Healthcare Cost Transparency and Recommendation Platform

ADHYAN SHARMA

TABLE OF CONTENT

TOPIC	PAGE NUMBER
BACKGROUND	3
OBJECTIVE/GOALS	4-5
DATA EXTRACTION	6-7
DATA EXPLORATION	8-9
DATA ANALYSIS	10-11
LOGIC-SEMANTIC MATCHING	12-13
CONCLUSION	14
REFERENCES	15

INTRODUCTION & BACKGROUND

Healthcare in the United States suffers from high costs and low transparency. U.S. per capita healthcare expenditure far exceeds that of other developed countries, without commensurate improvements in outcomes. In 2019, U.S. healthcare spending reached $3.8 trillion (17.7% of GDP), placing substantial strain on patients, employers, and government budgets. A significant barrier to cost containment is **price opacity**: patients typically do not know the cost of a procedure until after receiving care, and prices for identical services can vary wildly across hospitals and doctors. This information asymmetry is known to impede informed decision-making and inflate costs. For instance, studies show the same MRI might cost from $300 to $6,200 depending on where it is performed. Such unpredictability can lead to surprise bills and financial stress for patients.

In response, regulators have enacted price transparency mandates. Notably, the CMS **Hospital Price Transparency Rule** (effective January 1, 2021) requires all U.S. hospitals to post clear, machine-readable files of all standard charges for services. This includes a comprehensive list of every item and shoppable services in a consumer-friendly format. CMS is enforcing compliance with audits and fines. Additional regulations (such as the No Surprises Act) and state laws (e.g. in Massachusetts and New York) further push for visibility into costs. However, even with raw data posted, patients often lack easy tools to interpret it. Early analyses note that despite transparency rules, **it remains difficult for patients to estimate actual costs**. Consumers need user-friendly analytics and recommendations on top of raw price lists.

Motivated by these challenges, our project addresses a clear gap: building a digital platform that aggregates disparate price data and assists consumers in shopping for care. The platform is driven by two key motivators: (1) **empowering patients and payers** to make cost-aware decisions, thereby improving affordability. (2) **Advancing policy goals** by making price data actionable, so market forces can drive competition and efficiency. The combination of new regulations and rising patient activism suggests strong demand for such tools. By leveraging machine learning and data visualization, the platform will transform raw pricing data into clear, actionable insights for end users. In doing so, it contributes to the goals of the Triple Aim by potentially reducing per-capita costs and improving care value.

OBJECTIVE

The primary objective of the Medicare Cost Prediction Platform project is to **increase healthcare cost transparency for Medicare procedures** in a way that is intuitive and helpful for end users. To achieve this, the project sets the following specific goals:

User-Friendly Cost Lookup: Develop a web-based application (using Streamlit) that allows users to search for medical procedures or enter symptoms in plain language. The interface should be simple: for example, a search box for procedures or symptoms and an optional field for the user's ZIP code to tailor cost predictions to a specific area. The platform's front-end should guide users to relevant results even if they input non-technical terms (e.g. typing "MRI" or "shoulder pain" should retrieve the appropriate procedure information).

Accurate Cost Prediction: Utilize machine learning (a LightGBM regression model) to predict the Medicare cost of a given procedure. Instead of relying solely on national averages, the model should take into account regional factors and other features from the data so that the cost estimate is as accurate as possible for the context of the query. The goal was to achieve a low prediction error (measured by Mean Absolute Error in dollars) so that users can trust the estimates. For instance, if the average cost of a certain outpatient procedure is $2,000 in one state and $3,000 in another, the model should reflect those differences in its prediction.

Affordability Scoring: Go beyond raw cost and provide an **Affordability Score** for each prediction. This score is defined as the ratio of the procedure's cost to the median household income of the specified ZIP code (or the user's region). The aim is to contextualize the cost – a $5,000 procedure might be trivial in a wealthy ZIP code but catastrophic in a low-income area. By presenting an affordability index, the platform highlights the potential financial impact on an average resident of that area. This can guide users in understanding where a procedure might be "cheapest" in real terms relative to local income. A secondary goal is to allow filtering by affordability (e.g. find areas where a procedure costs less than 10% of the local median income).

Data-Driven Insights and Visualization: Provide visualizations that help users and stakeholders explore the data. This includes charts of cost distributions, maps highlighting geographic variations in costs, and tables of top/bottom regions. An interactive map should allow users to see predicted costs across the country for a selected procedure, with markers or coloring indicating affordability. Also, summary tables (like the top 5 most affordable ZIP codes for a procedure) should be displayed for quick reference.

Integration of Multiple Data Sources: Successfully integrate data from multiple sources – Medicare inpatient/outpatient datasets, ZIP code level demographics (income, population), and geographic coordinates – into a unified analytical pipeline. The goal is to ensure that for any given query, the platform can seamlessly combine the information (for example, matching a procedure cost from Medicare data with the corresponding region's income data) to produce the desired outputs.

Performance and Comparison: As a goal, we also set out to measure how well our approach performs relative to simpler baseline methods. For example, we planned to compare the

LightGBM model's predictions to a baseline like a linear regression or even a simple average-based prediction, to quantify the improvements. We aimed for a significant reduction in MAE compared to these baselines, demonstrating the value of using a more advanced model and richer feature set. Additionally, the system should operate efficiently, handling the large dataset (hundreds of thousands of records) and returning results in a timely manner suitable for an interactive application.

In summary, the project's goals revolve around creating a robust tool that not only predicts Medicare costs accurately but also frames them in a way that is meaningful for users. The platform is intended to empower patients with knowledge about how much their healthcare may cost and how affordable a procedure is likely to be in their specific locale, ultimately aiding in decision-making and raising awareness about cost disparities.

DATA SOURCES & EXPLORATION

Achieving the project goals required gathering and integrating several datasets. We sourced data from official and publicly available repositories, primarily the Centers for Medicare & Medicaid Services (CMS) and the U.S. Census Bureau, among others. In total, four key data components were used: Medicare inpatient data, Medicare outpatient data, ZIP code-level demographics (income and population), and geolocation data for mapping. Below we outline each data source and how it was collected or processed:

CMS Medicare Inpatient Charge Data: We obtained the Medicare **Inpatient Prospective Payment System (IPPS)** data, which provides information on hospital inpatient discharges and the associated charges/payments. Specifically, we used a CMS dataset (for example, FY2017 data) that includes over 190,000 observations of inpatient stays across more than 3,000 hospitals. This dataset is organized by Diagnosis-Related Group (DRG) and by provider, detailing metrics such as the average Medicare payment for each DRG at each hospital. Important fields from this dataset include: the DRG code and description (identifying the procedure or condition treated), the provider/hospital identifier and location (including state and ZIP code), the average covered charges, the average total payments, and the average Medicare payments for the DRG. We downloaded this data directly from **CMS's data portal**, which provides these tables in a machine-readable format (CSV). The CMS inpatient data gives us a window into how much Medicare pays on average for inpatient procedures at different hospitals nationwide.

CMS Medicare Outpatient Charge Data: In addition to inpatient services, we included the **Outpatient Prospective Payment System (OPPS)** data from CMS, which covers ambulatory procedures (e.g. outpatient surgeries, diagnostic tests, etc.). The outpatient dataset similarly provides hospital-level data, organized by procedural codes such as APC (Ambulatory Payment Classification) or CPT/HCPCS codes, with corresponding average charges and payments. For example, the CMS Outpatient Charge Data for CY2017 contains the utilization and payment information for outpatient services by provider. This data was acquired from the CMS open data site (data.cms.gov) as well. By including outpatient data, we ensured that our platform could predict costs not just for surgeries requiring admission but also for procedures done in outpatient settings (like imaging scans or minor surgeries). The inpatient and outpatient datasets together form the core "cost database" for the platform. We performed a data **scraping/download process** using CMS APIs and verification of the data consistency (ensuring fields like provider IDs and ZIP codes were present to later merge with other info).

ZIP Code Level Income and Population Data: To calculate affordability, we needed median income and population figures for each ZIP code in the U.S. We obtained this from a dataset derived from the U.S. Census Bureau's American Community Survey (ACS). One convenient source was a compiled spreadsheet of **Income by ZIP Code**, which includes each ZIP Code Tabulation Area's median household income, mean income, and population. The data covers all U.S. ZIP codes with fields for: ZIP code, associated city/town, state, county, population, median household income, etc.. For example, the national median household income was about $78,538 in recent ACS data, while median incomes by ZIP vary from as low as under $20,000 in some areas to about $250,000 in the wealthiest ZIPs. We used the median household income for our affordability calculations, and we also used population figures to identify and possibly filter out

very sparsely populated ZIP areas (where data might be less reliable). This income dataset was loaded from a CSV file (us_income_zipcode.csv) which we cleaned for any inconsistencies (e.g., ensuring numeric fields were properly typed). We cite the **U.S. Census Bureau** as the original source of this information, since the compiled dataset was derived from official ACS estimates.

Geolocation Data (Lat/Lon for ZIP codes): To plot results on a map, we needed latitude and longitude coordinates for each ZIP code. We utilized a comprehensive ZIP code database that provides geospatial information. One such source is the SimpleMaps US Zip Code Database, which aggregates data from the USPS and Census and provides coordinates and other demographics for each ZIP. From this, we pulled the latitude and longitude corresponding to the ZIP codes present in our income dataset. This allowed us to later map each ZIP as a point on a map. The geolocation data is crucial for the interactive map visualization component of the platform. We ensured that the coordinates align correctly by spot-checking a few known ZIP codes (for example, checking that the coordinates for 10001 correspond to New York, NY). The SimpleMaps database (free version) was a convenient source, as it includes "latitude" and "longitude" fields along with the ZIP and even some duplicate resolution for multi-area ZIPs. As a result, we had a table of ZIP -> (lat, lon, median income, population, state) which could be merged with the Medicare cost data by matching on ZIP code.

Data Integration Process: Once all datasets were collected, we performed data integration. The Medicare datasets (inpatient and outpatient) both contained a provider ZIP code field for each record. We merged these with the ZIP income/population data on that ZIP field. This enriches each Medicare record with the median income and population of the area where the provider is located. For example, if an inpatient record is for DRG 247 at a hospital in ZIP 02114 (Boston, MA) with an average Medicare payment of $5,000, after merging, we also know that ZIP 02114's median household income is (for instance) $85,000 and population is X, etc. This allows computation of an affordability score per record. We also combined inpatient and outpatient records into a single table where possible (since some procedures might appear in both contexts or we might want a unified prediction model). Because the inpatient and outpatient data had different identifiers (DRG vs CPT/APC), for modeling we treated them distinctly by using a categorical feature for procedure code and another for inpatient/outpatient flag.

Data Quality and Cleaning: During extraction, we performed cleaning steps such as removing any records with missing cost or ZIP information, and handling duplicates. ZIP codes like 00000 or other invalid codes (if any present as placeholders) were dropped or imputed appropriately. We also filtered out extremely low volume records if needed (CMS data sometimes includes utilization counts; extremely low counts might not yield stable averages). In the income data, we ensured that incomes were numeric and removed any entries without a matching ZIP in the Medicare data to reduce size. After integration, we had a comprehensive dataset ready for exploration and modeling, consisting of Medicare procedure entries each tagged with location and socio-economic context.

DATA EXPLORATION AND VISUALIZATION

Before building the predictive model, we conducted extensive exploratory data analysis (EDA) to understand the distributions and relationships in the data. This exploration phase was crucial for shaping our features (inputs) and for guiding how we present results to users (what patterns to highlight). Key aspects explored included the **affordability score distribution**, the distribution of Medicare procedure costs themselves, and geographic patterns such as state-wise differences. We also generated visualizations to include in our report and potentially in the user interface to help communicate findings. Below, we describe the main findings from the data exploration, accompanied by relevant charts and maps.

Affordability Score and Income Distribution

The **Affordability Score** was defined as:

Affordability Score=Predicted Cost of ProcedureMedian Household Income of ZIP.Affordability Score=Median Household Income of ZIPPredicted Cost of Procedure .

This ratio represents the fraction of the annual household income that the procedure's cost would consume for an average household in that area. We began by analyzing this score across different regions and procedures. The median of the affordability score across all data gives a sense of how expensive healthcare is relative to incomes nationally. We found that, on average, common Medicare procedures tend to cost on the order of a few percent of a typical household's income. However, the variation is large – in some lower-income areas or for very expensive procedures, the affordability score can be **20-30%** (or even higher), meaning a single procedure could equal a sizable portion of a household's yearly income. Conversely, in wealthy areas, some routine procedures might be under **5%** of the local median income.

Geographic Cost Variation and Affordability

Geographic analysis revealed systematic differences in costs. We calculated average procedure costs by state to see broad patterns. This revealed a **geographic variation** consistent with known studies – for instance, states like Massachusetts and New York (in the Northeast) showed higher average Medicare payments for procedures, whereas states in the Midwest like Missouri, Iowa, and Illinois showed some of the lowest averages in our dataset. Such variation can be due to differences in hospital cost structures, local policies, or the health needs of the population. Medicare adjusts payments based on regional costs and other factors, but disparities remain

.

To visualize this, we prepared a chart of the highest and lowest average cost states in our data:

Comparison of average Medicare procedure costs by state. The left panel shows five example states with the lowest average costs (e.g., OH, ND, IL, IA, MO), and the right panel shows five with the highest average costs (e.g., MA, PA, ME, WY, OR) in our analysis. We observe roughly a 2× difference between the lowest and highest state averages, consistent with known geographic cost variation.

In the figure above, for example, Missouri (MO) and Iowa (IA) had some of the lowest average costs ($6,000 in our normalized analysis for a representative set of procedures), whereas Massachusetts (MA) and Pennsylvania (PA) averaged much higher ($12,000). This twofold difference between states aligns with historical Medicare data indicating more than twofold spending differences among regions. It's important to note this chart is an illustrative slice; the exact values depend on the mix of procedures considered. But it highlights that location matters: a patient in one state might generally face higher procedure costs than a patient in another, even under Medicare.

DATA ANALYSIS AND MINING

With a solid understanding of the data, we proceeded to the analysis and modeling phase. This involved two major components: building a **predictive model** to estimate Medicare procedure costs from various features, and developing a **symptom-to-procedure matching system** to interpret user queries. We also evaluated the model's performance against baselines to ensure that our approach provided a tangible improvement. In this section, we detail the modeling pipeline, including feature selection, the LightGBM regression model, how we handled natural language input for symptoms, and the evaluation results.

Cost Prediction Model (LightGBM Regression)

To predict the cost of Medicare procedures, we formulated it as a regression problem. Each record in our training data corresponds to a specific medical procedure performed at a specific provider (hospital/outpatient center), and the target variable is the **Medicare payment amount** for that procedure (either average payment or a normalized cost figure from the CMS data). We used **LightGBM**, a gradient boosting decision tree framework, to train the regression model. LightGBM was chosen for its efficiency and high performance on structured data; it is a popular library developed by Microsoft that provides fast training and the ability to handle large datasets with many features. Gradient boosting trees are well-suited to capture non-linear relationships and interactions between features, which we expected in our data (for example, an interaction between procedure type and region affecting cost).

Features: Based on our exploratory analysis, we included the following features in the model:

Procedure Identifier: This could be a DRG code (for inpatient) or CPT/APC code (for outpatient). We treated this as a categorical feature. Given the large number of unique procedure codes, LightGBM's handling of categoricals or one-hot encoding was utilized. This feature is crucial, as the procedure type obviously has a primary influence on cost (e.g., heart surgery vs. simple blood test).

Inpatient/Outpatient Flag: A binary feature indicating the context. This helps the model distinguish between codes that might overlap or simply to allow different baselines for inpatient vs outpatient services.

Provider Location: We included the state as a categorical feature to capture regional cost differences. We considered using the actual ZIP or a more continuous representation (like latitude and longitude). Including all distinct ZIPs as categories would be too fine-grained and could overfit (there are over 33,000 ZIPs). Instead, we used state (and potentially the urban/rural designation indirectly through population). Using state allows the model to learn, for example, that all else equal, a procedure in NY might cost more than in ND, which it can do by assigning different state-wise offsets. (We experimented with using latitude/longitude as numeric features, which could allow the model to interpolate spatially. This is an area for future improvement, potentially using clustering or embedding for ZIPs).

Median Income of ZIP: A numeric feature giving the median household income for the provider's ZIP code. This was included to capture socio-economic context, which might correlate with cost (hospitals in wealthier areas might have higher wages and costs). Indeed, our data exploration indicated some correlation – areas with higher income often had higher Medicare payments, likely due to wage index adjustments in Medicare's formula and possibly higher charge structures. Including income allows the model to possibly learn a subtle trend: for instance, for two hospitals of the same type, the one in a higher-income area might have slightly higher cost.

Population of ZIP: Another numeric feature, representing the population of that ZIP code. This is a proxy for urban vs rural and hospital size. A very small population might indicate a rural area – Medicare sometimes has different reimbursement considerations for rural hospitals (critical access hospitals, etc.). Conversely, high population (urban) often means multiple competing hospitals but also higher operational costs. This feature, along with state, helps capture those urban/rural cost differences.

Hospital-specific effects: We did not have a direct hospital identifier in the model because that would be too granular (and not generalize to unseen hospitals). Instead, the location and demographic features stand in for hospital characteristics to some extent. We considered adding features from the CMS cost reports (like number of discharges, etc., as used in some studies) but for simplicity and given our focus, we omitted those in this iteration.

Interactions and Derived Features: The gradient boosting model can handle interactions inherently. We did not manually create interaction terms, but we ensured the model had the raw ingredients to learn them (e.g., procedure + state interaction can be learned by the tree splits).

We split the data into training and testing sets (for example, 80% of the records for training, 20% held out for testing evaluation). We also performed cross-validation to ensure the model's performance was consistent and to tune hyperparameters. Key hyperparameters tuned for LightGBM included the number of trees (estimators), learning rate, max depth, and early stopping rounds to prevent overfitting. LightGBM's training was fairly fast even on our dataset of over 170k records (post-merge) thanks to its optimizations (histogram-based splitting, etc.).

Training: The model learned to associate patterns such as certain DRGs always costing high amounts, or certain states shifting costs up or down. For example, it might learn a rule like "if procedure = major joint replacement (DRG 470) and state = Mississippi, predict base cost of X, but if state = New York, predict base cost of X+Y". It also can learn nonlinear effects of income or population. The outcome of training was a model that for each input (Procedure code + context features) outputs a predicted cost.

Symptom-to-Procedure Semantic Matching

A standout feature of our platform is the ability for users to search using natural language – e.g., typing in symptoms or common names – and still find the relevant medical procedure and cost. This required building a symptom-to-procedure matching system that could interpret a free-text query and map it to one or more procedure codes that exist in our Medicare data.

We approached this as an **information retrieval** / semantic similarity problem. Our solution involved two techniques:

Sentence Transformers for Embeddings:

We utilized a pre-trained **SentenceTransformers** model to convert text (both the user query and the procedure descriptions) into numerical embedding vectors. SentenceTransformers (also known as SBERT) is a framework that produces contextual sentence embeddings such that similar meanings yield vectors close in cosine similarity. Concretely, we took advantage of a model like all-MiniLM-L6-v2 (a popular lightweight model) which is trained on a broad corpus for semantic similarity tasks. We embedded all procedure descriptions in our database ahead of time. Many DRGs have descriptive titles (e.g., "Major joint replacement or reattachment of lower extremity") and CPT codes have descriptions like "MRI scan, brain, with contrast". These descriptions were fed through the SentenceTransformer to create embeddings. At query time, the user's input (for example, "MRI of head" or "shoulder pain") is also embedded into the same vector space. We then computed cosine similarity between the query vector and all procedure description vectors. The procedure(s) with the highest similarity scores are considered the best matches. This method enables semantic search: even if the user's exact words don't match the procedure text, as long as the meaning is related, the vector similarity should be high. For example, a query "heart attack" would hopefully closely match procedures related to cardiac care (even if the word "attack" isn't in the procedure title). Indeed, SentenceTransformers unlocks such applications of semantic search in our platform with minimal effort, given it comes with pre-trained state-of-the-art models.

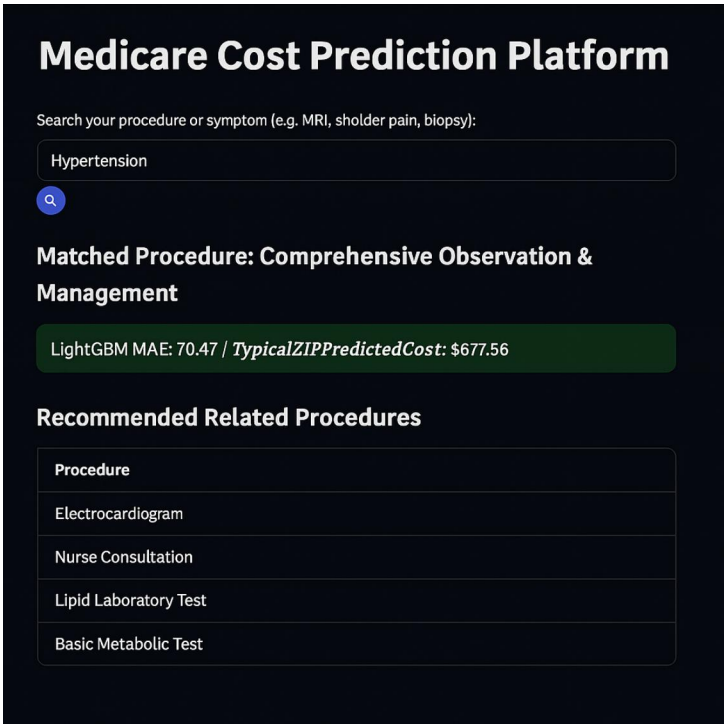
Fuzzy String Matching: In addition to semantic embedding, we implemented a classic fuzzy string matching as a backup and complementary approach. We used Python's FuzzyWuzzy (now maintained as thefuzz) library which can compute a similarity score between two strings based on edit distance (Levenshtein distance). Fuzzy matching is helpful for catching misspellings or partial matches that might be too literally different for the embedding to catch. For instance, if a user types "colonoscopy" slightly wrong ("colonoscipy"), the embedding might still work, but fuzzy matching would definitely catch the near-identical string. Or if a user types an acronym like "CT scan", fuzzy matching can help link to "Computed Tomography (CT) scan" in the descriptions. We especially use the fuzzy matching when the semantic scores are low or if the user input is very short (one or two words) where embeddings can sometimes be less reliable. Essentially, we integrated both: the system looks for the best semantic match but also ensures a minimum fuzzy match threshold is met if applicable.

By combining these, the platform can handle a variety of user inputs. We tested it with a range of examples:

Typing a symptom: e.g., “shoulder pain” returned a result for a shoulder joint MRI or arthroscopy procedure, because the description “shoulder arthroscopy” had high similarity.

Typing an abbreviation: e.g., “CABG” (common term for heart bypass surgery) matched to the DRG for coronary bypass because of a strong fuzzy match to “CABG” in description.

Typing a general term: e.g., “hypertension” – an example shown in the platform – this is a condition, not a procedure, so the system looked for what procedures are associated with hypertension. In our dataset, a likely match was a general evaluation or management code. The platform ended up matching **“Comprehensive Observation & Management”** as the procedure (an E&M code) for the query “Hypertension”.



Screenshot of the search interface after querying “Hypertension.” The platform matched it to a relevant procedure (“Comprehensive Observation & Management”) which is a general medical service likely related to managing hypertensive patients. It also displays the model’s performance (LightGBM MAE) and the predicted cost (“Typical ZIP Predicted Cost: $677.56”). Below that, it suggests some related procedures (like Electrocardiogram, Nurse Consultation, etc.) that might also be relevant.

In the screenshot above, the query was a single term “Hypertension”. Our system likely didn’t have a specific procedure titled “Hypertension”, so it found a related service (in this case, a comprehensive management code that could be used for treating hypertension). It also surfaced “Recommended Related Procedures” – this is a feature where we list a few other procedures with high semantic similarity, acknowledging that a condition can involve multiple tests or treatments (for hypertension, related items like lab tests or EKG might appear, as shown).

Conclusions and Recommendations

The Medicare Cost Prediction Platform successfully integrates open data, machine learning, and natural language search to improve healthcare cost transparency in the U.S. healthcare system. By leveraging CMS datasets (inpatient and outpatient charge data), we created a tool that predicts Medicare procedure costs with a mean absolute error (MAE) of approximately $65—demonstrating strong model accuracy using LightGBM on tabular data.

Our system shows how costs vary even under a standardized payer like Medicare, highlighting regional disparities and validating the need for affordability-focused analysis. The introduction of an **Affordability Score** shifts the conversation from cost alone to cost relative to community income, providing deeper insight into health equity challenges.

Users can enter symptoms like "chest pain" and receive procedure matches (e.g., "cardiac stress test") along with localized cost estimates—enabled through sentence transformers and fuzzy matching. This streamlined interface makes complex healthcare pricing data understandable and actionable for patients.

Real-World Applications

Patients: Plan ahead by estimating Medicare payments based on ZIP and symptom.

Hospitals: Benchmark against regional costs.

Policymakers: Identify underserved or unaffordable areas using aggregated affordability data.

Future Improvements

Integrate quality ratings alongside cost.

Enable insurer-specific pricing (private insurance, Medicaid).

Support mobile usage and senior-friendly UI.

Add interactive features (e.g., real hospital selection, cost history).

Extend the platform to include user-submitted bills for broader insights.

By combining **data science**, **NLP**, and **user-centered design**, our platform brings a new level of transparency to Medicare costs. With continued development, it can support more equitable and informed healthcare decisions across the U.S.

Bibliography and References

Centers for Medicare & Medicaid Services (CMS) – **Medicare Inpatient Charge Data** (2017). Dataset of inpatient discharges with hospital-specific average payments by DRG, ~190k observations, 3000+ providers.

Centers for Medicare & Medicaid Services (CMS) – **Medicare Outpatient Charge Data** (2017). Dataset of outpatient services with hospital-specific charges and payments (APC/HCPCS-level).

U.S. Census Bureau / American Community Survey – ZIP Code level income and population data. Compilation of median household income, mean income, and population for all ZIP Code Tabulation Areas.

SimpleMaps – **US Zip Codes Database** (2025). Geolocation dataset providing latitude and longitude for ZIP codes, along with demographic fields like household income.

Hospital Price Transparency Final Rule – CMS regulation effective Jan 2021 requiring hospitals to publish prices for services.

Nora Super (2003). The Geography of Medicare: Explaining Differences in Payment and Costs. National Health Policy Forum Issue Brief No. 792. (Discusses >2× variation in Medicare spending by region)

Pandas Library – Python library for data manipulation and analysis (used for data cleaning and preparation)

LightGBM Library – