Healthcare Price Finder: A Web-Based Application for Estimating Healthcare Medical Cost by Modeling Medicare Data With Machine Learning

Healthcare Price Finder: A Web-based Application for Estimating Healthcare Medical Cost by Modeling Medicare Data with Machine Learning

Zhejing Luo

A Thesis in the Field of Software Engineering

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

August 2020

Abstract

The pricing intransparency in the healthcare system of the United States has been a long-standing problem. This thesis describes the Healthcare Price Finder project, a web-based solution that helps users to estimate how much the medical procedures cost. Using the Medicare Provider Utilization and Payment Dataset between 2012 and 2017, an XGBoost regression machine learning model was trained to predict the cost of medical procedures The model enables the Healthcare Price Finder to provide medical cost estimates even when past pricing data is not available. A Healthcare Price Finder web application was also developed using JavaScript framework and libraries including React.js and Node.js, and provides an easy-to-use interface for average users to get cost estimates for medical procedures and look up the past prices from the Medicare Provider and Payment dataset. The current version of the Healthcare Price Finder tool was developed with California data, and can be scaled up to provide medical procedure charge amount estimates and past price lookups for the entire United States.

Dedication

Dedicated to my precious son Aris Yang, who made the grand entrance into our life and brought us immense love and joy at an unusually eventful time in history (which also happens to be the middle of my thesis project).

Acknowledgments

First and foremost, I would like to express my gratitude to my thesis supervisor, Dr. Zoya Kinstler, for her guidance throughout my thesis project. Dr. Kinstler provided crucial input that steered this project into the right direction, and her empathy and encouragement during this journey was invaluable to me. I would also like to thank my research advisor, Dr. Sylvain Jaume, who helped me with my project and for his valuable advice and support during the thesis process. My sincere thanks also goes to my thesis pre-work course instructor, Eric Gieseke, for helping me in shaping the idea of this project.

I am very grateful for the support I received from the accessibility and accommodations office staff, including Linda Suillivan, Caitlin Mason, and Leora Simo. Big thanks to my advisor Nada El-Newahy.

Last but not least, completing this thesis project would have been impossible without the support from my mother Zaihua Yang and my husband Ryan Giordano. Thank you for being the cornerstones of my life.

Table of Contents

Abstract i

Dedication ii

Acknowledgments iii

List of Tables iv

List of Figures v

Chapter I. Introduction 1

Chapter II. Background 2

Chapter III. Prior Work 4

3.1 Review of Existing Medical Price Estimator Solutions 4

3.1.1 ClearHealthCosts 4

3.1.2 FairHealth Consumer 9

3.1.3 Healthcare Bluebook 12

3.1.4 Mayo Clinic Cost Estimator 15

3.2 What Can Be Done Better? 18

Chapter IV. System Overview 20

4.1 System Architecture 20

4.2 Technology Choices 22

Chapter V. Data 25

5.1 Data source 25

5.2 Dataset Details 27

5.3 Data Cleaning 34

Chapter VI. Machine Learning 36

6.1 Machine Learning Price Prediction 36

6.2 XGBoost 36

6.3 Feature Encoding 39

6.4 Feature Selection 42

6.5 Model Parameter Tuning 52

6.6 Model Evaluation 54

6.7 Scaling with More Data 55

Chapter VII. Application Implementation 58

7.1 Application Architecture 58

7.2 User Interface 60

7.3 Relational Database 67

7.3.1 Payment Table 68

7.3.2 HCPCS Tables 69

7.3.3 Provider Tables 71

7.4 Front-End (Client) 74

7.5 Back-End (Server) 76

Chapter VIII. Summary and Future Work 78

References 81

List of Tables

Table 1. AWS tools used in the Healthcare Price Finder project. 23

Table 2. Total number of the rows in the full Medicare Provider Utilization and Payments dataset and the California subset by year. 26

Table 3. List of the columns in Medicare Provider Utilization and Payments dataset. 27

Table 4. Full list of features in the model training data. 40

Table 5: XGBoost regression model parameters and the optimized selected values for the medical price estimator model. 52

Table 6. 10-fold cross-validation results of the tuned XGBoost model. 54

Table 7. PAYMENT table columns in Healthcare Price Finder relational database. 69

Table 8. HCPCS table columns in Healthcare Price Finder relational database. 70

Table 9. HCPCS_AGGREGATED table columns in Healthcare Price Finder relational database. 71

Table 10. PROVIDER table columns in Healthcare Price Finder relational database. 72

Table 11. PROVIDER_COORDINATES table columns in Healthcare Price Finder relational database. 73

Table 12. PROVIDER_TYPE table columns in Healthcare Price Finder relational database. 73

List of Figures

Figure 1. ClearHealthCosts landing page. 6

Figure 2. ClearHealthCosts search results page. 7

Figure 3. ClearHealthCosts sample search result page when no past price is found. 8

Figure 4. FAIR Health Consumer landing page. 10

Figure 5. FairHealth Consumer search results page with the Related Costs section. 11

Figure 6. FairHealth Consumer search results page with Local Price Comparison. 12

Figure 7. Healthcare Bluebook search tool landing page. 13

Figure 8. Healthcare Bluebook search results page (free version). 14

Figure 9. Mayo Clinic Cost Estimator landing page. 15

Figure 10. Locations supported in Mayo Clinic Cost Estimator. 16

Figure 11. Mayo Clinic Cost Estimator search result page (no insurance selected). 17

Figure 12. System overview diagram of the Healthcare Price Finder project. 20

Figure 13. F scores in a trained XGBoost model with 100 decision trees and max tree depth of 5. 46

Figure 14. Feature permutation weights in a trained XGBoost model with 100 decision trees and max tree depth of 5. 46

Figure 15. F scores in a trained XGBoost model with 100 decision trees and max tree depth of 8. 49

Figure 16. Feature permutation weights in a trained XGBoost model with 100 decision trees and max tree depth of 8. 49

Figure 17. Overview flowchart of the Healthcare Price Finder Application. 59

Figure 18. Healthcare Price Finder application landing page. 60

Figure 19. Healthcare Price Finder main search page after entering location and a keyword in the search bar. 61

Figure 20. Healthcare Price Finder procedure price details page (when the past prices on the procedure at user's location is available). 63

Figure 21. Healthcare Price Finder procedure price details page (when the past prices on the procedure at user's location is not available). 65

Figure 22. Healthcare Price Finder provider details page. 66

Figure 23. Healthcare Price Finder relational database schema. 68

Figure 24. Healthcare Price Finder Application front-end core structure UML diagram. 74

Figure 25. Healthcare Price Finder Application Back-end server UML diagram. 76

Chapter I.

Introduction

The goal of this master's thesis project is to explore the possibility of predicting the price of medical procedures in the United States using the 2012-2017 Medicare Provider Utilization and Payment Data: Physician and Other Supplier dataset that was released to the public by the Center of Medicaid and Medicare Services, and use the discovery results to develop a user-friendly web-based application -- Healthcare Price Finder -- to grant anyone in America with the ability to get estimates of how much the healthcare procedure they need would cost, and easy access to the past medical pricing data at where they live.

Machine learning techniques are used to develop an XGBoost (Extreme Gradient Boosting) regression model for providing estimates of the medical procedures based on the healthcare procedure and the location selected by the user.

As a proof of concept, only the California subset of the source data is used in this project for a manageable scope and quick iterations. But the machine learning modelling techniques and the web application developed for this project can be scaled up to support medical price estimates in the entire United States, and contribute to increasing the pricing transparency in US healthcare.

Chapter II.

Background

Court records show that 66.5 percent of the bankruptcy filers from 2013 to 2016 were tied to medical issues, this makes medical bills the No.1 cause of personal bankruptcies in the United States (Himmelstein et al., 2019). One unique characteristic of the US healthcare billing system is pay-after-service, that people often have no idea how much their medical bill is going to be until they receive the bill long after the visit is completed. Because of this, healthcare consumers often have no means of negotiating or comparing the prices between healthcare providers prior to their visit, and are often forced to pay the expensive bills for the medical services they cannot afford.

This billing system is one manifestation of a multitude of transparency lacking in the US healthcare system, which has led to an unparalleled inefficiency -- United States is the country with the highest healthcare GDP spending in the world, while also having one of the lowest life expectancies among developed countries (Kumar et al., 2011).

Medicare is the US federal health insurance program for people who are 65 or older, certain younger people with disabilities, and people with End-Stage Renal Disease (permanent kidney failure requiring dialysis or a transplant, sometimes called ESRD). Centers for Medicare and Medicaid Services (CMS) Open Payments program is a part of the national transparency program Open Payments that collects and publishes information

about relationships between drug and medical supply manufacturers and distributors, on one side, and healthcare providers on the other. In the Open Payments program, Centers for Medicare and Medicaid Services (CMS) released the aggregated pricing data of all the claims filed to Medicare. However, this dataset is released to the public in large raw text files, a format that is difficult to access by the general public.

Currently multiple services exist to help average consumers to compare the prices of medical service by providing access to the healthcare price estimates based on payments data from CMS or other insurers. However, most of them only support looking up historical data, and are unable to extrapolate price estimates when past data is missing. In this project, I attempt to fill this gap by developing a machine learning model that is able to predict healthcare procedure prices even with missing data. I also created a web application to offer this model to the general public for estimating the healthcare costs.

Chapter III.

Prior Work

3.1 Review of Existing Medical Price Estimator Solutions

As of today, there are multiple medical price lookup and estimation tools online, however, many of them are behind paywalls or require some form of organization membership. Three (including UnitedHealth Group, Anthem, Aetna) out of the five biggest commercial insurance companies by market value as well as some of the smaller medical insurance companies like Kaiser Permanente provide some form of treatment cost estimators tool as a part of their benefits. However, these tools are only available to their insurance plan members, and non-members are unable to access them. There are also some companies providing treatment cost estimators for a fee, like IBM Watson's Treatment Cost Calculator. However, since I am aiming to build a tool that anyone with or without insurance can use free of charge, this section only discusses the existing medical procedure price estimator solutions online that are available for free to the general public.

3.1.1 ClearHealthCosts

ClearHealthCosts (clearhealthcosts.com) was founded by ex-New York Times journalist Jeanne Pinder in 2010, and has been partnering with multiple public media platforms to help people to understand how much their medical procedures cost.

ClearHealthCosts has received a lot of media attention and was featured on the news multiple times, and their CEO Jeanne Pinder presented " What if all US health care costs were transparent?" on TEDTalks Residency in December 2018.

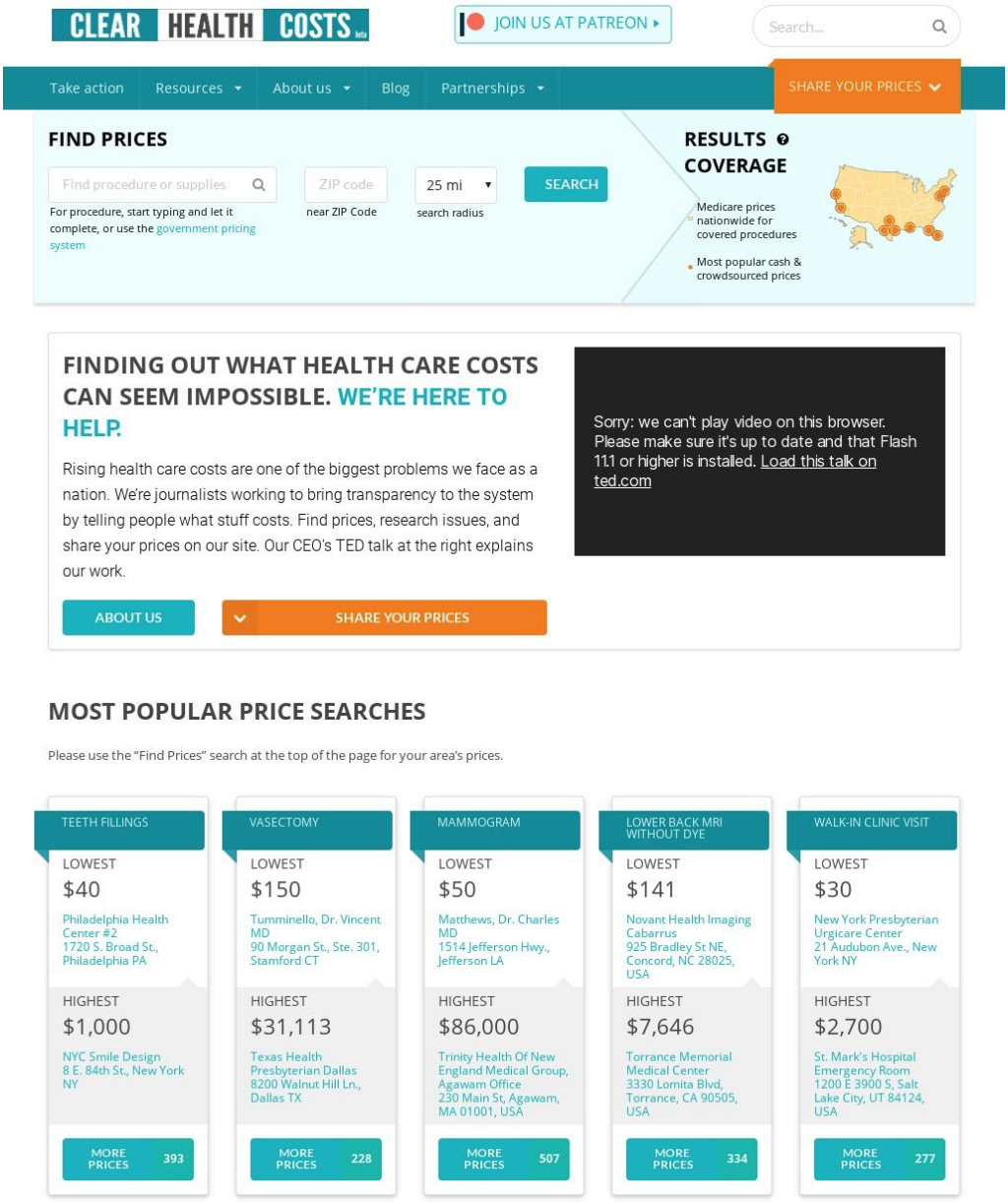

Figure 1. ClearHealthCosts landing page

According to the ClearHealthCosts website, they use crowdsourced healthcare prices from their independent reporting, from healthcare providers, from participating consumers and from databases.

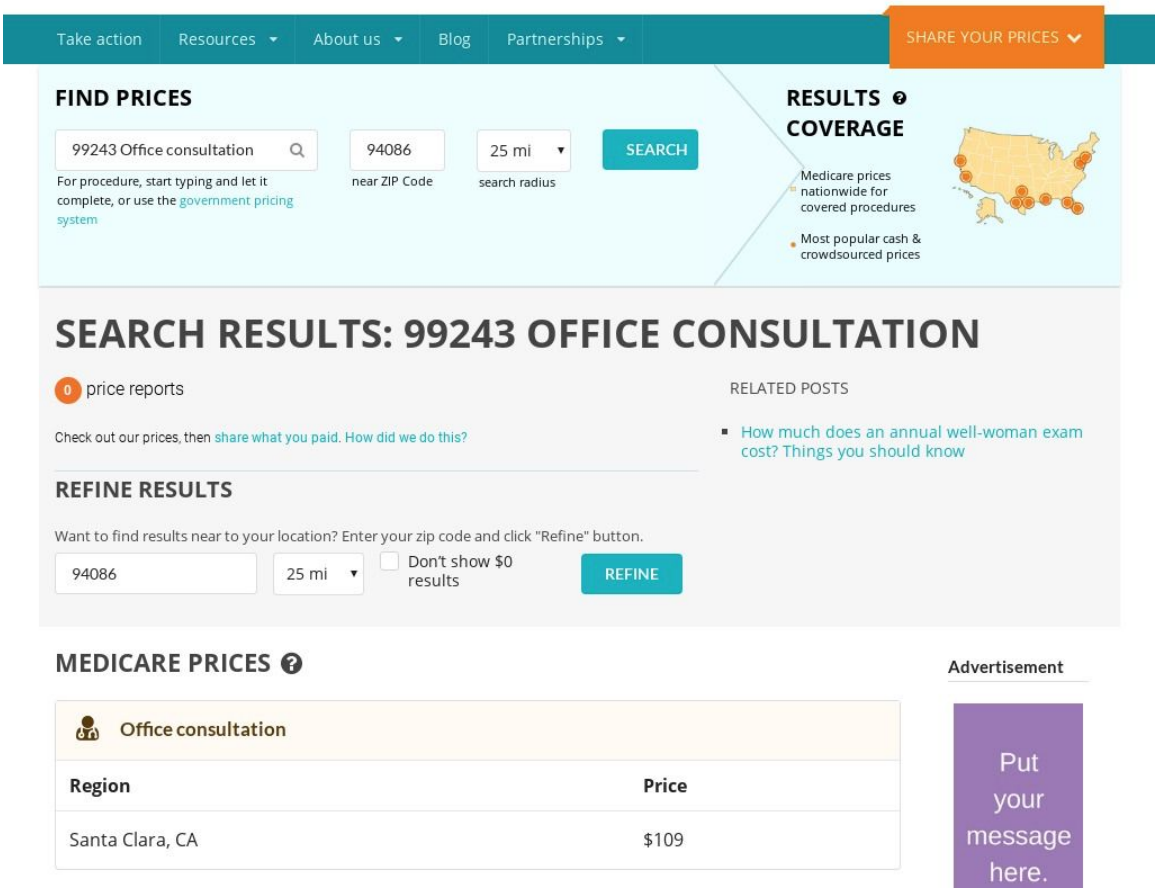

Figure 2. ClearHealthCosts search results page

ClearHealthCosts website has a straightforward search interface. User enters the procedure name in the search bar, and a dropdown list automatically appears and displays the most probable match in medical procedures. With a medical procedure selected, the user enters a zip code, chooses a search radius in miles, then clicks the Search button to

see the results. However, some of the healthcare procedure names came from user supplied data and contain errors or cannot be matched with any specific medical billing codes. The procedure names that are listed along with medical billing codes are often abbreviated, which can be challenging for average users who do not have a medical background to understand or even choose correctly.

