CAPSTONE PROJECT 2 REPORT

On

PREDICTING WHAT PHYSICIANS SHOULD BE PAID BASED ON HISTORICAL DATA FROM THE CENTERS FOR MEDICARE AND MEDICAID SERVICES AND THE US FOOD AND DRUG ADMINISTRATION USING GRADIENT BOOSTED TREES AND RANDOM FOREST ALGORITHMS IN PYTHON

By

Andrew Ogah

A report submitted in partial fulfillment of the requirement of the

Springboard Data Science Career Track Fellowship

April 13, 2018

Introduction

In 2013, regulators introduced The Physician Financial Transparency Reports or Sunshine Act mandating that companies manufacturing drugs, devices, and biological agents report individual payments of greater than $10, or $100 in aggregate annually, provided to US physicians. This act was partly to curb any potential conflicts of interest that can inappropriately influence physician practice. This move supported the need to control altered prescription behavior caused by increased interaction with pharmaceutical representatives. Added to these, in 2010, Congress also signed into law the Patient Protection and Affordable Care Act mandating residents in the US to either buy insurance or pay a fine if not covered by an employer-sponsored health plan, Medicaid, Medicare or other public insurance programs. With all these comes a trend of increasing premiums, deductibles, and co-payments for medical services, and increasing the costs of using out-of-network health providers rather than in-network providers. People - those visiting for a routine medical checkup or in the emergency room (ER) units, want to estimate current industry average pay for physicians to know their deductibles. This need is the driver for my analysis.

Materials and Methods

Data extraction and cleaning

The dataset used in this analysis are from the 2015 Medicare Part D data of about 10 million records, which includes 99.91% of total prescription claims having prescribers with a valid NPI submitted by the Part D plan sponsors during 2015. And, the 2015 Open Payment data of approximately 11.5 million records containing physician profile ID, demographic information and amount of payment received. Center for Medicare and Medicaid Services maintains both datasets on their website.

