Alain Tamazian, Deepak Jayan, Sofian Ghazali

DSCI550 Project Final Report: PAS Classification

Problem Definition: Nursing distribution in hospitals is determined in an inefficient, subjective manner. By implementing a combination of clustering analysis and multiclass classification for the patients' required acuity, we create a date-driven PAS. Our model will classify each patient's visit as one of the following levels of needed care: "minimum", "intermediate", "semi-intensive", and "intensive".

Description of Background: With increasing digitization in the healthcare industry, there is potential for the improvement of existing logistical procedures within hospitals. Currently, hospitals utilize a program called Patient Acuity System (PAS) that manages hospital resources, especially with regards to nurse workload (HCPro). This proposed system determines the appropriate amount of time and manpower nurses need to allot each patient, thereby, improving patient care through efficient resource management. However, there is still no specific, accurate method to create a PAS in the healthcare sector, and the existing ones are often poorly implemented. Therefore, many determinations regarding nurse resource allocation are still made on a subjective basis by individuals – such as the head nurse (HCPro). Therefore, our acuity classifier system will automatically advise the nursing department about the acuity for each patient, so they can better allocate their resources and enhance patient care.

Description of Dataset: This 2012 dataset is provided and copyrighted by Practice Fusion – a large electronic health record (EHR) company (Practice Fusion). The dataset can still be found online and – under the "free use" copyright exception – it can be utilized for the purpose of an (unpublished) academic project such as this. Before any preprocessing and feature deletion, the raw dataset we found is composed of 5 CSV files: "diagnosis", "medication", "patient", "physician_specialty", and "transcript". The datasets have 9948 rows of patient data. Together, there are 2494 attribute columns, of which many are binary, dummy variables. After a cursory elimination of variables that were clearly not relevant to our target question, our actual dataset was made up of 61 features from "diagnosis", "transcript" and "patient" files. The "diagnosis" file contains all the ICD9 diagnosis codes for different diseases in each patient's medical history – along with visit frequency columns. The "'transcript" file contains quantitative data of each patients' various health measures like the min. and max. of their blood pressure, respiratory rate, and temperature to name a few. We added the "age" feature to our dataset from the "patient" file. The values in this merged dataset are numeric, which are compatible with most algorithms and (as we hoped) were helpful in gauging the seriousness of a patient's condition.

Description of method used: The project was implemented through Python – with software libraries "Pandas" and "Numpy" being used for general dataset manipulations, "Matplotlib" and "Seaborn" for visualization, and mainly "Sklearn" for analytical algorithms. We started our data analysis by conducting data exploration for our 61 variables. Through univariate analysis, we discovered that – though many of the values are 1s and 0s – the 25 features in the "diagnosis" file were not categorical dummies as suspected. We verified that the features are discrete, quantitative variables giving the counts for the number of times each patient was previously diagnosed with a particular ailment. This meant we could use these features for clustering as well. During data exploration we didn't find any missing values. However, the IQR method showed that nearly all the features had many "outliers" – some in the 1000s. By visualizing the distribution, through the "seaborn" boxplot function, it was clear that the extreme cases were due to the high skew of most features. We used Pearson-Fischer calculation to quantify the skews. Only 12 out of the 61 variables are approximately symmetrical, with 39 high right skews and 3 high left skews. For bivariate analysis, the "sklearn.feature_selection" "r_regression" method gave us the Pearson correlation coefficient; 15 of the features are highly correlated (i.e. |r| > 0.7).

