Matias Y. Active deep learning to detect demographic traits in free-form clinical notes. J Biomed Inform. 2020;107:103436.
7. Gunjal H, Patel P, Thaker K, Nagrecha A, Mohammed S, Marchawala A. Text Summarization and Classification of Clinical Discharge Summaries using Deep Learning. 2020.
8. Ye J, Yao L, Shen J, Janarthanam R, Luo Y. Predicting mortality in critically ill patients with diabetes using machine learning and clinical notes. BMC Med Inform Decis Mak. 2020;20(11):1–7.
9. Lu H, Ehwerhemuepha L, Rakovski C. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. BMC Med Res Methodol. 2022;22(1):1–12.
10. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data. 2019;6(1):1–48.
11. Chlap P, Min H, Vandenberg N, Dowling J, Holloway L, Haworth A. A review of medical image data augmentation techniques for deep learning applications. J Med Imaging Radiat Oncol. 2021;65(5):545–63.
12. Shorten C, Khoshgoftaar TM, Furht B. Text data augmentation for deep learning. J Big Data. 2021;8(1):1–34.
13. Liu P, Wang X, Xiang C, Meng W. A survey of text data augmentation. In: 2020 International Conference on Computer Communication and Network Security (CCNS). 2020. p. 191–5.
14. Bayer M, Kaufhold MA, Reuter C. A survey on data augmentation for text classification. ACM Comput Surv. 2021.
15. Harvard University i2b2 Obesity. Contest 2008 Data [Internet]. [cited 2022 Apr 28]. Available from: https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/.
16. Solt I, Tikk D, Gál V, Kardkovács ZT. Semantic classification of diseases in discharge summaries using a context-aware rule-based classifier. J Am Med Inform Assoc. 2009;16(4):580–4.

17. Yang H, Spasic I, Keane JA, Nenadic G. A text mining approach to the prediction of disease status from clinical discharge summaries. J Am Med Inform Assoc. 2009;16(4):596–600.
18. Savova G, Clark C, Zheng J, Cohen KB, Murphy S, Wellner B, et al. The Mayo/MITRE system for discovery of obesity and its comorbidities. In: Proceedings of the i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data. 2008.
19. Ware H, Mullett CJ, Jagannathan V. Natural language processing framework to assess clinical conditions. J Am Med Inform Assoc. 2009;16(4):585–9.
20. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Advances in neural information processing systems. 2017. p. 5998–6008.
21. Xie Q, Dai Z, Hovy E, Luong T, Le Q. Unsupervised data augmentation for consistency training. Adv Neural Inf Process Syst. 2020;33:6256–68.
22. Sennrich R, Haddow B, Birch A. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:151106709. 2015.
23. Edunov S, Ott M, Auli M, Grangier D. Understanding back-translation at scale. arXiv preprint arXiv:180809381. 2018.
24. Luque FM. Atalaya at TASS 2019: Data augmentation and robust embeddings for sentiment analysis. arXiv preprint arXiv:190911241. 2019.