The search results page on ClearHealthCosts does not list providers nor the amount they charge individually, while it's known the amount charged for the same procedure from different healthcare providers can vary substantially. On the search results page, often only a "Medicare price" is listed, although it is unclear how this price is calculated. Another issue ClearHealthCosts search results is a lot of healthcare procedure searches lead to empty results -- only a "$0" is shown, especially with less common medical billing codes.

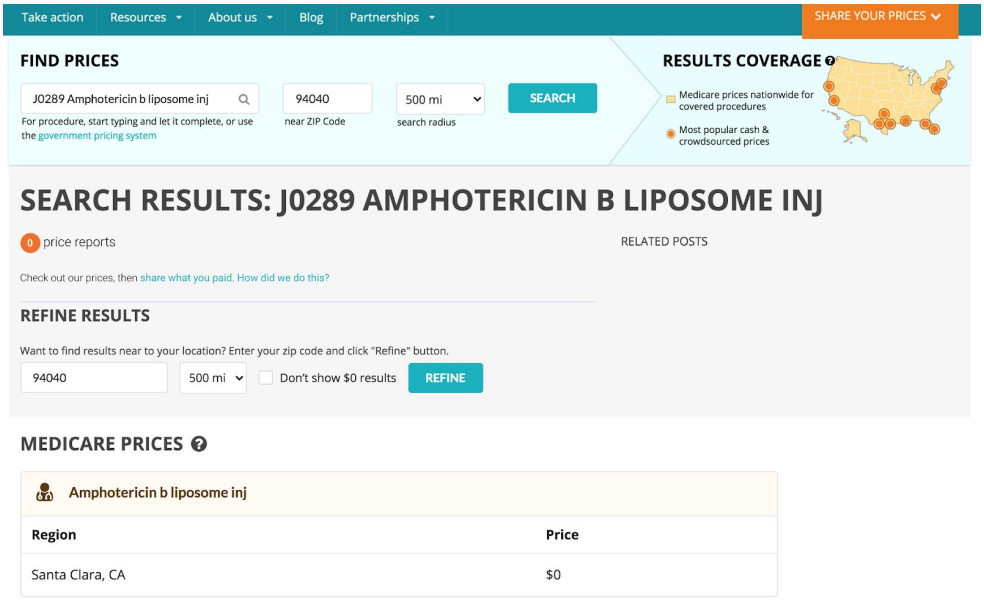

Figure 3. ClearHealthCosts sample search result page when no past price is found

While having the highest publicity among the healthcare price estimator tools, there are still places that can be improved in the presentation of the pricing data and the data quality.

3.1.2 FAIR Health Consumer

FAIR Health (www.fairhealthconsumer.org) is an independent national nonprofit organization founded in 2009 with the mission to help people to understand healthcare costs and health coverage and to bring transparency to healthcare costs and insurance. According to the FAIR Health's website, the cost estimates are based on claims for medical and dental services paid for by private insurance plans and includes more than 31 billion private health care claims and 20 billion Medicare claims for 10,000 services in all areas of the United States, dating back to 2002.

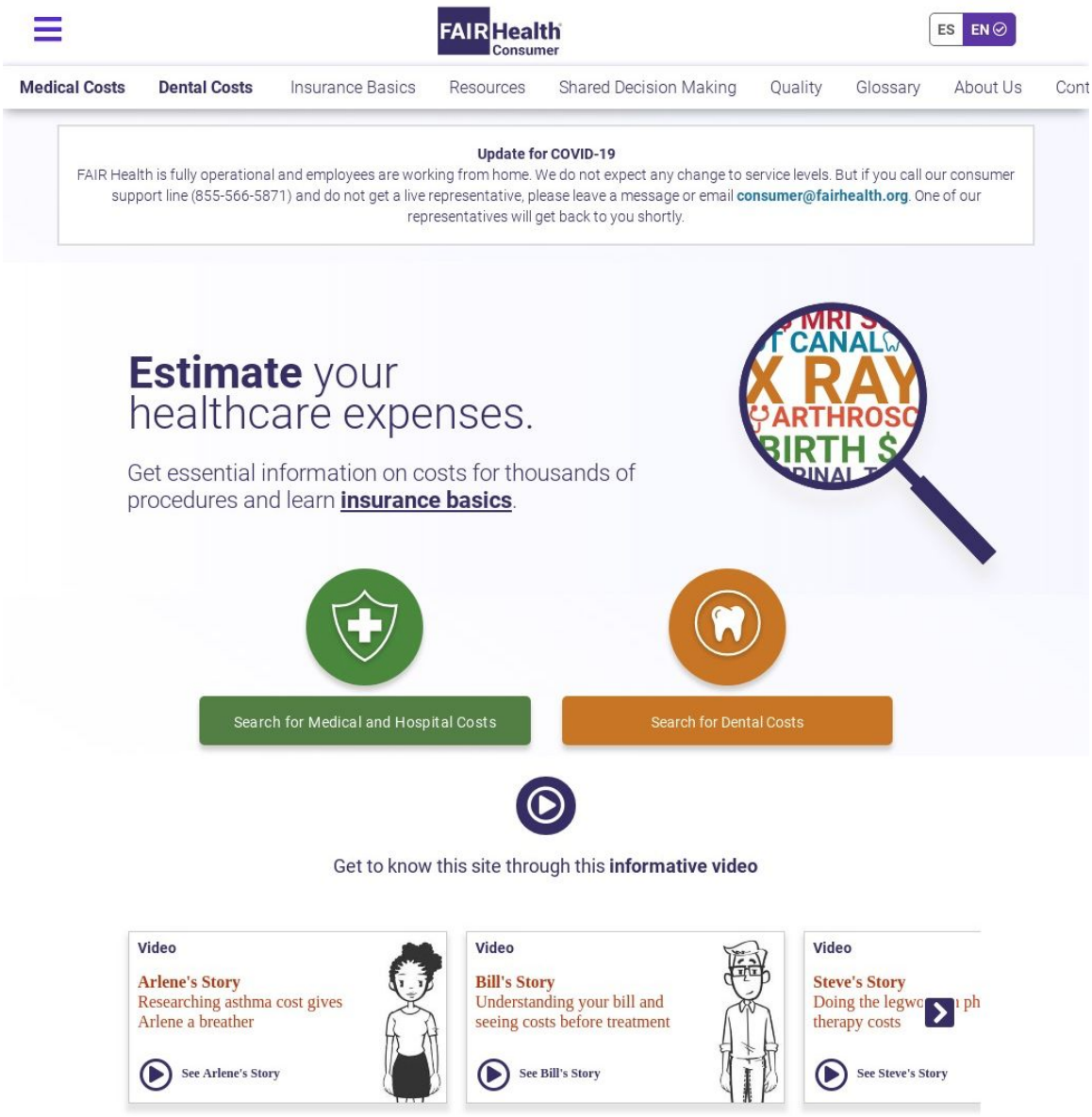

Figure 4. FAIR Health Consumer landing page

FAIR Health Consumer takes a step by step approach when asking users to enter information for the medical procedure price estimate search, with prompts as one or multiple pages at each step. Users are first asked to choose their insurance status, then

enter the provider zip code, and search for healthcare procedures by keywords or browsing the pre-defined categories.

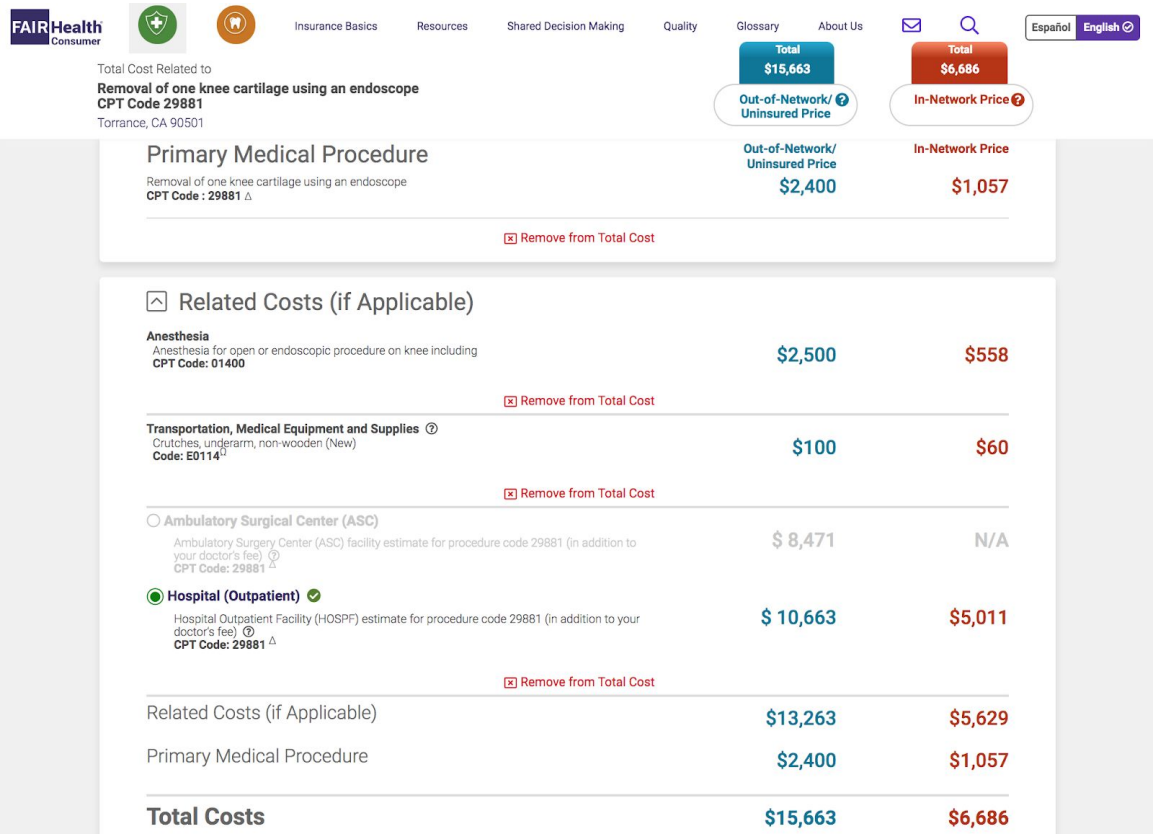

Figure 5. FairHealth Consumer search results page with the Related Costs section

The FAIR Health search results page presents the price estimates in two columns, first one for the Out-of-Network/Uninsured price, and the second one for In-Network Insured price. In addition to the price estimates of the selected healthcare procedure and location, FAIR Health Consumer also shows the estimated prices of the related medical procedures and the total cost, as well as the prices for the selected procedure in nearby cities. However, no healthcare provider specific data is shown on the results page, possibly due to reasons like the contractual agreements with data providers for anonymization, or intentional data presentation as a part of the product design. The FAIR Health Consumer price estimator tool stands out among competitors with it's sheer volume of historical data; however, gaps still exist on price estimates of the healthcare procedures that do not have sufficient historical data.

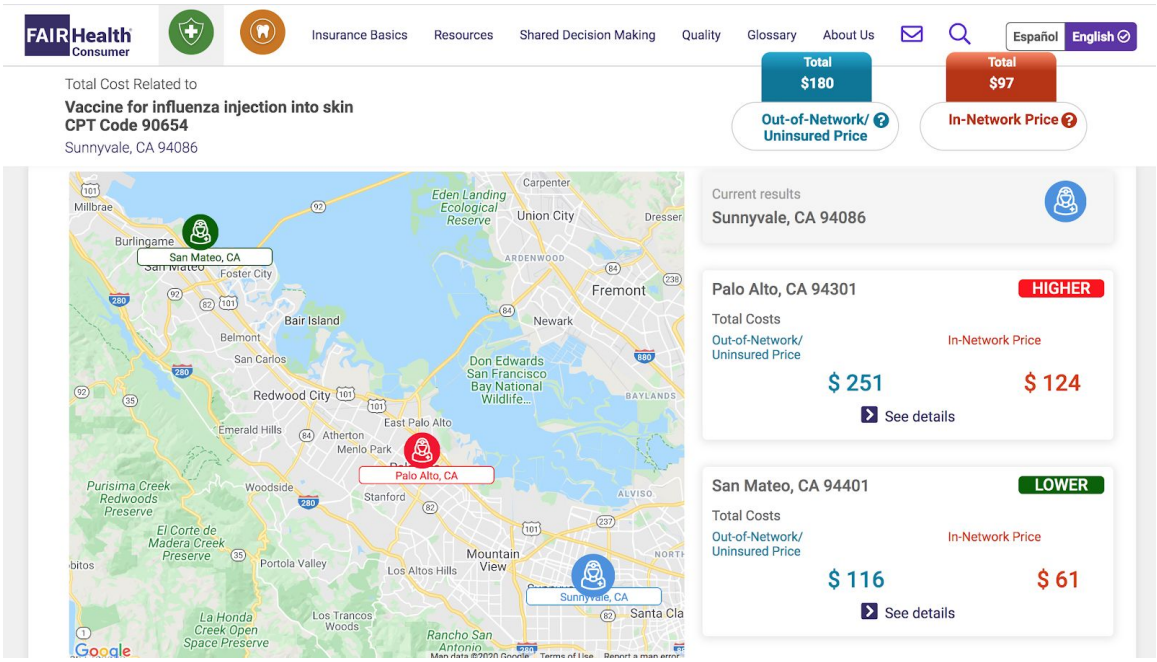

Figure 6. FairHealth Consumer search results page with Local Price Comparison

3.1.3 Healthcare Bluebook

Healthcare Bluebook (healthcarebluebook.com) is a Nashville based company that aims to help patients to select healthcare providers using quality and cost data. Healthcare Bluebook's medical price estimate tool is targeted at businesses to help their employees to find affordable medical care, but a free search tool is available to any healthcare consumer online.

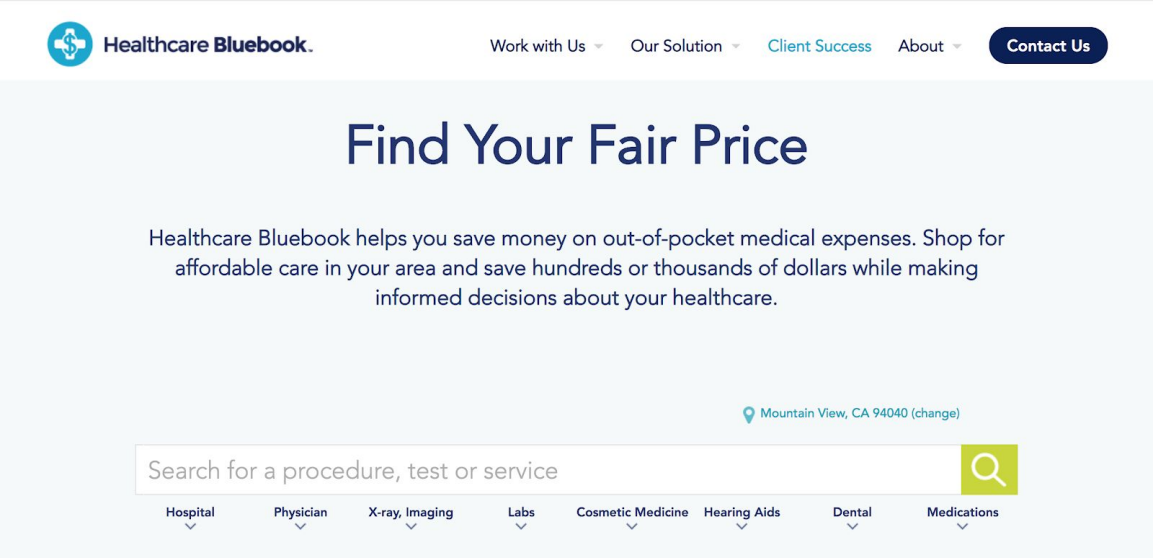

Figure 7. Healthcare Bluebook search tool landing page

On the search interface, the user enters the location city or zip code first, and types in the search bar to get the list of medical services that are likely matches or choose from the list of the medical service categories under the search bar, and eventually select one medical service.

The pricing estimates provided by Healthcare Bluebook are the total price of the full medical service including all the procedures and physician charges instead of individual procedures by HCPCS billing codes. In fact, the HCPCS procedure billing codes are never displayed in Healthcare Bluebook's search results. The search results show a "Fair Price" box for the expected price of the medical procedure, and a color-coded price range bar with the green rating for "At or Below Fair Price", yellow rating for "Slightly Above Fair Price", and red rating for the "Highest Price". It's unclear how the fair price and the price ranges are calculated. The free version of the Healthcare Bluebook does not include the pricing data for individual healthcare providers, but it is available in the premium version for business users. According to the screenshots of the premium version shown on the website, each healthcare provider is color-coded by the amount they expect to charge to help users to find providers who charge reasonable prices for their service. Healthcare Bluebook's pricing estimate result does not take medical insurance into consideration.

