Understanding the Causes of High-Cost Medical Malpractice Claims

Asher Labovich

Brown University

DATA 1030: Hands-on Data Science

Dr. Andras Zsom

December 10, 2023

Introduction

America spends more on healthcare than any other developed country. 2.4% of this spending – or approximately 1 in every 40 dollars – goes to medical malpractice lawsuits (McGough). To bring America's spending down to the rest of the developed world, it is necessary to understand what factors result in high-cost medical malpractice lawsuits. Within Kaggle's "Medical Malpractice" insurance dataset – adapted from the JMP case study dataset – there is a clear differential between "low-cost" medical malpractice claims – defined as <$225k, and "high-cost" claims, which are >$225k (Santello; JMP). Considering that the maximum claim is over $900k, predicting whether a given lawsuit will be large or small can prevent healthcare organizations and hospitals from facing financial difficulties and bankruptcies.

Exploratory Data Analysis

Initially, I planned to conduct a regression analysis on the dataset. However, considering there are no claims between ~200k-250k, the data lends itself towards a classification analysis on low-cost vs high-cost claims (Figure 1). The dataset contains 7 features: age, marital status, medical speciality, insurance type, attorney type, gender, and severity. The feature "insurance type" contains five different types of insurance – Private, Medicare/Medicaid, Workers Compensation, Unknown, and No Insurance. Of these five insurance types, two – Medicare/Medicaid and Workers' compensation – do not contain any high-cost claims whatsoever (Figure 2). Since both of those insurance types are public, these results suggest that public insurance programs are either banned from or choose not to pursue high-cost claims.

The dataset also contains an ordinal feature describing how severe a given medical malpractice incident was – ranging from 1 (emotional damage) to 9 (death). While one might assume that a higher severity would be positively correlated with a greater chance of a high-cost lawsuit, this is not the case (Figure 3). There indeed is a positive correlation up until severity reaches a peak at 7, but afterwards a lower percentage of claims are high-cost.

In both insurance type and severity, some sections of the feature contain zero high-cost claims whatsoever. Since tree-based models are exceptional at prediction when features can be easily split, early exploratory analysis suggests that the tree-based models will dominate prediction accuracy.

Methods

I began splitting the data into training and test sets, with an 80/20 split. Because only 14.4% of claims are high-cost, I stratified by the target variable to ensure both the training sets and testing sets contained a significant percentage of high-cost claims. Though the dataset contained both numerical (Age) and ordinal (Severity) variables, the latter was already organized by number, so there was no need to preprocess any ordinal variables. Therefore, all preprocessing was done via one-hot encoding or normalization via the standardscaler function in sklearn. Though I originally wanted to utilize GridSearchCV for hyperparameter tuning, this was futile: considering my dataset had ~80k observations, and many models contained ~1000 possible hyperparameter combinations, my computer did not have the computational power to create the necessary models to find the best possible hyperparameters. Such difficulties necessitated the use of RandomizedSearchCV, which only used 150 iterations, dropping the required time of hyperparameter tuning by an order of magnitude. In addition, assuming that "good" combinations occupy 5% of the hyperparameter space, a random search covering the entire space finds a good combination 95% of the time – so, the computation gain from RandomizedSearchCV far outweighs the theoretical decrease in accuracy (Zheng).

Within RandomizedSearchCV, I continued stratifying by target variable by applying shuffled StratifiedKFold as the cross-validation parameter. This ensured that no cross-validation set contained an outsized set of high-cost claims, a non-zero possibility had I chosen to use non-stratified KFold. The significant imbalance of my target variable also changed the type of scoring function used within GridSearch. If half the observations were high-cost and half were low-cost claims, accuracy would be the best metric – but in the case of significant imbalance,

accuracy is a worse scoring metric as it incentives the model to lean towards the more populous class. In this case, predicting whether a given case will be high-cost is much more important, since (by definition) high-cost claims impact overall healthcare costs significantly more than low-cost claims. Therefore, I chose to use the f1 score as a scoring metric, which does not take true negatives (low-cost cases) into account. However, I still chose to evaluate the final test on accuracy score, as a basic accuracy score is significantly more interpretable than an f1 score.

