Predictive Modeling with Medicare Data

Machine Learning for Cities: Final Project

Adelle Lin, Yuta Okazaki, and Andrew Fair

Table of Contents

Introduction & Purpose

Medicare is the United States' national social health insurance program for older adults. Medicare serves approximately 50 million Americans, the majority of which are 65 and older (some younger individuals with disabilities are also served).

Machine learning can be applied in healthcare to identify patients at risk for specific health outcomes. We analyzed a public-use Medicare dataset to attempt predictive modeling for several outcomes: diabetes, depression, and high-cost. We selected diabetes due to its widespread prevalence in the United States (much of it undiagnosed)[1], and the significant economic toll it takes on our healthcare system[2]. We chose depression because its co-occurrence with physical health conditions has been well documented[3], and we wanted to see if we could predict its occurrence based on physical and demographic features in our dataset. We sought to predict high-cost individuals because we believe a predictive algorithm for the costliest patients would be useful to insurance companies wishing to curb expenditures via preventive programs. We also initially attempted predictive models for death and end-stage renal disease, although we abandoned these efforts for reasons we will address in a following section.

Our hypothesis was that we could use machine learning models to effectively predict diabetes, depression, and high-cost patient outcomes.

Approach

Description of Data

The Centers for Medicare and Medicaid Services (CMS) makes available a synthetic public-use dataset[4] representing a 5% sample of 2008 Medicare beneficiaries, with their insurance claims from 2008 to 2010. This dataset is fully anonymized and "synthesized", meaning that while the data are based upon real beneficiaries, they are altered in such a manner to make them unlinkable to actual beneficiaries. CMS makes this dataset publicly available, primarily for data entrepreneurs to use for application development and research training purposes - indeed, the dataset is known as the Data Entrepreneurs' Synthetic Public Use File, or DE-SynPUF. The dataset represents more than 2 million individual beneficiaries, and includes beneficiary summary information, inpatient claims, outpatient claims, carrier claims, and prescription drug events.

While we initially planned to use the entirety of the DE-SynPUF data, it became apparent that running our models on >2 million patient records would tax our laptops unduly. Fortunately, CMS provides the dataset in 20 random subsamples. We selected the first of these subsamples, representing 116,352

---

[1] http://www.cdc.gov/diabetes/data/statistics/2014statisticsreport.html
[2] http://hcup-us.ahrq.gov/reports/statbriefs/sb160.jsp
[3] http://www.nyc.gov/html/doh/downloads/pdf/chi/chi26-9.pdf
[4] https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs/DE_Syn_PUF.html

patients. We used only the Beneficiary Summary table, which includes demographic characteristics; indicators for 11 chronic illnesses; reimbursement amounts for inpatient, outpatient, and carrier claims; and months of Medicare coverage. DE-SynPUF includes Beneficiary Summary tables for 2008, 2009, and 2010.

Data Storage and Pre-Processing

We loaded the dataset into a PostgreSQL database, and served it to Jupyter notebooks via ODBC connection. Initial pre-processing was performed on the fly while loading in the data, by way of SQL query. A death indicator was constructed by checking for patients with a listed death date in 2009 or 2010, so that 2008 data could be used predictively. We derived an age field from the listed date of birth, and we aggregated cost fields into sub-totals for inpatient, outpatient, carrier claims, and total costs.

Methods: Predictive Models Used

We ran four predictive models on our data. First, we constructed a naive Bayes classifier mixed model to accommodate both discrete and continuous inputs, adapting code provided in class; we used a training set to construct a probability distribution to apply against a test set. Next, we ran our data through a linear support vector machine (SVM) using sklearn, using a validation set to identify an optimal C value. We then applied a neural network classifier against our training and test set, using code furnished by the course instructor. We additionally ran a random forest model, using sklearn. Finally, we constructed a simple heuristic model from scratch.

To simplify evaluation of our various models, we created a module called evaluate.py to output a set of standarized set of evaluation results. Specifically, this function calculates outputs the following: true positives, false positives, true negatives, false negatives, accuracy, sensitivity/recall, specificity, precision, and F1 score. Additionally, for reference, it also outputs the true rate of the outcome of interest in the sample. Values we paid close attention to are sensitivity - out of all patients with diabetes, how many did the classifier classify correctly - and precision - out of the individuals whom the model predicted diabetes, how many were classified correctly.

Data treatment before running the models

Before applying the features into the models, we created dummy data for the demographic data and rescaled the medical expense data through whitening - for both the linear support vector machine and the random forest classifier. In addition, we sampled 10% of the entire dataset to enable the SVM workable in our personal computers where computing resource is not enough to deal with all data. Before running the Random Forest Classifier, we also converted the continuous data - coverage, inpatient, outpatient and carrier costs into binary variables by splitting the data at the median values.

In testing our models, we employed a 67%/33% training/test split.

Sub-Projects by Outcome

Outcome 1: Death and End-Stage Renal Disease

We focused our initial efforts at predicting death and end-stage renal disease. End-stage renal disease is an indicator available directly in the DE-SynPUF dataset. We created our own death indicator based on the number of people that were alive in 2008 and were deceased in 2010. Therefore we are predicting for death within two years of the 2008 record. However, we abandoned these two analyses early. Both of these were quite rare conditions, and we were initially pleased to find that, for instance, we were able to predict a death indicator with 95% accuracy. However, it quickly became apparent that our models were simply predicting that nobody dies, and were 95% accurate because 95% of the sample in fact did not die. Sensitivity was 0%. We subsequently focused our analyses on outcomes with higher prevalence in our sample - ie, we tried to avoid highly imbalanced classes.

Outcome 2: Diabetes

Unlike death and end-stage renal disease, diabetes are more balanced class, having 38% of diabetes data in the same dataset. We choose demographic data (age and race), chronic illness indicators (e.g. Alzheimer's and heart failure) excluding diabetes, medical expense data, as our features to predict whether a patient is diabetic or not.

Among the models, the random forest classifier performs the best, predicting diabetes 79.3% accuracy, 74.4% sensitivity, 82.1% specificity and 0.731 F1 score. As described below, other three models work well too, although they are slightly inferior to the random forest in this dataset. Based on the criteria, the random forest classifier earns the highest sensitivity and the modest precision, leading the highest F1 score.

Naive Bayes Classifier	SVM	Neural Network Classifier	Random Forest Classifier
Accuracy = 0.7949 Sensitivity = 0.6875 Specificity = 0.8597 Precision = 0.7474 F1 = 0.7162	Accuracy = 0.7956 Sensitivity = 0.6902 Specificity = 0.8601 Precision = 0.7515 F1 = 0.7195	Accuracy = 0.7812 Sensitivity = 0.7030 Specificity = 0.8289 Precision 0.7145 F1 = 0.7087	Accuracy = 0.7932 Sensitivity = 0.7442 Specificity = 0.8231 Precision = 0.7193 F1 = 0.7316

This random forest classifier can be simpler. We utilize a feature selection function from sklearn to extract some important features, finding 7 features out of 24 features that are more important than others. The features are carrier medicare, outpatient expense, Alzheimer's, heart failure, kidney disease, depression, and ischemic heart disease, all of which have higher importance score than the average of all the features. Fitting the model with the limited 7 features, the random forest classifier predicts better than the original, earning 79.9% accuracy, 76.8% sensitivity, 71.9% precision, and 0.743 F1 score. This means that, in practice, the model works well with less data, which helps us compute the prediction

faster: while the original random forest classifier takes 57.8 seconds to fit the data, the simple takes only 29.2 seconds.

Outcome 3: Depression

Depression was a trickier case due to the lower occurrence in patients - 21.24% however we still felt that this warranted some analysis due to the difficulty / subtleties in actually diagnosing depression[5]. We also chose demographic data (age and race), chronic illness indicators (e.g. Alzheimer's and heart failure) excluding depression, medical expense data, as our features to predict whether a patient is experiencing depression.

The results from all the tests showed a similar accuracy between 75% and 80% however sensitivity fluctuates pretty wildly, between 7% to 62%, indicating that it is hard to provide an accurate prediction on depression as a true positive. Another way to interpret this is that probability-based predictions work better for this subset of the data. However specificity is very high, in the high 90%s for SVM, Neural Network Classifier and Random Forest Classifier which indicates we could potentially predict with confidence if a person does not have depression.

Naive Bayes Classifier	SVM	Neural Network Classifier	Random Forest Classifier
Accuracy = 0.7694 Sensitivity = 0.6170 Specificity = 0.8105 Precision = 0.4675 F1 = 0.5319	Accuracy = 0.7888 Sensitivity = 0.1093 Specificity = 0.9797 Precision = 0.6013 F1 = 0.8744	Accuracy = 0.7891 Sensitivity = 0.0685 Specificity = 0.9831 Precision 0.5215 F1 = 0.1211	Accuracy = 0.8021 Sensitivity = 0.2786 Specificity = 0.9431 Precision = 0.5685 F1 = 0.3740

Using feature selection to extract important feature, 9 out of the 24 were selected, mostly the chronic diseases and also the costs. Comparing it to the feature selection through SVM, the common features are inpatient and carrier costs. Demographic data (neither age nor sex) did not seem to be much of an predictor for depression. Out of the chronic diseases, diabetes and alzheimer's were selected as the most prominent feature, which would also make sense intuitively.

Fitting the model with the limited 9 features, the Random Forest Classifier gave a very similar result to the original. Which means that in practice the model would work the same with less data, and would help compute the prediction faster. predicts better than the original, earning 79.9% accuracy, 76.8% sensitivity, 71.9% precision, and 0.743 F1 score. This means that, in practice, the model works well with less data, which helps us compute the prediction faster: while the original random forest classifier takes 34.5 seconds to fit the data, the simple takes only 16.9 seconds.

[5] http://www.ncbi.nlm.nih.gov/pubmed/9054902

Outcome 3: High-Cost

A high-cost indicator was constructed in the following manner: All annual costs were summed for each patient (inpatient + outpatient + carrier claims costs), and pandas was used to assign a percentile rank to each patient according to total annual cost. Patients with total costs exceeding the 80th percentile were given a value of 1 for high-cost, while the rest were assigned 0.

Models to predict high-cost were run against demographic features (sex, race and age) plus 11 chronic illness indicators (Alzheimer's, heart failure, kidney disease, cancer, chronic obstructive pulmonary disease, depression, diabetes, ischemic heart disease, osteoporosis, arthritis, and stroke). Age was the only continuous predictor; in instances where models required categorical inputs, age was split at the median.

The same models as above were applied: naive Bayes classifier, SVM, neural classifier, and random forest classifier. While random forest (with 2000 trees) and SVM result in the greatest accuracy, we consider naive Bayes classifier to yield the best performance due to its improved sensitivity/recall. We operate on the assumption that it is important to identify as many high-cost individuals as possible, pointing to the importance of high sensitivity.

Naive Bayes Classifier	SVM	Neural Network Classifier	Random Forest Classifier	Heuristic Model (threshold=3)
Accuracy = 0.8528 Sensitivity = 0.7637 Specificity = 0.8751 Precision = 0.6050 F1 = 0.6752	Accuracy = 0.8740 Sensitivity = 0.5848 Specificity = 0.9489 Precision = 0.7476 F1 = 0.6562	Accuracy = 0.8625 Sensitivity = 0.5420 Specificity = 0.9357 Precision = 0.6582 F1 = 0.5945	Accuracy = 0.8740 Sensitivity = 0.5893 Specificity = 0.9456 Precision = 0.7314 F1 = 0.6527	Accuracy = 0.8453 Sensitivity = 0.7343 Specificity = 0.8731 Precision = 0.5914 F1 = 0.6551

We decided to construct a heuristic model to see if we could accurately predict high-cost using a very simplified model. Sci-kit learn was used to extract the top 5 contributing features from the random forest model, and these 5 features were used to create a simple, summative heuristic model for high cost. Flat feature weights of 1 were used, with a maximum possible score of 5. The model was run through a loop of score thresholds from 1 through 5. Optimal results were obtained for a threshold of 3 (see table above). The heuristic model results appear comparable to naive Bayes classifier results.

Conclusions & Practical Implications

To assess the performance of our models, we can imagine a very naive model as a yardstick to measure against. A naive model might pick the same outcome in each case. For instance, it might always pick a negative outcome (not diabetes, not depression, not high-cost). Such a model would be 62% accurate in the case of diabetes, 79% in the case of depression, and 80% in the case of high-cost (as the prevalences of these outcomes are 38%, 21%, and 20%, respectively). We might consider our models successful if they exceed these baseline levels of accuracy.

Two of our three models surpass this baseline: diabetes and high-cost. This might suggest that diabetes and high-cost are more highly correlated with the predictors in our models than is depression. It might be that it is simply impossible to accurately predict depression based on only a simple set of demographic features and chronic illness indicators. It is also important to keep in mind that our models are based on cross-sectional data, and we cannot conclude causality or even temporal order of events from our results.

The performance of our models could be of practical use to potential clients such as healthcare providers and insurance companies. For instance, a hospital or other provider might be interested in a predictive algorithm for diabetes, in order to identify individuals at risk for this condition and either conduct diagnostic testing or start them on a preventive course. An insurance company might be interested in a model to predict high-cost patients, in order to implement a preventive program for a targeted selection of patients to curb costs. A heuristic model as described above might be particularly desirable to clients, as it is extremely easy to implement.

Which model to select for each given application is a subjective matter requiring domain expertise. For instance, a practitioner wishing to use machine learning to identify patients at risk for depression might prize a naive Bayes classifier model, which has the highest sensitivity, because it is essential to identify as many people who have depression as possible to avoid potential self harm. On the other hand, one might choose a model with both high sensitivity and specificity to predict an outcome like cancer, as it would be undesirable to send a patient for expensive diagnostic testing unnecessarily.

Limitations

The limitations in our dataset pertain to a few issues around the ability to run larger sets of data. As mentioned, for our SVM models, we took a smaller subset of the data in order to be able to process the data in a timely manner. This could potentially reduce the effectiveness of the model, especially for a disease like depression that is a rarer occurrence and might require larger sets of data to produce a better prediction.

By using the summary tables, we also only ended up using a small subset of the data actually available. For example, by using the summary for inpatient claims, we did not use data such as claim utilization day count, discharge date etc. which potentially could have made our predictions more robust. There are almost 300 features that have not been included in our analysis by purely using the summary tables.

Another limitation relates to the method that we chose to create binary data to run the Random Forest Classifier. We used the median point to split the data for continuous data categories where a more discerning method could be used for each feature where this occurred, such as within the cost categories.

Next Steps

As for next steps, we might consider using more of the dataset, adding more features into the dataset, and then introducing a validation process to select the best model. First, as we described in the limitations section, we need to apply our models to other samples from the DE-SynPUF dataset, each of which has about 120,000 records. To deal with these huge datasets, we could utilize more powerful computers such as CUSP's High Performance Computer (HPC) and its workstation. HPC is a Hadoop based computing network which is capable of distributing the large volume of data to many nodes and of computing it in parallel.

Secondly, we might add the variety of datasets by procuring other datasets from third parties. For instance, a life insurance provider may be interested in providing us with its data if it could predict an appropriate premium for a patient based on healthcare information. If that is the case, a database system, like PostgreSQL, comes into play in storing datasets permanently and in extracting and transforming them, known as Extract. The system could quickly combine dataset A with dataset B, aggregating their values, which would minimize our efforts in preprocessing the data in a more manual fashion.

The last step we would consider is a validation process. In our project, we split our dataset into a training and a testing set, and then we train models in the training set and find the model that performs the best on the testing set. Yet, in order to find the best model, we need to introduce the validation process between training and testing process. Adding 10 datasets as validation datasets, for instance, would facilitate us calculating the average accuracy for each model running with them. The model which earns the highest accuracy could be the best in these datasets.

Individual Contributions

Yuta developed the training and testing frameworks for our main machine learning models: naive Bayes classifier, SVM, and neural network classifier. He developed a composite naive Bayes Classifier function to accept both discrete and continuous variables seamlessly. Yuta focused particularly on the diabetes outcome analysis. He devised the next steps that we would pursue if we were to continue our project efforts.

Andrew set up the PosgreSQL database which allowed us to more efficiently extract data to be run through our various models. He aggregated the various cost datasets, which also increased efficiency of processing the data. Andrew focused on predicting cost as a label. Further to the 4 models that the group used to analyse the dataset, he also developed a heuristic model to validate the results that he had found, confirming that a simpler model can be used to produce accurate predictions. He laid out measures for evaluating the effectiveness of our ML models by developing a model evaluation function. Andrew brought domain knowledge of medicare data, drawing from his professional experience in public health, and initiated sharing of project information via Slack.

Adelle had put the group together early in the semester and worked with both Yuta and Andrew to test the various models and databases that they had developed. She also tested various methods of data treatment methods within each model and analysed the limitations of the project. Adelle single mindedly focused on trying to predict depression, she had dreams that machines could help society reach a utopian state where no one needed to feel alienated. But ultimately, humans are better at predicting depression and a good hug can go a long way.