Figures

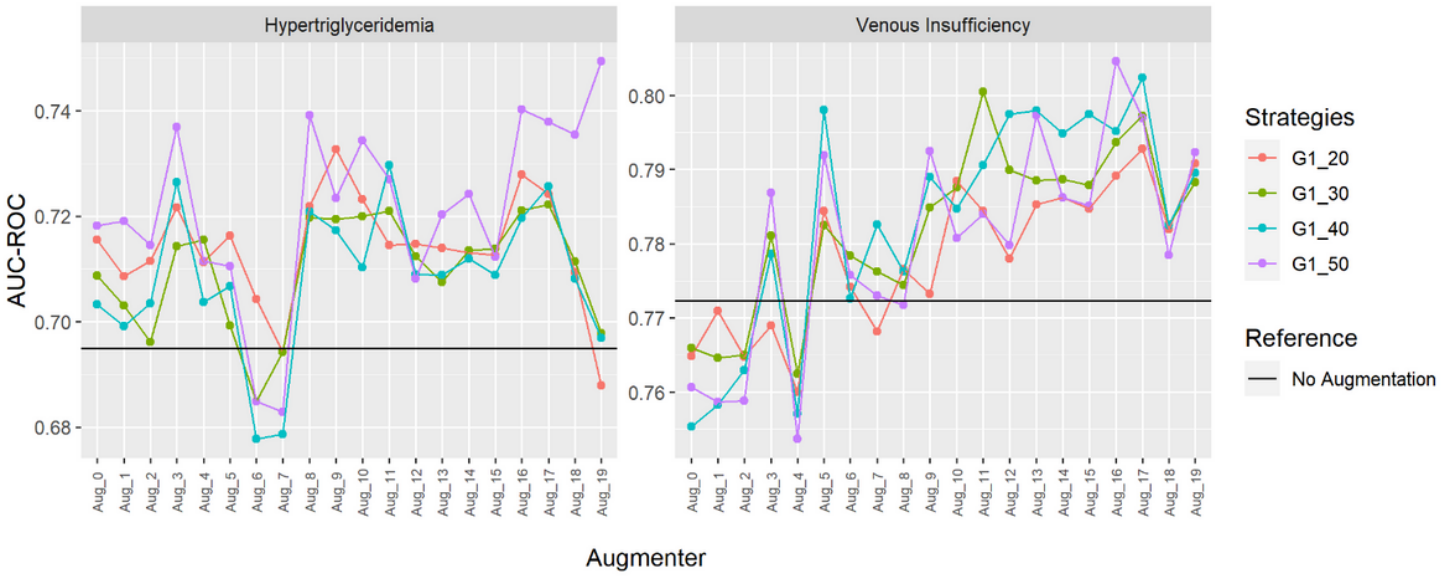

Figure 1

AUC-ROC for Group 1 (Disease Prevalence < 10%)

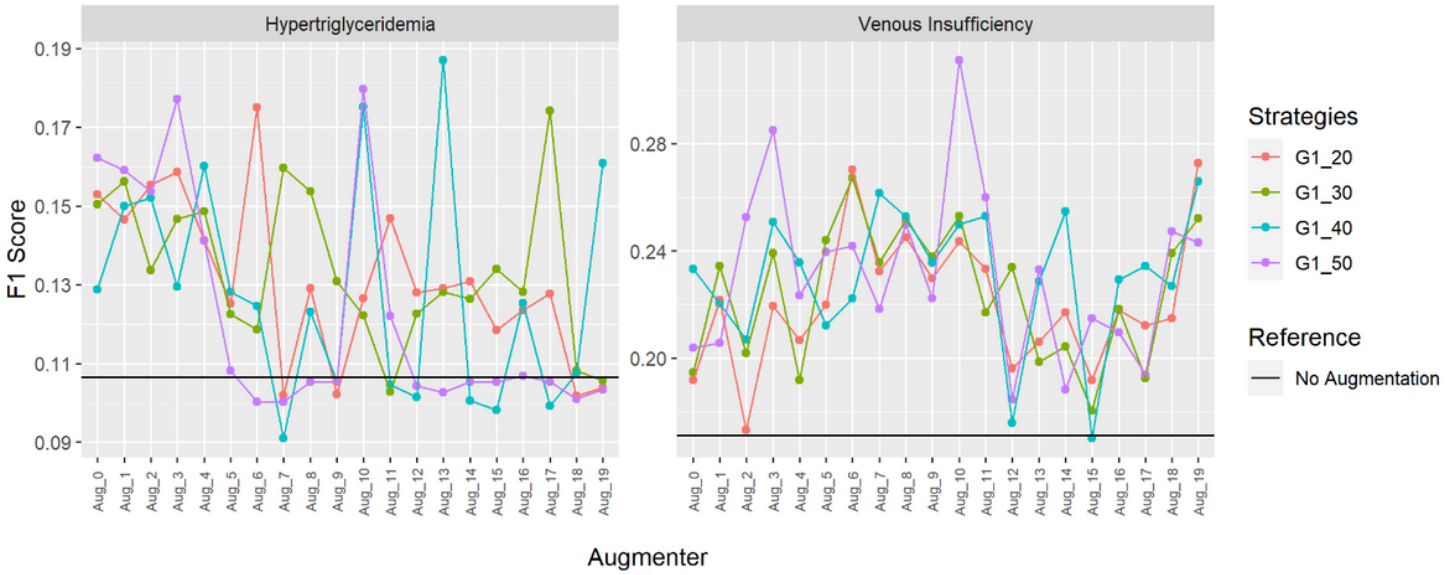

Figure 2

F1 Score for Group 1 (Disease Prevalence < 10%)