I repeated this process 4 times, choosing a different random state every time. In the final random state, I calculated and saved global SHAP values for each model, to help analyze which features were most important in predicting the test scores. I chose to use the SHAP generic "Explainer" for all models, regardless of whether they were a tree or not, so I could compare across models. Because my dataset contained significant missing values – 37% of the observations contained missing values – I began by working with XGBoost, which can natively handle and gain information from missing values. However, both variables with missing values were categorical, which meant that one-hot encoding would treat missing values as their own category, and so I could apply other methods without needing to impute any missing values. Specifically, I chose to use three other ML algorithms: random forest, k-nearest neighbors, and elastic-net logistic regression. While kernel-based SVC could have determined non-linear trends in my dataset, it would have also taken far too long: as Figure 4 shows, support vector machines grow faster than O(n) with the number of observations, which would require too much computational power. However, only one algorithm chosen (logistic regression) was linear, so any trends revealed by SVC would have likely already been found by one of the other algorithms.

For XGBoost, I trained the learning_rate (0.01-1), max_depth (3-12), n_estimators (25-75), gamma (0.01-0.2), and alpha/lambda (0.01-10) regularization parameters. For random forest, I trained significantly fewer parameters – n_estimators (25-100) and max_depth (4-15) only – simply because random forest does not use gradient descent and therefore has significantly fewer hyperparameters to choose from. For elastic-net logistic regression, I trained C (0.001-1000) and l1_ratio (0.1-0.9). For k-nearest neighbors, I trained n_neighbors (10-50) and weights (uniform, distance).

Results

Considering that so few lawsuits involved monetary compensation over $225,000, the baseline accuracy score was 85.6%. All four models had prediction accuracies above the baseline, though they differed fairly significantly from model to model. Table 1 summarizes the effectiveness of each model. By a significant margin, the worst predictor was logistic regression. It had an accuracy more than 2 points lower than the next best predictor, which is even worse after considering that the other predictors are all within 0.4 points of accuracy from each other. Logistic regression is also notable for being the only model to predict fewer than 50% of large suits correctly. Since it predicted small suits at a similar rate to other predictors (and higher than XGBoost) its failure comes entirely from its inability to predict these large suits. Logistic regression also took the longest time to predict values by far, often not converging within the maximum iterations despite allowing for many thousands of iterations. Since the best hyperparameter combination had the l1 ratio at 0.1, which was the smallest l1 ratio I tested, testing a purely ridge model might have had better results.

The other three estimators were close enough to each other that it is likely a small change in hyperparameter combinations might have changed the end leaderboard significantly. The best predictor was XGBoost, but it predicted small suits correctly less often than random forest, and large suits about as often as KNN. It only has a 0.1 point advantage over random forest, driven entirely by its 10 point advantage in the prediction of large suits. While initial exploratory data analysis suggested that tree-based models would perform significantly better, it does seem that k-nearest neighbors is capable of taking advantage of similar splits in the data, though via different mechanisms.

As Figure 5 shows, each model has a standard deviation so low it appears to be a single point – which follows from the fact that each predictor achieved practically identical scores in all 4 random states. This is to be expected when the number of observations in the training and testing datasets are so large. While no statistical test was conducted, the low standard deviations reveal that each model most definitely performed statistically significantly better than a baseline model.

By using the same SHAP explainer for each model, I was able to compare the six most important features across models (Figures 7-10). Private insurance appeared in three of the four models (minus XGBoost), and age appeared in the same number (minus logistic regression). Given figure 2, there was no surprise that private insurance was consistently one of the most important variables: by splitting across insurance types, a model could isolate all high-cost claims into just three out of the five types of insurance. However, age's importance was quite surprising. Figure 6 reveals why age played such a huge role in many of these models: older people filed exactly zero high-cost claims. This type of information is especially useful for tree models, as they can simply split by age and cut off and move all high-cost claims to one side,

increasing explanatory power. Both tree-based models had age as one of the top six important variables.