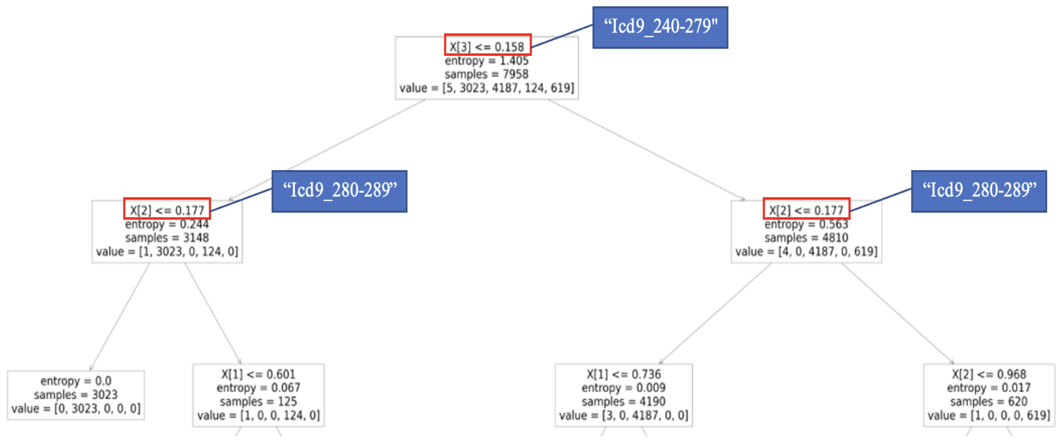
Although many outliers were identified, all available information about the data source and the values suggested they are all natural outliers; count data (like our ICD9 attributes) are notoriously right skewed. Since the outliers aren't errors, it would be improper to apply imputation or deletion. Instead, for outlier treatment, we used "Numpy" to apply variable transformation and bring the distributions as close to normal as possible, thus reducing the impact of the outliers. We applied log transformation to correct the distribution of highly, right skewed attributes. Since count data contains 0s, we implemented a common variation – log(x+1). For less right skewed ones, we used square or cube root transformations depending on their level of skew. To transform the left skews closer to normal, we used varying degrees of exponentiation. The new Fisher-Pearson coefficients showed that the transformations were effective at normalizing many of the extreme skews. This process yielded 29 approximately normal distributions and 24 high skews (which had significant improvements as well). Our next step was normalization – for which we used the "sklearn" "MinMaxScaler" function. Since clustering is conducted using (e.g.) Euclidean distance, the algorithms are sensitive to differences in feature magnitude. So, it is crucial to scale the variables within the same range to give equal weights when clustering and classifying. Before normalization, we first did a random 80:20 train:test split to prevent data leak from "unseen" test data. We normalized the training set and applied the fitted scaler on the test set. Clustering algorithms are susceptible to the curse of dimensionality – making feature selection especially significant. Since variables with extreme skews have relatively lower predictive value, we did feature deletion per that criterion. There are 10 features with |skew| > 2. We eliminated 8 of them and kept 2, for which there is strong domain indication of relevance for acuity – Icd9_140-239 for neoplasms and Icd9_280-289 for blood diseases (Kramer). Then, we identified and deleted 15 redundant features with |correlation| > 0.7.

We used medical domain insights to eliminate 23 more. An example of this is that, within the medical community, systolic is known to be a better indicator of serious, imminent health risk than diastolic BP (Harvard Health). After similar acuity considerations, we had 15 features: Age, RespiratoryRate_Min, Temperature_Std, SystolicBP_Max, DiastolicBP_Max, SystolicBP_Std, AcuteFreq, DiagnosisCount, Icd9_240-279, Icd9_290-319, Icd9_390-459, Icd9_460-519, Icd9_580-629, Icd9_140-239, and Icd9_280-289. To explore these ICD9 codes, refer to: "https://www.aapc.com/codes/icd9-codes-range/".

As mentioned, since clustering is unsupervised and our classifier will be relying on "artificial" labels, it would have been especially ineffective to simply create a single model and rely on it as our final one. On the other hand, simply comparing all possible feature-combination models would be highly inefficient. Instead, we found it most appropriate to create and evaluate 10 models based on different combinations of approximately 4-6 features (from our pool of 15), which we believed would be effective for clustering the data into groups of acuity – i.e. our target classes. Despite all attempts to normalize distributions, a significant portion of our features were still skewed and had many outliers. So, we used the DBSCAN clustering algorithm, known for being robust to outliers. The DBSCAN() function is available through the "sklearn.cluster" library. DBSCAN can't handle high dimensions very well, which is why we used around 5 features. The "minPts" parameter (which sets the minimum cluster size) is 2 times the number of model features – a common rule of thumb (Lin). The Ɛ parameter determines the min. distance between data points necessary to cluster them together. The technique we used to find the proper Ɛ, was by visually approximating the point of max. curvature of our dataset's k-distance graph – which here is between 0.24 and 0.3 (Lin). For each of the 10 DBSCAN models we tested, we selected Ɛ within this range that formed four clusters. The two models that didn't satisfy this condition were eliminated. Next, for each model, we calculated an internal evaluation of the consistency within each models' clusters. After eliminating the noise, we used the "silhouette_score" method from "sklearn" to measure cluster cohesion and separation. However, the silhouette coefficient – like other established cluster evaluation algorithms – is better with globular clusters, and may underestimate with DBSCAN.