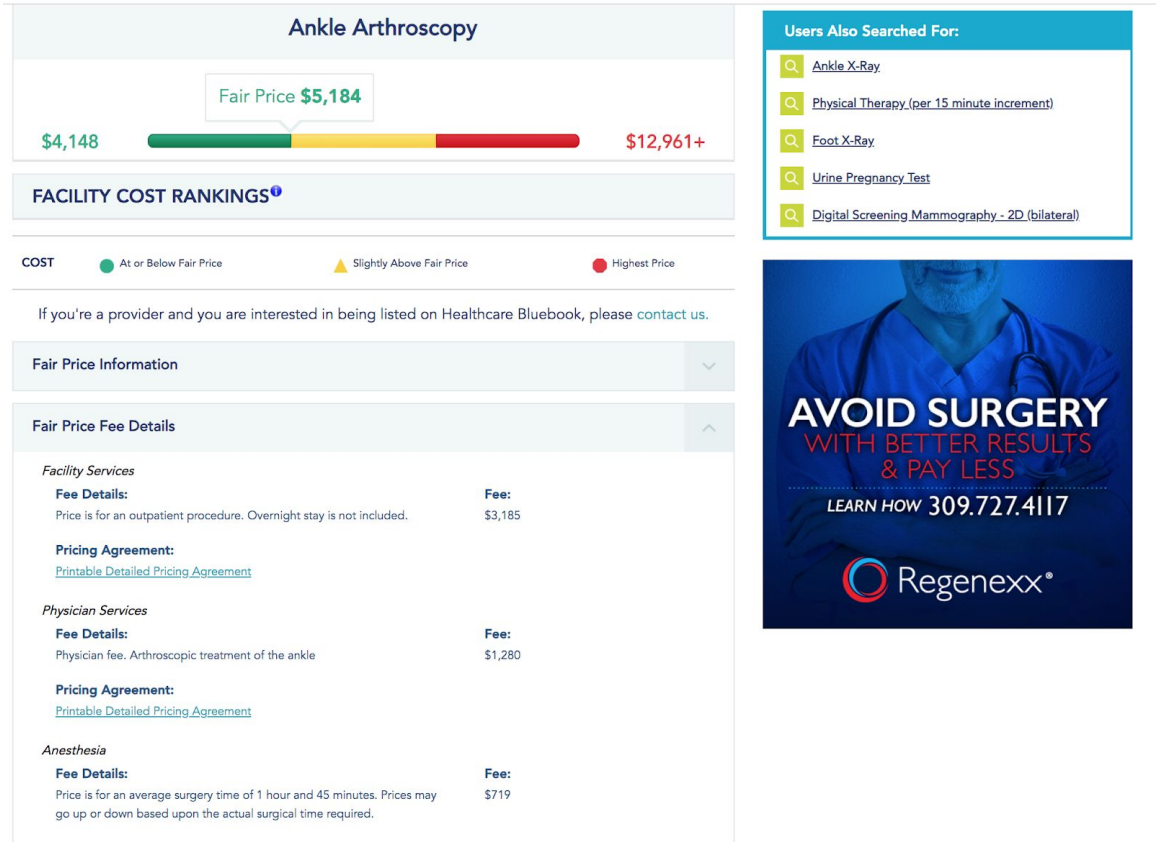

Figure 8. Healthcare Bluebook search results page (free version)

3.1.4 Mayo Clinic Cost Estimator

Mayo Clinic is a nonprofit research medical center currently based in three major locations: Rochester, Minnesota; Jacksonville, Florida; and Scottsdale, Arizona. They have made the price of their cosmetics procedures and medical services and procedures accessible to the public through a Cost Estimator Tool (costestimator.mayoclinic.org). This review only covers the medical services and procedures estimation tool.

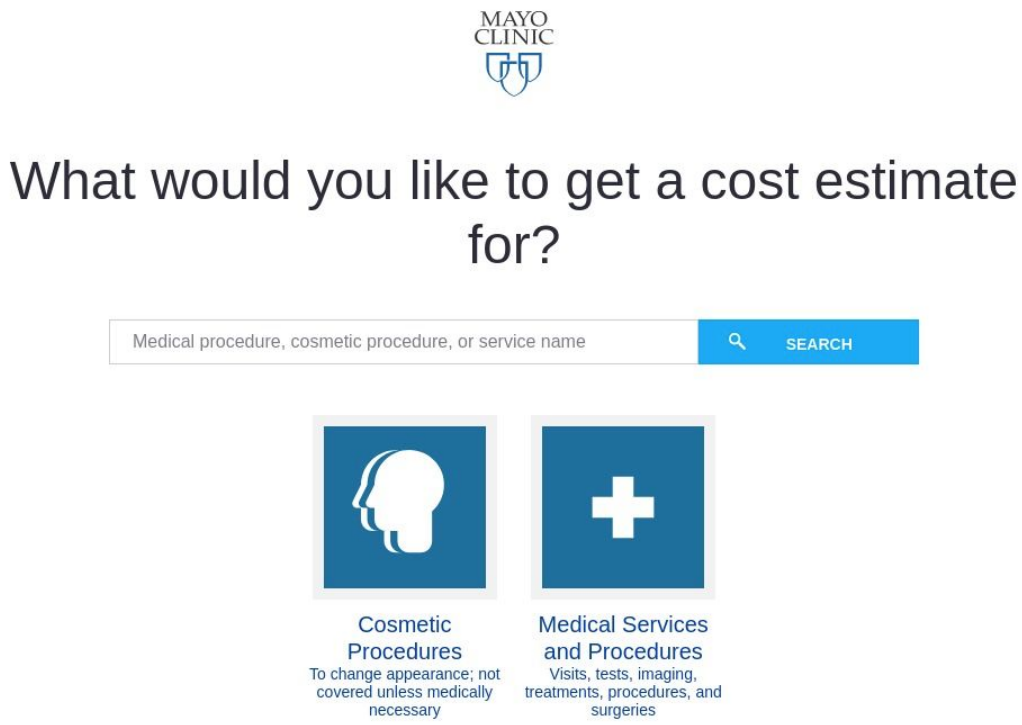

Figure 9. Mayo Clinic Cost Estimator landing page

The user starts the price estimates search on a simple interface by entering the medical procedure name in the search bar. However, users have to navigate through as many as 10 steps to select one of the 3 Mayo Clinic locations, enter their insurance information, choose their medicare participation status, etc. on different pages in this tool before reaching the actual price estimates page. It's also revealed at a very late step that this cost estimator tool is only able to provide prices from the three Mayo Clinic locations and may not be suitable for all users. It is possible to improve the user experience of Mayo Clinic Cost Estimator search by combining some of the steps and alert users of the limitation of the tool as early as possible.

Figure 10. Locations supported in Mayo Clinic Cost Estimator

On the search results page of the Mayo Clinic Cost Estimator tool, only an estimated average total cost amount is displayed, rest of the items are all shown as "N/A". The cost estimates results are based on full procedures instead of HCPCS medical billing codes. Only the procedures available at Mayo Clinics are included in the search, which means this tool is unable to provide estimates for some of very common medical procedures. The design and the available data in the Mayo Clinic Cost Estimator tool make it highly useful for a very specific consumer group -- users who want to find out how much some medical procedures would cost before and after insurance at one of the three Mayo Clinic locations. It is not suitable for average users who just want to get a quick estimate of how much the healthcare procedure they need would cost.

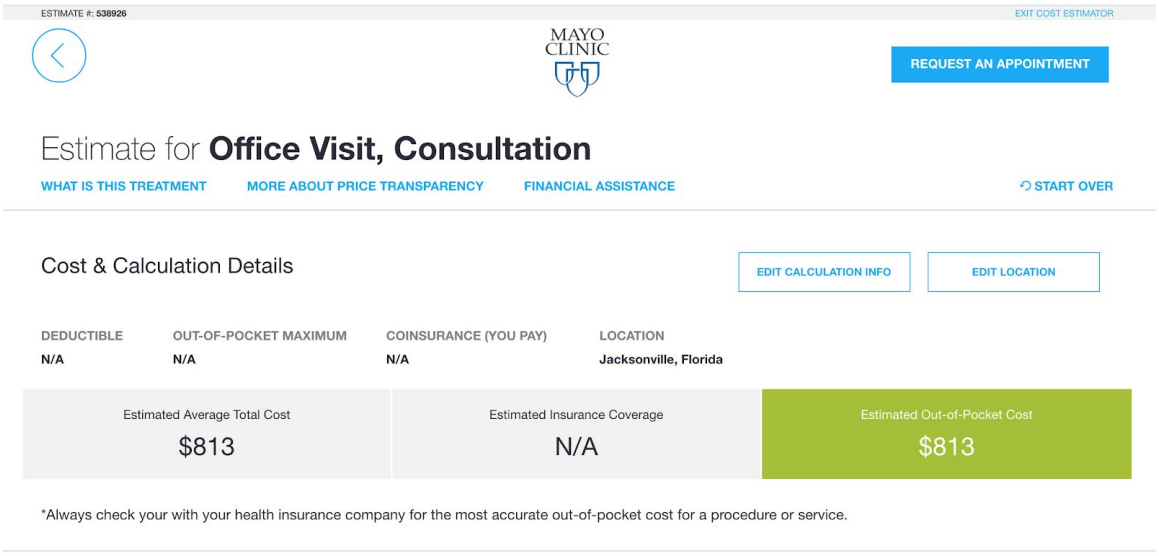

Figure 11. Mayo Clinic Cost Estimator search result page (no insurance selected)

3.2 What Can Be Done Better?

In order to stand out competitively against previous medical procedure price estimation solutions, the Healthcare Price Finder application should aim to develop some unique features that other reviewed medical price estimator solutions cannot provide, while also including as many of their strengths as possible in a reasonable range.

One common weakness among the reviewed price estimator tools is all of them are strictly based on the past pricing data, which makes them incapable of providing medical price estimates when past data is unavailable, and weakness can lower the user experience significantly. I plan to close this gap in the Healthcare Price Finder application by using machine learning modelling to provide estimates for the medical procedure costs even when past data is not available.

Another thing I noticed is none of the reviewed tools presents healthcare provider level pricing data for free. Because of this, their functionality is limited to providing generic price estimates at the selected locations, but cannot tell the users which healthcare providers should be avoided for charging much higher than peers on the selected procedure. In order to help users to choose a healthcare provider who charges reasonable amounts for their services, I plan to make the provider-level medical pricing data available for any user to look up, and maybe include provider-centered pages to help users to look up provider information like the address for planning their visit. In future developments, this part can be fledged into provider recommendations features.

The Healthcare Price Finder application should be designed with an easy-to-use interface with simple steps in the search process since it's a common strength among the past medical price estimator solutions.

Chapter IV.

System Overview

4.1 System Architecture

This section explains the overall structure of the proposed Healthcare Price Finder project, from preprocessing the data that supports the machine learning model to the final website that displays the price of the healthcare procedures at the selected location to the users. The project is constructed with four main components -- machine learning, data storage, web application front end, and web application back end. Below shows the architectural chart of all the components integrated.

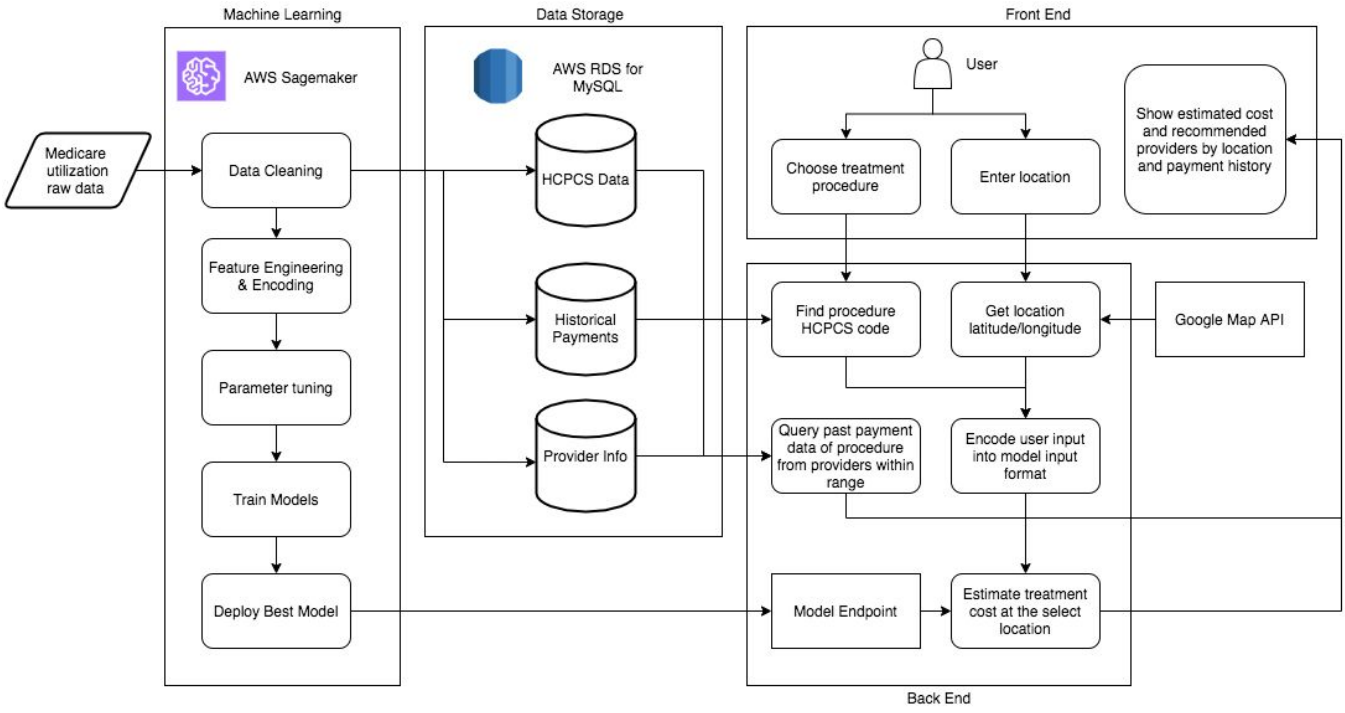

Figure 12. System overview diagram of the Healthcare Price Finder project

The Machine Learning component includes data preprocessing, which cleans the Medicare Utilization and Payments raw data and passes cleaned data to the Data Storage component. The cleaned data then goes through feature encoding and engineering to select the key columns from the dataset that are essential for creating an effective machine learning model. The model parameters go through further calibration and cross-validation to optimize the prediction accuracy, and the selected parameters are used to build the most accurate machine learning model as measured by model Root Mean Square Error (RMSE) metric and achieved within based on the chosen parameters (see section 6.4 Feature Selection). The final model is deployed as an API endpoint, which responds to user queries of healthcare procedure and location with estimated provider charge amounts.

The Data Storage component stores the cleaned data in a relational database. The Medicare data is modelled into three categories of tables in star-schema -- the main fact table for historical payments, and dimensional tables for the Healthcare Common Procedure Coding System (HCPCS) data and healthcare providers data.

The Front-End component renders the interface for the user to interact with the application, like selecting the healthcare procedure they need and entering their location, and passes the user inputs to the Back-End server component for retrieving relevant data. The Back-End component handles the requests sent by the Front-End clients, connects to the different services like Google Maps API and XGBoost model deployed on an AWS

Sagemaker endpoint, and responds to the Front-End request with relevant data. Finally, the Front-End presents the results on the user interface.

4.2 Technology Choices

Python was selected as the main programming language used in this project. Compared to other data processing tools and programming languages, Python stands out with the following qualities:

1. Python is the most commonly used programming language in the Machine Learning field, and it is well-supported across platforms with abundant documentation.
2. Python has multiple powerful libraries including Pandas, NumPy, and IPython that are easy to use and saves time during development.
3. Python is a powerful tool for data wrangling and cleaning.
4. Python is highly interoperable and works well as the "glue" by offering integration with other tools.

XGBoost (eXtreme Gradient Boosting) is an implementation of the Gradient Boosted Decision Trees algorithms for predictive modelling (Chen & Guestrin, 2016). Gradient Boosted Decision Trees is one of the "ensemble learning" machine learning algorithms. Ensemble machine learning methods involve iteratively generating multiple models, measuring their performance, and building new models to reduce the errors in the previous models; the iteratively created models are combined into an ensemble of the models for making predictions. The XGBoost implementation stands out with some

major strengths including being fast, scalable, and outperforming other algorithms. It has been empirically proven to be highly effective in data science studies and won a wide array of machine learning competitions on Kaggle. In this project, XGBoost is used to create a model that predicts the medical billing price by the medical procedure and location that user enters. More information on details of the application of the XGBoost machine learning techniques in this project can be found in Chapter VI.

The medical procedure price predictive model and the data storage part of the application were implemented and hosted on Amazon Web Services (AWS) platform. The following tools were used:

Table 1. AWS tools used in the Healthcare Price Finder project

AWS Tools	Description
Sagemaker	A cloud machine learning platform with the ability to build, train, and deploy machine learning models quickly. Used for training and hosting the healthcare price estimator model in this project.
Relational Database Service (RDS)	A web service for setting up, operating, and scale distributed relational databases in the cloud. AWS RDS for MySQL is used in this project for data table storage and supporting data querying.
Simple Storage Service (S3)	A scalable object storage service on the cloud. S3 is used to store the raw data files and the trained model in this project.