Outlook

With more computational power, I would have put more time into hyperparameter tuning for XGBoost and random forest. Both models had test scores that were practically identical, but neither tested the full list of hyperparameters. For random forest, the criterion, max_features, and min_impurity decrease hyperparameters might have reduced any overfitting and increased the test score over that of XGBoost. For XGBoost, already the most powerful model tested for this project, adding more hyperparameters such as grow_policy, min_child_weight, and subsample may have done the same, increasing accuracy and the gap over the other models even more.

Considering the surprise of age as an important factor in many of the models, I would have liked to create a SHAP graph of the age feature. SHAP feature graphs can reveal non-linear relationships between features and the target variable, which would be very useful in determining whether the probability that a case is high-cost decreases monotonically as a person's age goes up, or whether there is a peak.

Though running a support vector machine was initially impossible, splitting up the training dataset into smaller subsets would have reduced the training time. Splitting up the testing set into 10% or 5% of the total training data, and only training on 40-50% of the data might have been a significant enough reduction to allow for an SVC. Brown University also has OSCAR, a supercomputer with access for students, which most definitely has enough power to train a smaller SVC.

Considering the superiority of many models over a baseline prediction, hospitals and insurance companies should consider applying machine learning predictors to understand whether a given suit will require significant compensation, and how to plan for such events.

Appendix:

Figure 1: The distribution of claims by price of claim

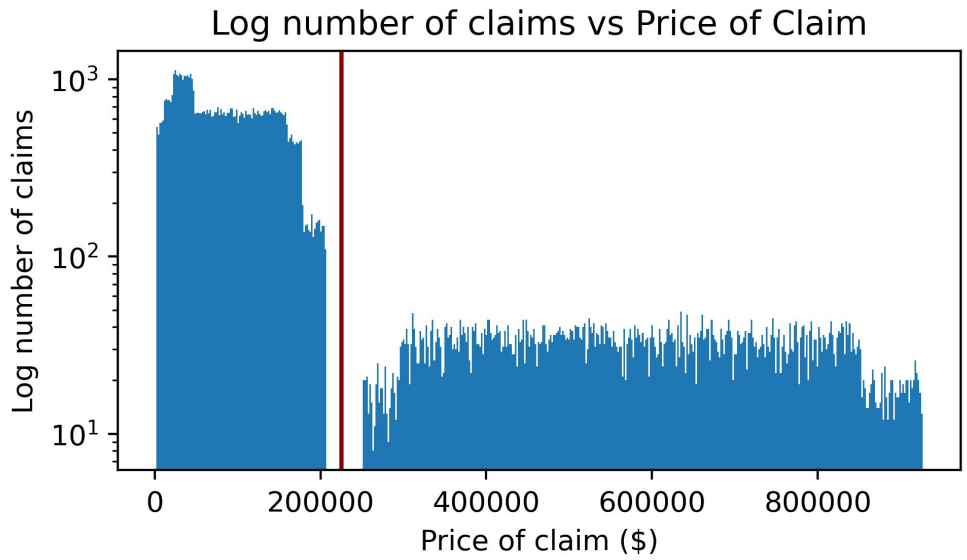


Figure 2: Distribution of claim amount by insurance type

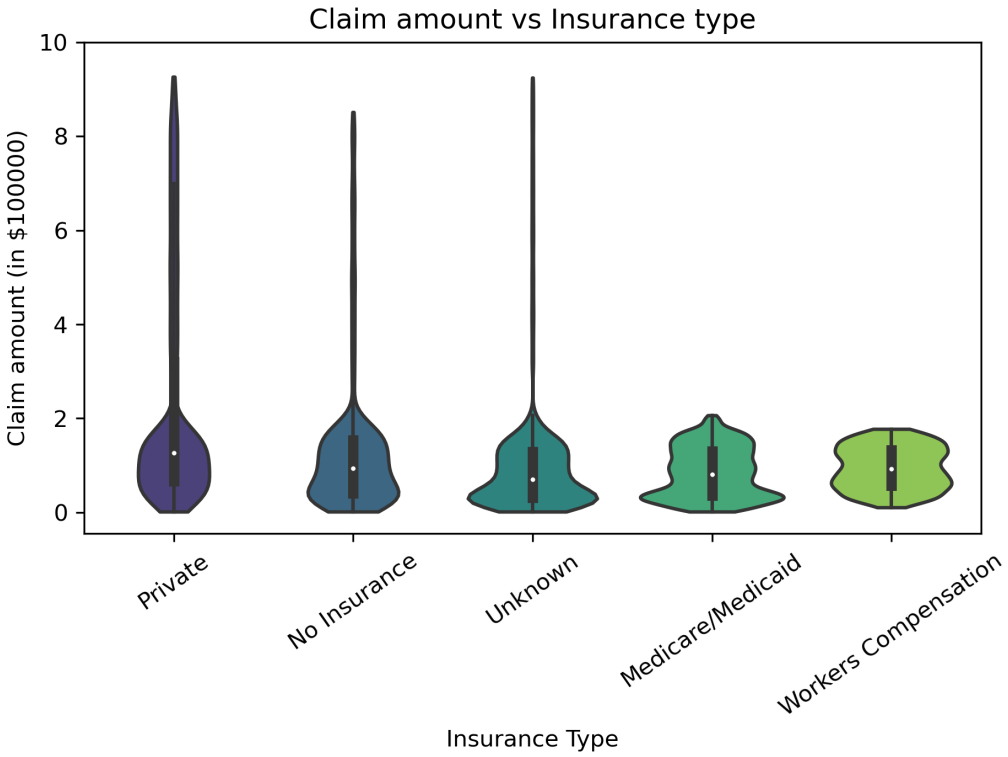


Figure 3: Percent of high-cost claims within each severity type

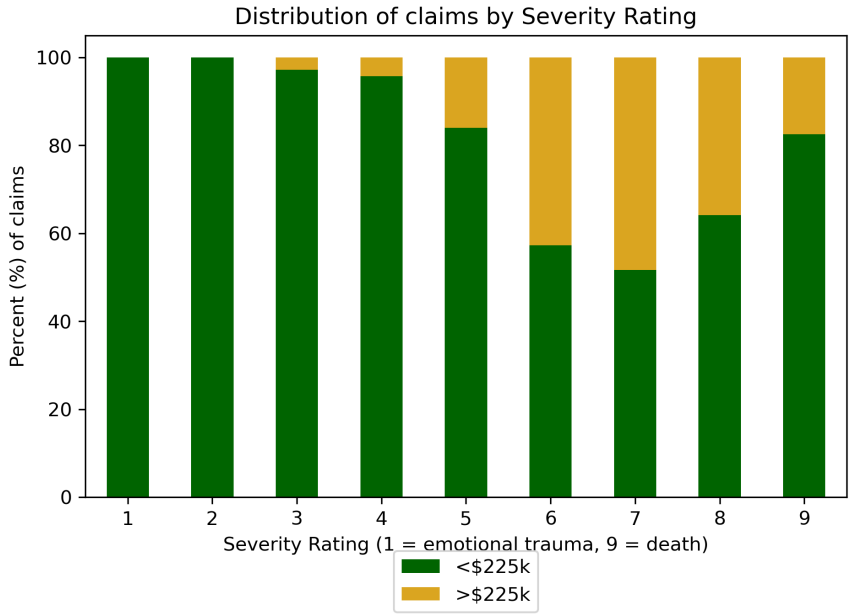


Figure 4: Speed to run a support vector machine by number of observations

Table 1: A summary of the accuracy of the different machine learning algorithms trained.

