Benjamin Spiegel

Medicare Capstone Milestone 2

10/09/2019

I obtained the data for my dataset from cms.gov using two different csv files for the year 2016. The first file was titled ‘[medicare-physician-and-other-supplier-national-provider-identifier-npi-aggregate-report-calendar-year-2016.csv](http://localhost:8888/edit/Desktop/springboard/Springboard-20190719T193509Z-002/Springboard/Kagle_Datasets/Medicare_NPI/medicare-physician-and-other-supplier-national-provider-identifier-npi-aggregate-report-calendar-year-2016.csv).’ It included 55 different features for all Medicare contracted doctors. These features provided information ranging from professional background of physicans and on demographic data of each doctor’s patient population. This demographic information included percentage of patients for each doctor who fit into demogaphic groups (age, race, income), and each doctor’s percentage of patients with specific health designations and conditions. Because the files were CSV files I used the ‘pd.read\_csv()’ function.

The second file titled ‘ [Part D Prescriber PUF NPI Drug, CY2016](http://download.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/PartD_Prescriber_PUF_NPI_DRUG_16.zip)’ included drug prescription data for every medication that a doctor prescribed over the year 2016. Consolidating the data to the most useful features I used pandas *iloc* function and pandas *drop* function to create a new dataframe including only ‘total\_drug\_cost’ (the total cost of medicare prescription) ’total\_30\_day\_fill\_count’ (total number of monthly prescriptions filled), ‘specialty description’, and each doctors’ ‘National Provider Identifier’ .

Using pandas’ groupby function I combined all data for different drugs into one row per doctor. By doing this I now had data on each doctors’ total number of 30 day prescriptions written and total cost to medicare of all medications written using medicare’s part D drug plan. After combing all of the rows from the Part D drug based dataframe described above into one row for each doctor, I merged this dataset with the first described dataset using pandas’ function ‘dataframe.merge()’. The code for this is shown below:

|  |
| --- |
| dr\_cost = dr\_cost.groupby('National Provider Identifier')  drug\_sum = dr\_cost.sum()  drug\_sum = drug\_sum.merge(dr\_df, how = 'inner', on = 'National Provider Identifier') |

After merging the files, I created a dictionary with each medical specialty as a key and each specialty’s respective average ‘total\_drug\_cost’ as a value. By using the map function, I then created a feature that was equal to mean prescription cost of each physician’s specific specialty. I created this feature as an initial method for recognizing and calculating the relationships between specialty and total drug costs.

|  |
| --- |
| drug\_avg = df\_fill.loc[:,['specialty\_description','total\_drug\_cost']]  drug\_avg = drug\_avg.groupby(by ='specialty\_description').median()  nested\_dict = drug\_avg['total\_drug\_cost']  nested\_dict = nested\_dict.to\_dict()  dr\_df['specialty\_description\_vector'] = dr\_df['specialty\_description'].map(nested\_dict) |

Although creating a feature equal to the median annual drug cost of a doctors’ specialty was helpful in predicting physicians drug cost, I wanted to take it a step further and make sure the machine learning algorithm was able to handle the specialties as categorical data. As there were over 70 different specialties in the data, the top algorithms could not handle features with this many categorical variable. My solution to this was to filter the data so that only the 10 specialties were included. I chose to use the 10 specialties with the most doctors prescribing over 3,000 medications. I did this because I wanted to choose specialties with the largest number of doctor with meaningful data. As opposed to simply the largest number of doctors who had made any prescription. This avoided working with specialties with a large proportion of physicians who were creating random noise due to their lack of prescription volume.