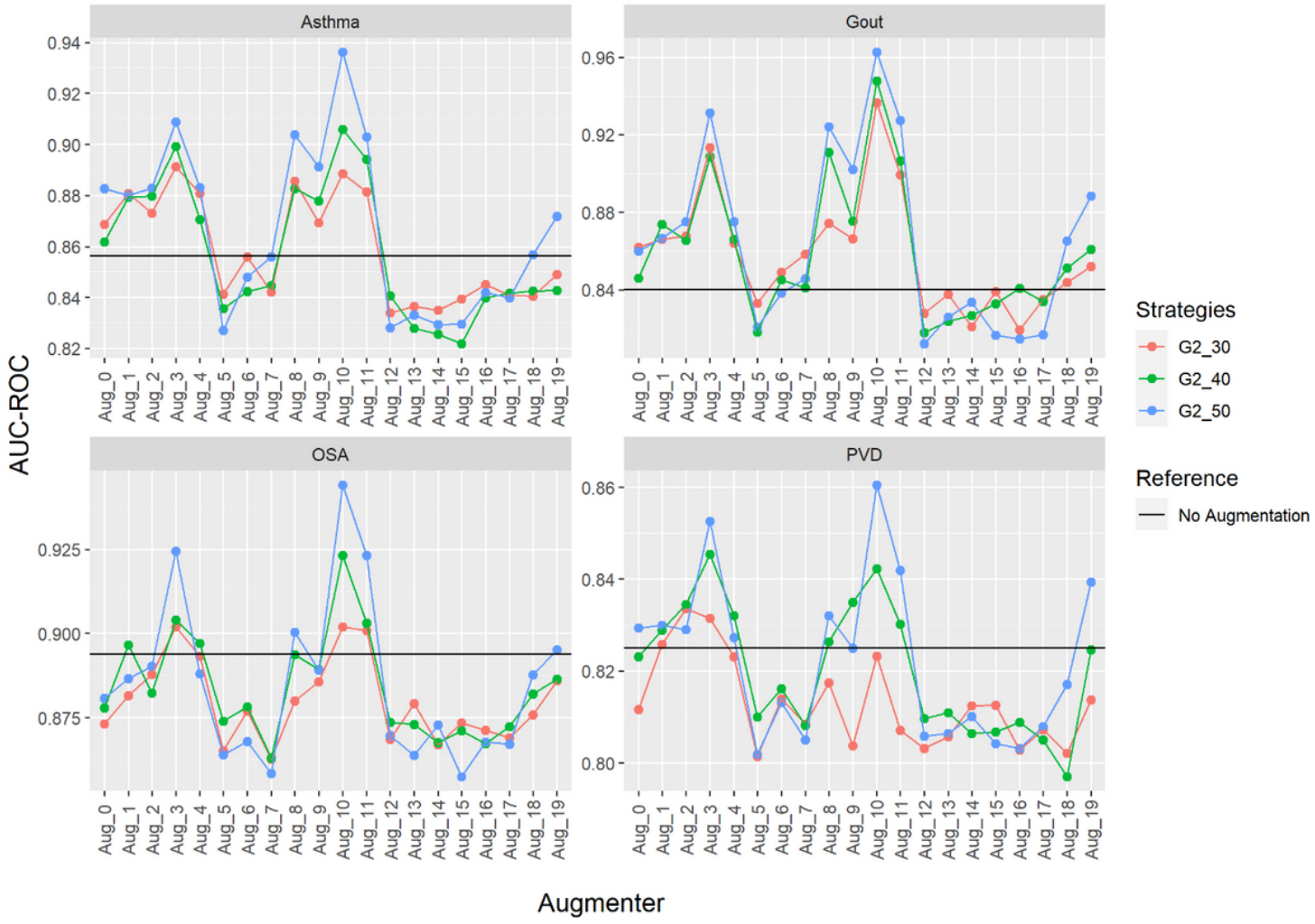

Figure 3

AUC-ROC for Group 2 (Disease Prevalence between 10% - 20%) (Part 1)