I chose to use AWS mainly because AWS is almost the industry standard -- it is known as the best overall cloud computation platform with rich functionalities. With a

large user base, the AWS platform is often better documented with real examples compared to other cloud computation platforms. I also have worked with AWS in the past, which makes the development process easier. AWS allows users to pay only for the actual amount of computing resources that were consumed, and scales up or down based on the user demand easily for a reasonable price. Many tools commonly used for data processing and other web services are streamlined together on the AWS platform, which simplifies the setup process and provides better compatibility and reliability.

JavaScript and its frameworks like React.js, Node.js, and some smaller libraries are used to implement the Healthcare Price Finder web application. Google Maps API geocoding service is used here to determine the geographical location of addresses, calculate the distance between the users and healthcare providers, and display the providers in range on the interactive map.

Chapter V.

Data

5.1 Data Source

The source data, Medicare Provider Utilization and Payment Data: Physician and Other Supplier, is a part of the open datasets released to the public by US Center Medicare and Medicaid (https://www.cms.gov/) for transparency. It collects the utilization and payments for procedures and services by healthcare providers to the federal Medicare insurance plan members without any personal identifying information of the members. Physicians and other healthcare providers determine the amount they charge for the services and procedures provided, and Medicare pays by a pre-negotiated rate.

The dataset contains 100% of the processed Medicare claims from the healthcare providers and payment (allowed amount and payment amount) information from calendar year 2012 to 2017, organized by National Provider Identifier (NPI) and Healthcare Common Procedure Coding System (HCPCS) code, and place of service.

It should be noted that healthcare providers who did not file any Medicare claims and HCPCS codes that were not covered in the Medicare claims between 2012 and 2017 are missing from this dataset. However, it is possible to add them to the model by supplementing the Medicare Provider Utilization and Payments dataset with the official complete HCPCS codes list and the National Provider Registry dataset from the National Plan Provider Enumeration System.

Since more than 90% of the Medicare enrollees are 55 years or older, this source dataset has more data on medical procedures that are more commonly used by the older population, and less data on procedures mainly utilized by the younger population, like the pediatric surgeries. Because of this demographic bias, the source data does not reflect the pattern of healthcare utilization of the general population. However, the health spending data in 2016 showed that people who are age 55 and over made up 29% of the population but accounted for 56% of all, indicating the aging population tends to have more healthcare needs. Therefore, the Medicare Provider Utilization and Payments dataset still represents the medical needs of a substantial percentage of the total population, thus still highly useful for studying the pricing and utilization of a large number of medical procedures.

For the proof of concept of the Healthcare Price Finder project, we only used the data for the state of California which was 7.8% of the full United States dataset.

Table 2. Total number of the rows in the full Medicare Provider Utilization and Payments dataset and the California subset by year

Year	Entire US	California State	% of CA data in US
2012	9,153,272	720,674	7.87%
2013	9,287,876	724,547	7.80%
2014	9,316,307	726,474	7.80%
2015	9,497,891	740,947	7.80%
2016	9,714,896	759,715	7.82%
2017	9,847,443	779,555	7.92%

Total	56,817,685	4,451,912	7.84%

5.2 Dataset Details

Below is a table listing all the columns present in the Medicare Provider Utilization and Payments dataset since 2012, and the description of each column. The column naming conventions are slightly different across the years, and only 2017 column names are shown here.

Table 3. List of the columns in Medicare Provider Utilization and Payments dataset

	Column Name	Field Name (from 2017)	Data Type
1	National Provider Identifier (NPI)	npi	Plain Text
2	Last Name/Organization Name of the Provider	provider_last_org_name	Plain Text
3	First Name of the Provider	provider_first_name	Plain Text
4	Middle Initial of the Provider	provider_mi	Plain Text
5	Credentials of the Provider	credentials	Plain Text
6	Gender of the Provider	provider_gender	Plain Text
7	Entity Type of the Provider	entity_code	Plain Text
8	Street Address 1 of the Provider	provider_street1	Plain Text
9	Street Address 2 of the Provider	provider_street2	Plain Text
10	City of the Provider	provider_city	Plain Text
11	Zip Code of the Provider	provider_zip	Plain Text

12	State Code of the Provider	provider_state	Plain Text
13	Country Code of the Provider	provider_country	Plain Text
14	Provider Type	provider_type	Plain Text
15	Medicare Participation Indicator	medicare_participation_indicator	Plain Text
16	Place of Service	place_of_service	Plain Text
17	HCPCS Code	hcpcs_code	Plain Text
18	HCPCS Description	hcpcs_description	Plain Text
19	HCPCS Drug Indicator	hcpcs_drug_indicator	Plain Text
20	Number of Services	line_srvc_cnt	Numeric
21	Number of Medicare Beneficiaries	bene_unique_cnt	Numeric
22	Number of Distinct Medicare Beneficiary/Per Day Services	bene_day_srvc_cnt	Numeric
23	Average Medicare Allowed Amount	average_medicare_allowed_amt	Numeric
24	Standard Deviation of Medicare Allowed Amount (before 2014 only)	stdev_medicare_allowed_amt	Numeric
25	Average Submitted Charge Amount	average_submitted_chrg_amt	Numeric
26	Standard Deviation of Submitted Charge Amount (before 2014 only)	stdev_submitted_chrg_amt	Numeric
27	Average Medicare Payment Amount	average_medicare_payment_amt	Numeric
28	Standard Deviation of Medicare Payment Amount (before 2014 only)	stdev_medicare_payment_amt	Numeric
29	Average Medicare Standardized Amount (after	average_medicare_standard_amt	Numeric

	2016 only)		

1. **npi** – National Provider Identifier (NPI) for the performing provider on the claim. The provider NPI is the numeric identifier registered in NPPES (National Plan and Provider Enumeration System).

2. **provider_last_org_name** – When the provider is registered in NPPES as an individual (entity type code='I'), this is the provider's last name. When the provider is registered as an organization (entity type code = 'O'), this is the organization name.

3. **provider_first_name** – When the provider is registered in NPPES as an individual (entity type code='I'), this is the provider's first name. When the provider is registered as an organization (entity type code = 'O'), this will be blank.

4. **provider_mi** – When the provider is registered in NPPES as an individual (entity type code='I'), this is the provider's middle initial. When the provider is registered as an organization (entity type code = 'O'), this will be blank.

5. **credentials** – When the provider is registered in NPPES as an individual (entity type code='I'), these are the provider's credentials. When the provider is registered as an organization (entity type code = 'O'), this will be blank.

6. **provider_gender** – When the provider is registered in NPPES as an individual (entity type code='I'), this is the provider's gender. When the provider is registered as an organization (entity type code = 'O'), this will be blank.

7. **entity_code** – Type of entity reported in NPPES. An entity code of ‘I’ identifies providers registered as individuals and an entity type code of ‘O’ identifies providers registered as organizations.

8. **provider_street1** – The first line of the provider’s street address, as reported in NPPES.

9. **provider_street2** – The second line of the provider’s street address, as reported in NPPES.

10. **provider_city** – The city where the provider is located, as reported in NPPES.

11. **provider_zip** – The provider’s zip code, as reported in NPPES.

12. **provider_state** – The state where the provider is located, as reported in NPPES. The fifty U.S. states and the District of Columbia are reported by the state postal abbreviation. The following values are used for all other areas: 'XX' = 'Unknown' 'AA' = 'Armed Forces Central/South America' 'AE' = 'Armed Forces Europe' 'AP' = 'Armed Forces Pacific' 'AS' = 'American Samoa' 'GU' = 'Guam' 'MP' = 'North Mariana Islands' 'PR' = 'Puerto Rico' 'VI' = 'Virgin Islands' 'ZZ' = 'Foreign Country'

13. **provider_country** – The country where the provider is located, as reported in NPPES. The country code will be ‘US’ for any state or U.S. possession.

14. **provider_type** – Derived from the provider specialty code reported on the claim. For providers that reported more than one specialty code on their claims, this is the specialty code associated with the largest number of services.

15. **medicare_participation_indicator** – Identifies whether the provider participates in Medicare and/or accepts assignment of Medicare allowed amounts. The

value will be ‘Y’ for any provider that had at least one claim identifying the provider as participating in Medicare or accepting assignment of Medicare allowed amounts within HCPCS code and place of service. A non-participating provider may elect to accept Medicare allowed amounts for some services and not accept Medicare allowed amounts for other services.

16. **place_of_service** – Identifies whether the place of service submitted on the claims is a facility (value of ‘F’) or non-facility (value of ‘O’). Non-facility is generally an office setting; however other entities are included in non-facility.

17. **hcpcs_code** – HCPCS code used to identify the specific medical service furnished by the provider. HCPCS codes include two levels. Level I codes are the Current Procedural Terminology (CPT) codes that are maintained by the American Medical Association and Level II codes are created by CMS to identify products, supplies and services not covered by the CPT codes (such as ambulance services). CPT codes, descriptions and other data only are copyright 2016 American Medical Association. All rights reserved. CPT is a registered trademark of the American Medical Association (AMA).

18. **hcpcs_description** – Description of the HCPCS code for the specific medical service furnished by the provider. HCPCS descriptions associated with CPT codes are consumer friendly descriptions provided by the AMA. CPT Consumer Friendly Descriptors are lay synonyms for CPT descriptors that are intended to help healthcare consumers who are not medical professionals understand clinical procedures on bills and patient portals. CPT Consumer Friendly Descriptors should not be used for clinical

coding or documentation. All other descriptions are CMS Level II descriptions provided in long form. Due to variable length restrictions, the CMS Level II descriptions have been truncated to 256 bytes. As a result, the same HCPCS description can be associated with more than one HCPCS code.

19. **hcpcs_drug_indicator** –Identifies whether the HCPCS code for the specific service furnished by the provider is a HCPCS listed on the Medicare Part B Drug Average Sales Price (ASP) File.

20. **line_srvc_cnt** – Number of services provided; note that the metrics used to count the number provided can vary from service to service.

21. **bene_unique_cnt** – Number of distinct Medicare beneficiaries receiving the service.

22. **bene_day_srvc_cnt** – Number of distinct Medicare beneficiary/per day services. Since a given beneficiary may receive multiple services of the same type (e.g., single vs. multiple cardiac stents) on a single day, this metric removes double-counting from the line service count to identify whether a unique service occurred.

23. **average_Medicare_allowed_amt** – Average of the Medicare allowed amount for the service; this figure is the sum of the amount Medicare pays, the deductible and coinsurance amounts that the beneficiary is responsible for paying, and any amounts that a third party is responsible for paying.

24. **stdev_Medicare_allowed_amt** – Standard deviation of the Medicare allowed amounts. The standard deviation indicates the amount of variation from the average

Medicare allowed amount that exists within a single provider, HCPCS service, and place of service. Note: This variable has been removed beginning with calendar year 2014 data.

25. **average_submitted_chrg_amt** – Average of the charges that the provider submitted for the service.

26. **stdev_submitted_chrg_amt** – Standard deviation of the charge amounts submitted by the provider. The standard deviation indicates the amount of variation from the average submitted charge amount that exists within a single provider, HCPCS service, and place of service. Note: This variable has been removed beginning with calendar year 2014 data.

27. **average_medicare_payment_amt** – Average amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service.

28. **stdev_medicare_payment_amt** – Standard deviation of the Medicare payment amount. The standard deviation indicates the amount of variation from the average Medicare payment amount that exists within a single provider, HCPCS service, and place of service. Note: This variable has been removed beginning with calendar year 2014 data.

29. **average_medicare_standardized_amt** – Average amount that Medicare paid after beneficiary deductible and coinsurance amounts have been deducted for the line item service and after standardization of the Medicare payment has been applied. Standardization removes geographic differences in payment rates for individual services, such as those that account for local wages or input prices and makes Medicare payments across geographic areas comparable, so that differences reflect variation in factors such

as physicians' practice patterns and beneficiaries' ability and willingness to obtain care. Note: This variable is available starting with the calendar year 2014 data.

5.3 Data Cleaning

The raw Medicare Provider Utilization and Payment Data: Physician and Other Supplier datasets from the CMS Open Data website has many inconsistencies. For example, the source datasets are formatted slightly differently across the years, and some columns have data quality issues like human error from the data entry, or contain multiple forms of the same information. In order to combine the datasets from all the years as the training data for further processing in the next steps, additional data cleaning work is required. Data cleaning aims to identify and remove errors and duplicated data, in order to create a reliable dataset. Performing data cleaning improves the quality of the training data for analytics and enables accurate decision-making.

Data cleaning steps applied to the raw dataset include:

1) Remove all the data outside of the United States.
2) Use column names from the 2017 Medicare dataset as the standard column names. The data from all previous years are all mapped to the standard column names.
3) Remove all the columns from the raw datasets that aren't present in all the years. This includes "Standard Deviation of Medicare Allowed Amount", "Standard Deviation of Submitted Charge Amount", and "Standard Deviation of Medicare Payment Amount" from 2012 data and 2013 data,

and “Average Medicare Standardized Amount” for all the datasets from 2014 and after.

4) Remove duplicate rows from the datasets.
5) For 9-digit zip codes in provider_zip column, only the first 5 basis digits which indicate the destination post office or delivery area are kept. Anything after the 5 basis digits is truncated.
6) Add a year column to distinguish the data from each year.
7) Capitalize all the letters in the credentials column and remove all the non-letters. eg) “M.D.” and “md” both become “MD” and recognized as the same title.
8) Change all the provider specialty names in provider_type column to the version in the dataset from the most recent year (2017 at the time of this project).
9) Extract California dataset using “state” column (STATE = ‘CA’).

Chapter VI.

Machine Learning

6.1 Machine Learning Price Prediction

Machine Learning (ML) is a subset of the Artificial Intelligence (AI) methods that enables a system to learn from the data and detect patterns for automated probabilistic model building without being explicitly programmed (Murphy, 2012). Analytical models created with ML methods have been successfully applied in multiple fields to solve problems. ML is especially useful for enabling fast analysis of a large amount of data. In this project, I use ML techniques to estimate the billing price of medical procedures based on the Medicare utilization and payments data from CMS from 2012 to 2017. Specifically, the Gradient Boosted Decision Trees method XGBoost is used.

6.2 XGBoost

XGBoost (eXtreme Gradient Boosting) is an implementation of the Gradient Boosted Decision Trees (GBDT) machine learning method (Chen and Guestrin, 2015). Gradient Boosting (GB) is an approach that falls under the ensemble learning category in machine learning. Decision Trees (DT) algorithms build tree-like models of decisions and possible outcomes composed of a root node, internal nodes and leaf nodes (end nodes), starting from a root node and branch out through the internal nodes and eventually reach leaf nodes for outcomes. GBDT yields an ensemble of decision tree models where each

decision tree model learns from the results from the previous tree to enhance the prediction effectiveness.

XGBoost algorithm is outlined as below:

1) Regularized learning:

Given a dataset with n samples, there are independent variables x_i ($x_i \in \mathbb{R}^m$), and each of these variables has m features . For each of these variables x_i, there are corresponding dependent variables y_i ($y_i \in \mathbb{R}$) . A tree ensemble model uses the independent variables x_i and K additive functions to predict the dependent variable $y_i^{\wedge}$:

$$y_i^{\wedge} = \sum_{k=1}^{K} f_k(x_i), f_k \in F \quad \text{Eq. (1)}$$

F is the space of classification and regression trees (CART), and f_k corresponds to an independent tree structure with leaf scores.

With l as a loss function that measures the gap between the prediction $y_i^{\wedge}$ and the target y^i, and Ω is a term for penalizing the complexity of the model, the goal of each tree in the ensemble model is to minimize the regularized objective $\mathcal{L}(\phi)$ in Eq. (2):

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad \text{Eq. (2)}$$

For any independent tree structure f, T is the number of leaves, and ω_i is the weight score of the i^{th} leaf. The penalty the tree complexity Ω is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\omega_i\|^2 \quad \text{Eq. (3)}$$

2) Gradient Tree Boosting

Let $\hat{y}_i^{(t)}$ be the prediction of the i-th instance at the t-th iteration, a new tree f_t that improves the results the predicted results the most can be trained and added to the ensemble model to minimize the model training objective $\mathcal{L}^{(t)}$:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \Omega\left(f_t\right) \quad \text{Eq. (4)}$$

By expanding Eq (1) ~ Eq (4), the optimal weight for ω_j^* of leaf j, and the corresponding optimal values $\mathcal{L}^{\sim t}(q)$ can be written as:

$$\omega_j^* = -\frac{\sum_{i \in I_j} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right)}{\sum_{i \in I_j} \partial_{\hat{y}^{t-1}}^2 l\left(y_i, \hat{y}^{t-1}\right) + \lambda} \quad \text{Eq. (5)}$$

$$\mathcal{L}^{\sim t}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right)\right)^2}{\sum_{i \in I_j} \partial_{\hat{y}^{t-1}}^2 l\left(y_i, \hat{y}^{t-1}\right) + \lambda} + \gamma T \quad \text{Eq. (6)}$$

In practice, it is often difficult to compute this value for all the possible tree structures q exhaustively. Instead, a greedy algorithm that starts from one leaf and iteratively adds branches to the tree replaces this formula. Let I_L be the instances sets of left nodes and I_R the instances sets of right nodes after the split, then all the nodes $I = I_L \cup I_R$, and the loss reduction $\mathcal{L}_{\text{split}}$ after the split is converted as below:

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right)\right)^2}{\sum_{i \in I_L} \partial_{\hat{y}^{t-1}}^2\left(y_i, \hat{y}^{t-1}\right) + \lambda} + \frac{\left(\sum_{i \in I_R} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right)\right)^2}{\sum_{i \in I_R} \partial_{\hat{y}^{t-1}}^2\left(y_i, \hat{y}^{t-1}\right) + \lambda} - \frac{\left(\sum_{i \in I} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right)\right)^2}{\sum_{i \in I} \partial_{\hat{y}^{t-1}} l\left(y_i, \hat{y}^{t-1}\right) + \lambda}\right] - \gamma \quad \text{Eq. (7)}$$

Furthermore, two techniques are used in XGBoost algorithms to prevent overfitting. The first technique is shrinkage, which lowers the influence of each individual tree and leaves space for future trees to improve the model. The second technique is subsampling on features, in addition to the traditional random forest row subsampling. Both techniques are applied in the XGBoost model created in this project. More details on the XGBoost algorithm can be found in a study conducted by Chen and Guestrin in 2016.

