SIADS 696 Milestone II Project Report

Predicting Healthcare Costs by Clustering Current Procedural Terminology Codes in Medicare Claims Data

Adrienne Martz, Martin Ho

Github Project

Introduction

There is a healthcare crisis in the United States where resources are not being used efficiently and emphasis on patient outcomes has diminished considerably. But properly incentivizing those in the system, so that it evolves into one that accomplishes the intended goals can help to solve this problem. The Centers for Medicare and Medicaid Services (CMS) is currently developing alternative payment models to reward clinicians that more effectively deliver value-based care to patients. CMS has been working on these models and variations of them for well over a decade with some success. CMS has announced a new initiative to improve patient outcomes through evidence-based prevention to reduce costs (Center for Medicare & Medicaid Services, 2025).

Our project further explored the Bundled Payments for Care Improvement (BPCI)[1] payment model by using unsupervised learning to cluster procedures identified by Current Procedural Terminology (CPT)[2] codes in claims data and use supervised learning to estimate costs for patients by predicting what bundle of procedures they would most likely need to treat their condition(s). Previous versions of the BPCI model had an acute inpatient stay at a hospital as the start of an episode of care. We included all claims for a patient and created a preventative care indicator to see if outcomes were better for those who received preventative care.

Our approach would allow for clustering of procedures where comorbidities exist and links between other diagnoses could be observed as well as looking at if there is compliance with preventative measures. Part of our goal was to create clusters for chronic conditions that have ongoing care which could reveal multiple ways to treat the diagnosis, and we could analyze what is the most cost-effective way to treat the condition(s) and perhaps prevent an acute inpatient stay from ever occurring. A long term outcome would be to lessen the burden on the healthcare system freeing up resources, so more people can receive care.

We discovered that both the clustering and classification models performed well in both data representation and prediction tasks, with ~70% variance of data explained and achieving a test recall of ~97%, showcasing that there is potential for a more accurate method to predict bundled payments for chronic conditions from existing conditions. Extending this research could create more accurate cost estimations for care and procedures, but this was omitted in our analysis due to a lack of time. Nevertheless, our work demonstrates that predictive tools can be built for estimating the cost of care using past claims information. Future work could address the selection bias in what procedures are included in a claim that may occur because of billing departments being directed to maximize claim amounts.

Related work

CMS researchers created a technical user guide for the synthetic Public Use Files (PUFs) (Center for Medicare & Medicaid Services, 2023) released by CMS which provides background information on the synthetic data generation process and includes a comparison of the attributes of the synthetic data versus real data. While we are using different data released by CMS, the guide gives insight as to how the data were generated (CMSgov, 2025) which can help ensure the models we create do not just pick up on the methodology used to create the synthetic data. The researchers conducted a Comorbidity Analysis which looked at ICD-10 diagnosis codes across all claim service types (e.g. inpatient, carrier, hospice, etc.). A finding was that the synthetic data would typically only use one specific code for a given diagnosis while the real data would use several more granular codes representing variation in a disease. In a second phase of their analysis, to overcome the differences in the use of diagnosis codes, they grouped ICD-10 codes together according to Chronic Conditions Warehouse (CCW) chronic conditions

[1] Terminology definitions are in Appendix A

[2] CPT codes are HCPCS Level I codes, so CPT and HCPCS will be used interchangeably. Refer to Appendix A for a summary

definitions. We intended to compare our unsupervised learning results with the CCW definitions to evaluate how well our clusters were formed, but will have to do so in a follow-up analysis.

Data Source(s) and Preprocessing

Our primary data source is the Beneficiary Claims Data API (Center for Medicare & Medicaid Services, n.d.) that is hosted and maintained by CMS. It consists of 5 main endpoints containing synthetic data of submitted claims, claim response, patient information and explanation of benefits. The raw data retrieved was in NDJSON format, of which each newline is a separate claim for a unique patient. Our team obtained a valid sandbox API key and pulled the files from the API to serve as static files. As this API is continuously updated, our data is a snapshot.

We ran into significant challenges with the API as even though this was not included in the main documentation, the API had total record limits verified through subsequent pulls and by others in a forum discussion (BCDA Team, 2024). As a result, we could not merge the claims data with the other datasets as it did not contain the same patients in both files. We hoped that just using the claims data would be enough, but in the end there were only 2,607 patients after filtering the data and 44 CPT codes. The intention was to use a large volume of claims and get a good sense of all procedures done, but that was not realistic given the data.

As part of our modeling efforts, we had to acquire significant domain knowledge especially in regards to the numerous coding systems used in the data. These different coding systems are described in Appendix A. The following is a description of our main claims dataset:

Description	Partially adjudicated claims. Stores financial and clinical details on professional and institutional claims. The Claim is used by providers and payors, insurers, to exchange the financial information, and supporting clinical information, regarding the provision of health care services with payors
File Type	NDJSON
Key Variables	CPT/HCPCS procedure codes, ICD-10 diagnosis codes, DRG diagnosis codes, gender, birthdate, total value of claim
Number of Entries	178,761
Documentation	https://hl7.org/fhir/R4/claim.html

Feature engineering

After unnesting the claims NDJSON file, the preprocessing removed outliers and rows with incomplete data. Patients could be identified by their patient medicare number, so when examining their rows of claims, you could see that the fields were not always filled in consistently. For instance, a patient's first name might have the full name, but another row would just have an initial (Appendix B Figure 1). Birthdate was not always filled in, so the value from another row was used. If there was no birthdate to use to calculate age, those patients were dropped. We also removed rows where there were not any CPT codes filled in or where it appeared to be placeholder text. Because the data are partially adjudicated, the provider can start to bill the claim without it containing all the necessary information and later in the process, all information is filled in for the claim to be fully adjudicated.

After preprocessing, we essentially took the data from long format with a patient having multiple rows for each of their claims to wide format by combining fields from different claims rows into list columns. CPT codes from all claims for a patient were combined into a list column in order of their treatment. The same was done for ICD-10 and DRG codes which are the code types identifying a patient's diagnosis. An intended feature was time in between claims to see if having certain procedures sooner improved outcomes, but we were unable to add in that component. We calculated the number of claims and kept patients that had at least the 20th percentile which resulted in having at least 10 claims. We also looked at the combined CPT column length and dropped patients with more than 670 codes. It did not seem

realistic that a patient would have that many and we suspected there might be an issue with the way the synthetic data were created. As mentioned above, we created a preventative care indicator using a list of preventative care procedures identified by Blue Cross Blue Shield (Blue Cross Blue Shield, n.d.).

Once we had a patient level file, we created features for the unsupervised learning model. Initially we created a column corresponding to each HCPCS code in the combined HCPCS list column (i.e. HCPCS_1 for the first HCPCS code in the list). We label-encoded these columns and proceeded to create a baseline K-Means model. When looking at the results of what features were most important for a PCA component, we realized that we had inadvertently created features with lots of missing values. Columns toward the end of the HCPCS list (for example, HCPCS_255), would be mostly encoded missings which would stand out to the model even though there was no value in that information. We switched to thinking of our combined HCPCS list as a document and treated it as a bag of words problem. Each feature was TF-IDF encoded to create a sparse matrix. TF-IDF encoding would also take into account the frequency of the procedure which was an element we also wanted to incorporate. We kept gender as a label encoded column. For a complete list of features, see Appendix C.

The resulting features dataset was used in the unsupervised learning and the results were used as data for the supervised model, where we split the data into training and testing datasets for the supervised model. As a result, there was only one set of features that needed to be engineered.

Unsupervised Learning

Methodology

Model Selection

Having switched to TF-IDF encoding, we had hoped that we would have enough patient claim histories to see distinct bundles of procedures used to treat a diagnosis. That lead us to choose the following models to explore:

- **K-Means** - is an iterative algorithm that partitions the dataset into a chosen number of distinct non-overlapping clusters where each data point belongs to only one group. As we were not sure of the shape and structure of the data, we choose this as our base model to get a good sense of the data in the feature space and how much separation there is between clusters
- **DBSCAN** - is a density based clustering algorithm that clusters data points that are close together in dense regions and marks outliers as noise in areas of low density in the feature space. DBSCAN performs well when clusters are not just circular and convex and removes outliers by identifying them as noise and not assigning them to a cluster. After looking at our initial K-Means results, the clusters that formed were very dense, so trying a density based algorithm seemed a good choice to see if there was improvement in the model metrics
- **Gaussian Mixture Model (GMM)** - is a probabilistic model that assumes data points are generated from a mixture of several Gaussian (normal) distributions with unknown parameters. GMM performs soft clustering by assigning each point a probability of belonging to multiple clusters and as a result clusters are not necessarily spherical and can overlap. Looking at the K-Means results, it seemed that soft clustering might help in the areas where there is a lot of spread and overlap and perhaps a model that allowed for overlapping might perform better

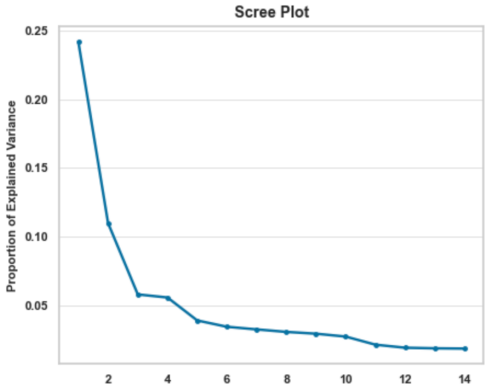
Some other model types we briefly considered were kernel K-means which allows for non-spherical clusters and robust sparse K-means as in the end our features were represented by a sparse dataset.

Standardization

Standardization involves scaling features so that they have a mean of zero and a standard deviation of one. Distance based clustering methods benefit from using standardized data. Standardization is also useful in reducing the influence of outliers in the data. PCA and DBSCAN are sensitive to different data scales and it's best practice to standardize the data. As GMM is assuming the data follow normal distributions, standardization is also preferred.

Dimension Reduction

Principal component analysis (PCA) is a linear transformation of the features where the number of components retained in the analysis are those that explain the most variance in the data. A way to visually display the explained variance by component is using a Scree plot. The elbow method recommends retaining all components before the point where the curve starts to flatten.



Looking at the scree plot, the shape of the curve does not form a perfect elbow, but the curve flattens out between components 5 and 6 resulting in the first **5** components explaining about **50%** of the variance. There is another dip between components 10 and 11 before the curve flattens again. Using **11** components results in explaining **68%** of the variance.

Figure 1: Scree plot shows two dips at 5 and 11 components

We chose to use 11 components over 5 components since there is a significant increase in the explained variance of 18% with only a minor increase in the number of principal components, so there is not a lot of risk of overfitting. As a comparison, for a further increase of 18% in explained variance, 12 more principal components would have to be included which greatly increases the number of features with limited upside. A quick look at what features most strongly influence the components reveals some interesting results:

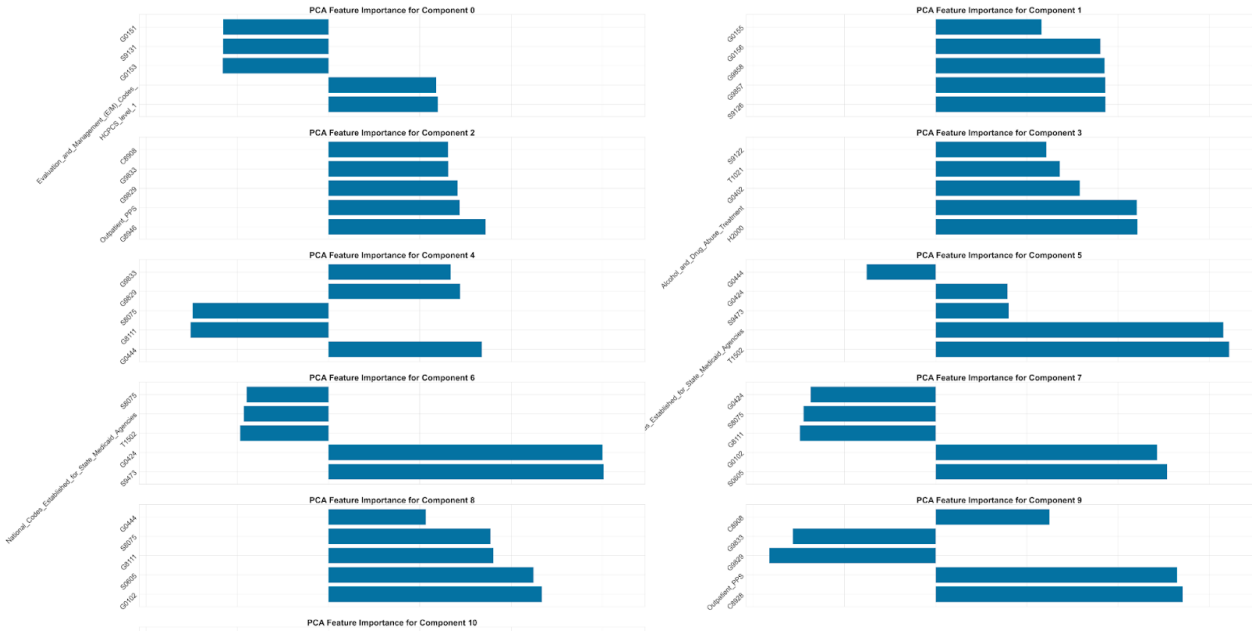


Figure 2: Feature contributions for different principal components

There are some features that have a negative loading which stands out in particular for components 1 and 4. A larger absolute value indicates that the feature significantly influences the component. A positive sign implies that the variable and the principal component are positively related and a negative sign indicates an inverse relationship. Some of the features that have a negative loading are:

HCPCS Code	Description
G0151	Services by a qualified physical therapist
S9131	Physical therapy
G8111	Receive Mammogram

Table 1: A sample of HCPCS codes and their descriptions

Intuitively it would seem that some of these would have a positive relationship. However, as there were limited HCPCS codes in the dataset, the data most likely do not mirror real life claims data, so interpretation is limited. Follow-up analysis with more data would show more accurate relationships.

We used the same PCA reduced dataset for all models. During model development, we did hyperparameter tuning to find the optimized parameters for each model type. For all models, we used silhouette score as our comparison metric (discussion is below). For DBSCAN and Gaussian, code was written to be similar to GridSearch with Cross Validation (GridSearchCV) as it did not have Silhouette score as a scoring metric. We selected the parameters with the most influence to optimize our models.

K-Means	DBSCAN	GMM
`init` - method to initialize centroid	`metric` - method for calculating distance	`init_params`- method to initialize weights
`n_clusters` - number of clusters	`eps` (epsilon) - maximum distance between two samples for one to be considered in the neighborhood	`n_components` - number of mixture components
`n_init` - number of times the k-means algorithm is run with different centroid seeds.	`min_sample` - number of samples in a neighborhood for a point to be a core point	`n_init` - number of initializations to perform

Table 2: Hyperparameters optimised in various unsupervised learning techniques

Our selections for the K-Means model are described in detail below in our sensitivity analysis. For DBSCAN, we were forced to choose euclidean as the metric as cosine caused an issue. Due to time constraints, we were not able to investigate this. Epsilon is the most important parameter for the DBSCAN model, so in our version of GridSearchCV we looked at the range [0.75, 1, 1.25, 1.5] and coupled that with looking at the range [5, 10, 15] for min_sample. The optimized solution was eps = 1.25 and min_sample = 15. For our GMM, we only evaluated kmeans++ as the initialization method as the method is superior to others as it allows for faster convergence and higher cluster quality. Using our version of GridSearchCV, we looked at the ranges [7, 8, 9, 10] for n_components and [5, 10, 15] for n_init and the optimized parameters were 8 and 5 respectively.

Unsupervised Evaluation

The table displays results with optimized hyperparameters:

Metric	K-Means 8 Clusters	DBSCAN 5 Clusters	Gaussian 8 Clusters
Euclidean	0.612	0.6527	0.3757
Cosine	0.7029	0.6347	0.4181

Table 3: Results of All Models Using Silhouette Score

To evaluate our models, we used silhouette score which measures how well the clusters are separated. For each data point, the silhouette score is calculated based on two values. Cohesion or intra-cluster distance is the average distance to other points in the same cluster and separation or nearest cluster distance is the average distance to points in the nearest neighboring cluster. Scores range from **-1** to **1** where the closer to **1** the better. 0 indicates that there are overlapping clusters and -1 means there are

data points in the wrong clusters. Silhouette score is best when trying to find the optimal number of clusters, so it seemed appropriate for optimizing our K-Means model.

Silhouette score has metric type as a parameter. Euclidean measures the straight line distance between two points in space and is sensitive to absolute distances between points which given the shape of our data did not seem appropriate. Cosine measures the angular cosine similarity between two vectors and works better with high dimensional data. We ultimately choose to encode our data using TF-IDF, so cosine was the preferred metric when evaluating model performance, but we also calculate using euclidean as another comparison point.

With further model development, we saw that there were very dense clusters, so other evaluation methods like Calinski-Harabasz Index and Davies-Bouldin Index may have been better to use as they are better at evaluating dense clusters. Due to time constraints, we were not able to explore these other methods.

Below are scatterplots of the clustering results for each model:

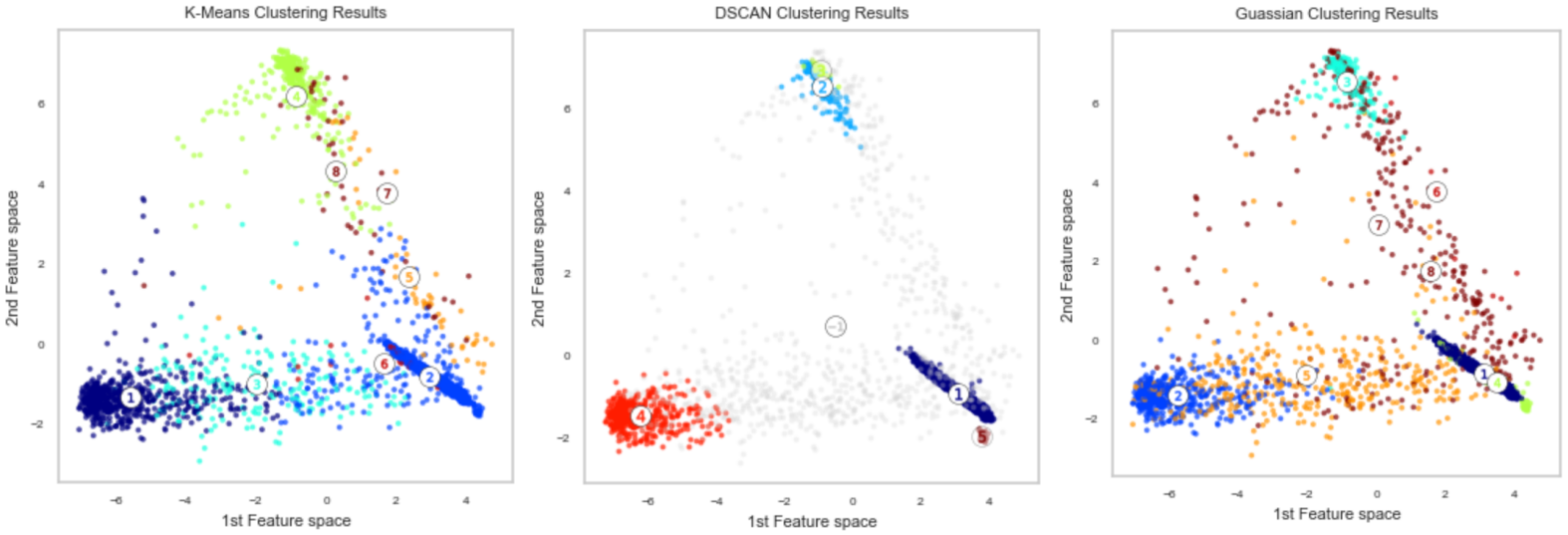


Figure 3: Cluster Plots

From the table above, K-Means performs best using the cosine metric. We determined K-Means was our best model, but DBSCAN removes a lot of the noise, which is represented by the grey points, so that may be why it performed best when using the euclidean method. All three models pick up on the same underlying structure. The structure of the data is distinct with the v-shape, and we would have liked to dig into the cause of that and perhaps refine our features, but due to time constraints, we were not able to and would do so in a follow-up analysis.

Sensitivity Analysis & Hyperparameter Tuning

To find the optimized parameters for our K-Means model, we looked at silhouette plots and scores for a range of clusters and number of times the algorithm was run with different centroids. For the initialization method, we decided to go with kmeans++, which is the default. The following our are results:

Metric	K-Means 6 Clusters			K-Means 7 Clusters			K-Means 8 Clusters			K-Means 9 Clusters			K-Means 10 Clusters		
	n_init = 5	n_init =10	n_init = 15	n_init = 5	n_init =10	n_init = 15	n_init = 5	n_init =10	n_init = 15	n_init = 5	n_init =10	n_init = 15	n_init = 5	n_init =10	n_init = 15
Euclidean	0.5774	0.5774	0.5774	0.4433	0.5982	0.5855	0.612	0.612	0.5986	0.4861	0.6178	0.6178	0.4891	0.4891	0.4891
Cosine	0.6804	0.6804	0.6804	0.6398	0.69949	0.6813	0.7029	0.7029	0.68856	0.6754	0.7024	0.7024	0.6745	0.6745	0.6745

Table 4: Sensitivity Analysis Using Silhouette Score

Looking at the table it appears that changing the number of clusters has a greater impact on model performance than changing the number of times the algorithm is run as some metrics stay the same for a given cluster when it changes.

Supervised Learning

The supervised learning task is a multi-class classification problem where the target classes and number of clusters were the results of the unsupervised learning task. Our supervised model aims to predict which cluster a patient belongs to based on the set of procedures a patient has already undergone. This would allow for better estimation of the total cost of a patient's treatment based on others who have undergone similar treatment. While documentation indicated that there would be individual costs for each procedure in the claim, ultimately it was just the total cost of the claim. We attempted to extract procedure costs by looking at claims where one or a few procedures were billed. With the dataset being smaller than expected and lack of consistency in total costs between patients who undergo similar treatments and care, we were not able to complete the analysis at this time.

All features follow those created from the unsupervised learning analysis. Exploration involved 6 different models each with different methodologies and characteristics for classification. Since this is a multi-class classification problem, we used the One vs. Rest Classification process to adapt traditionally binary classifiers to fit the context. Justification for each model is stated below:

- **Baseline** - serves as the benchmark for subsequent models. All predicted values are the most frequent class in the training dataset
- **Logistic Regression** - serves as the entry-level model for classification problems to see if there is any improvement over the baseline. Using the linear combination of HCPCS and a sigmoid function to predict the presence of a patient in a cluster
- **Decision Trees** - a tree-based implementation to contrast Logistic Regression
- **Random Forest** - an improvement over single decision trees. Random Forest employs bagging to improve data diversity amongst its many trees to make predictions more robust
- **XGBoost** - a state-of-the-art improvement over the Random Forest model that primarily uses boosted trees to improve performance
- **Feed-forward Neural Network** - a simple mutli-layer perceptron implementation to test the performance of a non-linear classifier on data

Methodology

The optimal number of cluster values was deemed to be 8 from our unsupervised learning task, however, each cluster contains a different number of entries, making this dataset imbalanced. Therefore, to maintain class distribution amongst the training (80%), validation (10%) and testing (10%) datasets, a stratified split approach was used. Each model was trained using the training data, and validated with the validation set.

A subset of models underwent a K-fold stratified shuffle split wherein 10 folds were used. This method preserves the percentage of samples for each class in the multiclass classification setting and shuffles the data to ensure that the model cannot learn implicit relationships within the ordered data. This method further validated the initial run using a single train validation split ensuring that the results obtained are more consistent and robust.

All models during initial exploration are scored against 4 metrics: Accuracy, Precision, Recall and F1 and this choice is described below. Finally, from the results of the experiment, only the highest scoring model underwent subsequent sensitivity analysis and failure analysis. This was done due to time limitations, and further investigation should be conducted to confirm the validity of this result.

Metrics

The selected metrics represent different aspects of the model being evaluated. Because of the imbalance within the dataset, a wider variety of metrics are used

1. **Accuracy** - accounts for the general performance of the model. This value though less robust alone, when compared to the score of the baseline model, a better comparison can be made
2. **Precision** - precision is key since the model is dealing with diagnoses which to achieve trust should be the primary metric optimised for. We want to accurately predict the class of diagnosis it falls under to prevent wrongly estimating cost since variations in treatment can greatly vary between treatment types. Here each class score is weighted on the support (the number of true instances for each class), which accounts for the imbalance within the dataset.
3. **Recall** - recall allows us to measure the coverage, which in general measures on average how many of each class label was correctly classified. While important for a single diagnosis case, most patients that are making claims are already about to undergo a set of procedures, meaning that they are already receiving the expected care without relying on this model. However, observing this metric is still important to ensure that all classes, on average, are still being fairly represented during predictions. This metric is also weighted
4. **F1 Score** - F1 score provides a good balance between recall and precision. This is done to ensure that for each model results are not skewed towards one metric. This metric is also weighted

Results

The table below describes the results of initial model exploration on the validation set:

	Baseline	Logistic Regression	Decision Tree	Random Forest	XGBoost	FFN
Accuracy	51.34	97.32	94.25	96.17	98.08	75.48
Precision	26.36	97.0	93.97	95.91	98.10	81.73
Recall	51.34	97.32	94.25	96.17	98.08	65.13
F1	34.83	97.06	94.05	95.93	98.04	-

Table 5: Results validation set from initial exploration in percentage (%)

All models outperformed the baseline model with strategy most frequent. As the model architecture complexity increases the performance of the models improves. Since the dataset was split in a stratified manner, all classes are represented during this exploration. However, as seen in the baseline model, some classes are overrepresented which might have some impact on the overall performance. XGBoost, as expected, performs the best due to its ability to iteratively improve its trees through boosting. Finally, the Feedforward Neural Network (FFN) does not perform as well, likely due to a lack of data. Neural networks require a large quantity of data and more iterations to converge. Further studies can be conducted to evaluate the relevance of this model when more data is present. A detailed breakdown of the cross validation splits and confusion matrices for each model can be found in Appendix D Figures 16-24.

Training Curves

The training curves suggest that more data could enhance the model. Looking at sampling between 10% to 100% of the training data to predict the validation data, at 100% the model maxes out in all metrics. Sufficient data would suggest a plateau before utilising the full dataset. Furthermore due to a class imbalance, classes that are less represented might not appear in the lower percentages, therefore the analysis does not fully capture the extent of the data.

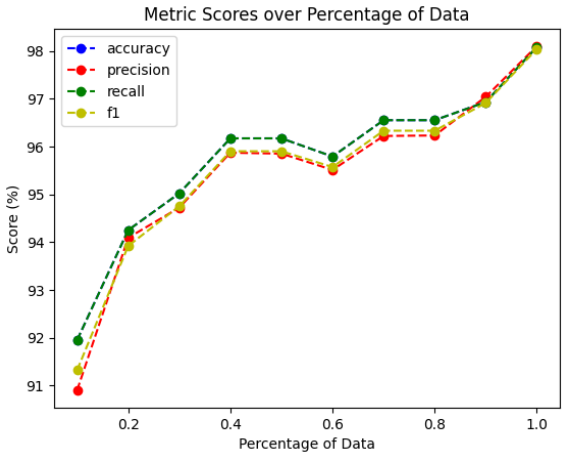


Figure 4: The lack of a plateau at 100% of training data suggest more data can improve performance

Oversampling methods were not used to supplement the original dataset since there is no good way to represent a sequence of claims. However, using the transformed feature space other methods like SMOTE could be a valid alternative to extending underrepresented classes. Ultimately, the best way would be to expand our dataset, and since the API is updated biweekly, scheduled data loaders could be created to increase the amount of data available.

Feature Importance and Ablation Analysis

HCPCS codes represent different procedures and services billed in the claim and some might have stronger indicative power when separating between clusters of diagnosis. Here feature importance represents how much each HCPCS code and other features contributes to the model's predictive power, as well as helping with interpretation of results.

Decision trees select the cutoff points by minimising the gini coefficient across all features. The algorithm recursively splits until the terminating condition is met. A feature is deemed important if the impact of the split on the variance of the gini coefficient compared to its parent node is large. The plotted tree provides a natural visual representation of feature interactions, making it easy to trace the path for which a claim is classified under.

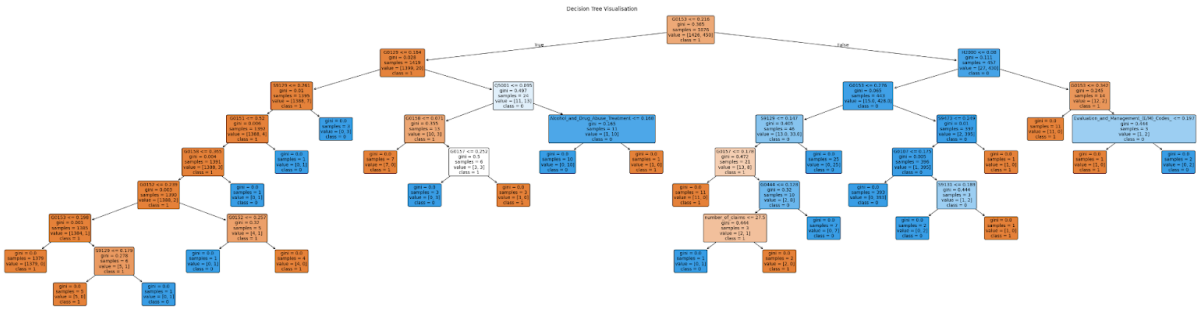


Figure 5: Plotted decision tree for model interpretation

Many of the HCPCS codes conditioned on are within the G0XXX range which fall under Procedure/ Professional Services. While the model performed decently well, this method provides little clarification of how the model is making predictions since each feature represents the same category. Other features like H2000 (Alcohol and Drug Abuse Treatment) appear, but infrequently which suggests that the feature is not as impactful. This is further confirmed by the feature importance graph (Appendix D Figure 22-23) where majority of the importance is assigned to G0153 (services performed by a qualified speech-language pathologist); other methods were employed to bridge this gap.

To identify how each feature impacts the model directly, Partial Dependence Plots (PDP) were used. It shows the marginal effect one or two features have on the predicted outcome. It is constructed by fixing all other features at their average then making predictions for each value in the selected feature's range. Since this task is a multiclass problem, each class must be observed on separate graphs. A feature is considered more important if changing its value causes large fluctuations in the output.

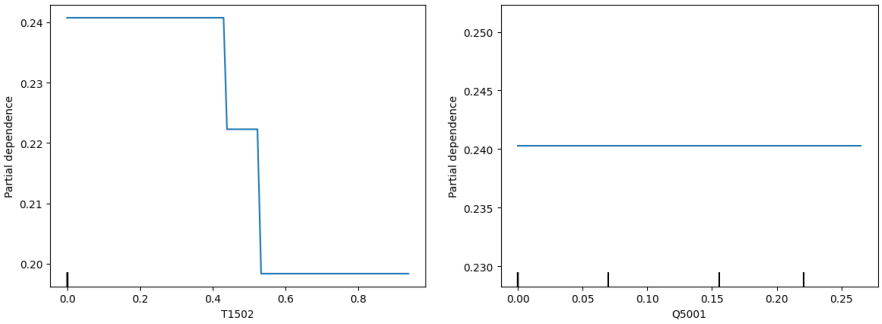
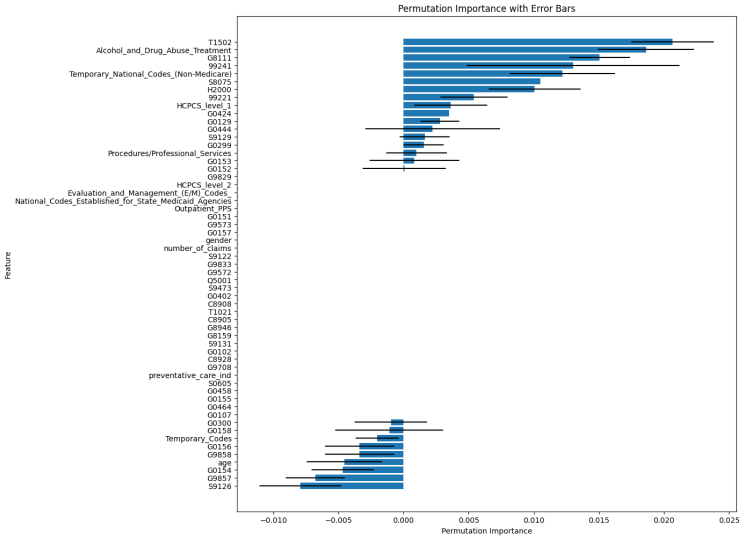


Figure 6: Comparison of PDP Plots for T1502 (left) and Q5001 (right)

We observe that features like T1502 (administration of medication) cause the partial dependence to decrease quite significantly as the value of the feature increases, while features like Q5001 (hospice or home health care provided in patient's home/residence) do not change regardless of the value it is set. This implies that features like T1502 display a stronger marginal effect on the predicted outcome making it a stronger feature. While the method is easy to interpret, since it fixes the other features at their average, it assumes independence between features, which is often unrealistic. Correlations between variables and feature interactions are not captured by the plot, and since the model is still a culmination of multiple features, this method is limited. However, it does provide an intuitive representation of the average prediction if all the values of the selected feature are varied.

To circumvent this independence assumption, Permutation Importance Plots improve upon the PDP plot by preserving the feature interactions. The selected column is randomised while holding all other features constant. If the deviation from the original model's prediction is large after the features have been randomised, it means that the feature was heavily relied on for predictions.



T1502 is rated the highest importance, which is a similar result as the PDP plot. Many of the features of higher importance represent one instance of a larger group of procedures. These are represented by their differing starting letter or number ("T", "G", "9", "S", etc.). These individual features might be the strongest contributors when separating between the clusters. The error bars also indicate the variance of the importance where some features like G0444 might not have a statistically significant impact since its standard deviation covers the zero value. A drawback of this method is that each feature's importance is tagged to the model performance. A feature that is unimportant to a bad model, might be important to a good model, therefore, careful calibration of the model was undertaken prior to this analysis.

Figure 7: Permutation importance plot

Ultimately, observing how different feature importance is assigned between models suggests that each model assigns weights slightly different and that feature interaction likely plays a large role in the predictive power of the model.

Sensitivity Analysis & Hyperparameter Tuning

Sensitivity analysis aims to test the robustness of a model by varying the values of hyperparameters and checking for major fluctuations in performance. Here we employ a variety of techniques to not only check, but also identify optimal values for the XGBoost model. Hyperparameter tuning was only performed on the best performing model during the initial exploration phase. The following were optimised because of their known contribution to model performance and nature of our data

1. **Number of Estimators:** the number of boosted trees to create. Increasing the number of trees will provide a large boost to any dataset to improve generalisation. Balancing this is critical to prevent under and overfitting
2. **Learning Rate:** controls the step size shrinkage for each tree. Controls how fast the model learns. This has a strong interaction with the number of estimators since a smaller learning rate will require more trees to optimise
3. **Max Depth:** the maximum depth of each tree created. A larger tree would make it possible to better separate the data points. Our data contains a large number of features that can be split on, therefore increasing the max depth would allow the tree to make more detailed splits
4. **Column sampled when creating a tree:** fraction of feature columns that the tree can split on. This introduces some regularisation to prevent overreliance on a single feature, forcing the model to generalise better
5. **Gamma:** minimum loss reduction required to make a new partition. This ensures that each split is more meaningful since a larger value ensures more information is gained.
6. **Alpha:** L1 regularisation term. This parameter is likely the most critical due to our data being sparse
7. **Lambda:** L2 regularisation term. To complement the other regularisation term and serves the same purpose

First GridSearchCV was used to permute a variety of set values for each hyperparameter to discover how a range of different parameters interact with each other to affect the model. This approach is simple since it allows us to specify any valid values desired to be run against a comprehensive cross validation training loop. A drawback of this approach is that it neither provides the impact of each hyperparameter nor an intelligent way of searching for better hyperparameters. The process is merely a personal heuristic selection.

	learning_rate	max_depth	n_estimators	reg_alpha	mean_test_score	std_test_score
Best Parameters	0.1	10	100	0	98.12	0.006292
Worse Parameters	0.01	3	10	5	95.73	0.006801

Table 6: Results of GridSearchCV

Unsurprisingly, a higher learning rate together with a larger number of trees produced a better model. From the training data analysis, we observe that there is less than sufficient data to validate the model, therefore, higher parameters were required to achieve a better score.

To gain a more comprehensive understanding of the impact of each hyperparameter, a manual search across a single hyperparameter's space was conducted by scoring the model against the validation dataset to observe the change in performance over values. Learning Rate and Alpha (L1) Regularisation showed the greatest change in performance as the values varied.

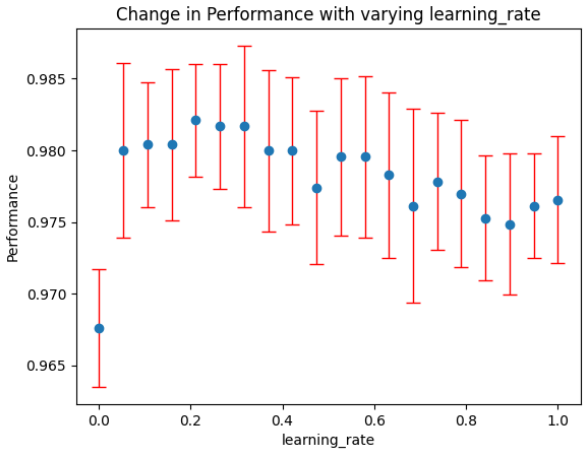


Figure 8: Model performance peaks at ~0.2 learning rate

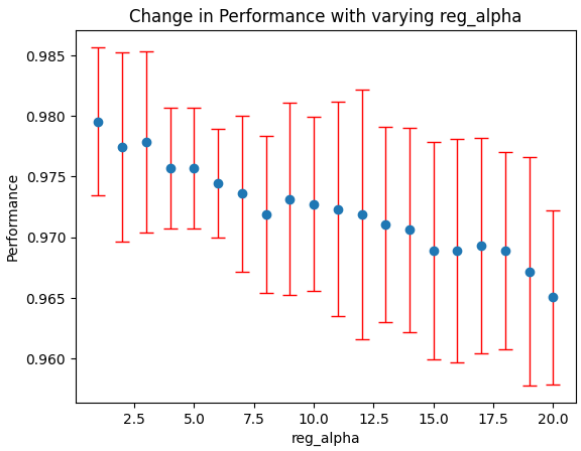


Figure 9: Model performance degrades as the value increases

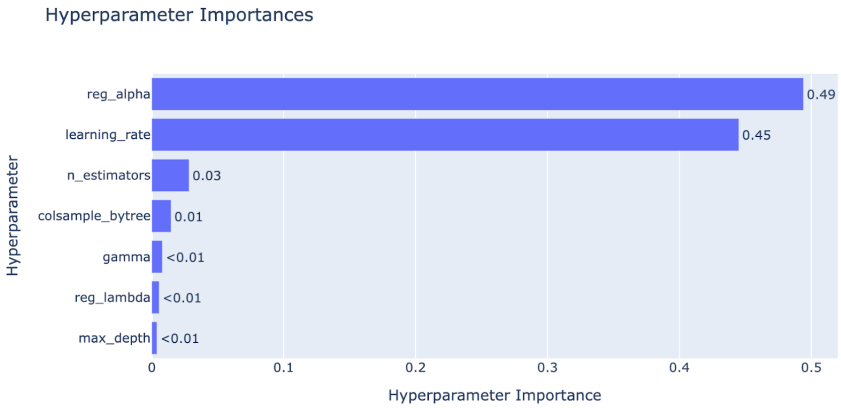
The learning rate achieved the greatest performance at ~0.2 while alpha value was at 0. This corresponds to the results found by the Grid Search, which suggests that these parameters are likely to be ideal for this problem.

Finally to leverage a more intelligent hyperparameter search, the Optuna package leverages Tree-structured Parzen Estimator (TPESampler) to intelligently search parameters values associated with the best objective values. In doing so it retrieves the best set of parameters without random guessing. This greatly reduces search time and allows for a wider range of hyperparameters to be searched.

	learning_rate	max_depth	n_estimators	colsample_bytree	gamma	reg_alpha	reg_lambda
Best Parameters	0.15865	9	179	0.43684	0.17851	0	5

Table 7: Optimal hyperparameters as found by Optuna

Many of the optimal values are similar to those observed in the previous analyses. L2 regularisation (reg_lambda) was introduced to control the impact of features. With the dataset containing many features, this is to be expected. The largest difference would be the increase in the number of trees required. This suggests that previous iterations of the search were underfitting the model, which could only be obtained by expanding the search space.



The hyperparameter importance plot, created using the fANOVA function in Optuna, corresponds to previous findings. The optimal L1 regularisation value was found to be 0, but still has high importance which suggests that all features have some minute contribution, and is better controlled by L2 regularisation by not fully removing any features.

Figure 10: Hyperparameter importance ranked by fANOVA scoring

Final Results

The final results obtained by the XGBoost model with optimal hyperparameters on the test set are:

	Validation Set	Test Set
Accuracy	98.08	97.32
Precision	98.10	97.12
Recall	98.08	97.32
F1	98.04	97.15

Table 8: Final results of XGBoost with optimal hyperparameters

The model performs well on the test set, expectedly lower compared to the validation set. However, some incorrect predictions were observed which are explored below.

Failure Analysis

There are still some entries where the model fails to make the right prediction. Analysing a single instance reveals insights into how the model made predictions and potentially identifies problematic entries. One entry that was incorrectly assigned (expected: 0, predicted: 1) upon inspection showed significant difference from the entries that were correctly predicted, suggesting that the distribution of values are already skewed. Particularly, the number of claims in the incorrect entry (99) is significantly larger compared to the average of correct entries (28.2)[3]. Other features like 99241 and Procedures/Professional_Services largely differed from value to mean value. From the feature importance analysis (Figure 7 above), these are also key variables used in the prediction as well.

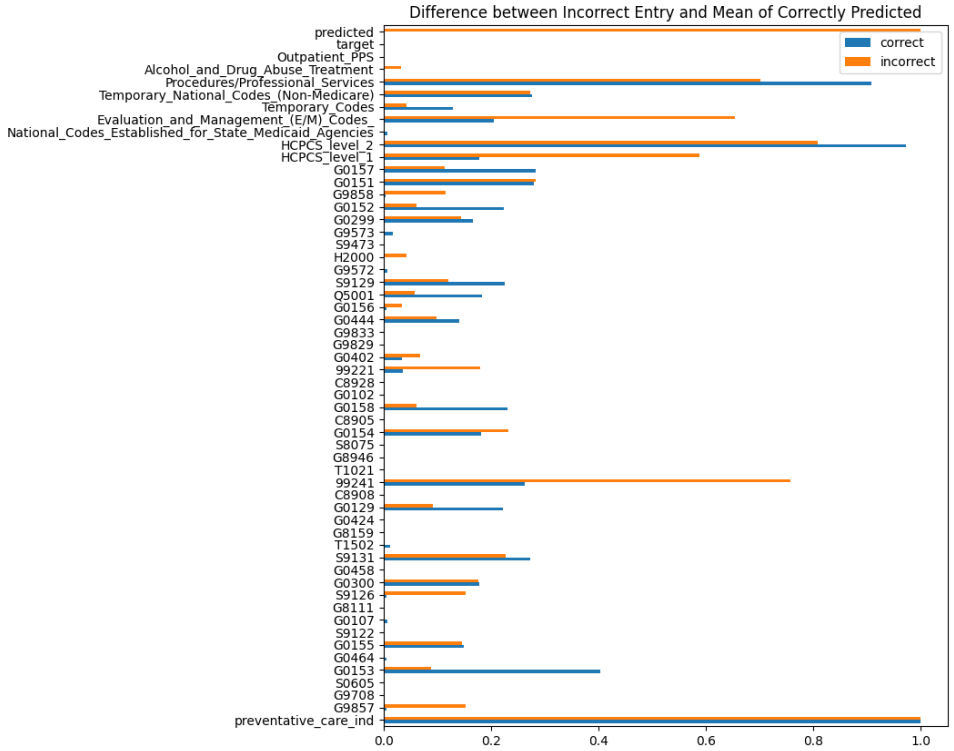


Figure 11: Difference in feature values for incorrectly predicted entry and average of correctly predicted entries

[3] The number of claims feature has been removed from the figure as the scale would make seeing the other features difficult

Comparing this entry to a correctly predicted one using a force plot further reinforces this claim where the different features cause the wrong outcome from the model.

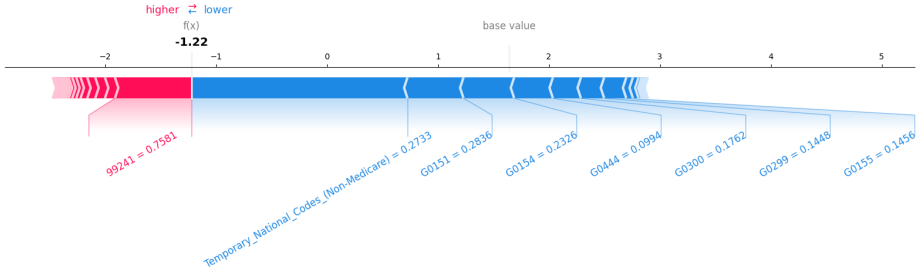


Figure 12: Force plot of an incorrectly predicted entry

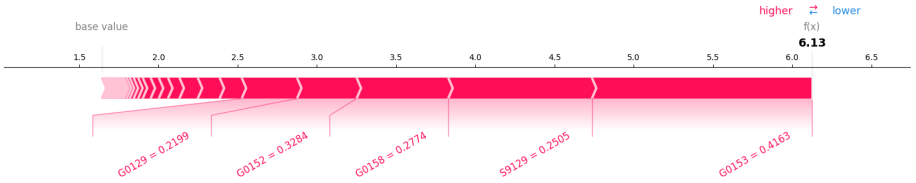


Figure 13: Force plot of a correctly predicted entry

Feature G0153 for example causes predicted values to decrease rather than increase in the incorrect entry. Given how well the model performs on the test set, this error can be classified as an edge case with how widely it differs from the other entries in the same class. Additional preprocessing steps such as filtering entries based on features interquantile range could reduce the change of outliers appearing in the dataset. Alternatively, outlier detection models such as Isolation Forests, or using Local Outlier Factor to remove these entries would improve the models generalisability.

Discussion

The largest finding was learning that the BDCA API does not allow for all records from their synthetic datasets to be retrieved. The documentation indicated there would be ~30,000 enrollees which should have been sufficient patient claims histories to do the analysis. Additionally, there are ~11,000 HCPCS codes and in our final analytic dataset there were only 44 present. The other challenge we faced is that not all fields in the FHIR documentation were in the raw NDJSON and this limited the features we could create. We struggled with feature engineering and incorrectly chose to label-encode the features we did have. After running an initial baseline K-Means model, we noticed this was not the correct encoding method and corrected for it by switching to TF-IDF encoding.

With all these challenges, it became clear that the intention of our unsupervised model to group bundles of procedures would not be possible. Given that we were so far along into the project and due to other challenges, we could not go back and reassess the initial data pull. With more time, we could create a larger analytic dataset by pulling data from the API on a more frequent basis since it is updated biweekly. A detailed comparison between the synthetic data and known characteristics of the Medicare claims population would be thoroughly researched, so we could determine how realistic the data are and the extent to which we can draw proper conclusions from our analysis. As an example of what we intended to show, here are word clouds that were generated from the bundles that we did cluster in our unsupervised learning model using MDC description which is the highest level of diagnosis code granularity:

Figure 14: Word Clouds using MDC Descriptions

What we had hoped is by looking at the word clouds, comorbidities or interesting combinations of diagnoses would present themselves. We would also look across word clouds to see how many occurrences of the diagnoses are present in other clouds and investigate if there were multiple bundles of procedures used to treat the same conditions. As the number of HCPCS in the dataset was so small, there were not many MDC categories that were in the data, but you can see how some distinct words pop in addition to HLTH and BLOOD. This could be refined further to remove more common words. Feeding the clusters into the supervised model would allow us to estimate the costs of each cluster and we could determine if one treatment plan was more cost effective than the other. This was a component we were not able to address in this analysis and would be the focus of an additional analysis given more time and resources.

Slightly surprising was the performance of the model, even on less complex architectures, performance was still good. Our opinion is that due to the large number of features and minimal entries, the model is able to create very specific decision boundaries and trees that strongly separate the classes, somewhat akin to overfitting. While we did not prove this, public opinion recommends that the number of entries be significantly more than the number of features (Doug, 2011) to prevent overfitting, which in this context is unattainable due to API. Feature importance and failure analysis were challenging due to the number of available features, which made it difficult to pinpoint sources of importance. In more complex models, often feature interaction is used intrinsically to create new signals for the model to pick up, so with a larger number of features, interactions are unseen and hard to showcase.

To circumvent this, we utilise a wide variety of feature importance techniques and failure analysis plots that target different aspects of interpretation. For example, the PDP plots showed a single features contribution, while permutation importance focused on measuring interaction importance. Instead of using a single technique as the determining factor, multiple techniques provided deeper insight into how the model was functioning allowing us to make more concrete conclusions. Given more time, there are two improvements we would make. As mentioned above, increase the dataset size which should reduce class imbalance and prevent overfitting. Second, a deeper dive into failure analysis where each feature is studied more in depth and in context, providing stronger interpretation on how the model is making decisions for greater clarity.

Ethical Considerations

If this same analysis or an extension of it is done with actual patient data, it would be critical to respect patient privacy. Even with using actual patient data, we may still have limited inferential power and since this greatly impacts a patient's life, it will always be better to err on the side of calculating larger bundle amounts then less.

References

BCDA Team. (3 July, 2024). *Google Groups*. Retrieved from Confirmation of file sizes: https://groups.google.com/g/bc-api/c/g-Y1HWKHyVU?pli=1

Blue Cross Blue Shield. (n.d.). *anthembluecross.com.* Retrieved from ACA preventive care coding guidelines: https://www.anthembluecross.com/content/dam/digital/docs/provider/commercial/general/ACA-preventive-care-coding-gdlns.pdf

Center for Medicare & Medicaid Services. (May, 2023). *Data.CMS.gov.* Retrieved from Synthetic Medicare Enrollment, Fee-for-Service Claims, and Prescription Drug Event Data Public Use File: https://data.cms.gov/sites/default/files/2023-05/d51e1218-68c3-4c7c-9598-0b81f22fe903/User%20Guide%20-%20CMS%20Synthetic%20RIF%20Files%20May%202023_AM508_v2.pdf

Center for Medicare & Medicaid Services. (13 May, 2025). *CMS.gov*. Retrieved from CMS Innovation Center Strategy to Make America Healthy Again: https://www.cms.gov/priorities/innovation/about/cms-innovation-center-strategy-make-america-healthy-again

Center for Medicare & Medicaid Services. (n.d.). *Beneficiary Claims Data API*. Retrieved from Partially Adjudicated Claims Data.

CMSgov. (7 May, 2025). *Beneficiary FHIR Data*. Retrieved from Synthetic Data Guide: https://github.com/CMSgov/beneficiary-fhir-data/wiki/Synthetic-Data-Guide

Doug. (11 November, 2011). *Stack Overflow*. Retrieved from Optimal Feature-to-Instance Ratio in Back Propagation Neural Network: https://stackoverflow.com/questions/8090316/optimal-feature-to-instance-ratio-in-back-propagation-neural-network

Appendix

Appendix A: Overview of Code Systems and Terminology

The code systems are in order of least granularity to most.

Major Diagnostic Category (MDC)[4]:

MDCs are a classification system used to group patients based on their primary diagnosis. MDCs were created by two physicians to facilitate better billing and reimbursement processes for Medicare. There are typically 25 mutually exclusive principal diagnosis areas each representing a specific organ system or medical condition.

MDC	DESCRIPTION	MDC	DESCRIPTION
01	DISEASES & DISORDERS OF THE NERVOUS SYSTEM	14	PREGNANCY
02	DISEASES & DISORDERS OF THE EYE	15	NEWBORNS & OTHER NEONATES WITH CONDTN ORIG IN PERINATAL PERIOD
03	DISEASES & DISORDERS OF THE EAR	16	DISEASES & DISORDERS OF BLOOD
04	DISEASES & DISORDERS OF THE RESPIRATORY SYSTEM	17	MYELOPROLIFERATIVE DISEASES & DISORDERS
05	DISEASES & DISORDERS OF THE CIRCULATORY SYSTEM	18	INFECTIOUS & PARASITIC DISEASES
06	DISEASES & DISORDERS OF THE DIGESTIVE SYSTEM	19	MENTAL DISEASES & DISORDERS
07	DISEASES & DISORDERS OF THE HEPATOBILIARY SYSTEM & PANCREAS	20	ALCOHOL/DRUG USE & ALCOHOL/DRUG INDUCED ORGANIC MENTAL DISORDERS
08	DISEASES & DISORDERS OF THE MUSCULOSKELETAL SYSTEM & CONN TISSUE	21	INJURIES
09	DISEASES & DISORDERS OF THE SKIN	22	BURNS
10	ENDOCRINE	23	FACTORS INFLUENCING HLTH STAT & OTHER CONTACTS WITH HLTH SERVICES
11	DISEASES & DISORDERS OF THE KIDNEY & URINARY TRACT	24	MULTIPLE SIGNIFICANT TRAUMA
12	DISEASES & DISORDERS OF THE MALE REPRODUCTIVE SYSTEM	25	HUMAN IMMUNODEFICIENCY VIRUS INFECTIONS
13	DISEASES & DISORDERS OF THE FEMALE REPRODUCTIVE SYSTEM		

Medicare Severity Diagnostic Related Group (MS-DRG)[5]:

The MS-DRGs within each MDC are defined by a particular set of patient attributes which include principal diagnosis, specific secondary diagnoses (major complication or comorbidity (MCC) or complication or comorbidity (CC)), procedures, sex, and discharge status. A leading character is used for more specific classifications.

MS-DRG	Description	MS-DRG	Description
A0021-A0999	Ambulance and Other Transport Services and Supplies	M0001-M0005	MIPS Value Pathways
A2001-A2029	Matrix for Wound Management (Placental, Equine, Synthetic)	L0112-L4631	Orthotic Procedures and Services
A4100	Skin Substitute Device	M0010	EOM (Enhancing Oncology Model) Enhanced Services

[4] Center for Medicare & Medicaid Services. (2005). *CMS.gov.* Retrieved from Downloads: https://www.cms.gov/medicare/medicare-fee-for-service-payment/acuteinpatientpps/acute-inpatient-files-for-download-items/cms022605

[5] Center for Medicare & Medicaid Services. (n.d.). *CMS.gov.* Retrieved from ICD-10-CM/PCS MS-DRG v37.0 Definitions Manual: https://www.cms.gov/icd10m/version37-fullcode-cms/fullcode_cms/P0002.html

A4206-A8004	Medical And Surgical Supplies	M0075-M0301	Miscellaneous Medical Services
A9150-A9999	Administrative, Miscellaneous and Investigational	M1003-M1070	Screening Procedures
B4034-B9999	Enteral and Parenteral Therapy	M1106-M1143	Episode of Care
C1052-C1062	Other Therapeutic Procedures	M1146-M1425	Other Services
C1600-C1606	Surgical, Imaging Devices and Grafts	P2028-P9615	Pathology and Laboratory Services
C1713-C9901	Outpatient PPS	Q0035-Q9998	Temporary Codes
E0100-E8002	Durable Medical Equipment (DME)	R0070-R0076	Diagnostic Radiology Services
G0008-G9999	Procedures/Professional Services	S0012-S9999	Temporary National Codes (Non-Medicare)
H0001-H2041	Alcohol and Drug Abuse Treatment	T1000-T5999	National Codes Established for State Medicaid Agencies
J0120-J8999	Drugs Administered Other than Oral Method	U0001-U0002	Coronavirus Diagnostic Panel
J9000-J9999	Chemotherapy Drugs	V2020-V2799	Vision Services
K0001-K0900	DME Medicare Administrative Contractors (MACs)	V5008-V5364	Hearing Services
K1004-K1037	Components, Accessories, and Supplies	A1-XU	Modifiers for HCPCS Codes

International Classification of Disease (ICD)[6]:

ICD is a global coding system for causes of death published by the WHO. It was created to provide consistent classification for causes of death, so that mortality statistics could be standardized. A single selected cause can be chosen and is called the underlying cause of death.

International Classification of Disease - Clinical Modification (ICD-CM)[6]:

ICD-CM is a standardized code system used by healthcare providers to diagnose patients by coding diseases and medical conditions. ICD-CM is maintained by CDC's National Center for Health Statistics (NCHS) and builds on the ICD codes. Each ICD-CM diagnosis code is categorized as a major complication or comorbidity (MCC).

International Classification of Disease - Procedure Coding System (ICD-PCS)[7]:

ICD-PCS is a medical classification system in the US used for inpatient hospital procedures. They are composed of seven characters, where each is an axis of classification which specifies information about the procedure performed. ICD-PCS is maintained by CMS. Where ICD-CM classification focuses on medical conditions, ICD-PCS is focused on procedures used to treat conditions

Healthcare Common Procedure Coding System (HCPCS)[8]:

HCPCS consists of two levels:

HCPCS Level I codes are **Current Procedural Terminology (CPT®)** codes, a numeric coding system maintained by the American Medical Association (AMA). The AMA publishes yearly updates.

---

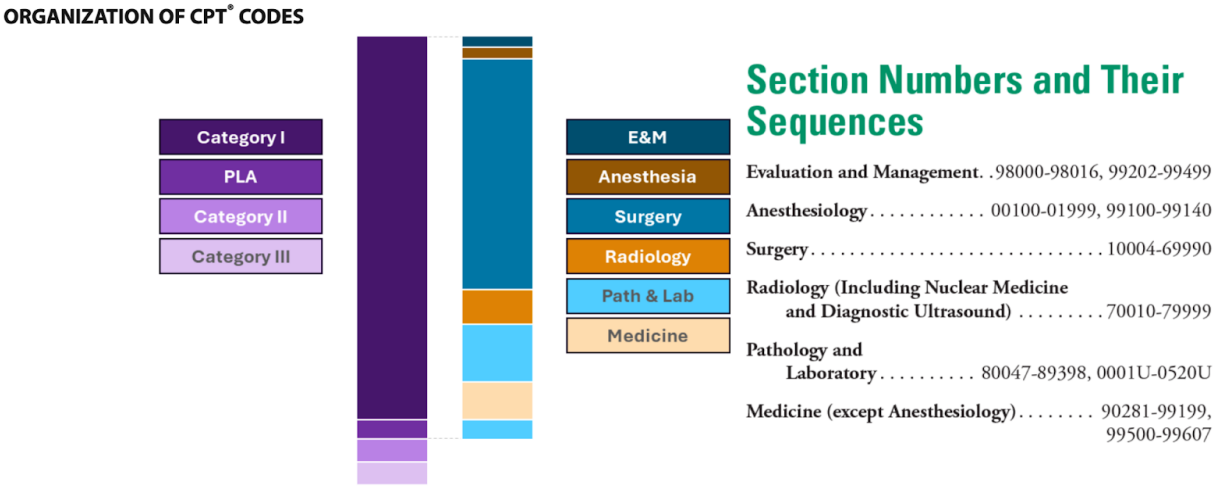
[6] Center for Medicare & Medicaid Services. (18 June, 2025). *CMS.gov.* Retrieved from Overview of Coding and Classification Systems: https://www.cms.gov/cms-guide-medical-technology-companies-and-other-interested-parties/coding/overview-coding-classification-systems

[7] Center for Medicare & Medicaid Services. (1 April, 2025). *CMS.gov*. Retrieved from ICD-10-PCS Official Guidelines for Coding and Reporting 2024: https://www.cms.gov/files/document/2024-official-icd-10-pcs-coding-guidelines-updated-12/19/2023.pdf

[8] Center for Medicare & Medicaid Services. (20 August, 2025). *CMS.gov*. Retrieved from Healthcare Common Procedure Coding System (HCPCS): https://www.cms.gov/medicare/coding-billing/healthcare-common-procedure-system

I. **CPT Category I**: codes are five numbers with descriptive terms used primarily to identify medical services and procedures provided by physicians and other health care professionals for which they bill public or private health insurance programs. Codes are 5 numeric digits.

II. **CPT Category II[9]:** codes are used for performance management. Codes consist of a single alphabetical letter followed by 4 numeric digits.



HCPCS Level II codes are a standardized coding system that is used primarily to identify products, supplies, and services not included in the CPT® codes, such as ambulance services or durable medical equipment, prosthetics, orthotics, and supplies (DMEPOS) when used outside a physician's office. HCPCS Level II codes consist of a single alphabetical letter followed by 4 numeric digits and are maintained by CMS.

Terminology

Adjudicated Claim: The claim adjudication process in medical billing is when the insurance payer reviews a claim submitted by the healthcare organization and determines the extent of their responsibility to pay for the medical services by reviewing the claim and determining benefit requirements and coverage. This allows the payer to adjust the amount they will pay to the healthcare provider, but could also lead to a significant reduction of the billed amount or denial of the claim altogether. The determination and amount owed are given to the healthcare provider[10].

Bundled Payments for Care Improvement (BPCI) was an initiative by the CMS which linked payments for the multiple services beneficiaries received during an episode of care. Traditionally, Medicaid would make payouts to each individual service throughout a course of treatment leading to fragmented care and minimal coordination across providers leading to opaque costs incurred by patients. BPCI aimed to provide better coordination by bundling services together to create a more transparent view over services and their cost. These models aimed to increase quality and care coordination at a lower cost to Medicare[11].

---

[9] American Medical Association. (2025). *ama-assn.org*. Retrieved from Organization of CPT® Codes: https://platform.ama-assn.org/ama/#/documents/cpt/cpt-readme

[10] One Source Medical Billing. (2025). *onesourcemedicalbilling.com*. Retrieved from What Is Claim Adjudication? A Critical Phrase In Medical Billing: https://onesourcemedicalbilling.com/what-is-claim-adjudication-in-medical-billing/

[11] Center for Medicare & Medicaid Services. (12 April, 2021). *CMS.gov*. Retrieved from Bundled Payments for Care Improvement (BPCI) Initiative: General Information: https://www.cms.gov/priorities/innovation/innovation-models/bundled-payments

Appendix B: Data Processing


```
"name": [
  {
    "family": "Legros616",
    "given": [
      "Dirk334"
    ],
    "text": "Dirk334 Legros616 ([max 10 chars of
      first], [max 15 chars of last])"
  }
],
"resourceType": "Patient"
```

```
"name": [
  {
    "family": "Runolf",
    "given": [
      "M."
    ],
    "text": "M. Runolf ([first initial], [max 6
      chars of last])"
  }
],
"resourceType": "Patient"
```


Figure 15: Different claim entries represent names differently. Full name (left) initials (right)

Appendix C: Unsupervised Learning Features List

gender	G0158	G0153	G0300	99221	G0157	HCPCS_level_2
age	G9829	H2000	G0155	G9708	G0299	HCPCS_level_1
number_of_claims	G8946	C8905	S9473	G8159	99241	Outpatient_PPS
preventative_care_ind	T1021	G9573	S9122	G8111	G0154	National_Codes_Established_for_State_Medicaid_Agencies
	S9126	G0458	G0129	Q5001	S9131	Temporary_National_Codes_(Non-Medicare)
	G0152	G0424	G9858	G0156	S0605	Temporary_Codes
	G9572	S8075	S9129	G0151	G0464	Alcohol_and_Drug_Abuse_Treatment
	G9857	C8908	G0444	T1502	G0402	Evaluation_and_Management_(E/M)_Codes_
	G0107	G0102	C8928	G9833		Procedures/Professional_Services

Appendix D: Additional Supervised Learning Plots

Individual Model Confusion Matrix and Fold Statistics

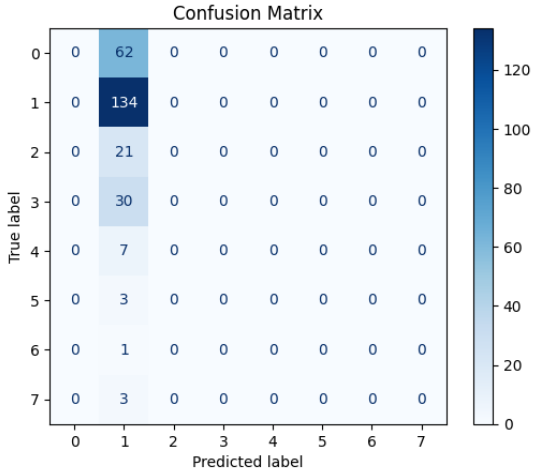


Figure 16: Baseline model confusion matrix

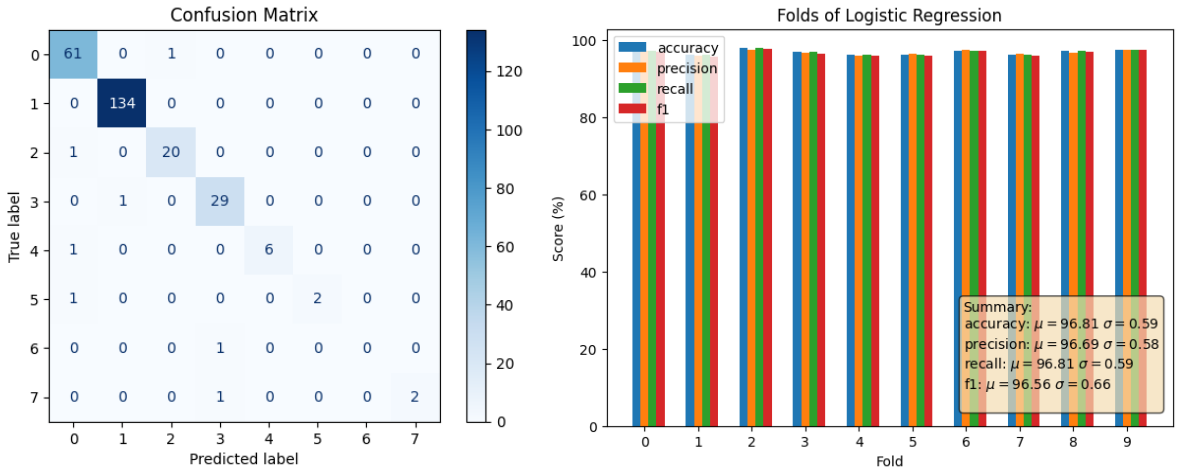


Figure 17: Logistic regression model confusion matrix and K-fold results

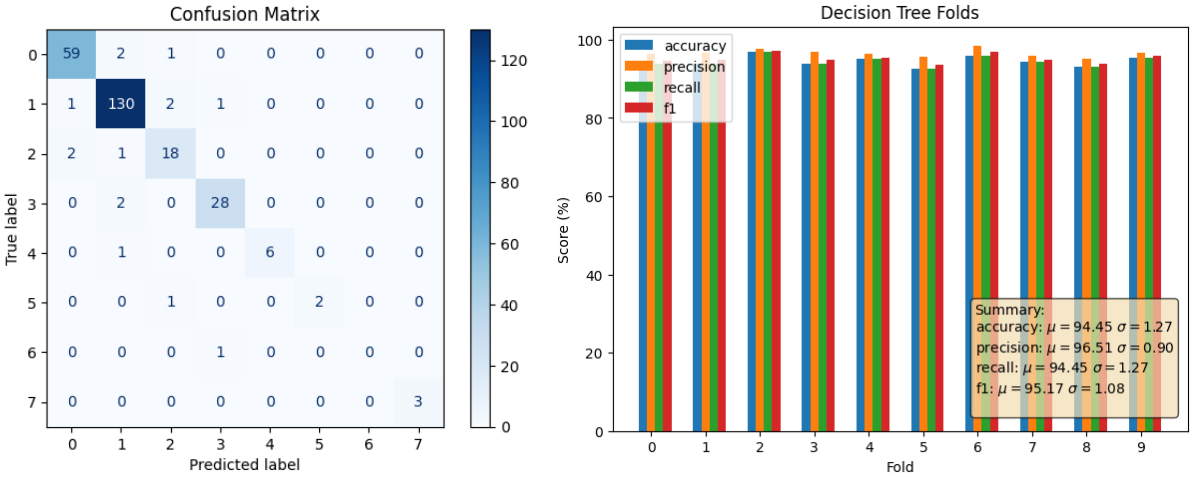


Figure 18: Decision tree model confusion matrix and K-fold results

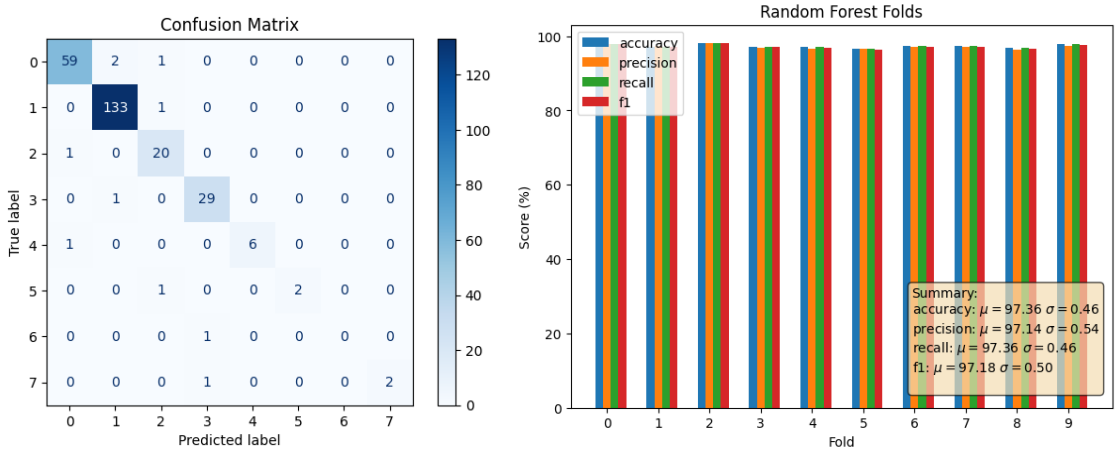


Figure 19: Random forest model confusion matrix and K-fold results

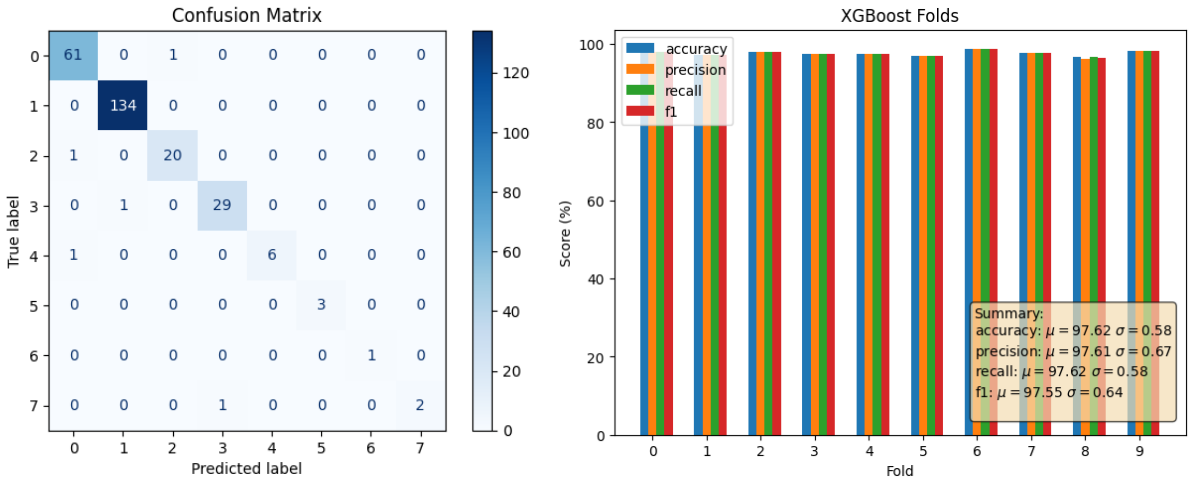


Figure 20: XGBoost classifier model confusion matrix and K-fold results

Feedforward Neural Network Training Curves

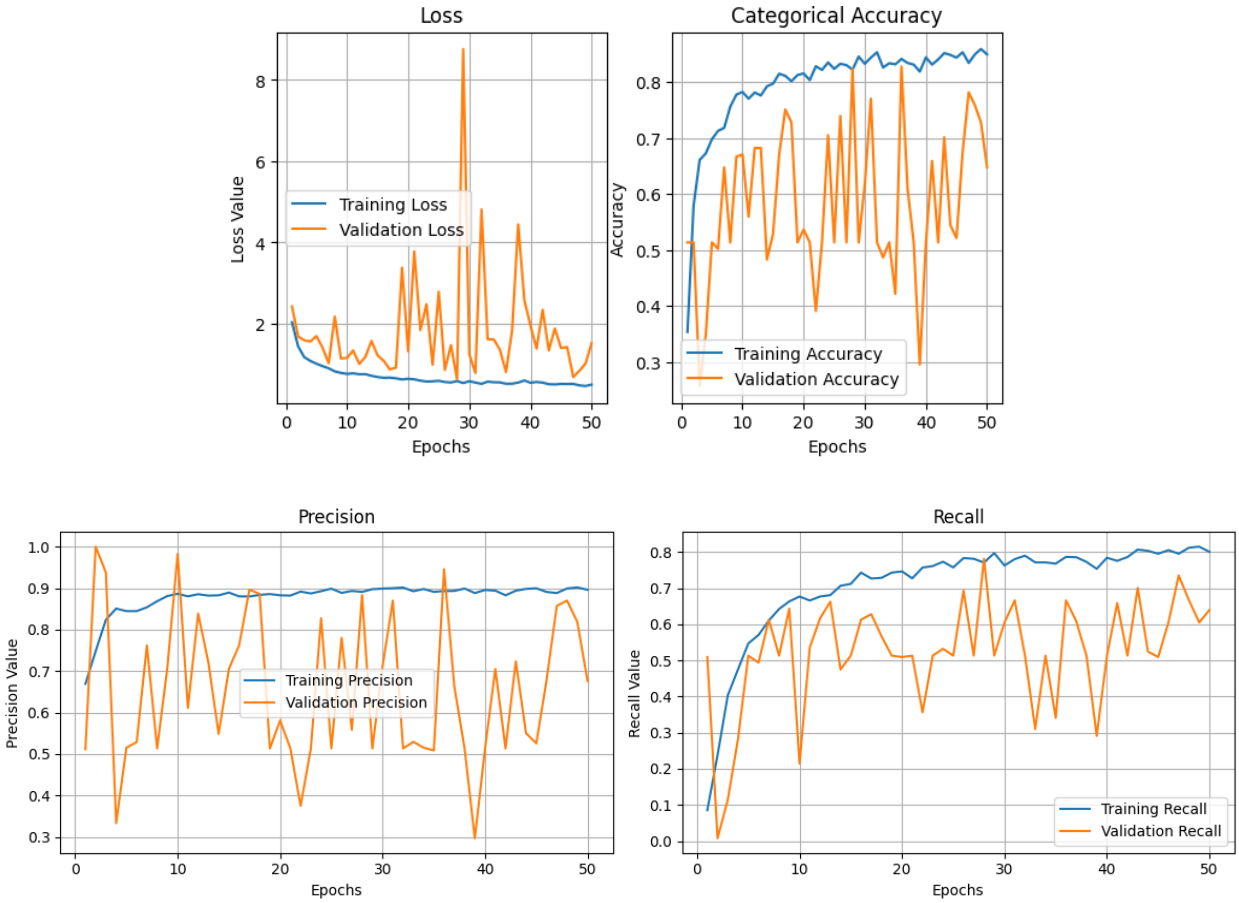


Figure 21: Training and validation metrics across epochs

Feature Importance Plots

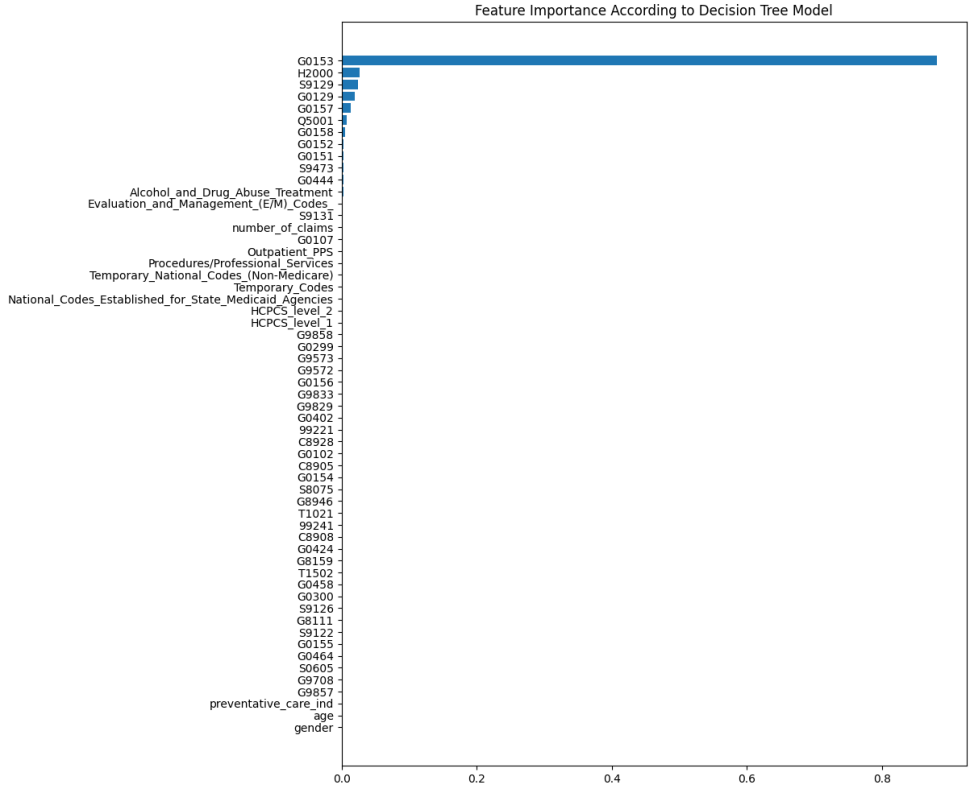


Figure 22: Feature importance plot from decision tree model

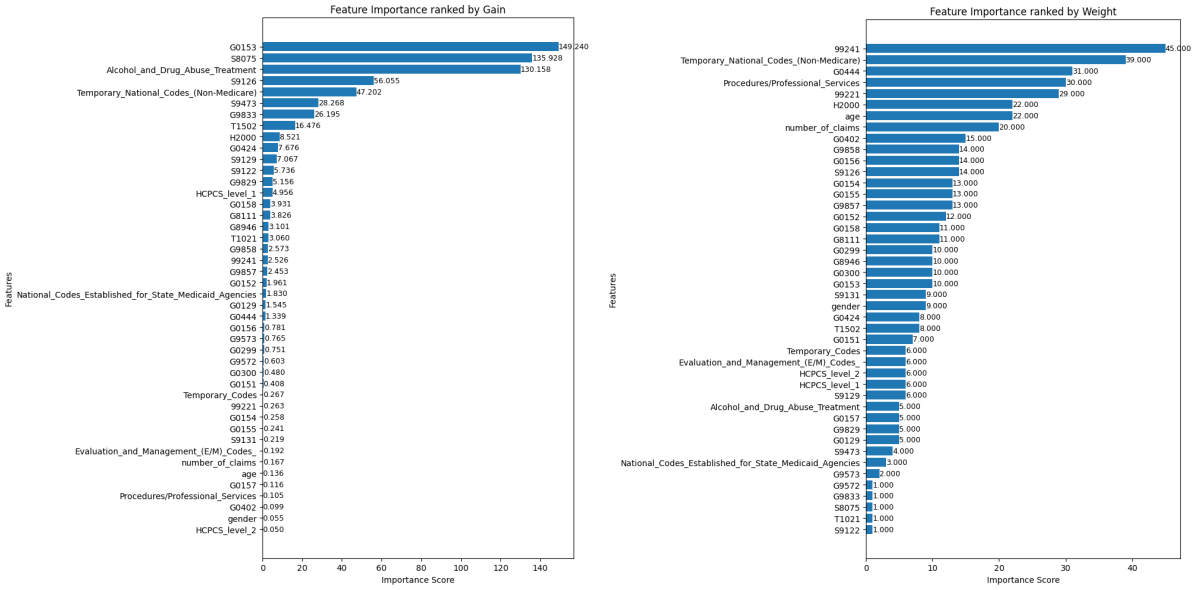


Figure 23: Feature importance plots from XGBoost classifier for 2 different split conditions gain and weight

Final Model Confusion Matrix

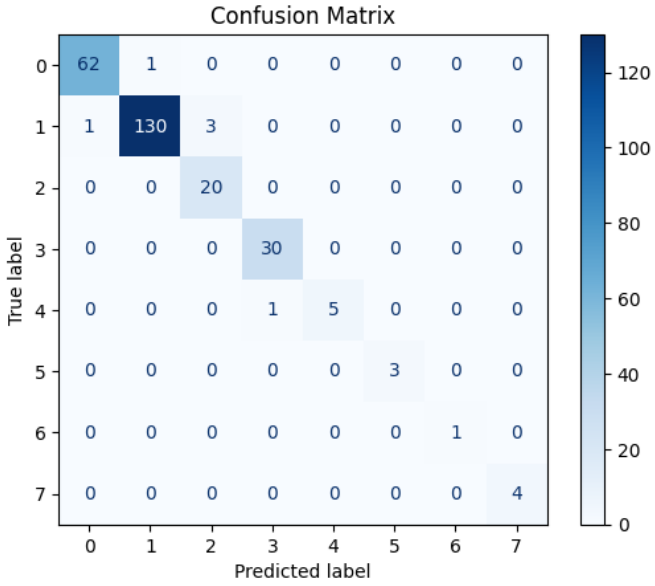


Figure 24: Optimised XGBoost classifier model confusion matrix