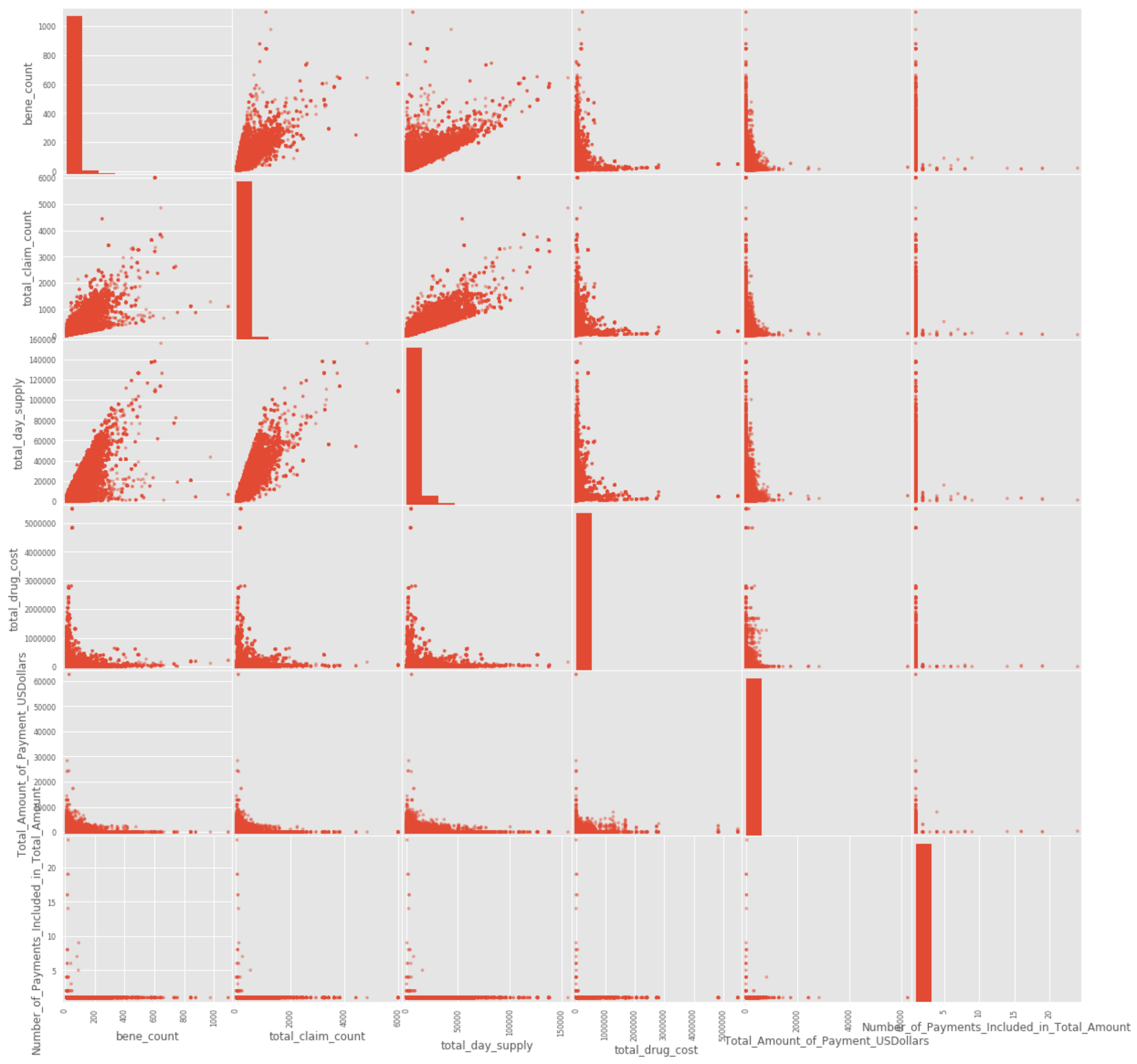
Analyzing the features of both datasets, I noticed that the IDs are not right keys for merging both in into one, because not all registered doctors received payments for a particular prescription. So, I engineered a new feature by combining the first, and last names of the physician. This new feature, in addition to the city and state where a medical practitioner is located now acted as this mapping key. I also created a new feature for the state based on regions situated, because the original variable had a high cardinality issue with uneven classes. At the time of writing this report, the drug names and medical specialties had similar problems. I contemplate using a fuzzy matching algorithm with variable edit distances to re-engineer the drug name feature.

Considering that the dataset is massive, I decided to select a random sample of 10% from the merged dataset. Further cleaning the dataset, I removed noisy columns like the first and last name, 'NPI' - national provider identifier, and 'physician profile ID,' because they are assigned values and do not represent a random measure on each observation. They introduce high dimensionality and cause unnecessary overfitting of the data. I also dropped the drug name for now because of the issue mentioned earlier to be re-added after feature engineering. The 'name of associated covered drug or biological1' feature was redundant since we already have a drug name that covers that, so dropped that too. Because the record was still too extensive for my machines, I decided to remove all unknown values. This act is not a good practice because you may have some information that could be of value in some discarded data. I will revisit those when fine-tuning the model.

At this point, it is advisable to create an out of sample test set data. This act usually prevents what is called an information snooping bias. The premise is that the human brain is known to be a fantastic pattern detection system, which means that it is highly prone to overfitting. So, we set the test set data aside that we will not touch at all, and continue with the training set data. The test data set is used as an out of sample dataset to validate the model.

Exploratory Data Analysis

Exploring the dataset a little bit and looking closely at the numerical features using some plots, we see little or no trends in the dataset. Most of the variables are either slightly left or right skewed, but nothing severe since the spread of the data points is not overly high. Skewness translates to the presence of some outliers in the distribution, which sometimes may not bode well for linear models, and may need normalization to provide better results. I have decided to leave all these in here, for now, to see how the models perform and will require revisiting if models don't meet expected outcomes.