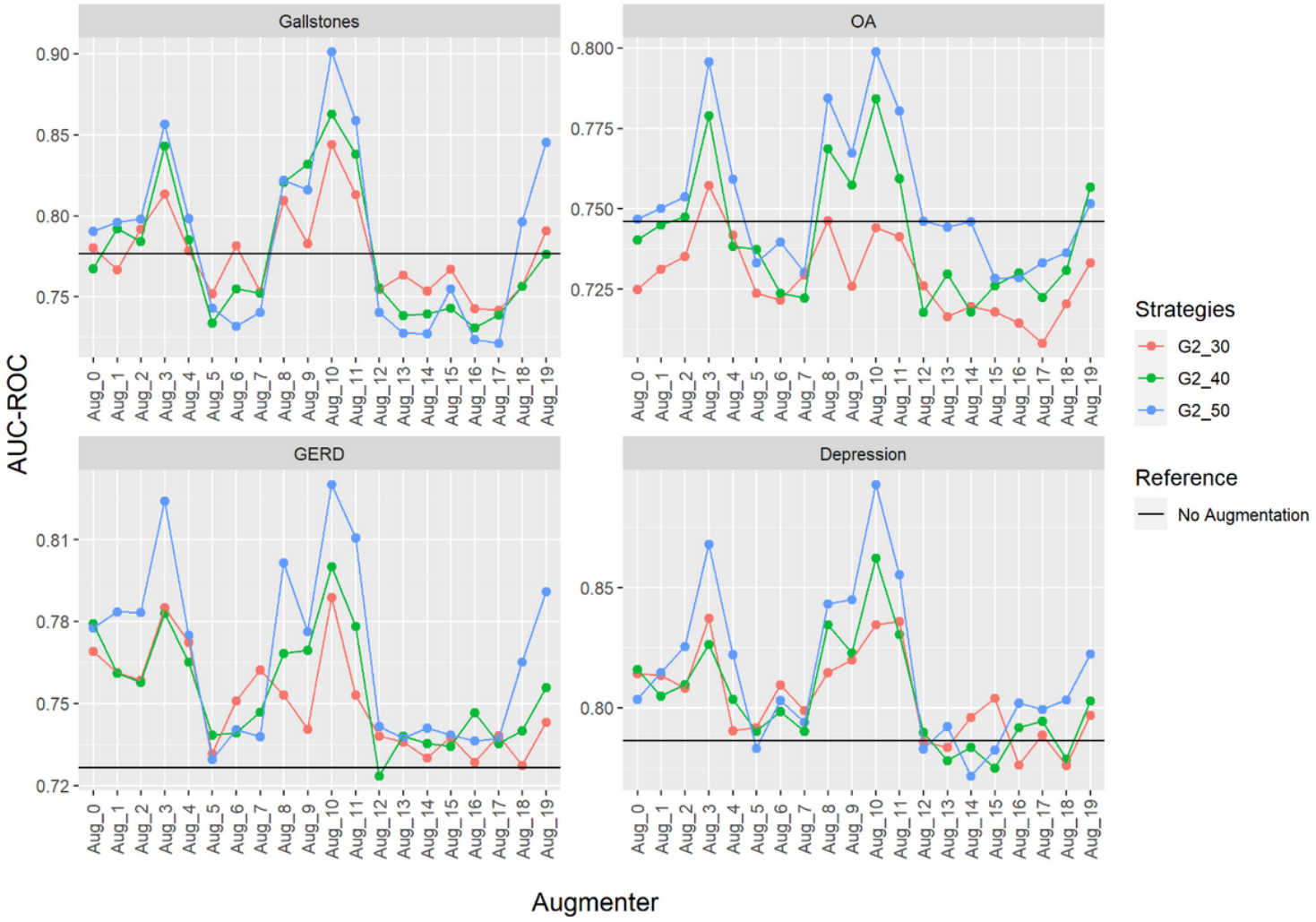

Figure 4

AUC-ROC for Group 2 (Disease Prevalence between 10% - 20%) (Part 2)

Figure 5

F1 Score for Group 2 (Disease Prevalence between 10% - 20%) (Part 1)

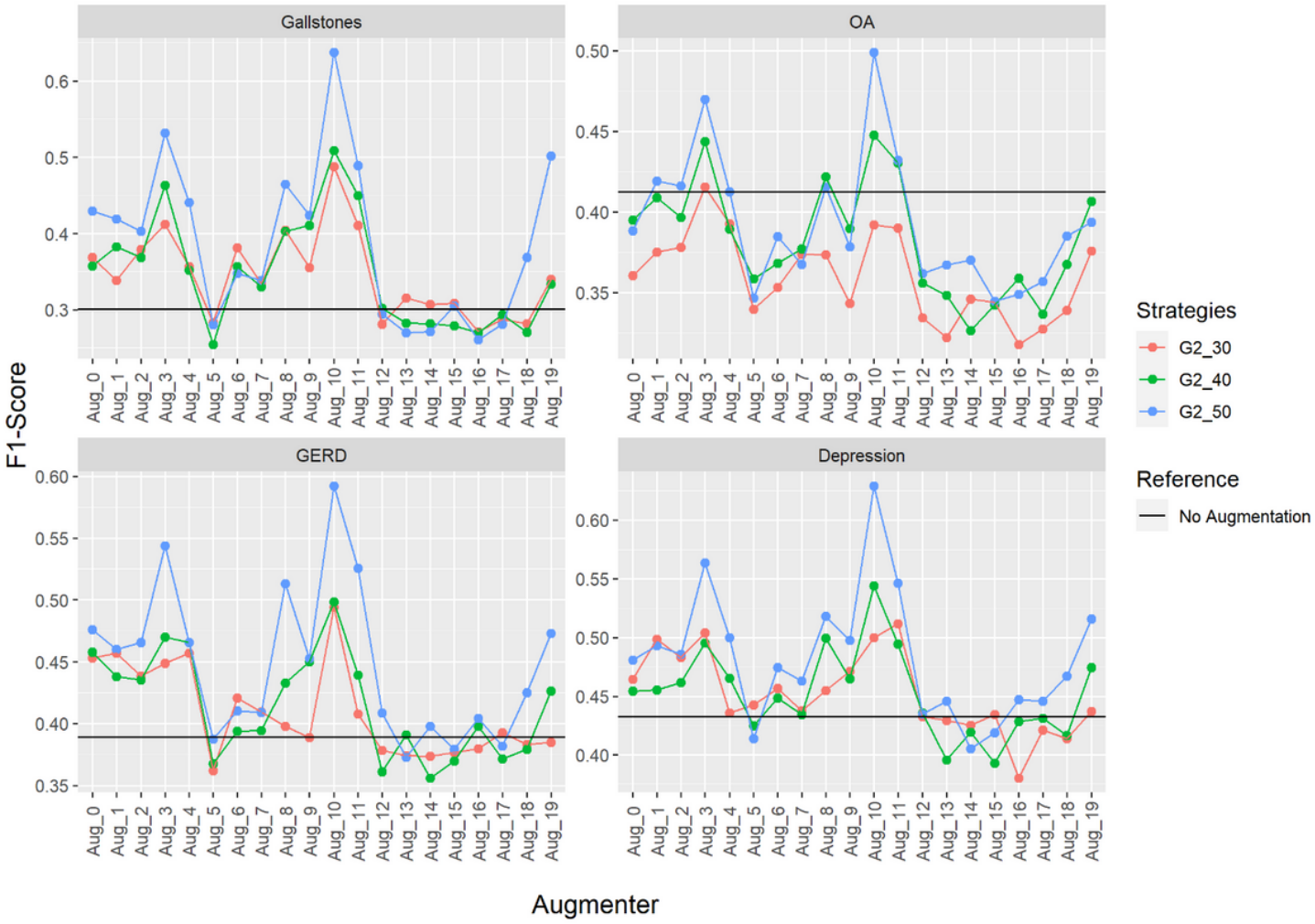

Figure 6

F1 Score for Group 2 (Disease Prevalence between 10% - 20%) (Part 2)