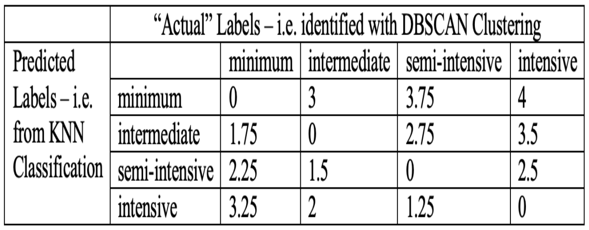
When running the DBSCAN algorithm and assigning the cluster labels to each instance, we had to consider whether to split the dataset into train/test sets after clustering or before. The "after" approach is simpler, since the instances – that will make up the classification's test set – are included in the clustering process and are automatically labeled by the DBSCAN algorithm. With the "before" approach, since the DBSCAN algorithm doesn't have a built-in "predict()" method, we had to manually find which of the four clusters each test instance would fit into best – using the Euclidean metric from the "SciPy" library. This is done using the epsilon distance that we set for each model; per DBSCAN, if a data point is a distance of ε (or closer) from a cluster's core point, the instance is at least a boundary point of that cluster. If it is too far from all of the clusters, the algorithm treats it as an outlier (i.e. "noise"). With this reasoning, we assigned each test instance the class label of the cluster whose core point it is closest to. For all 10 clustering and (subsequent) classification models, we tested and compared both options based on the same training set. Often the classifier using test data that was split after the clustering yielded a higher test accuracy. However, for our final models, we still chose to use test data that was split before the clustering process. The purpose of the test set is for it to show how our classifier handles new/unknown data. We believe that without splitting the dataset before clustering, there could be data leakage, since the test data is considered when creating the clusters and subsequently assigning all labels. Thus, we can't fully trust our results, since evaluating our classifier on such a test set may end up overestimating our model's generalization ability. Our results will already be biased since we don't use ground truths, but rather labels based on our own (limited) domain knowledge. So, we decided that splitting before clustering would yield more reliable/safe classification metrics. Though, without real ground truths to compare to, we can't be certain which approach is more accurate.

At the end of our clustering stage, we conducted a domain-focused analysis of the clusters and identified whether they are representative of acuity. An intuitive, efficient way we found to interpret the cluster meanings is through a decision tree – using binary binned, qualitative transformations of the clustering features (Liu). The tree classifier will be fitted, tested, and plotted on the training set the clustering is done on. Each tree class equals a cluster. By following the nodes and branches, we saw what combination of feature splits defined each cluster. Then it's just a matter of verifying whether those clusters' traits align with the concept of acuity. During our healthcare research, we found approximately 50% of admitted patients are low-acuity (Poon). We must take into account that for the four clusters to be representative of acuity classes, the two upper and lower levels must already be relatively balanced. To best describe this process, we can examine the sample output for clustering model 9 (shown in the figure below) – the labels for which become the stand-in of our optimal classifier's acuity classes.



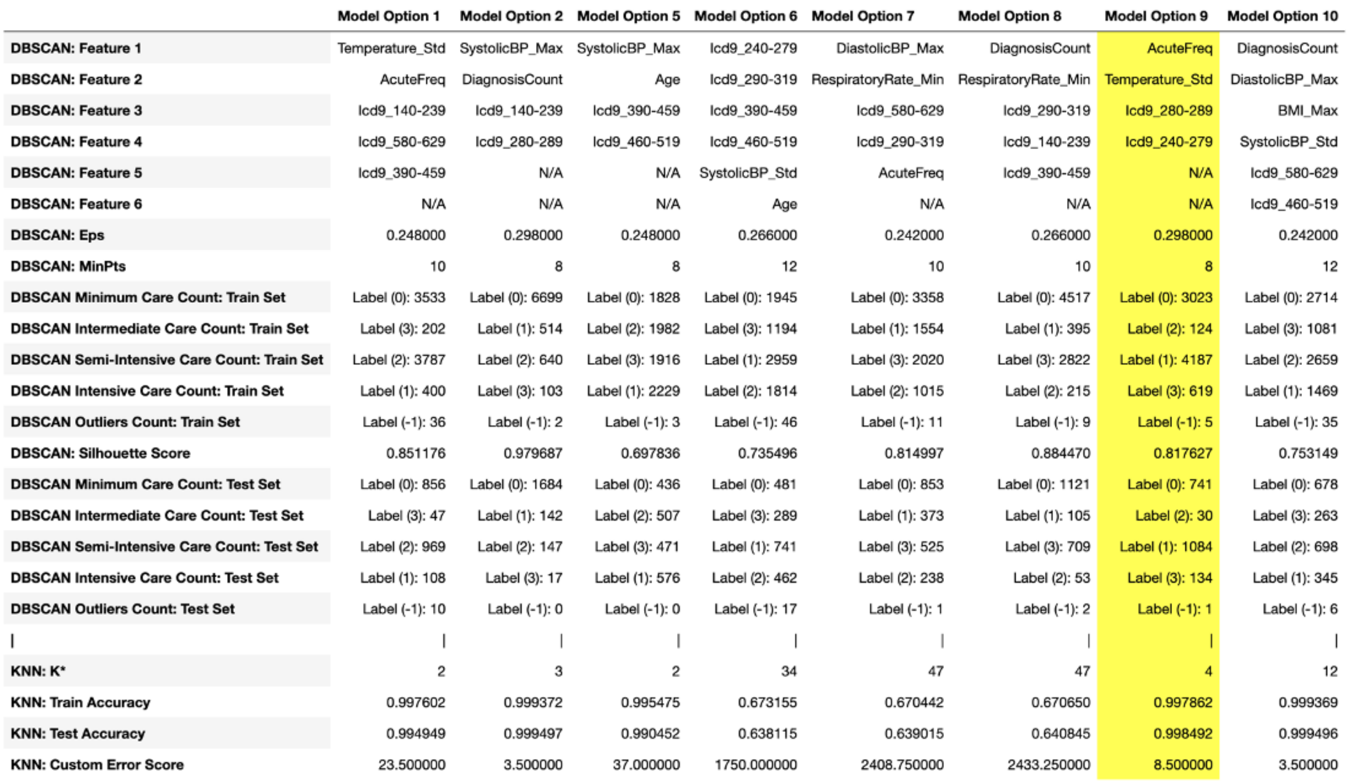
Using the “sklearn” module’s “DBSCAN” and “DecisionTreeClassifier” algorithms, we created and interpreted our model. Using hyperparameters of Ɛ = 0.298 and minPts = 8, DBSCAN created 4 clusters as needed. Our chosen features for clustering model 9 were: AcuteFreq, Temperature_Std, Icd9_280-289, and Icd9_240-279. The training error rate of the decision tree was 0, which means this model (i.e these features) accurately predicts the labels created by the clustering. The decision tree chose two key features that have the highest information gain measure, which would best split our data and give us insight into the cluster labels. The root node is based on a split of the “Icd9_240-279” diagnosis count, which is the medical code for “Endocrine, Nutritional and Metabolic Diseases, and Immunity Disorders” – which we will refer to as ENM/I disorders (for brevity) . The nodes of both first branches are based on the split of the “Icd9_280-289” feature for “Diseases of the Blood and Blood-forming Organs". DBSCAN designated only 5 samples as outliers which is a good sign. Since the values have been transformed, the quantitative variable’s split point doesn’t have any specific meaning. However, it’s comparative implication, for our cluster interpretation, is nonetheless clear. By inspecting the path to each leaf (while ignoring the DBSCAN “noise”), we learn the following. Class 1 (i.e. cluster 1) represents “minimum care”: derived from patients with lower cases of ENM/I disorders and lower cases of blood diseases. Class 2 (i.e. cluster 3) represents “intermediate care”: derived from patients with lower cases of ENM/I disorders but higher cases of blood diseases. Class 3 (i.e. cluster 2) represents “semi-intensive care”: derived from patients with higher cases of ENM/I disorders but lower cases of blood diseases. Class 4 (i.e. cluster 4) represents “intensive care”: derived from patients with higher cases of ENM/I disorders and higher cases of blood diseases. Despite being simplistic, these results support our domain knowledge regarding hospitals and patients' acuity. The two main determinants for the clustering include a range of diseases/disorders that are significant, broad indicators of patient health (Golden; Kramer). Also, we see that the 4 acuity classes have 3023, 124, 4187, and 619 samples – ordered from minimum to intensive. Though not equal, the distribution between the upper and lower acuity levels are nonetheless reasonable, and in line with what one could see in a hospital (Poon).

With each patient having an (artificial) acuity label, we could create our classifier. We use KNN algorithm from “sklearn.neighbors" since it’s effective with quantitative data. Feature selection was much simpler for the classifier since during the clustering process we already found the 15 most relevant attributes for patient care. While KNN is affected by the “curse of dimensionality” less than DBSCAN, 15 is still too many. So, we finalized our classifiers’ feature selection by considering which attributes were most significant for the 10 clustering models and will therefore be more valuable as predictors. Each of the 10 KNN models (with differing labels) are fitted on the train set with the following ten features: SystolicBP_Max, Temperature_Std, DiagnosisCount, Age, Icd9_460-519, Icd9_580-629, Icd9_140-239, Icd9_280-289, Icd9_240-279, Icd9_580-629, and Icd9_390-459. For each model we tuned the k hyperparameter to find the optimal k, within a range of 1-50. Finally, we calculated the train and test accuracy – as well as a testing misclassification value based on our own scoring metric. The table below shows the weighted value that we added for each of the 12 possible error types. The more serious the error (from a medical standpoint), the higher the number. As an example, to show the reasoning behind the weights, we consider that predicting that a patient requires minimum care when they actually require intensive care (4) is more detrimental to their health than the inverse (3.25). We compared the value from this scoring metric as well, when determining our "optimal" classifier (among our models). Ideally, a more accurate scoring formula would need to be constructed using specialized domain knowledge as well as insights from a math expert.



		"Actual" Labels – i.e. identified with DBSCAN Clustering			
Predicted Labels – i.e. from KNN Classification		minimum	intermediate	semi-intensive	intensive
	minimum	0	3	3.75	4
	intermediate	1.75	0	2.75	3.5
	semi-intensive	2.25	1.5	0	2.5
	intensive	3.25	2	1.25	0

Experiment: Analysis Results: All these steps are repeated approximately 10 times for the differing, 4-6, clustering features we chose. All of the relevant clustering and classification results (and parameters) are displayed in the table below. Two of the 10 models that we tested aren't included since they weren't able to meet our first condition of successfully creating four clusters with DBSCAN. Their clustering sets were: {SystolicBP_Max , DiastolicBP_Max, BMI_Max, Temperature_Std, Age, Icd9_460-519} and {SystolicBP_Std, BMI_Max, Icd9_140-239, Icd9_280-289, Icd9_240-279}.