For regression models, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are two of the most commonly used accuracy metrics for continuous results. MAE shows the average magnitude of the errors without considering their direction between the prediction results and the actual data, where all the differences are weighted equally. RMSE measures the average magnitude of the errors quadratically by taking the square root of the squared differences between the prediction results and the actual data, and gives a higher penalty to large errors. In this project, larger errors are particularly undesirable compared to smaller errors for users looking for an estimate of billing amount of the healthcare procedures, therefore RMSE is selected as the scoring metric in all the machine learning models, and MAE is only used as a reference metric in the interpretation of the final predictive model.

6.3 Feature Encoding

After removing the unique name columns and description columns from the full Medicare Payments dataset, the following 12 columns are used as features:

Table 4. Full list of features in the model training data

	Column Name	Field Name in Cleaned Dataset	Feature Type	Cardinality
1	Year	YEAR	Numeric	6
2	National Provider Identifier (NPI)	NPI	Categorical	106,746
3	Credentials of the Provider	CREDENTIALS	Categorical	1,557
4	Gender of the Provider	GENDER	Categorical	3
5	Entity Type of the Provider	ENTITY_CODE	Categorical	2
6	City of the Provider	CITY	Categorical	742
7	Zip Code of the Provider	ZIP_CODE	Categorical	1390
8	State Code of the Provider	STATE	Categorical	1
9	Provider Type	PROVIDER_TYPE	Categorical	90
10	Medicare Participation Indicator	MEDICARE_PARTICIPATION	Categorical	2
11	Place of Service	PLACE_OF_SERVICE	Categorical	2
12	HCPCS Code	HCPCS_CODE	Categorical	5530

Most of the features available are categorical, and 5 out of 12 features have a cardinality greater than 500. Since the XGBoost implementation package requires all the input data to be numeric or boolean, all the categorical columns have to be encoded first before being used for model training.

Two encoding methods were considered in the project. The first one is label encoding, which sorts all the unique values in a column and converts each value into an integer number ordinally. The second one is one-hot encoding, which consists of replacing each categorical feature with a series of new features, where each new feature represents a unique value in the original categorical feature, with value 1 indicating existent, and value 0 indicating non-existent.

With many of the machine learning models, one-hot encoding is often recommended over other ordinal encoding methods like label encoding. The issue with label encoding is it may lead to the predictive models falsely assuming the unique categories are ordered like the integers they convert into. This can cause errors or unexpected results especially in regression models when the relationship between the categories and target values are non-linear. One-hot encoding binarizes each category and prevents this ordinal error from label encoding; however. when a categorical feature has a high number of unique values, expanding it with one-hot encoding creates a large and sparse data, which requires an unreasonable amount of computing power to train a machine learning model.

Both encoding methods were tested with the data in this project, and the disadvantages described above with one-hot encoding were significantly taxing on the model performance. The source dataset contains multiple high cardinality columns like ZIP_CODE, NPI, and HCPCS_CODE, and one-hot encoding causes the dataset to grow more than 300 times in size, rendering the model fitting extremely difficult and expensive to perform. Different methods like clustering and splitting the dataset were tested to

reduce the dimensionality of the dataset, but none of them mitigates this issue without sacrificing model performance or adding too much complexity.

Compared to the test model results yielded with one-hot encoding, label encoded data performs well in XGBoost without significant decrease in model performance and takes much less computing resources. Further investigation confirmed that the decision tree based algorithms like XGBoost are tolerant of label encoded categorical features, since decision trees can make repeated splits on the same features to handle numeric-encoded features in a non-linear manner. Moreover, two categorical features -- HCPCS_CODE and ZIP_CODE -- in the source dataset, are also "ordered" to a certain degree while being categorical, as similar healthcare procedures often have similar codes in the HCPCS coding system, and locations that are geographically close also have similar zip codes. The linearity in label encoding enables the XGBoost to establish connections between values that are similar and take this connection into consideration while fitting models. Using label encoding also allows the model to work with the categorical values that are not present in the training dataset, as long as they are included as a part of the possible values in the label encoder. As a result, label encoding is selected as the categorical value encoding method used in this project.

6.4 Feature Selection

Feature selection is an approach of improving the effectiveness of machine learning models by identifying and removing the features that do not help much in making predictions. In this project, features are selected based on the feature importance

and effectiveness estimated from multiple rough training models, as well as their relevance of features in medical price estimator product design -- it must be feasible to determine the value of all the feature columns in the machine learning model input when users look up the price estimates for medical procedures. For example, there are multiple types of credentials for mental health professionals practicing in the psychotherapy, and it's often difficult for clients to find out the which type of the qualification a provider has prior to the visit, therefore making credentials a poor feature from product design perspective, even if it is highly relevant to the model outcome.

In XGBoost regression models, the Feature Importance metric (F-score) is the number of times a feature appears in a tree. However, this metric is biased towards features that have high cardinality, since the number of possible decision tree splits grows nonlinearly with higher cardinality. The calculation of F-score in XGBoost regression models is outlined as below:

- Inputs: fitted random forest predictive model m, tabular dataset (training or validation) D.
- For each feature j (column of D):
    - For each tree t in the random forest predictive model $1, ..., T$, compute $n_{t,j}$, the number of the times feature j appears in t.
    - Compute feature importance weight w_j for feature f_j defined as:

$$w_j = \sum_{t=1}^{T} n_{t,j}$$

In order to reduce the bias introduced by the high cardinality features, a different metric Feature Permutation Weight which measures how prediction score decreases when a feature is not available is also considered during the feature selection process. The feature permutation importance algorithm is outlined as below:

- Inputs: fitted predictive model m, tabular dataset (training or validation) D.
- Compute the reference score s of the model M on data D (RMSE is used as the reference score in this regression model).
- For each feature j (column of D):
    - For each repetition k in $1, ..., K$:
        - Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\tilde{D}_{k,j}$.
        - Compute the score s of model m on corrupted data $\tilde{D}_{k,j}$.
    - Compute importance i_j for feature f_j defined as:

$$i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j}$$

I first trained a rough XGBoost regression model to predict the billing charge amount in XGBoost with all the features to get an estimation of the feature importance. 80% of the dataset is used as the training data for fitting the model, and 20% of the dataset as the test data for validating the model. Most of the default parameter values are used for the first feature selection rough model, except max_depth is set at 5 instead of

the default value 3 to better accommodate the columns that have a large number of unique values, and subsample set at 0.8, The values of every model parameter listed below. More information on each parameter can be found in Chapter 6.5 Model Parameter Tuning.

- colsample_bylevel: 1
- colsample_bynode: 1
- colsample_bytree: 1
- gamma: 0
- learning_rate: 0.1
- max_delta_step: 0
- max_depth: 5
- min_child_weight: 1
- n_estimators: 100
- reg_alpha: 0
- reg_lambda: 1
- subsample: 0.8

With these parameters, A XGBoost model is fitted with a Root Mean Square Error (RMSE) of 760.21. Since the model parameters are not optimized, and the total number of the nodes in a model created with above parameters is much smaller than the number of unique values in the feature columns, there is still room for the model to be improved, resulting in a smaller RMSE value. More discussion on evaluating models

using RMSE metric can be found in Section 6.4 Model Evaluation. The F score and the feature permutation weights in the model are shown below:

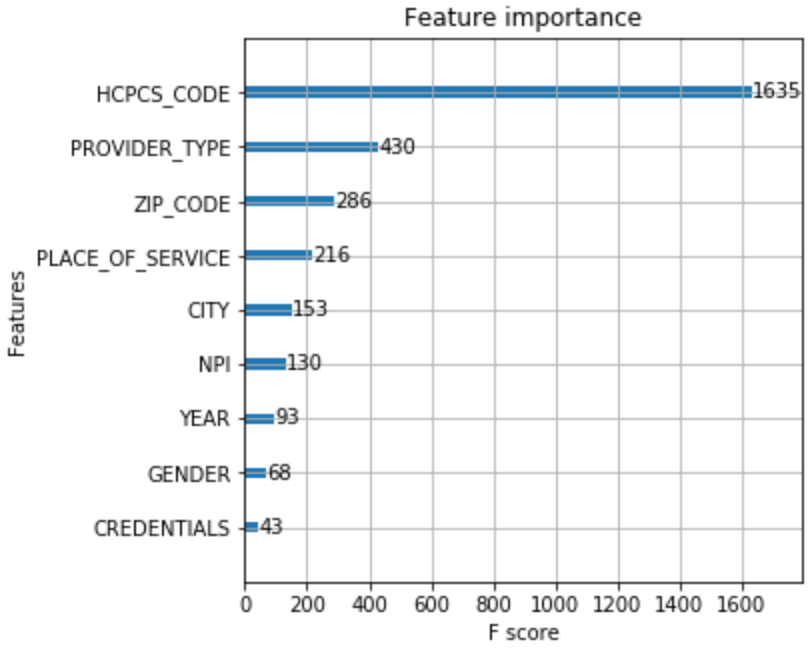

Figure 13. F scores in a trained XGBoost model with 100 decision trees and max tree depth of 5

Weight	Feature
0.6338 ± 0.0074	HCPCS_CODE
0.2436 ± 0.0051	PROVIDER_TYPE
0.0991 ± 0.0073	PLACE_OF_SERVICE
0.0349 ± 0.0021	GENDER
0.0245 ± 0.0013	ZIP_CODE
0.0144 ± 0.0010	NPI
0.0143 ± 0.0005	CREDENTIALS
0.0115 ± 0.0031	YEAR
0.0080 ± 0.0012	CITY
0 ± 0.0000	STATE
0 ± 0.0000	ENTITY_CODE
0 ± 0.0000	MEDICARE_PARTICIPATION

Figure 14. Feature permutation weights in a trained XGBoost model with 100 decision trees and max tree depth of 5

HCPCS_CODE and PROVIDER_TYPE are the top two features by both feature importance evaluation methods, and have a higher importance score than the rest of the features, indicating they are the strongest predictor for billing price of the medical procedures.

Four of the top five features in are the same (HCPCS_CODE, PROVIDER_TYPE, ZIP_CODE, and PLACE_OF_SERVICE) are the same in this model with max tree depth of 5, F score differs has CITY as one of the top 5 features, while feature permutation weight has GENDER in the top 5 features. While F score shows feature CITY is the 5th most splitted on, it has one of the lowest scores in feature permutation weights. The GENDER feature is the 4th place in the feature permutation weights, but was not splitted on many times.

HCPCS_CODE is the most important feature in both feature evaluation methods. In the F-score bar chart, the HCPCS_CODE feature was splitted on 1,635 times, much lower than 5,530 -- the cardinality of the HCPCS_CODE feature. GENDER only has 3 unique values, but was split 68 times. This indicates the parameter set up in this model is too simple for effectively accommodating the high number of unique values in this dataset effectively, settings in this rough model setup a simple model, since less than 30% of the unique values in the most important categorical feature HCPCS_CODE were covered in decision tree splitting. When this happens, the model tends to put the features

that have fewer values and can effectively slice the data into different categories, which is may be the reason that GENDER feature was one of the least used by F score, but has a high feature weight.

Both CITY and ZIP_CODE features are for geographical locations, with the CITY feature being a collection of ZIP_CODE supersets and therefore has a much lower cardinality than the ZIP_CODE feature. The disparity of CITY's F-score and feature permutation weight also indicated it was more effective than ZIP_CODE for slicing the dataset with its lower cardinality, but only has a minuscule effect on the model outcome once removed.

In order to gain better understanding of the feature importance in models that sufficiently cover the unique categorical values, another XGBoost regressor model is trained with similar settings as the above model, except the max tree depth is adjusted from 5 to 8. The new model has a RMSE of 590.09, better than the previous result which has an RMSE of 760.21. The feature importance score results in this model are shown below:

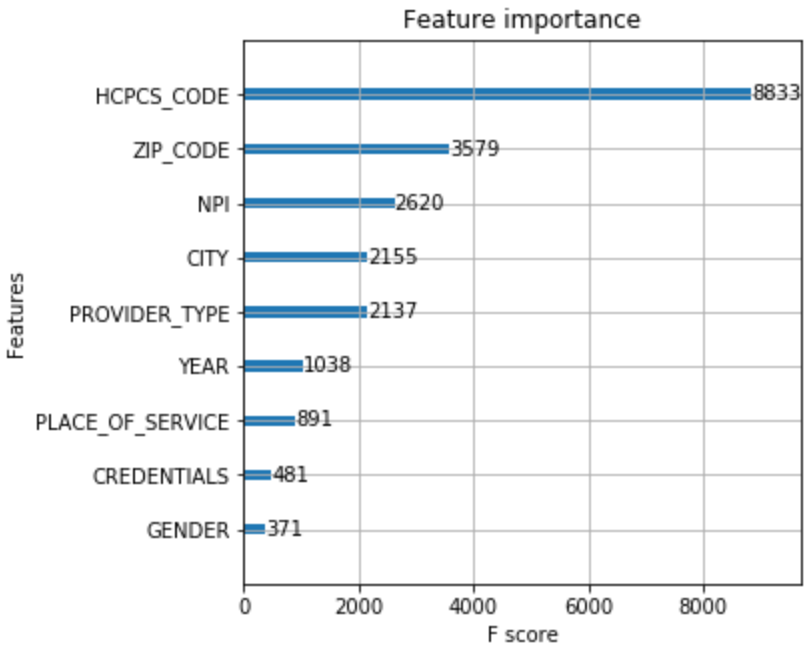

Figure 15. F scores in a trained XGBoost model with 100 decision trees and max tree depth of 8

Weight	Feature
1.1459 ± 0.0125	HCPCS_CODE
0.5039 ± 0.0216	PROVIDER_TYPE
0.1648 ± 0.0226	PLACE_OF_SERVICE
0.0790 ± 0.0032	ZIP_CODE
0.0412 ± 0.0019	NPI
0.0350 ± 0.0011	CITY
0.0267 ± 0.0059	YEAR
0.0241 ± 0.0012	GENDER
0.0186 ± 0.0009	CREDENTIALS
0 ± 0.0000	STATE
0 ± 0.0000	ENTITY_CODE
0 ± 0.0000	MEDICARE_PARTICIPATION

Figure 16. Feature permutation weights in a trained XGBoost model with 100 decision trees and max tree depth of 8

In the new XGBoost model with max tree depth of 8, HCPCS_CODE is still the most important feature in both feature importance evaluation methods. F score shows HCPCS_CODE was splitted on 8,833 times, inferring there are enough nodes to allow some HCPCS_CODE values to be splitted on multiple times, since the F-score is greater than the HCPCS_CODE's cardinality of 5,530. 4 out of the top 5 features in feature permutation weight ranking are consistent as the ones in the model with tree depth of 5 in both feature importance evaluation methods -- HCPCS_CODE, PROVIDER_TYPE, PLACE_OF_SERVICE, and ZIP_CODE.

NPI feature has the highest cardinality 106,746 and gets a much higher rank by both feature importance evaluation methods in complex models, the high cardinality limits the possibility of this feature being used effectively in simpler models for predicting the charge amount. Whether to select NPI as one of the features used during predictive model building implies very different goals from the product design perspective. Models that include the NPI feature and the geographical features predict the price of the medical procedures on an individual healthcare provider level. If the NPI feature is excluded, the models provide estimates of the medical procedure prices on the geographical location alone. In this project, the main product goal is to help users to get an idea of how much they would be charged for the selected healthcare procedure at a certain location, instead of an estimate from every healthcare provider. Since each physician is only licensed to provide a limited range of services, estimating how much an optometrist would charge for an open heart surgery is meaningless. For the charge

amount of specific individual healthcare providers, historical data are better indicators of how much a physician would charge for procedures and if a physician is able to provide services of a certain procedure. With above considerations, NPI should not be included in the list of the features in the final model.

Features including MEDICARE_PARTICIPATION, ENTITY_CODE, and STATE that are not listed in the chart have a feature importance score of 0, which means they weren't used at all during tree splitting and had no effect on the end predictive model. This implies Medicare Participation status and the entity type of the healthcare providers have no effect on how much they charge patients for their service. Since this model uses California data, the State information is the same for all of the rows and does not contribute to the predictive model for the billing price. However, if multi-state data is used as the training data, the state feature will have a higher impact in the model. In this project, these three features with a feature importance score of zero can be removed from the training dataset.

With these considered, these 4 features out of 12 feature columns are selected as the training data for building XGBoost models that estimates the medical procedure billing prices -- HCPCS_CODE, PROVIDER_TYPE, PLACE_OF_SERVICE, ZIP_CODE. Other features are dropped due to their low impact on the predictive model or the mismatch with the product goal.