Correlation matrix showing relationship between each predictors and the target – Total Amount of Payments

Statistical Inference

With the knowledge that most predictors have a very weak association with our target variable, the question we now have is: are these predictors significant enough in explaining variability in the Total Amount of Payment - the target variable? To answer this question, we usually fit a multivariate linear regression model to the dataset. And before doing that, we should ensure that the data, most especially our predictors meet the assumptions for the hypothesis of a linear model. Which are:

- sample must be reasonably random,
- data must be from a normal distribution or large sample (need to check $n \geq 30$),
- the observations are independent.
- the true relationship is linear. Check that the scatter plot is roughly linear and that the residual plot has no pattern.
- the standard deviation of the response y about the true line is the same everywhere. Look at the residual plot and check that the residuals have roughly the same spread across all the x-values.
- for any fixed value of x, the response y varies normally about the true line. Check a histogram or stemplot of the residuals.

Since we are trying to spend more time on prediction and analyzing the data than tweaking the abnormalities in the data, I have chosen to go the way of ensemble models that are robust against linearity, skewness, and collinearity. They also provide features that can identify which models are essential in explaining or predicting the target variables.

Building the machine learning algorithm.

My approach to this was comparing different algorithms to see how they performed using the mean squared error (MSE) as a yard stick. The MSE of an estimator measures the average of the squares of the erros or deviations- that is, the difference between the estimator and what is being estimated. Since our dataset is significantly imbalanced, care was taken to balance the training dataset before training the model because most models especially the linear ones do not perform well such. The Gradient Boosted Trees model was selected because it performed better than the other two models (Linear Regression and Random Forest model) used. We now carried out the prediction on the imbalanced out of sample data we created at the start of the analysis. This approach shows if the model generalizes well to any dataset it has not seen before, balanced or imbalanced.

The model was also cross-validated to ensure we are not overfitting the training dataset using the StratifiedKFold and KFold validation techniques that splits the training set into 10 distinct subsets called folds. Then, it trains and evaluate the model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The Scikit-Learn cross-validation features was used, which expects a utility function (greater is better) rather than a cost function (lower is better) of the MSE. Stratification is generally a better scheme, both in terms of bias and variance, when compared to regular cross-validation because it rearranges the data as to ensure each fold is a good representative of the whole.

In addition to checking for accuracy in terms of the MSE, I also tried to fine-tune the model by fiddling with the hyperparameters to find the great combination of hyperparameter values. The Scikit- Learn's GridSearchCV feature was used to automate this process. All I needed to do was tell it which hyperparameters I need it to experiment with, and what values to try out, and it will evaluate all possible combinations of hyperparameter values using cross-validation.

Results and Conclusion

With the Gradient Boosted Trees algorithm, we achieved a root mean squared error (RMSE) score of approximately $281.02 with a standard deviation of $58.57, and all our cross-validated score were not far off. Remember, with RMSE or MSE, lower is better. Our linear model was greatly overfitting the 'specialty_description' feature because of the high cardinality. Without that feature in here, I got value similar to what we have in the ensemble regressors. That shows ensemble regressors handle well with high cardinality data. This also means that feature needs re-engineering. That was also the same reason why the 'drug name' feature was left out.

##
##
#

Regarding model performance, we see that:

- for cases of all nationality of terrorism perpetrator group same as the nationality of the target(s)/victim(s), represented as 0 with index 0, we see that the model got the prediction of 184 observations correctly, and seven wrong.
- for cases of the nationality of terrorism perpetrator group differing from the nationality of the target(s)/victim(s), represented as 1 with index 1, we see that the model got the prediction of 80 observations correctly, and 11 wrong.

These results are useful knowing sufficiently well that there are factors that can still improve it that were left out at the start of my analysis due to the need for re-engineering because of the high cardinality and multicollinearity and outlier issues in the model. I will need to revisit those to see if we can improve on this model. Few things I will be looking at are:

- grouping classes in each predictor to about five or six, so that a feature like ‘provstate’ will be now be divided into regions: Northeast, West, South, Midwest, and Pacific, and ‘gname’ will be grouped into political, racial, religious or social causes.
- I would also engineer a feature by combining the date into a datetime format, so the ‘iyear,’ ‘imonth,’ ‘iday’ will just be one column.
- doing some research that help create this features, so domain knowledge is important here
- looking closely at outliers and multicollinearity issues. Questions have to be asked if removing some data points make sense or not.
- Considering hyperparameter tuning.