Figure 7

AUC-ROC for Group 3 (Disease Prevalence between 40% and 50%)

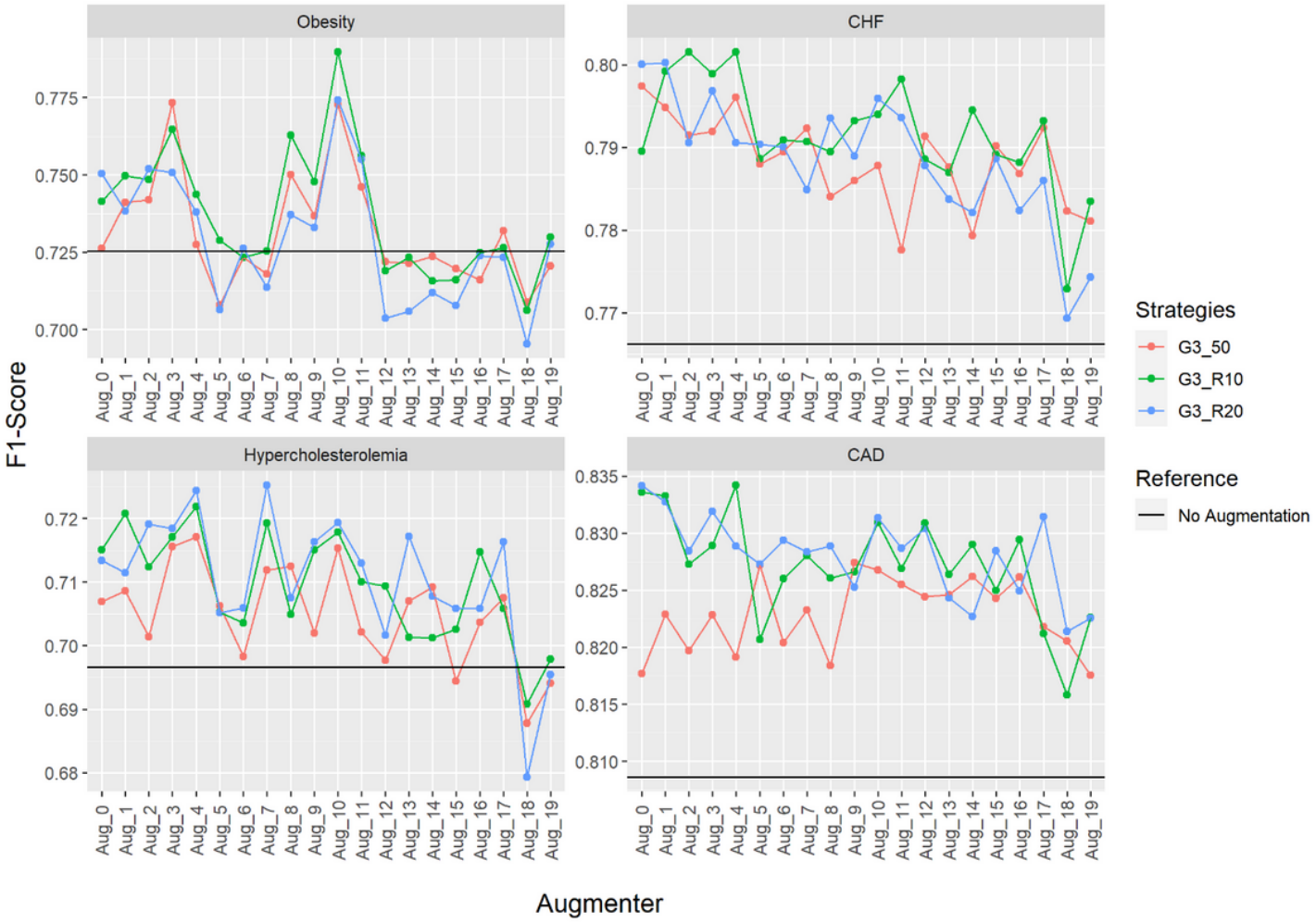

Figure 8

F1 Score for Group 3 (Disease Prevalence between 40% and 50%)