6.5 Model Parameter Tuning

In XGBoost, multiple model parameters determine the specification of the predictive model, and can be modified to optimize the model performance. In addition to increasing prediction accuracy, tuning the model parameters can also reduce the effects of overfitting, which happens when models become over-complex and take random noise in this specific set of the training data and consider it as meaningful indicators, and failing to predict other cases reliably.

Multiple model parameters are available in XGBoost for tuning for better model performance. This project uses a combination of random search and gradient descent to tune the model parameters for the optimized outcome, and 5-fold cross-validation is used to confirm the robust behavior of models under each set of parameters. After extensive searching, the optimal XGBoost model parameters values selected are:

Table 5: XGBoost regression model parameters and the optimized selected values for the medical price estimator model

Parameter	Select Value	Description
n_estimators (alias: num_round)	1000	The number of iteration rounds for boosting. This is also the number of the decision trees built during the model fitting process.
learning_rate (alias: eta)	0.07	Step size of the weight shrinkage of the features after every boosting step. This parameter modifies the impact of each decision tree and makes the boosting process more conservative. When the learning rate is too high, it's easy to jump over the optimal, but a learning rate too low can take too long to converge or get stuck in a local optimum.

gamma (alias: min_split_loss)	0	Required minimum loss reduction for making a further split on a leaf of the decision tree. A higher gamma value gets a more conservative the model.
max_depth	10	The maximum depth of a decision tree. While a higher maximum depth value improves the model performance, setting this parameter too high makes the model more complex and more likely to overfit.
subsample	0.8	Percentage of the training data that the XGBoost randomly samples every time when building a decision tree during a boosting iteration. Setting this parameter lower than 1 helps to prevent overfitting.
min_child_weight	5	Minimum sum of instance weight needed in a leaf node (child). If the decision tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In XGBoost regression models, this parameter determines the minimum number of instances needed to be in each leaf node. The larger min_child_weight is, the more conservative the algorithm will be.
alpha (alias: reg_alpha)	0.5	L1 regularization term on weights. Increasing this value will make the model more conservative.
lambda (alias: reg_lambda)	0	L2 regularization term on weights. Increasing this value will make the model more conservative.
colsample_bytree	1	The subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.
colsample_bylevel	1	The subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.
colsample_bynode	1	The subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level.

6.6 Model Evaluation

Cross-validation is a technique for evaluating machine learning models by training several models using the same parameters on subsets of the available data, and evaluating the model performance on the complementary subset of the data.

The k-fold cross-validation method is used to perform the validation of the XGBoost model created in the Healthcare Price Finder Project. In k-fold cross-validation, the training data is splitted into k subsets (also called folds), and the machine model is trained on k-1 folds of data, and evaluated the model performance using the data fold that was not included in the training. This process is repeated on all the possible k combinations, with a different fold of data reserved for validation every time. 10-fold cross-validation is used here to evaluate the XGBoost regression model selected in the parameter tuning process, and below table shows the results:

Table 6. 10-fold cross-validation results of the tuned XGBoost model

Reserved Fold	RMSE
1	410.01787451
2	376.00552359
3	407.3985799
4	367.79329907 (lowest)
5	400.9424476
6	401.69931103
7	418.92752339
8	476.95150865

9	520.24082873
10	553.99132506 (highest)

The cross-validation results showed the model performance under the selected parameters is consistent with RMSE values around 400 in most of the cases, with the highest RMSE at 553.99132506 and the lowest at 367.79329907. These RMSE values translate to MAE values of around 100, indicating the XGBoost model trained with the selected parameters is able to provide estimates for healthcare procedures within 100 dollars difference on average.

6.7 Scaling with More Data

Due to the budget limitations, the healthcare price estimator model created in this project only uses California Medicare Provider Utilization and Payments data from 2012 to 2017 as a proof of concept. According to the 2019 US census, California's population makes up 12.9% in the population of the entire United States, and California Medicare utilization data makes up 7.83% of the entire US Medicare Payment utilization dataset. The full estimator tool needs to be capable of providing healthcare price estimates for the entire United States, which involves scaling up with about 12 times more data.

Another scaling challenge is to incorporate the data from future years as CMS releases more Medicare utilization data in the future. YEAR was not selected as a significant feature with the model built based on 2012-2017 Medicare Provider Utilization and Payments data, however, this may need to change as data from more years

become available in the future, and the model should be recalibrated to provide better predictions.

Two approaches can be used to scale up the model with more data:

- One model for the entire US
    - Pros: this approach is easy to manage and integrate with the front-end healthcare price application, since one endpoint is able to handle all the healthcare price estimating requests. This approach also covers uncommon healthcare procedures better, since the model can use the pricing data by HCPCS_CODE from other states as a reference if there's no past pricing data of such procedure at the selected location.
    - Cons: the amount and the cost of computing resources required to create the predictive model may grow faster than linearly in relative to the amount of the data. However, it is possible to mitigate this challenge by optimizing the data, like using clustering and bucketing on high cardinality features like ZIP_CODE or HCPCS_CODE to reduce the amount of the unique values the model needs to process.
- Multiple models in parallel with by region or types of healthcare service
    - Pros: the amount of the computing resources needed for this divide-and-conquer approach is more predictable, as it increases linearly with the amount of the data. This approach provides higher

flexibility if the training dataset updates often, since each model covers one segment of the source data, and only the models that correspond to the segment of the dataset that was modified need to be updated.

- Cons: Training and deploying multiple models requires more work, and an additional coordinating layer needs to be set up in the application to match each healthcare price estimate query to the corresponding predictive model.

In practice, more tests need to be performed to determine which approach is more appropriate depending on the cost and product design when scaling up to the full source data.

Chapter VII.

Application Implementation

7.1 Application Architecture

With the healthcare procedure price estimator model in place, I create an interactive application for users to easily search for the estimated prices of procedures in their area, as well as look up past payment data as a reference. The Healthcare Price Finder application consists of three parts: Front-end (client), Back-end (server), and Relational Database. The Front-end (client) part renders the application interface, handles the interactions between the application and the user, and presents the pricing data to the user. The Back-end (server) part processes and responds to the requests from the Front-end clients, connected to the databases and external services including the model endpoint deployed on AWS Sagemaker and Google Maps API. The Relational Database part stores the pre-processed data supporting the Healthcare Price Finder application in multiple tables.

The JavaScript library React is used to build the application user interface, and Node.js is used to build the backend server which supports the application. The source code of both the front-end and backend of the application is written in TypeScript, a superset of JavaScript that is more strongly typed and provides object-oriented programming that native JavaScript does not have, resulting in applications that are more

structured and maintainable. TypeScript source code compiles into JavaScript code during the build process. Multiple JavaScript libraries were also introduced for implementing key features in the application. The data supporting both the front and the backend of the application is stored in a MySQL relational database on Amazon RDS.

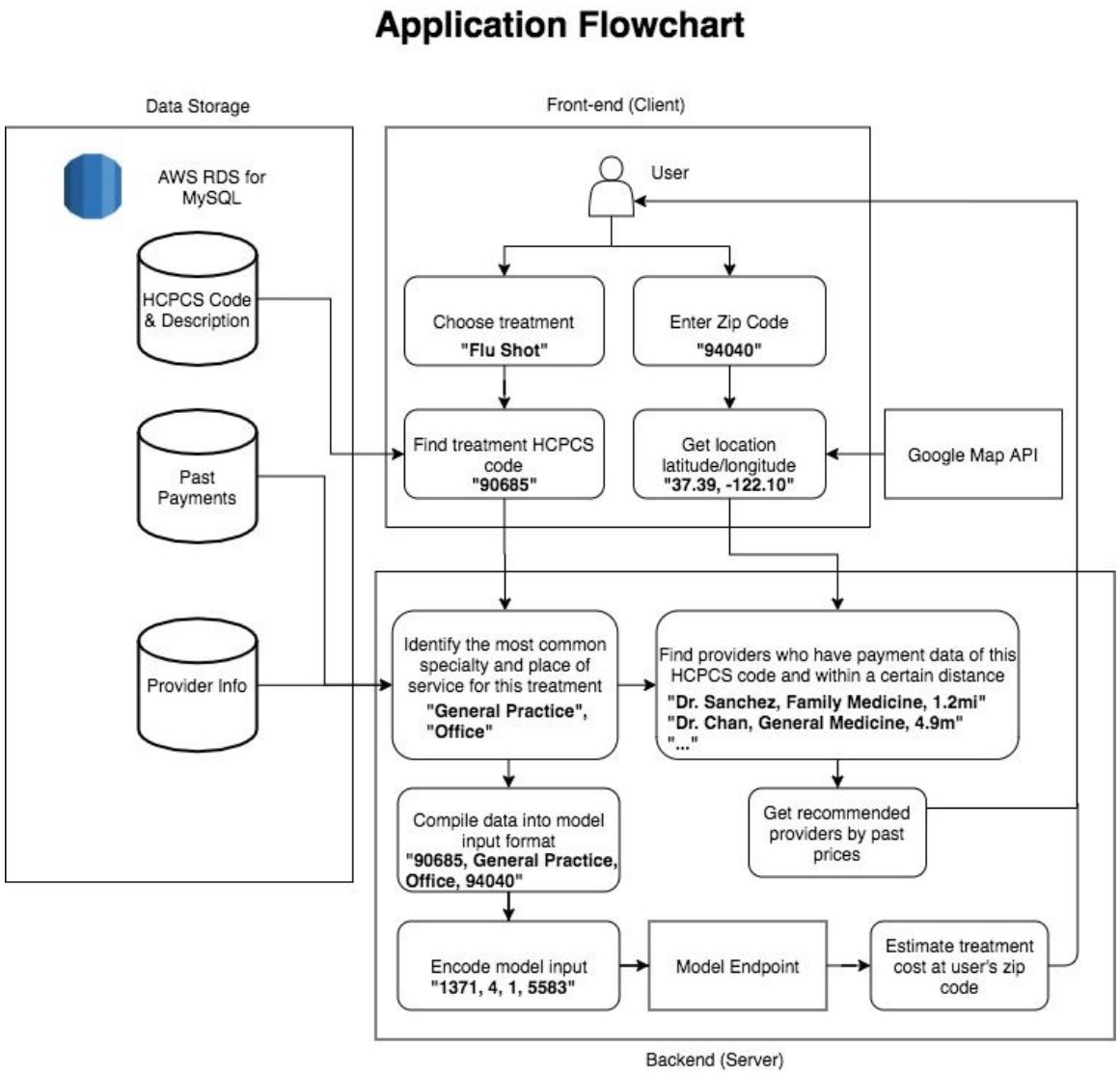

Figure 17. Overview flowchart of the Healthcare Price Finder Application

7.2 User Interface

I try to make the search process as simple and intuitive as possible for the users. With this in consideration, the application experience was designed to be lightweight and straightforward. Users only need to go through three easy steps to get get a provider charge amount estimate and see the past prices of the procedure they are looking for:

1. Enter the zip code of their location.
2. Enter keywords for the healthcare procedure they are looking for.
3. Choose a healthcare procedure.

Figure 18. Healthcare Price Finder application landing page

For average users, not everyone knows ahead the exact name of the healthcare procedures they are looking for, and even if they do, it's often difficult to memorize some

of the medical terminologies that are long and difficult to spell correctly. Therefore, fuzzy search is utilized in healthcare procedure search to list the likely matches using both the HCPCS code and HCPCS description for healthcare procedures as users type on the keyboard real-time. On determining the user location, users also do not need to enter their full addresses, just entering their zip code or giving the web application retrieve their location automatically is sufficient.

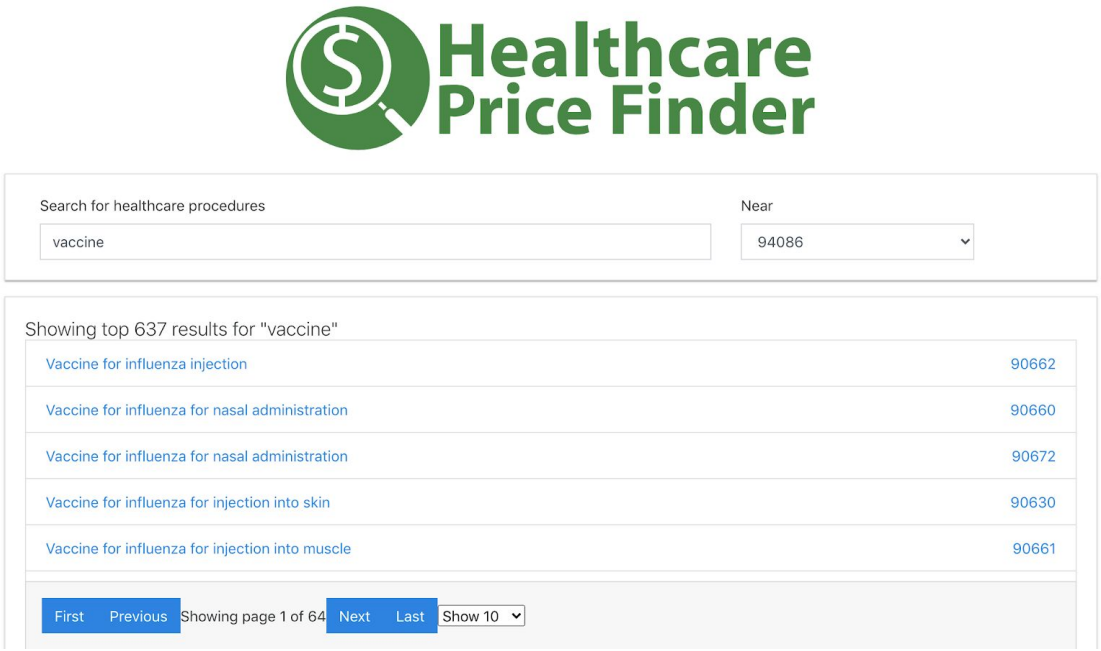

Figure 19. Healthcare Price Finder main search page after entering location and a keyword in the search bar

Although the price estimator machine learning model requires additional input including the place of service and provider specialty, it's often difficult for users to find out the place of service in advance and identify the correct provider specialty from a long

list of possible values, therefore the place of service and provider specialty are autofilled in the backend for machine learning model input by defaulting to the most probable value based on the patterns in the historical payments data for the selected HCPCS code.

The healthcare procedure prices estimate and past prices data are presented together on the healthcare procedure details page after the user selects the healthcare procedure and location. The price estimate at the user's zip code produced by the machine learning model is displayed on the top as "Estimated Uninsured Price", and the averages of the most recent charged prices and the actual amount with Medicare insurance on the selected healthcare procedure in the 300 miles radius are also shown on the top box as "Average Uninsured Price" and "Average Insured Price". The Medicare Utilization and Payments Dataset includes the healthcare procedure prices from each provider across multiple years, instead of showing the charge amount from every year, only the charge amount of the most recent year is shown on the HCPCS procedure details page.

On the first look, it may look excessive to include the past prices of the procedure when the estimated charge amount predicted by the model is available. However, I chose to present the data this way because the average price of the procedure in the 300 miles radius serves as a quick reference of the "fair price" around the user's location, while may not be the best indicator of how much the physicians at the selected zip code actually charge due to factors like the difference in individual physician's experience level and local living expenses. The average price with Medicare is included as a reference actual paid price for users who have healthcare insurance, an amount that varies little within a

state regardless how much the provider charges since the insurance pays the physicians at a pre-negotiated fixed amount by the performed procedure or offers a discount on the amount physicians charge. With this presentation of the prices, both users with and without insurance are able to benefit from the price data that is the most suitable to their healthcare insurance status.

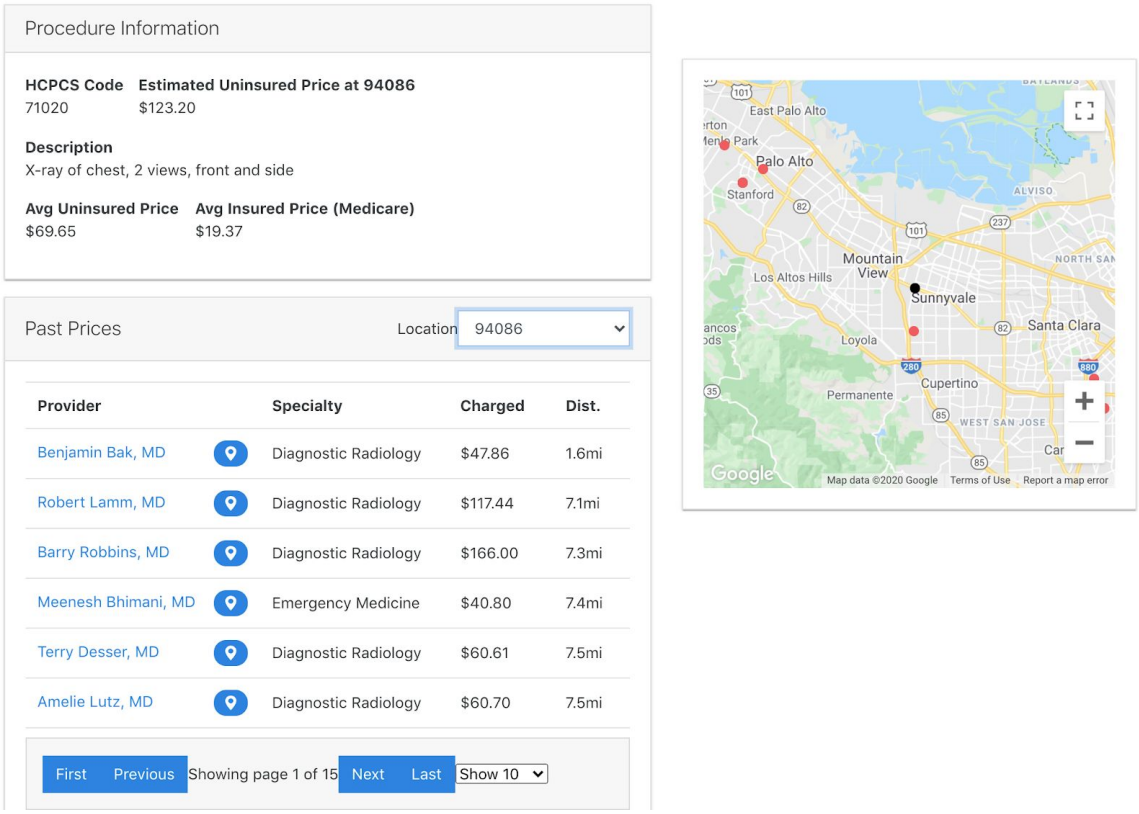

Figure 20. Healthcare Price Finder procedure price details page (when the past prices on the procedure at user's location is available)

Every provider who has filed Medicare claims from 2012 to 2017 on the healthcare procedure selected by the user within the 300 miles radius are listed on the

procedure prices details page, with the provider name, specialty, the most recent charge amount for the select procedure, and the distance to the user's zip code. The provider list is sorted ascendingly by the distance from the user's zip code by default to help users to find nearby providers. The sorting order can be easily changed by clicking on the column header, in case the user wants to find the cheapest option on the medical procedure they are interested in. The map made available by Google Map API is focused on the user's location (black dot) by default, and clicking the icon next to the listed provider names re-focuses the map on the selected provider's location.