Once I had filtered the data to ten specialties I then used pandas’ get\_dummies function to create dummy variables for each of the 10 specialties so that my algorithms would be able to recognize relationships between specialties and other things. Below is the code used to implement the steps described above

|  |
| --- |
| **from collections import Counter**  **spec\_df = df\_fill[df\_fill['total\_30\_day\_fill\_count']>=3000]**  **V\_count = Counter(spec\_df.specialty\_description).most\_common(10)**  **spec\_list = []**  **for object in V\_count:**  **spec\_list.append(object[0])**  **top\_spec = df\_fill[df\_fill.specialty\_description.isin(spec\_list)]**  **dummy\_df = pd.get\_dummies(top\_spec['specialty\_description'])**  **df\_fill = pd.concat([top\_spec,dummy\_df], axis = 1)** |

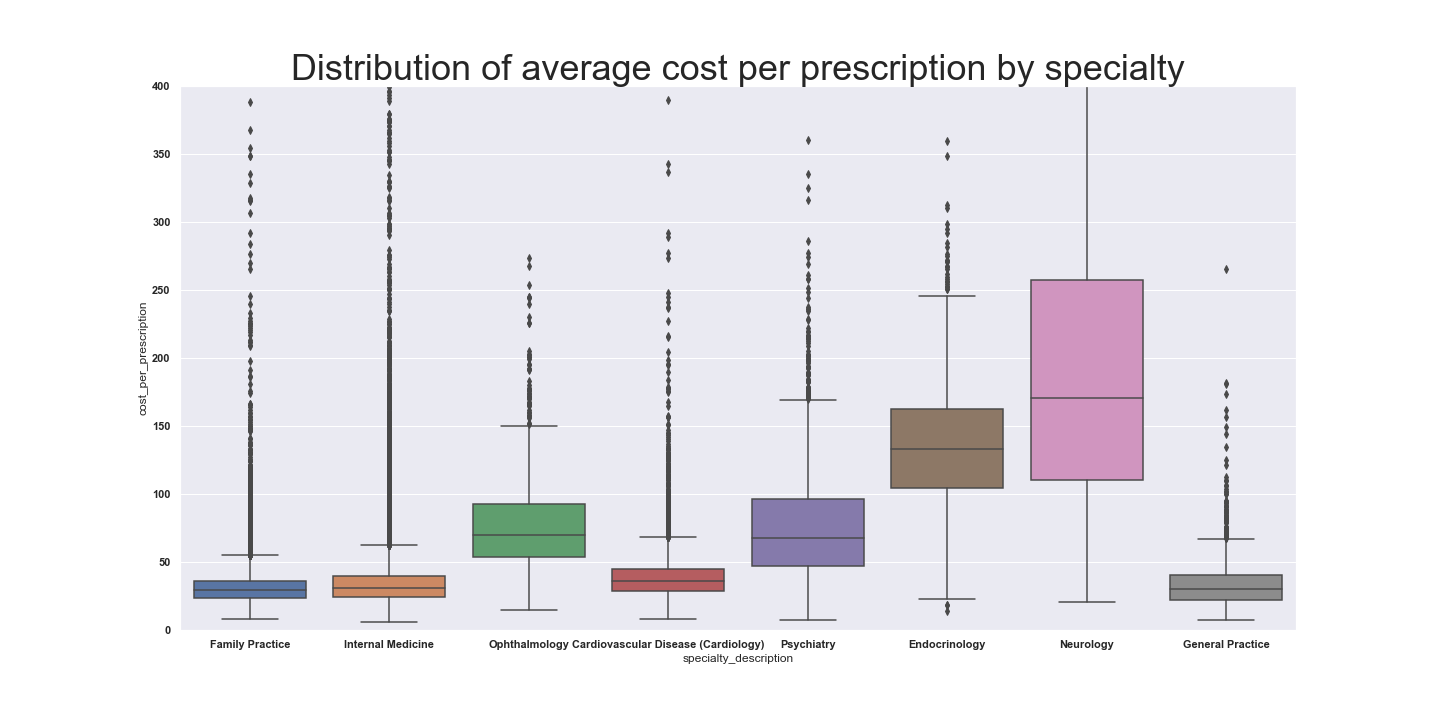
Next I dealt with missing values. I did this because of the inability of the machine learning algorithms to run data with nan values. In order to do so, I used the pandas’ function ‘dropna()’. I created a threshold of 44 values as the minimum amount of values a row could have without being dropped. This decreased the DataFrame from 622705 to 397766 rows. For those rows that were not dropped but still had na values, I ran pandas’ ‘fillna’ function. I replaced each na value with the median value of it’s column. The code used is below:

|  |
| --- |
| Df = df.dropna(thresh = 44 )  df = df.fillna(df.median()) |

Before using any machine learning on the dataset needed to replace any categorical values with numerical values. I did this for the feature ‘Gender of the Provider’. I created a dictionary (genderdict) where Male and Female were keys with values of 0 and 1 respectively. I then used remapped the feature with the code dataframe[‘Gender of the Provider’].map(genderdict).