Model	% Accuracy	% of small suits predicted correctly	% of large suits predicted correctly
Logistic Regression	89.2	96	48
KNN	92.4	95	76
Random Forest	92.7	**97**	66
XGBoost	**92.8**	95	**77**

Figure 5: Accuracy scores with error bars for each model

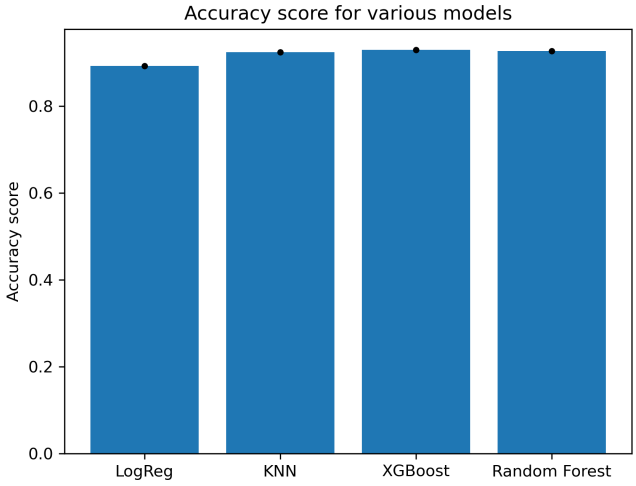


Figure 6: The effect of age on whether a given claim is large or small

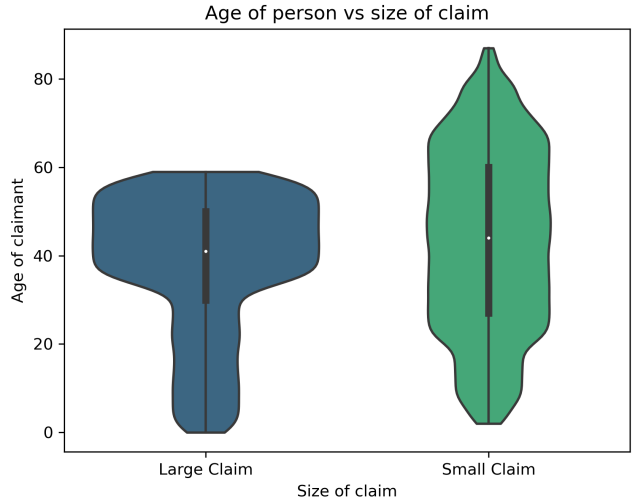


Figure 7: Feature importances for Logistic Regression

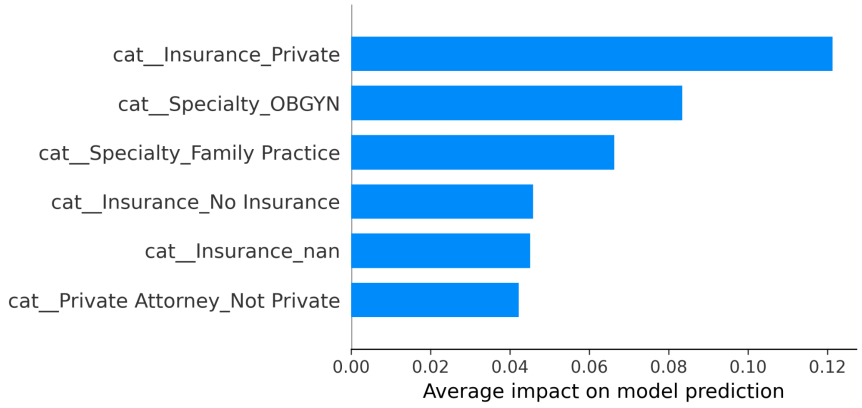


Figure 8: Feature importances for KNN

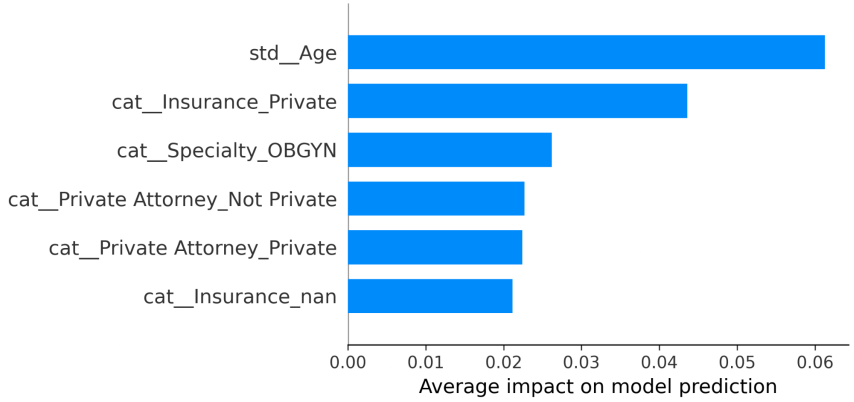


Figure 9: Feature importances for Random Forest

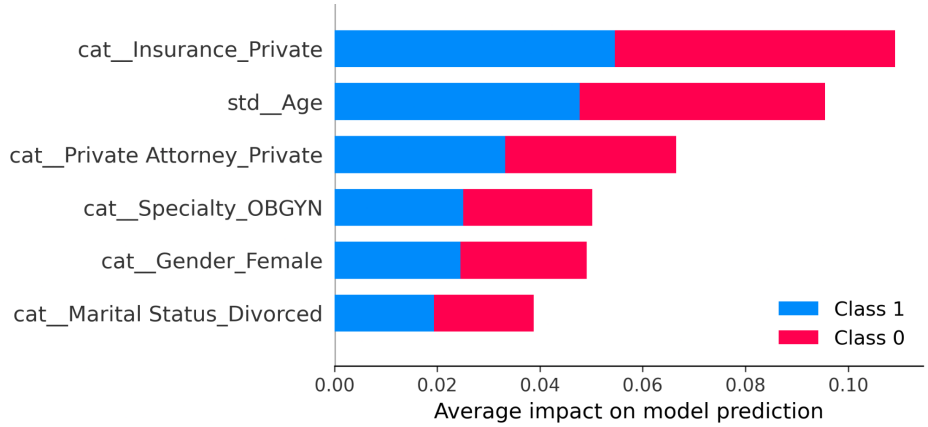
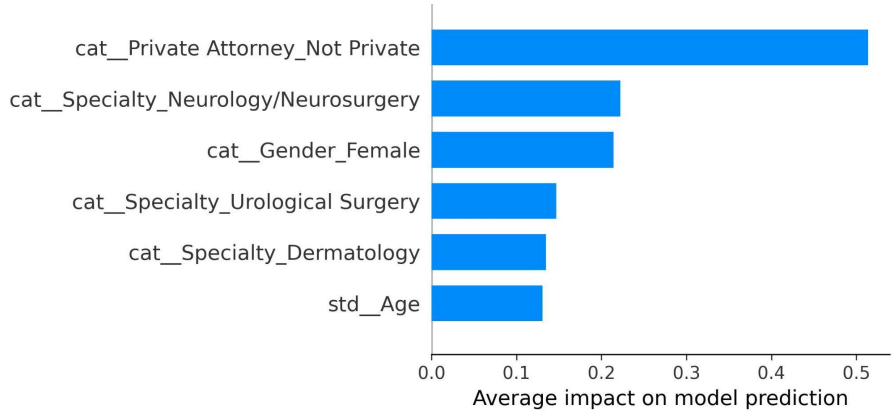


Figure 10: Feature importances for XGBoost



References

Case study library. JMP. (n.d.). https://www.jmp.com/en_us/academic/case-study-library.html

McGough, M., Telesford, I., Rakshit, S., Wager, E., Amin, K., & Cox, C. (2023, February 15). How does health spending in the U.S. compare to other countries?. Peterson-KFF Health System Tracker. https://www.healthsystemtracker.org/chart-collection/health-spending-u-s-compare-countries/

Santello, G. (2023, September 25). Medical malpractice - insurance dataset. Kaggle. https://www.kaggle.com/datasets/gabrielsantello/medical-malpractice-insurance-dataset/data

Zheng, A. (2015, May 27). How to evaluate machine learning models: Hyperparameter tuning. Wayback Machine. https://web.archive.org/web/20160701182750/http://blog.dato.com/how-to-evaluate-machine-learning-models-part-4-hyperparameter-tuning