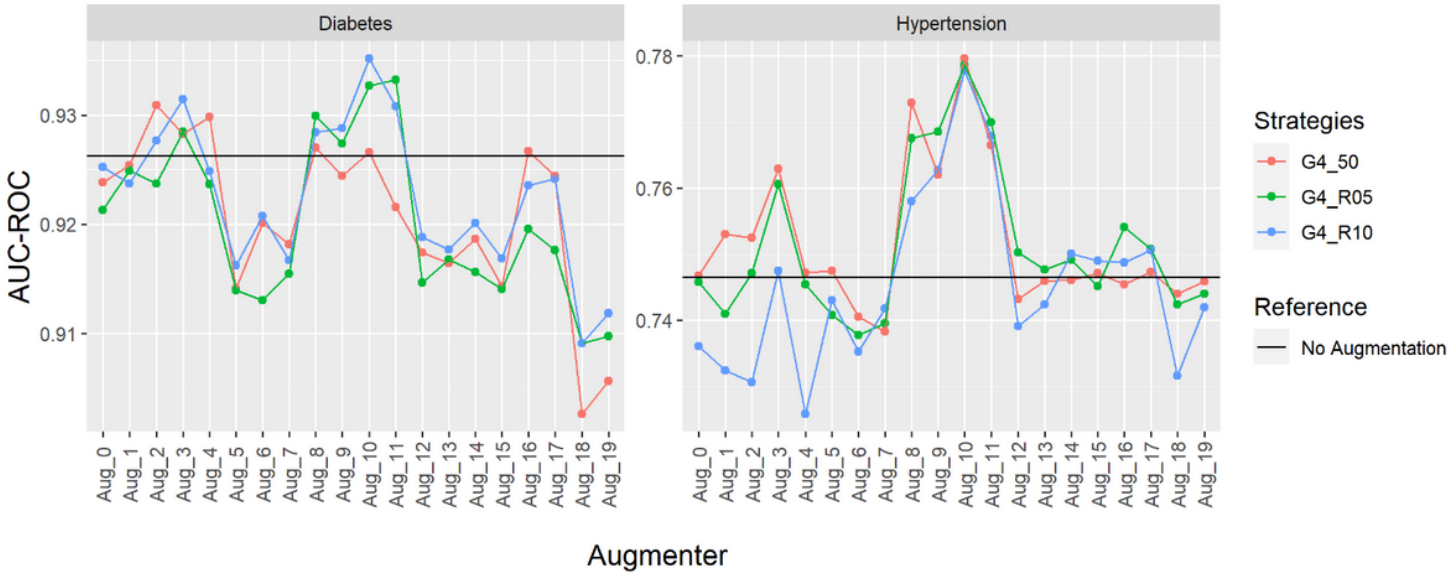

Figure 9

AUC-ROC for Group 4 (Disease Prevalence between 60% and 80%)

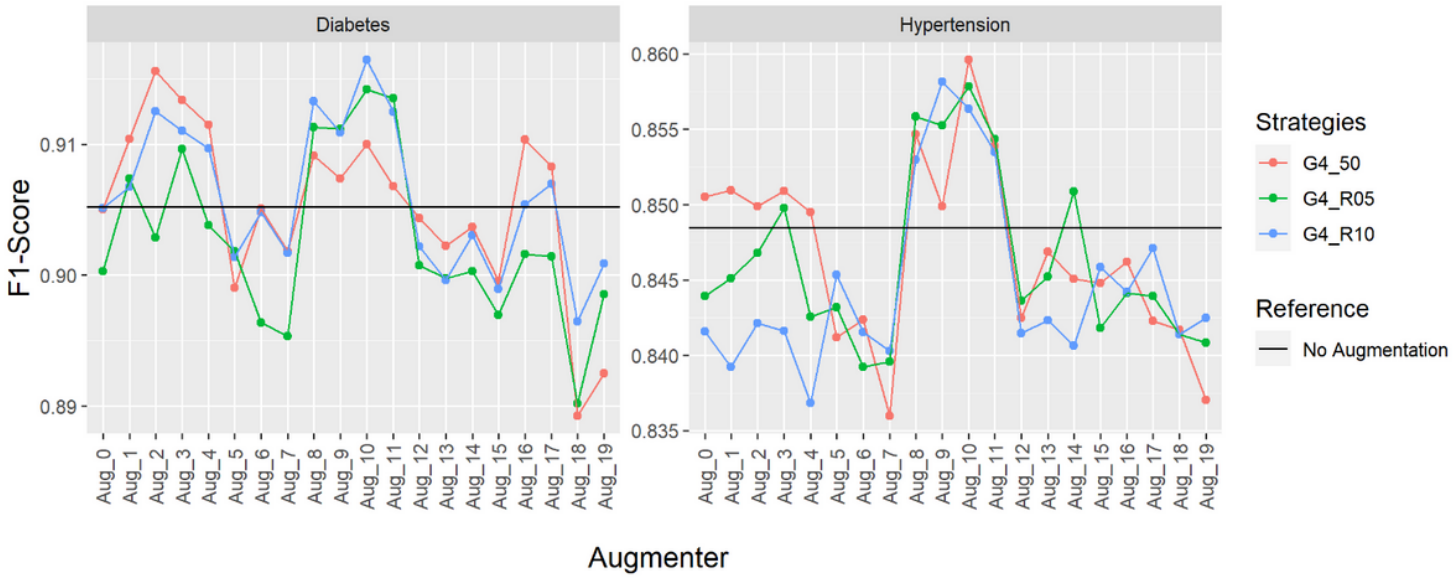

Figure 10

F1 Score for Group 4 (Disease Prevalence between 60% and 80%)