***Data Exploration***

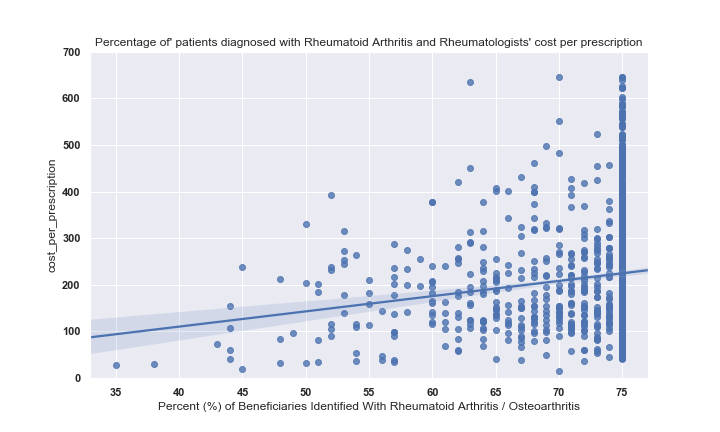
After obtaining and organizing the data I looked at Data exploration. The first visualization I created of the data, looked at how distribution of drug costs differed according to medical specialties.

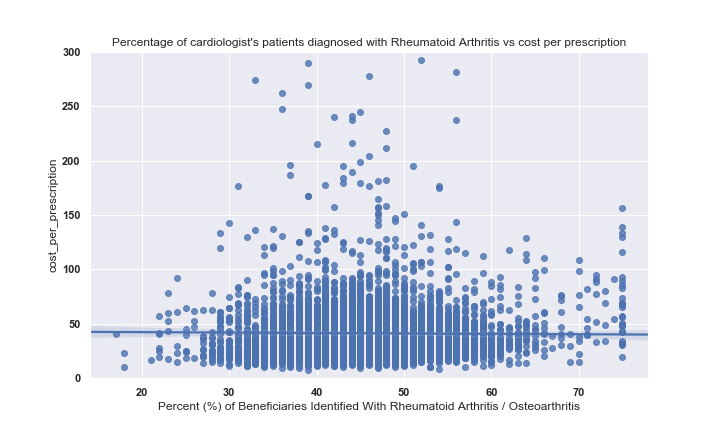


Looking at the plot, one can see that a doctor’s medical specialty has a large bearing on range of the average cost per prescription he/ she writes. Of the eight specialties in the plot, it can be seen that Neurology and Endocrinology have the highest average prescription costs. Similarly, it can be seen that Psychiatry and Ophthalmology are also quite high. On the other hand, primary care specialties and Cardiology had much lower average costs per prescription. This has to with the fact that many specialist write prescriptions for a larger percentage of brand name medications, medications that are often targeted to specific chronic illnesses.

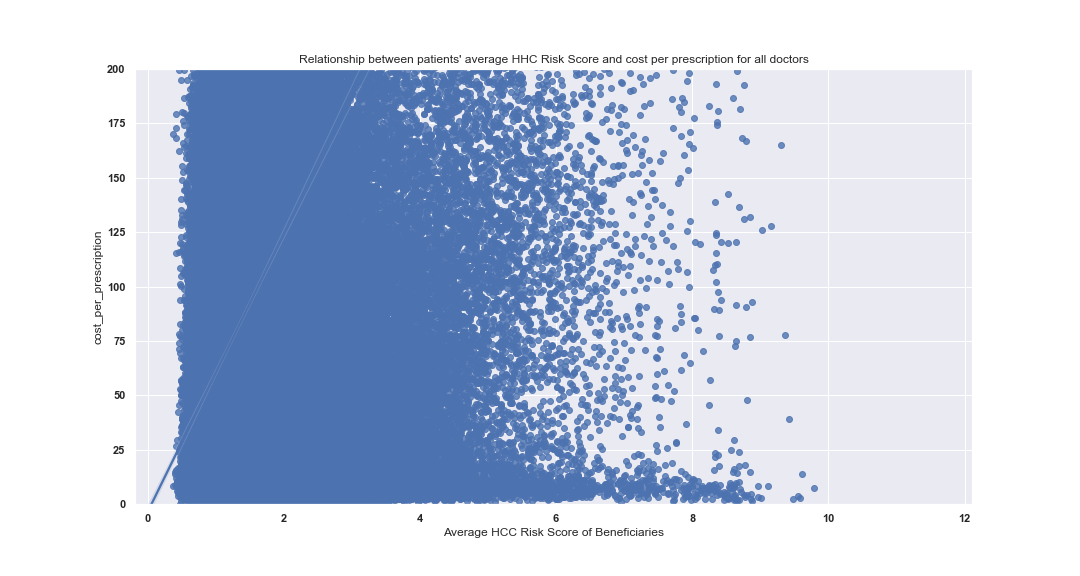
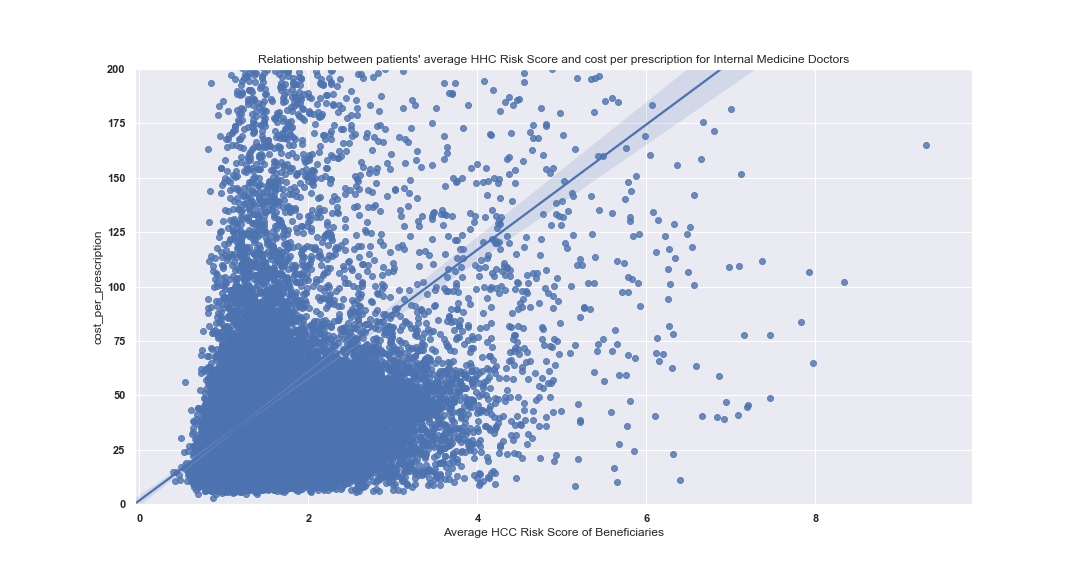
Next I tried to look at the relationship between the percentage of physician’s patients with specific health conditions and their average cost per prescription. I found however that it was necessary to isolate the distinct specialties to find a correlation. The reason for this is likely because individuals with chronic illness go to specialists that relate to their illnesses. Drugs that physicians prescribe are related to the illnesses they treat and not the other illness the patient has. Therefore, illness that specialist treat are much more correlated with the drugs they prescribe and their costs. For example, the percentage of an ophthalmologist’s percentage of patients with schizophrenia will not be correlated with prescription drug costs while a psychiatrists will. . A supporting plot of this theory can be seen in the plots below showing how the percentage of patients with Rheumatoid Arthritis effect specific specialties. While percentage of patients with Rheumatoid Arthritis has a large effect on the average prescription costs of Rheumatologists. It has very little relation to the average prescription costs of Cardiologist, who do not treat Arthritis.

Additionally, considering the extremely variability between different specialties and their average drug costs, the fact that specialty is so highly predictive with drug costs overrides and makes less clear the correlations with percentage of patients with chronic illnesses





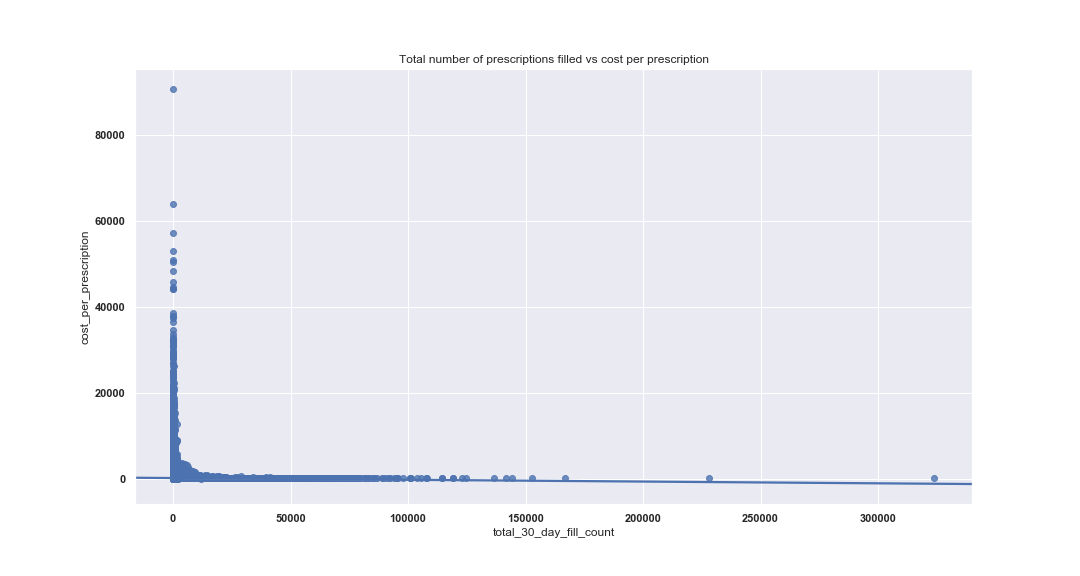
Finally, I looked at how ‘HCC score’ affects drug cost. HCC score is “a risk adjustment model that is used to calculate risk per individual on a basis of how cumulative chronic illnesses  predict future healthcare costs. Essentially specific chronic illnesses have specific HCC values depending on their average costs and adding up these different chronic illness HCC values results in a specific HCC score. Physicians with Higher average HCC scores for their patient’s population were correlated with higher average costs. Of interest, family practice physician’s average prescription drug cost correlation with HCC is weaker than that of all doctors combined. One could presume this is because patients who are very sick are going to be getting much of their care and prescription from specialists. Below are two charts demonstrating the relationships described above.



Supporting the theory above, you can see that the slope for cost per prescription is higher when comparing HCC risk score without filtering doctor types, while internal medicine (primacy care physicians) had a much weaker slope.

I also found that to demonstrate the correlations of chronic illness to average cost per prescription it was helpful to filter out doctor’s who barely prescribed any prescription. This could be a result of these doctors prescribing so little medications that just one or two expensive medications could dramatically increase the average costs of the medications the prescribe as a whole.

Additionally, a doctor prescribing so few medications is less important to the underlying problem of high drug costs because they prescribe so little and make up a small amount of overall drug costs. Regardless of the reason it is clear that physicians who prescribed very little medications are more likely to have significantly higher average drug costs. Below is a plot demonstrating such.



The plot above demonstrates the noise by created by physicians who do not prescribe many prescriptions in Medicare’s Part D drug plan, the machine learning algorithms are very powerful and can recognizing this trend and take it into consideration when making predictions.

Recognition of the correlation between percentage of patients with a specific chronic illness and a medical specialties average drug cost was dependent on and varied depending on the specialty gave some direction on designing the machine learning model. It made clear the importance of the model’s being able to recognize specialties as categorical variables. Specifically through the creation of the dummy variables.

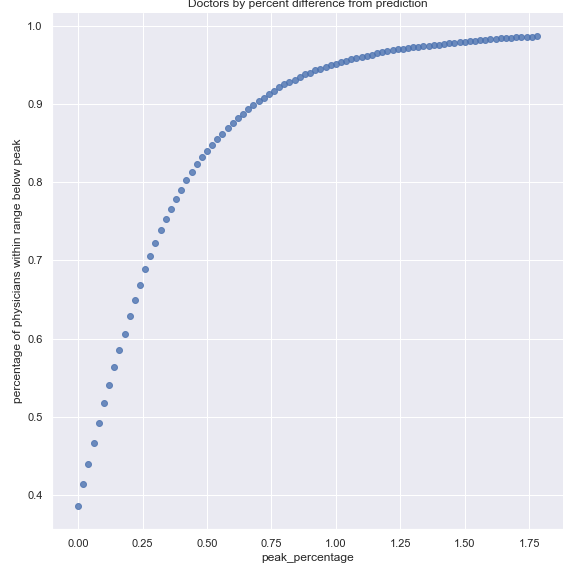
Once I had completed this, I began putting together the code for scikit learn’s RandomForestRegressor. First I ran GridSearchCV to find the best hyperparameters for my RandomForestRegressor estimator given my particular data. I then split the data into training and testing subsets. I did this using the code below:

|  |
| --- |
| X = all\_med.drop('total\_drug\_cost', axis = 1)  X = X\_initial.drop('National Provider Identifier', axis = 1)  y = all\_med.total\_drug\_cost  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = .20) |

Finally, I used random forest on the data set to create predictions for the y\_values. I used RandomForestRegressor.score() to find out what my accuracy score was. The score value for RandomForestRegressor tells the Coefficient of Determination for the model. It tells how much of the variance for the dependent variable the model is able to explain by the independent variables. After running Random forest on the data we got a Coefficient of Determinitation of .65, this is a reasonable score. Below is the code described above:

|  |
| --- |
| rf **=** RandomForestRegressor**(** n\_estimators **=** **200,** max\_features **=** **30,** max\_depth **=** **25)**  rf**.**fit**(**X\_train**,** y\_train**)**  rf\_y\_pred **=** rf**.**predict**(**X\_test**)**  rf\_SCORE **=** rf**.**score**(**X\_test**,** y\_test**)**  **print(**rf\_SCORE**)** |

Finally as a way of indicating how close my prediction’s were to the actual y\_test values, I calculated and plotted the distribution of physicians actual total drug cost values compared to their predicted costs. The slope was not normal. The resulting plot and the code for said plot are below:

****

|  |
| --- |
| index **=** **-.20**  dict\_perc **=** **{}**  **for** percent **in** range**(100):**  cnt**[0]** **=** **0**  cnt**[1]** **=** **0**  perc **=** **(**percent\_dif **<** index**).**astype**(**'int'**)**  **for** num **in** perc**:**  cnt**[**num**]** **+=** **1**  value **=** cnt**[1]/(**cnt**[1]+**cnt**[0])**  dict\_perc**[**index**]** **=** value  index **+=.02**  Prescription\_plot **=** pd**.**DataFrame**.**from\_dict**(**dict\_perc**,** orient **=** 'index'**)**  sns**.**lmplot**(**x **=** 'peak\_percentage'**,** y **=**'percentage of physicians within range below peak'**,** data **=** Prescription\_plot**,** fit\_reg **=** False**,** aspect**=1,** height**=8)** |

As a means of figuring out how effective the model was, I calculated the percentage of physicians that whose total cost of prescriptions was less than a specific percentage of the prediction. Below are the values for these distinct percentages:

Coefficient of Determinitation of .65

Within 120 percent of predicted prescription drug costs = .627 percent of physicians

Within 140 percent of predicted prescription drug costs = .781

Within 150 percent of predicted prescription drug costs = .832

Within 175 percent of predicted prescription drug costs t = .911

Within 200 percent of predicted prescription drug costs = .948

Within 250 percent of predicted prescription drug costs = .979

Within 300 percent of predicted prescription drug costs = .989