Since the Medicare Provider Utilization and Payment source dataset only covers the healthcare procedures in the Medicare claims filed between 2012 and 2017, not all the medical procedures that each physician can provide are covered. It's impossible to get the prices for all the procedures at each healthcare provider, nor is it possible to list all the physicians who can provide the procedure selected by the user just by using the Medicare Provider Utilization and Payment Dataset. Past payments data may not exist for some less common procedures in certain locations. However, the machine learning model is still able to provide an estimate using the past payments data from other locations of similar procedures and the price trends of other procedures at the selected location.

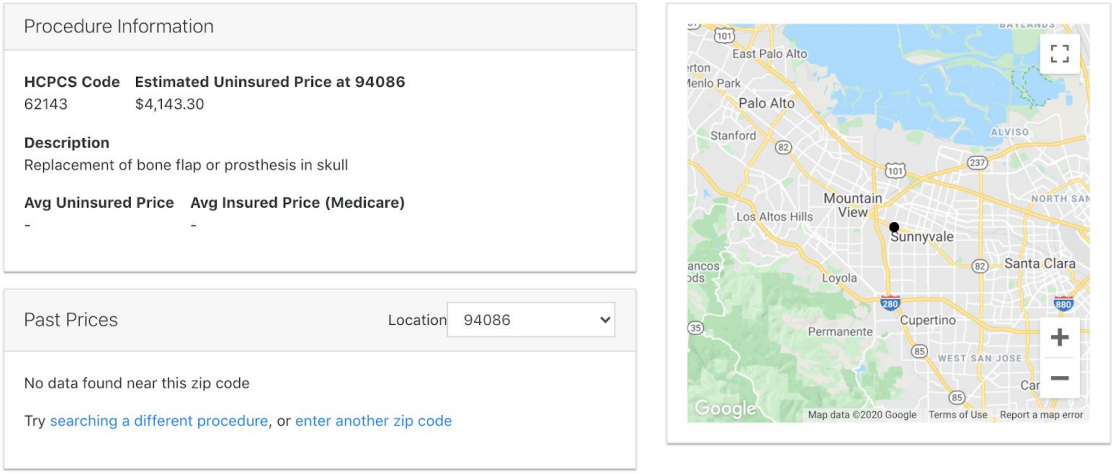

Figure 21. Healthcare Price Finder procedure price details page (when the past prices on the procedure at user's location is not available)

If a user is interested in learning more about a provider in the historical prices list, clicking on the provider name takes the user to a Provider Details page that includes additional information like the Provider's address and the full list of the uninsured and Medicare insured prices of the medical procedures by this provider. The side panel lists other providers nearby with the same specialty to help users to find other physicians who can provide similar medical services but aren't listed on the previous Procedure Prices Details page because they did not file a claim on the specific procedure that the user looked up. It should be noted that the list of the providers in the current version of the application is not exhaustive -- many healthcare providers never filed any claims to Medicare, therefore did not appear in the Medicare Provider Utilization and Payment Dataset. In the future, further application development on data integration with the full National Provider Index Registry dataset will enable the application to present a more

complete list of the physicians and enhance the healthcare provider recommendation functionalities.

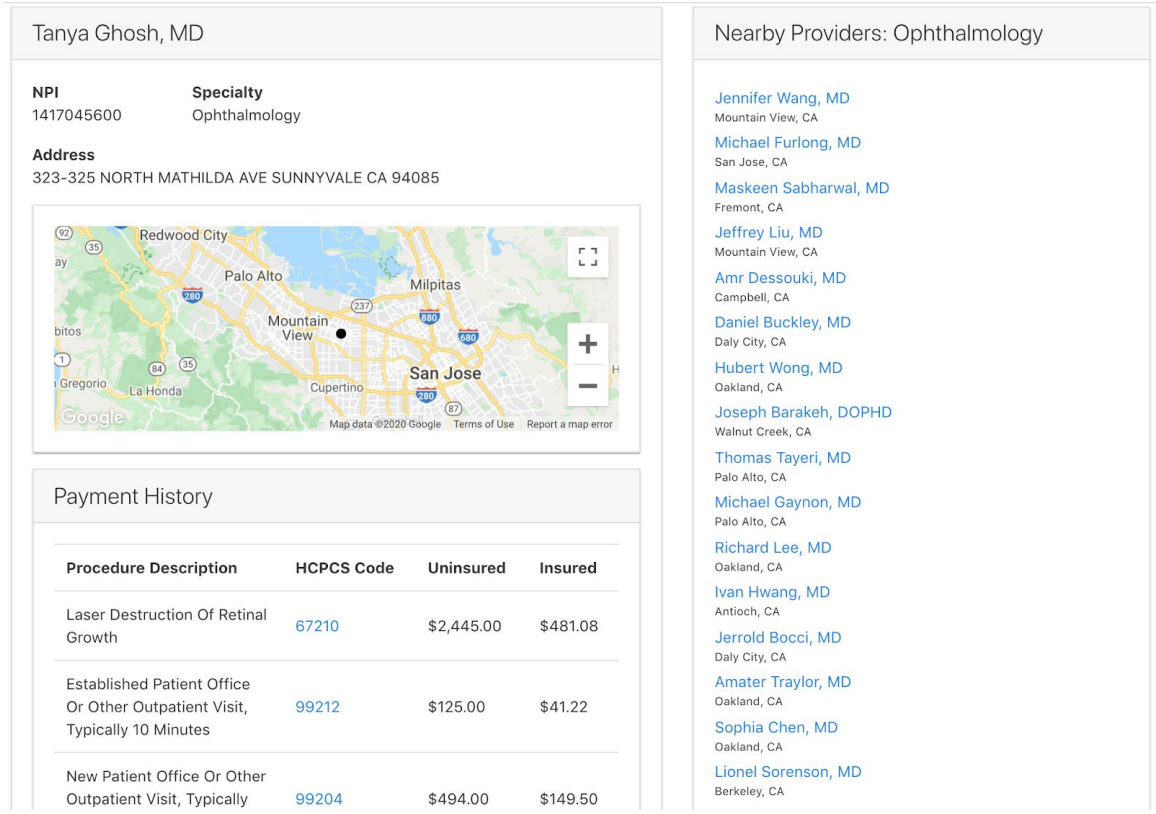

Figure 22. Healthcare Price Finder provider details page

There's still a lot of room for further development on the current version of the Healthcare Price Finder application. Future project development plans are discussed in detail in Chapter 8.

7.3 Relational Database

While it's convenient to use the cleaned dataset formatted as one big table with all the columns in feature selection, it's inefficient to use the same schema to support the front-end part of the Healthcare Price Finder web application. Features like the HCPCS code search and location lookup require fast response to small and lightweight data pulls, and querying from a big table is slow and expensive. In order to allow the medical procedure price estimate application to access and retrieve the data efficiently, the application data is modelled into three groups of smaller tables in a star schema:

- Payment fact table
- HCPCS tables
- Provider tables.

The data used in the application is modelled and aggregated from the Medicare Provider Utilization and Payment Data: Physician and Other Supplier 2012-2017 cleaned dataset, with additional data including the provider coordinates retrieved externally through Google API. These data tables supporting the application are stored in MySQL databases on Amazon Web Services RDS.

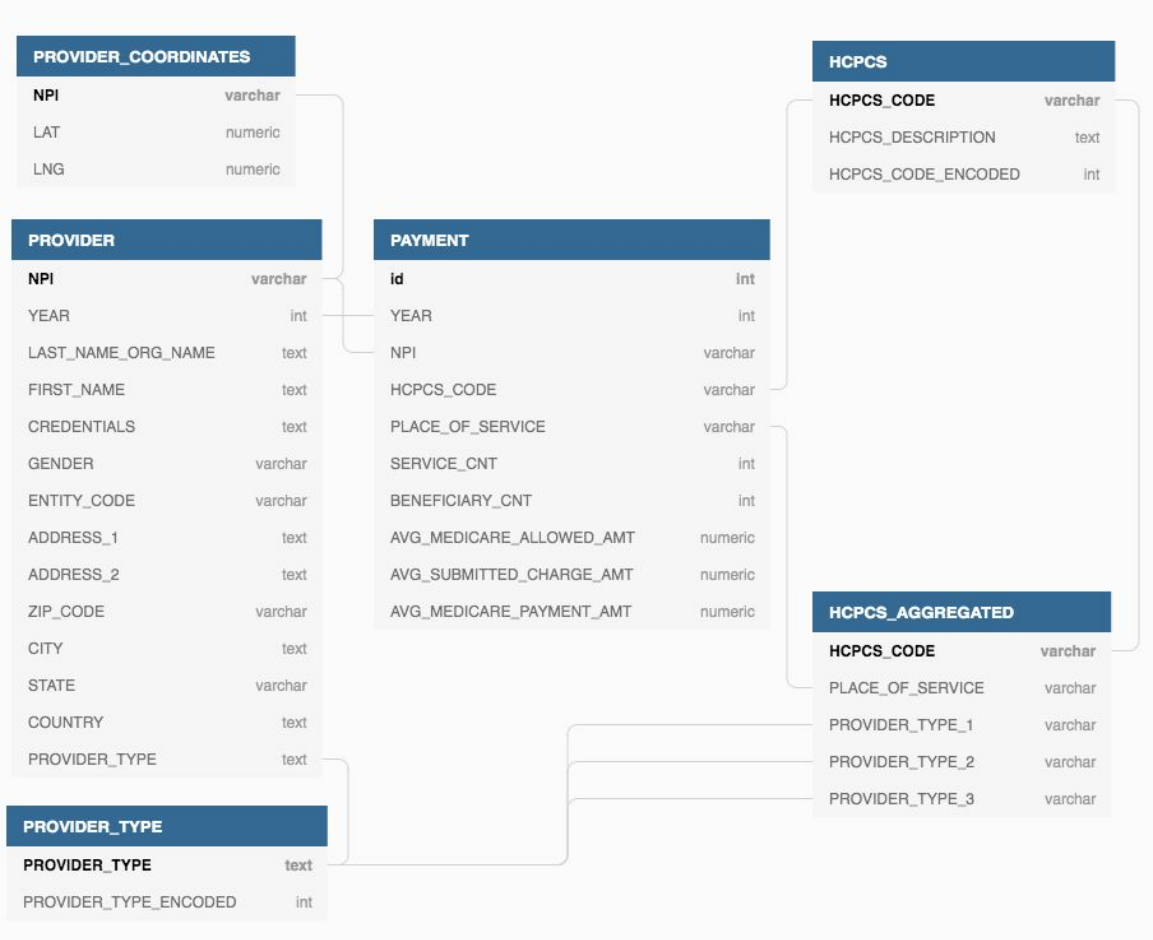

Figure 23. Healthcare Price Finder relational database schema

7.3.1 Payment Fact Table

Table Name: PAYMENT

- Primary Key Column: id
- Foreign Key Column(s): NPI (Provider tables), HCPCS_CODE (HCPCS tables)

This is the central table that stores information that directly pertains to the medicare payments. NPI and HCPCS Code columns are foreign key columns for provider tables and HCPCS code tables.

Table 7. PAYMENT table columns in Healthcare Price Finder relational database

	Column Content	Column Name	Data Type
1	id	id	int
2	Year	YEAR	int
3	National Provider Identifier (NPI)	NPI	varchar
4	Place of Service	PLACE_OF_SERVICE	varchar
5	HCPCS Code	HCPCS_CODE	varchar
6	Number of Services	SERVICE_CNT	int
7	Number of Medicare Beneficiaries	BENEFICIARY_CNT	int
8	Average Medicare Allowed Amount	AVG_MEDICARE_ALLOWED_AMT	Numeric
9	Average Submitted Charge Amount	AVG_SUBMITTED_CHARGE_AMT	Numeric
10	Average Medicare Payment Amount	AVG_MEDICARE_PAYMENT_AMT	Numeric

7.3.2 HCPCS Tables

There are two tables for HCPCS codes -- HCPCS and HCPCS_AGGREGATED. HCPCS_CODE is the primary key for all the HCPCS tables.

Table Name: HCPCS

- Primary Key Column: HCPCS_CODE
- Foreign Key Column(s): N/A

This dimension table stores the information on the medical procedures, with the HCPCS Code as the key column. It should be noted that only the HCPCS codes that appeared in the Medicare Payments source dataset are included. Only the most recent HCPCS code information is retained.

Table 8. HCPCS table columns in Healthcare Price Finder relational database

	Column Content	Column Name	Data Type
1	HCPCS Code	HCPCS_CODE	varchar
2	HCPCS Description	HCPCS_DESCRIPTION	text
3	Encoded HCPCS Code (for machine learning model input)	HCPCS_CODE_ENCODED	int

Table Name: HCPCS_AGGREGATED

- Primary Key Column: HCPCS_CODE
- Foreign Key Column(s): PROVIDER_TYPE_1, PROVIDER_TYPE_2, PROVIDER_TYPE_3

The columns in this table are aggregated based on the available Medicare payments data, and updates regularly with the addition of new data. It shows the most frequent values in the PLACE_OF_SERVICE column and the top 3 most frequent values

in PROVIDER_TYPE column in the Medicare Provider Utilization and Payments dataset from 2012 to 2017.

Table 9. HCPCS_AGGREGATED table columns in Healthcare Price Finder relational database

	Column Content	Column Name	Data Type
1	HCPCS Code	HCPCS_CODE	varchar
2	Place of Service	PLACE_OF_SERVICE	varchar
3	Provider Type	PROVIDER_TYPE_1	varchar
4	Provider Type	PROVIDER_TYPE_2	varchar
5	Provider Type	PROVIDER_TYPE_3	varchar

7.3.3 Provider tables

There are three tables for the provider data -- PROVIDER, PROVIDER_COORDINATES, and PROVIDER_TYPE, NPI is the primary key for PROVIDER and PROVIDER_COORDINATES tables, while the PROVIDER_TYPE table uses the text PROVIDER_TYPE column as the primary key.

.

Table Name: PROVIDER

- Primary Key Column: NPI
- Foreign Key Column(s): PROVIDER_TYPE

Table 10. PROVIDER table columns in Healthcare Price Finder relational database

	Column Name	Field Name	Data Type
1	Year	YEAR	int
2	National Provider Identifier (NPI)	NPI	varchar
3	Last Name/Organization Name of the Provider	LAST_NAME_ORG_NAME	text
4	First Name of the Provider	FIRST_NAME	text
5	Credentials of the Provider	CREDENTIALS	text
6	Gender of the Provider	GENDER	varchar
7	Entity Type of the Provider	ENTITY_CODE	varchar
8	Street Address 1 of the Provider	ADDRESS_1	text
9	Street Address 2 of the Provider	ADDRESS_2	text
10	City of the Provider	CITY	text
11	Zip Code of the Provider	ZIP_CODE	varchar
12	State Code of the Provider	STATE	Plain Text
13	Country Code of the Provider	COUNTRY	Plain Text
14	Provider Type (Specialty)	PROVIDER_TYPE	Plain Text

Table Name: PROVIDER_COORDINATES

- Primary Key Column: NPI
- Foreign Key Column(s): N/A

In addition to the provider columns from the raw data, the provider dimensional table also includes two additional columns -- latitude and longitude -- which were retrieved through the Google Map API using the provider address.

Table 11. PROVIDER_COORDINATES table columns in Healthcare Price Finder relational database

	Column Content	Column Name	Data Type
1	National Provider Identifier (NPI)	NPI	varchar
2	Latitude of the Provider's most recent Address	LAT	numeric
3	Longitude of the Provider's most recent Address	LNG	numeric

Table Name: PROVIDER_TYPE

- Primary Key Column: PROVIDER_TYPE
- Foreign Key Column(s): N/A

Table 12. PROVIDER_TYPE table columns in Healthcare Price Finder relational database

	Column Content	Column Name	Data Type
1	Provider Type (Specialty)	PROVIDER_TYPE	varchar
2	Encoded Provider Type (for machine learning model input)	PROVIDER_TYPE_ENCODED	int

7.4 Front-End (Client)

The Front End part of the Healthcare Price Finder application was developed with React.js as a medium that handles the interaction with the users and sending the requests to the backend server.