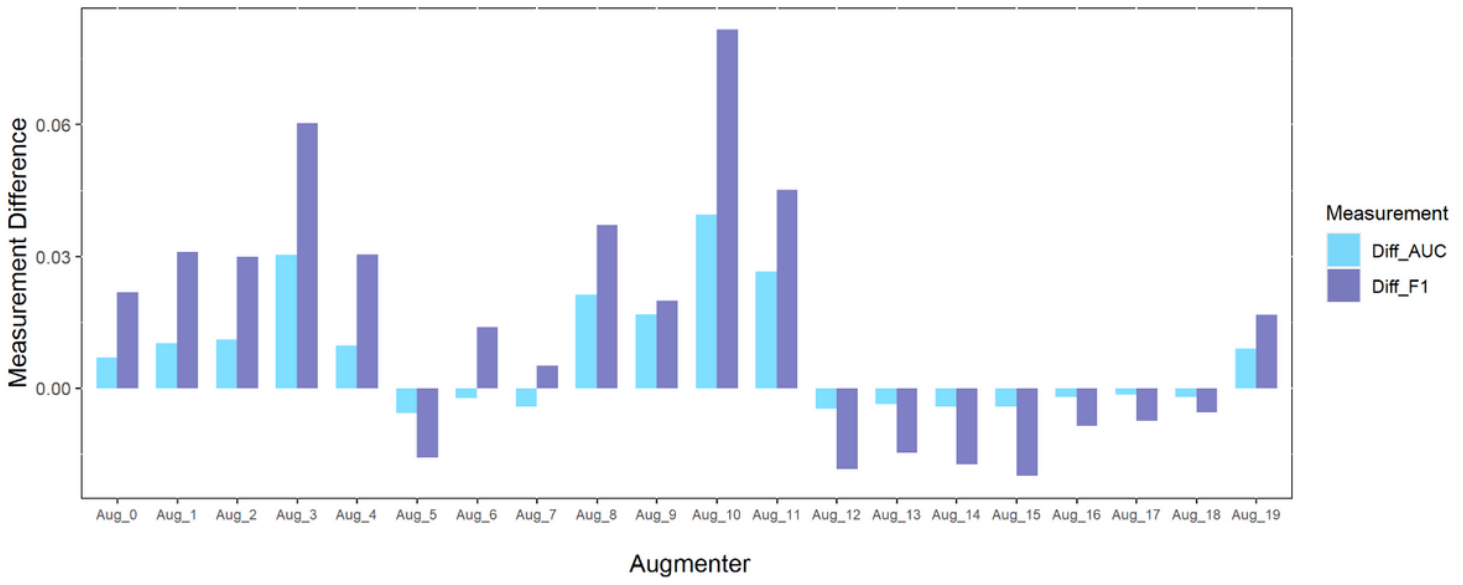

Figure 11

Performance Difference from No Augmentation

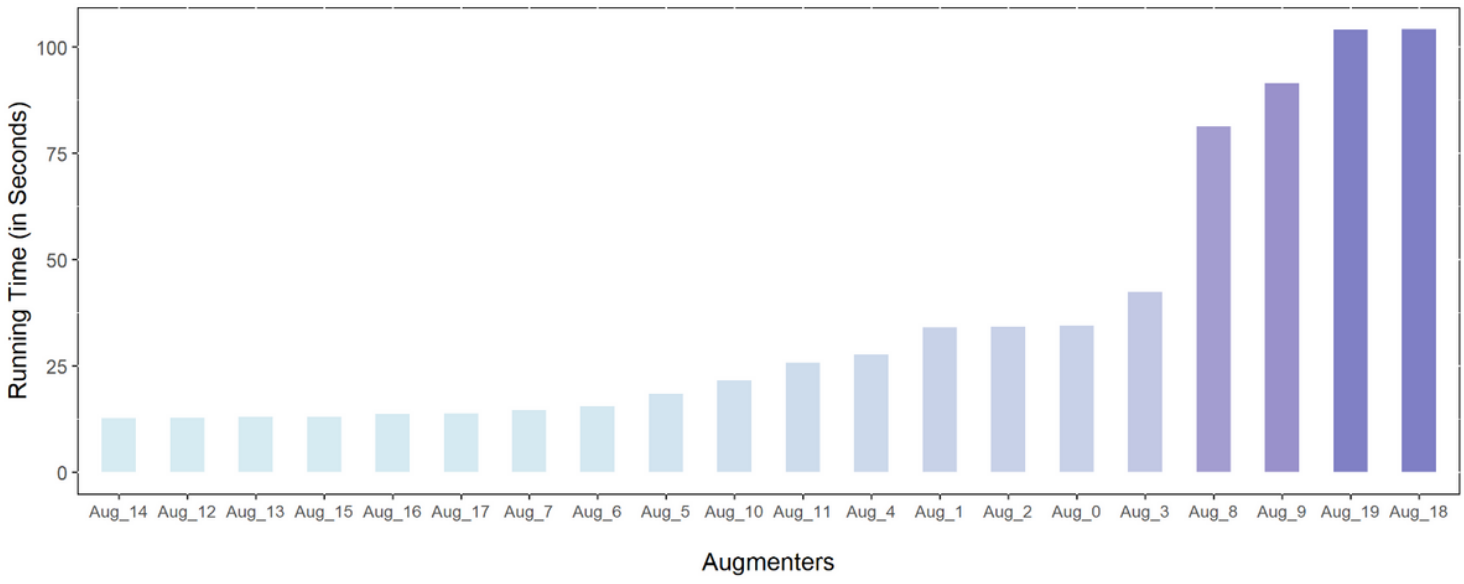

Figure 12

Average Running Time per 1000 Notes by Augmenter

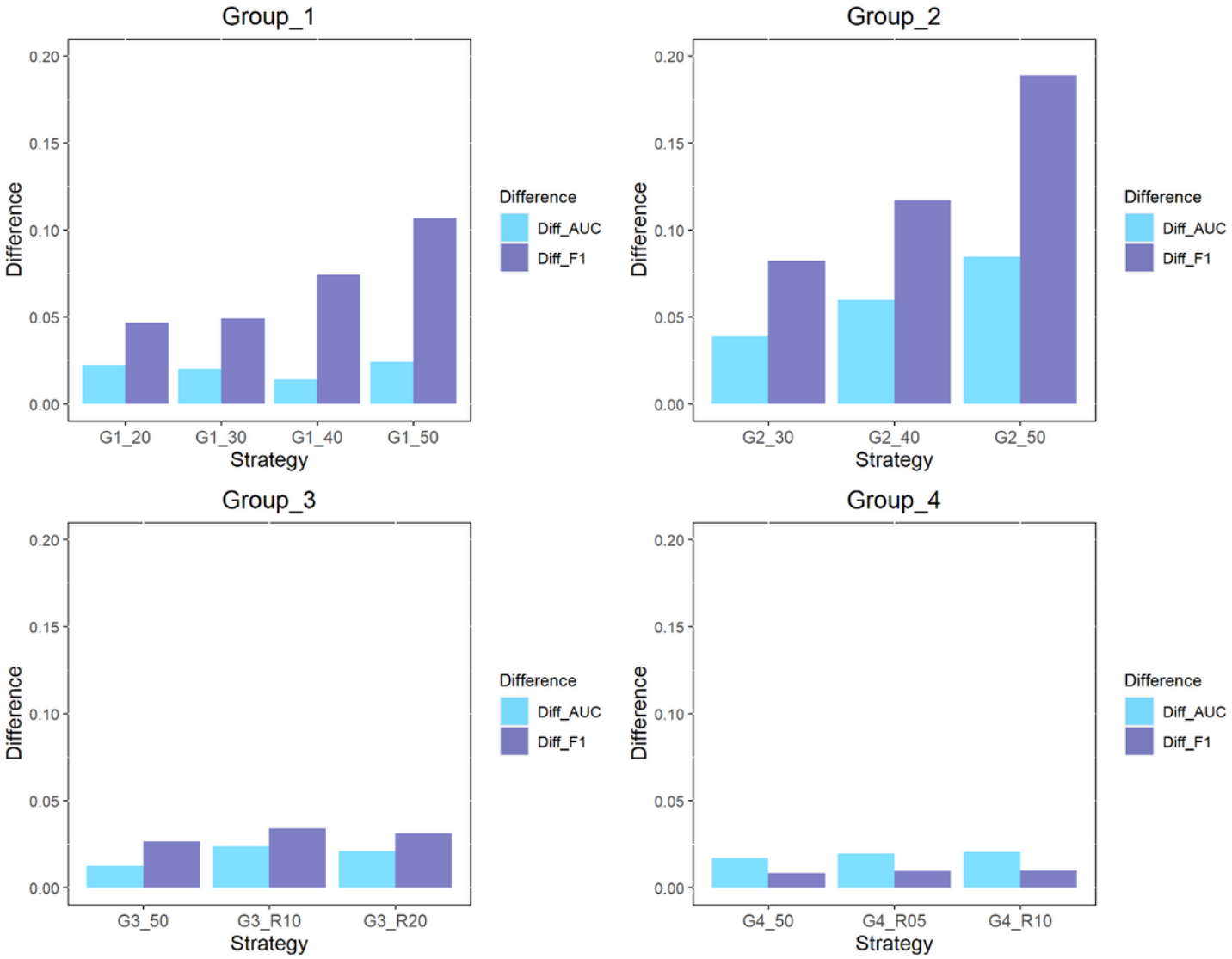

Figure 13

Performance Measurement Difference between Augmenter 10 and No Augmentation