	Model Option 1	Model Option 2	Model Option 5	Model Option 6	Model Option 7	Model Option 8	Model Option 9	Model Option 10
DBSCAN: Feature 1	Temperature_Std	SystolicBP_Max	SystolicBP_Max	Icd9_240-279	DiastolicBP_Max	DiagnosisCount	AcuteFreq	DiagnosisCount
DBSCAN: Feature 2	AcuteFreq	DiagnosisCount	Age	Icd9_290-319	RespiratoryRate_Min	RespiratoryRate_Min	Temperature_Std	DiastolicBP_Max
DBSCAN: Feature 3	Icd9_140-239	Icd9_140-239	Icd9_390-459	Icd9_390-459	Icd9_580-629	Icd9_290-319	Icd9_280-289	BMI_Max
DBSCAN: Feature 4	Icd9_580-629	Icd9_280-289	Icd9_460-519	Icd9_460-519	Icd9_290-319	Icd9_140-239	Icd9_240-279	SystolicBP_Std
DBSCAN: Feature 5	Icd9_390-459	N/A	N/A	SystolicBP_Std	AcuteFreq	Icd9_390-459	N/A	Icd9_580-629
DBSCAN: Feature 6	N/A	N/A	N/A	Age	N/A	N/A	N/A	Icd9_460-519
DBSCAN: Eps	0.248000	0.298000	0.248000	0.266000	0.242000	0.266000	0.298000	0.242000
DBSCAN: MinPts	10	8	8	12	10	10	8	12
DBSCAN Minimum Care Count: Train Set	Label (0): 3533	Label (0): 6699	Label (0): 1828	Label (0): 1945	Label (0): 3358	Label (0): 4517	Label (0): 3023	Label (0): 2714
DBSCAN Intermediate Care Count: Train Set	Label (3): 202	Label (1): 514	Label (2): 1982	Label (3): 1194	Label (1): 1554	Label (1): 395	Label (2): 124	Label (3): 1081
DBSCAN Semi-Intensive Care Count: Train Set	Label (2): 3787	Label (2): 640	Label (3): 1916	Label (1): 2959	Label (3): 2020	Label (3): 2822	Label (1): 4187	Label (2): 2659
DBSCAN Intensive Care Count: Train Set	Label (1): 400	Label (3): 103	Label (1): 2229	Label (2): 1814	Label (2): 1015	Label (2): 215	Label (3): 619	Label (1): 1469
DBSCAN Outliers Count: Train Set	Label (-1): 36	Label (-1): 2	Label (-1): 3	Label (-1): 46	Label (-1): 11	Label (-1): 9	Label (-1): 5	Label (-1): 35
DBSCAN: Silhouette Score	0.851176	0.979687	0.697836	0.735496	0.814997	0.884470	0.817627	0.753149
DBSCAN Minimum Care Count: Test Set	Label (0): 856	Label (0): 1684	Label (0): 436	Label (0): 481	Label (0): 853	Label (0): 1121	Label (0): 741	Label (0): 678
DBSCAN Intermediate Care Count: Test Set	Label (3): 47	Label (1): 142	Label (2): 507	Label (3): 289	Label (1): 373	Label (1): 105	Label (2): 30	Label (3): 263
DBSCAN Semi-Intensive Care Count: Test Set	Label (2): 969	Label (2): 147	Label (3): 471	Label (1): 741	Label (3): 525	Label (3): 709	Label (1): 1084	Label (2): 698
DBSCAN Intensive Care Count: Test Set	Label (1): 108	Label (3): 17	Label (1): 576	Label (2): 462	Label (2): 238	Label (2): 53	Label (3): 134	Label (1): 345
DBSCAN Outliers Count: Test Set	Label (-1): 10	Label (-1): 0	Label (-1): 0	Label (-1): 17	Label (-1): 1	Label (-1): 2	Label (-1): 1	Label (-1): 6
|	|	|	|	|	|	|	|	|
KNN: K*	2	3	2	34	47	47	4	12
KNN: Train Accuracy	0.997602	0.999372	0.995475	0.673155	0.670442	0.670650	0.997862	0.999369
KNN: Test Accuracy	0.994949	0.999497	0.990452	0.638115	0.639015	0.640845	0.998492	0.999496
KNN: Custom Error Score	23.500000	3.500000	37.000000	1750.000000	2408.750000	2433.250000	8.500000	3.500000

We analyzed and compared these results for the 8 models to find which cluster-then-label combination yielded the best KNN acuity classifier. Since we carefully selected the features (for all 8 DBSCAN models), the contextual evaluation of decision trees' paths indicated the clusters to be reasonable stand-ins for degrees of acuity. After matching each cluster to a care level, we then considered acuity distributions; relevant results are in the "Count" rows. Studies suggest that distribution of low and high acuity patients is approximately 50% (Poon). Since our dataset is from a random selection of EHR records, our two lower and upper care classes should be similarly distributed. So, we eliminated model 2 since its "Minimum Care" class made up the vast majority of our patients, which doesn't align with our domain knowledge. Then we looked at how much of our data was interpreted as noise by the DBSCAN algorithm (for each model). Since silhouette score and classifying is done after removing outliers, we consider this metric first. Models 1, 6, and 10 have a much higher number of "noise" – e.g. 63 vs 3. This is a crucial initial factor; it isn't as important for a model to have very high classification accuracy if many instances were first removed by DBSCAN for not fitting any cluster. We also theorized that a clustering which fails to label many patients, won't give a good representation of acuity. Next, since our goal is to find the best classifier, we compared the performance of the remaining KNN models (5, 7, 8, 9). Clearly classifiers 7 and 8 have a low accuracy and high weighted error score. Finally, with the various metrics – like DBSCAN silhouette score and KNN accuracy – we determined that Model Option 9 is quantitatively superior. Although we have good reasoning for choosing Model 9 as our "optimal" PAS classifier, given real ground truths the best model could differ. It should be noted that since some of the project algorithms have an element of randomness, there will be slight variation in these specific values when the code is rerun. Nonetheless, our general conclusions should remain unchanged.

Observation and Conclusion: In this data science experiment, we implemented an unsupervised cluster-then-label approach on an EHR patient dataset, to create a Patient Acuity System classifier. After data exploration, we applied different feature transformations to scale and balance the distribution of our attributes. Using domain-based reasoning along with our findings from univariate and bivariate analysis – specifically attribute skew and correlation – we reduced our 61-feature dataset to 15. Since our PAS needs to be trained with labeled patient instances, we used clustering to create and assign classes that we believe represent levels of acuity. Clustering isn't effective at high dimensions and we couldn't know what features (among our 15) would create the best clusters that are a reasonable stand-in for the four levels of patient care: "minimum", "intermediate", "semi-intensive", and "intensive". So, instead, we created 10 different clustering models with different ~5-feature combinations – to then pick the best. Since our data had many outliers, we used the DBSCAN algorithm for the clustering; for each of the 10 options, we conducted hyperparameter tuning for ε and trained our models. Then, using a decision tree, we got comprehensible descriptions of the four clusters' meanings. Interpreting their intrinsic structure and looking at the distribution among the "classes", we eliminated the models whose clusters did not align well with the concept of acuity. We also compared the number of instances DBSCAN discarded as "noise" and examined the clusters' silhouette coefficients. Finally, we trained and tested KNN classifiers (using the ten best predictors) to find the accuracy and weighted error score for the different models. We used these aforementioned criteria to compare the models and select our best PAS – KNN model 9.

Unfortunately, though our methodology is sensible, we can't be certain of our conclusion. The main reasons are the inherent challenges of unsupervised data and our lack of domain expertise. Without knowing the real acuity labels, the results given by our classifiers are based on simplifications of acuity, which overestimate model performance. Although we can't fully trust the classifiers' accuracy to be as good in a real-world scenario, we still expect it to be useful. When there are no ground truths available, having an actual domain expert is an especially crucial component of data science. Since unsupervised learning has a significant degree of subjectivity, the final decisions at various stages of the project – like feature selection, determining cluster meanings, and creating a weighted error score – were predicated on our understanding of the medical field. Ultimately, labeling from clustering can't replace real ground truth. We need to be cognisant that our results, though logical, are a biased representation based on our specific methodology and limited domain knowledge. For a more definitive evaluation of our models, we would need at least some ground truths to employ semi-supervised learning within our process.

References:

Golden, Sherita H., et al. "Prevalence and incidence of endocrine and metabolic disorders in the United States: a comprehensive review." *The Journal of Clinical Endocrinology & Metabolism* (2009).

Harvard Health. "Which Blood Pressure Number Is Important?" *Harvard Health*, 15 Feb. 2021, www.health.harvard.edu/staying-healthy/which-blood-pressure-number-is-important.

HCPro, Inc. "Patient Classification Systems to Coordinate Patient Care - Www.Hcpro.Com." *Hcpro*, 6 Oct. 2021, www.hcpro.com/NRS-279130-975/From-the-staff-development- bookshelf-Patient-classification-systems-to-coordinate-patient-care.html.

Kramer, Andreas H., and David A. Zygun. "Anemia and red blood cell transfusion in neurocritical care." *Critical care* 13.3 (2009): 1-22.

Lin, Ching-Heng, et al. "Applying density-based outlier identifications using multiple datasets for validation of stroke clinical outcomes." *International journal of medical informatics* 132 (2019).

Liu, Bing, Yiyuan Xia, and Philip S. Yu. "Clustering through decision tree construction." *Proceedings of the ninth international conference on Information and knowledge management*. 2000.

Poon, Sabrina J., Jeremiah D. Schuur, and Ateev Mehrotra. "Trends in visits to acute care venues for treatment of low-acuity conditions in the United States from 2008 to 2015." *JAMA internal medicine* 178.10 (2018): 1342-1349.

Practice Fusion. 2012 "Practice Fusion Analyze This! 2012 - Prediction Challenge | Kaggle." *Kaggle*. www.kaggle.com/c/pf2012/overview.