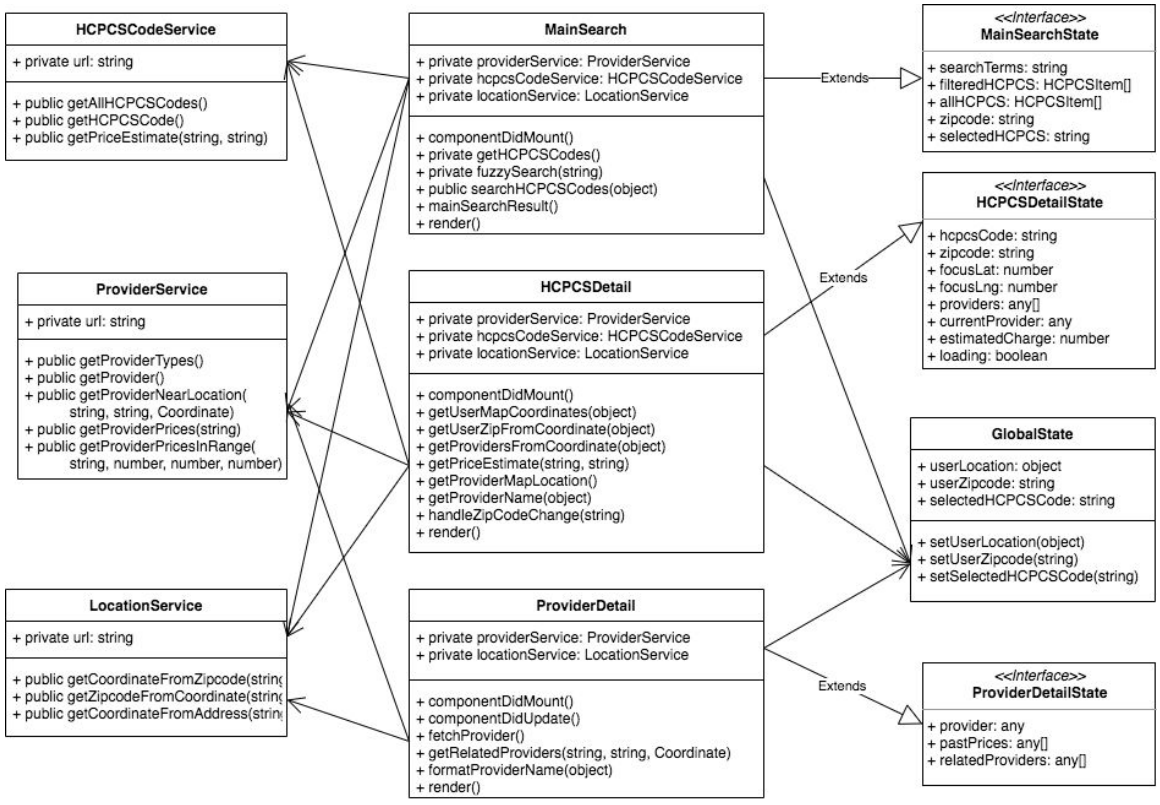

Figure 24. Healthcare Price Finder Application front-end core structure UML diagram

External class components like React.js, Fuse.js and components that build the application interface are not included in this diagram for visual clarity.

The application currently only has 3 pages -- Main Search, Healthcare Procedure Details, and Provider Details. Each page is a separate JavaScript class with variables and functions and handles the user interactions specific to the page, connects to the relevant services on the server, as well as renders the HTML code for each web page. The 3 pages are connected via React Router Navlink which activates and directs users to different pages of the application based on user inputs.

Considering the user privacy, none of the user-specific information is ever passed to the server. On the client side. Global State class that is accessible to all the page classes is used to store user information locally, including the user's location, selected healthcare procedure, and possibly other information like the username and saved providers with the future developments of the application. Each page also stores page-specific values in their own classes, and connects to HCPCSService, ProviderService, and LocationService which communicates with the server using Axios.js, a promise based HTTP client for the browser and Node.js. The Main Search page uses Fuse.js to support the fuzzy search on the healthcare procedures using the HCPCS code and description.

Multiple elements based on the native JavaScript and components and visual components from React.js are customized and used on different pages of the application. This includes modals, layouts, paginated tables, maps are used to build the application interface.

7.5 Back-End (Server)

The backend of the Healthcare Price Finder Application is developed using Node.js, an open-source, asynchronous event-driven JavaScript runtime environment that executes JavaScript code outside a web browser. Four Context classes (AWSContext, HCPCSContext, ProviderContext, LocationContext) and three Controller classes (HCPCSController, ProviderController, LocationController) are the core components of the back-end server, with an external MySQL database, an AWS Sagemaker endpoint, and Google Maps API each supporting a different type of data handling of the application.

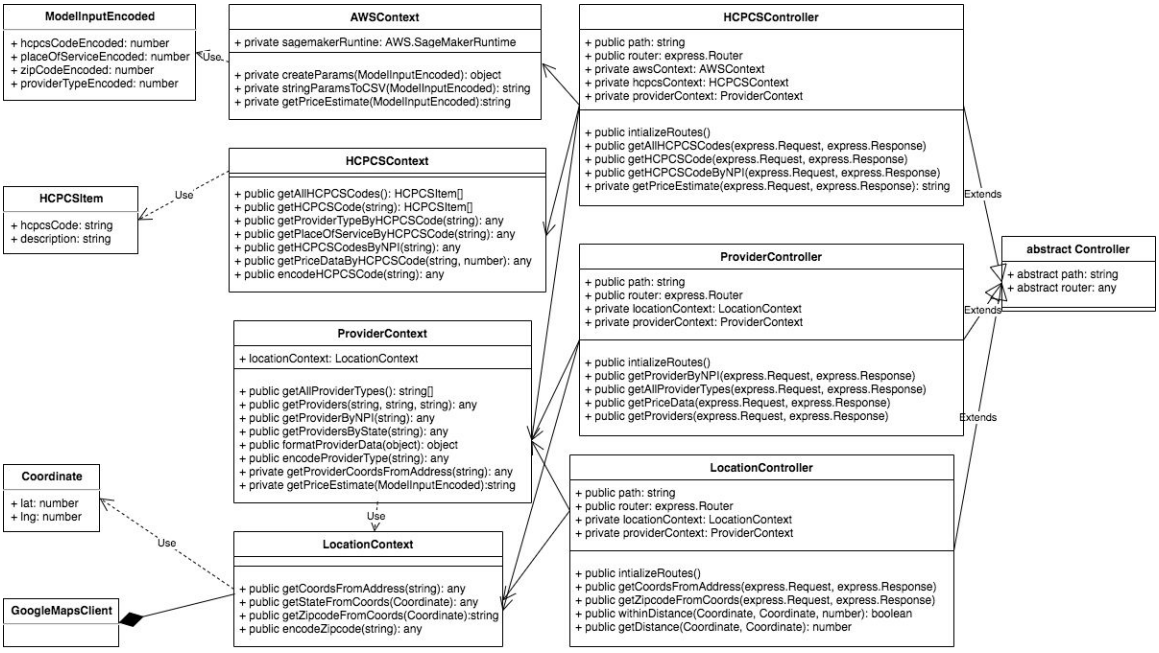

Figure 25. Healthcare Price Finder Application Back-end server UML diagram

Context classes handle the communication with the database and external services. Most of the functions in HCPCSContext class and ProviderContext class connect to the MySQL database and use SQL queries to retrieve the healthcare procedure data and provider data. LocationContext connects to Google Maps API and handles geocoding that retrieves the exact latitude and longitude from the address strings, as well as reverse geocoding that identifies the state and zip code from the coordinates. AWSContext connects to the healthcare procedure price estimator model deployed on AWS Sagemaker, organizes the user's inquiry data into the model input format and sends to the AWS Sagemaker endpoint as a request, and receives the predicted prices in response. Functions which encode the user inquiry data into the model input format are also defined inside the context classes.

Controller classes uses Express.js to accept the requests from the front-end client application, then call the Context classes to get the data needed to complete the user's request, process the data retrieved through Context classes, and route the processed result as responses back to the client part of the Healthcare Price Finder application. An abstract Controller class is defined first and extended as the separate controller classes including HCPCSController, ProviderController, and LocationController to handle different types of data requests.

Chapter VIII.

Summary and Future Work

In efforts to improve the pricing transparency in the US healthcare system, I attempt to create a Healthcare Price Finder tool that allows people to get an estimate of how much they will be charged for medical services before their visit.

I used the Medicare Provider Utilization and Payment dataset between 2012 and 2017, and developed an XGBoost regression machine learning model on AWS Sagemaker that is able to predict the prices of medical procedures even when past data is missing. I also created Healthcare Price Finder web application so average users can get healthcare procedure price estimates and look up the past prices data at their location easily.

While multiple online services that allow users to look up the past prices of the healthcare procedures exist, none of them is able to provide a price reference when the past prices are not available. Healthcare Price Finder attempts to address this issue by introducing machine learning techniques to provide price predictions when past pricing data is missing.

The Healthcare Price Finder tool still has a lot of room for additional development. The current version is a proof of concept, only California is supported in the medical procedure price estimator machine learning model and the web application.

In future iterations, the application will be scaled up to support medical price estimates anywhere in the United States.

Given the high complexity nature of the healthcare data, there is still a lot of work that can be done to improve the data quality and completeness in Healthcare Price Finder. One approach is to supplement the Medicare Provider Utilization and Payment dataset with other external datasets. The current Medicare Provider Utilization and Payment dataset does not include healthcare providers who did not filed claims to Medicare between 2012 and 2017, and the information of the provider whose situation has updated (eg. relocation, retirement, name change) since last being included in the source dataset is not accurate. By performing lookups with the National Provider Identifier, it is possible to replace the outdated provider data from the Medicare Provider dataset with the latest provider data from National Plan and Provider Enumeration System (NPPES) in the application, and include the providers that have never filed Medicare claims in the past. With better provider data, additional provider recommendation features can be developed.

The Center of Medicare and Medicaid Services is expected to release Medicare Provider Utilization and Payments data from more years in the future, together with the recent hospital price transparency rule expected to be implemented in January 2021, more healthcare procedure pricing data will become available. Taking these additional pricing data into consideration in the project can help the price estimator machine learning model to make more accurate price predictions, and provide users with more historical prices as references.

The source dataset used in this project is only one part of the data released by the Center of Medicare and Medicaid Services, past drug prices and inpatient hospital charged prices have also been shared with the public. It's possible to utilize the similar approach in this project to use these additional datasets to expand the predictive model and web application to cover the drug price estimates and inpatient treatment estimates.

In addition to the data and predictive modelling, further work can also be done on the web application side to refine the presentation of the healthcare pricing information and improve the user experience. Possibilities include: identify the HCPCS codes for medical procedure that often happen together (eg. surgeries and anesthesia), automatically recommend relevant healthcare procedures when users select one. HCPCS codes that refer to similar procedures (eg. 87502 and 87504 both refer to "detection test of influenza virus") can also be batched together to reduce the number of the similar items in the healthcare procedure search results.

Healthcare is something that all of us will interact with in our lifetime, and tools designed to combat the pricing intransparency in the US healthcare system have the potential of impacting a lot of people directly. Although the Healthcare Price Finder tool presented in this thesis is still an early version, I plan to continue the development and release it in the future in hope to help people to make financially informed healthcare decisions, and contribute to improving the well-being of the general public.

References

Amazon Simple Storage Service (Amazon S3). Amazon. https://aws.amazon.com/s3/.

Amazon Web Services Development Team. (2019). *Amazon SageMaker: Developer Guide*. https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-dg.pdf.

Amazon Web Services, Inc. (2007). *Amazon Relational Database Service Documentation*. Amazon. https://docs.aws.amazon.com/rds/index.html.

Amazon Web Services, Inc. (2015). *AWS General Reference*. Amazon. https://docs.aws.amazon.com/general/latest/gr/Welcome.html.

Association, A. M. (2015). *Cpt Professional*. American Medical Association Press.

Axios. *axios.js*. GitHub. https://github.com/axios/axios.

Breiman, L., Last, M., & Rice, J. Random Forests: Finding Quasars. *Statistical Challenges in Astronomy*, 243–254. https://doi.org/10.1007/0-387-21529-8_16

Center of Medicare and Medicaid Services. (2019). *Medicare Provider Utilization and Payment Data: Physician and Other Supplier*. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier.

Center of Medicare and Medicaid Services. *2018 Medicare Enrollment Section*. CMS. https://www.cms.gov/research-statistics-data-systems/cms-program-statistics/2018-medicare-enrollment-section.

Center of Medicare and Medicaid Services. *What's Medicare?* Medicare. https://www.medicare.gov/what-medicare-covers/your-medicare-coverage-choices/whats-medicare.

The Centers for Medicare and Medicaid Services, Office of Enterprise Data and Analytics. (2019, June 3). *Medicare Fee-For-Service Provider Utilization & Payment Data Physician and Other Supplier Public Use File: A Methodological Overview*. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/Medicare-Physician-and-Other-Supplier-PUF-Methodology.pdf.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

ClearHealthCosts. Clear Health Costs. (2012, April 5). https://clearhealthcosts.com/.

Estimating your treatment costs. Estimating your treatment costs | Kaiser Permanente ®. https://info.kaiserpermanente.org/html/estimating_your_treatment_costs/.

Facebook. React – A JavaScript library for building user interfaces. https://reactjs.org/.

FAIR Health Consumer. FAIR Health. https://www.fairhealthconsumer.org/.

Find Your Fair Price. Healthcare Bluebook – Your Guide to Fair Pricing for Healthcare Services. https://www.healthcarebluebook.com/ui/consumerfront.

Forbes Magazine. *The Biggest Health Insurers In The U.S.* Forbes. https://www.forbes.com/pictures/flil45gmeg/the-biggest-health-insur/.

Get cost estimates before choosing care. UnitedHealthcare. https://www.uhc.com/individual-and-family/member-resources/health-care-tools/cost-estimator.

Google. *Geo-location APIs | Google Maps Platform | Google Cloud*. Google. https://cloud.google.com/maps-platform/.

Himmelstein, D. U., Lawless, R. M., Thorne, D., Foohey, P., & Woolhandler, S. (2019). Medical Bankruptcy: Still Common Despite the Affordable Care Act. *American Journal of Public Health*, *109*(3), 431–433. https://doi.org/10.2105/ajph.2018.304901

Korobov, M., & Lopuhin, K. *Permutation Importance*. Permutation Importance - ELI5 0.9.0 documentation. https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html.

Kumar, S., Ghildayal, N. S., & Shah, R. N. (2011). Examining quality and efficiency of the US healthcare system. *International Journal of Health Care Quality Assurance*, *24*(5), 366–388. https://doi.org/10.1108/09526861111139197

Manage your health care costs. Aetna. https://www.aetna.com/individuals-families/using-your-aetna-benefits/manage-health-care-costs.html.

Mayo Foundation for Medical Education and Research. *Mayo Cost Estimator*. Mayo Clinic. https://costestimator.mayoclinic.org/.

Mckinney, W. (2017). *Python for data analysis: data wrangling with pandas, numpy, and ipython*. O'Reilly Media.

Murphy, K. P. (2012). *Machine Learning: a probabilistic perspective*. MIT Press.

MySQL. https://www.mysql.com/.

Nielsen, D. (2016). *Tree Boosting With XGBoost Why Does XGBoost Win "Every" Machine Learning Competition?* (thesis).

Node.js. https://nodejs.org/.

Pinder, J. *What if all US health care costs were transparent?* TED. https://www.ted.com/talks/jeanne_pinder_what_if_all_us_health_care_costs_were_transparent.

Reinhardt, U. E., Hussey, P. S., & Anderson, G. F. (2004). U.S. Health Care Spending In An International Context. *Health Affairs*, *23*(3), 10–25. https://doi.org/10.1377/hlthaff.23.3.10

Risk, K. *What is Fuse.js?* Fuse.js. https://fusejs.io/.

Rossum, G. V. (2000). *Python reference manual*. iUniverse.com, Inc.

SQLite Home Page. https://sqlite.org/index.html.

Treatment Cost Advisor Tool - Calculate Cost of Treatment: BCBS of WNY. https://www.bcbswny.com/. https://www.bcbswny.com/content/wny/find-a-doctor/treatment-cost-advisor.html.

Typed JavaScript at Any Scale. TypeScript. https://www.typescriptlang.org/.

Wikimedia Foundation. (2020, June 18). *Medical billing*. Wikipedia. https://en.wikipedia.org/wiki/Medical_billing.

XGBoost Developers. *XGBoost Documentation*. XGBoost Documentation - xgboost 1.2.0-SNAPSHOT documentation. https://xgboost.readthedocs.io/en/latest/index.html.

Zayas, C. E., He, Z., Yuan, J., Maldonado-Molina, M., Hogan, W., Modave, F., Guo, Y., & Bian, J. (2016). Examining Healthcare Utilization Patterns of Elderly Middle-Aged Adults in the United States. Proceedings of the ... *International Florida AI Research Society Conference.* Florida AI Research Symposium, 2016, 361–366.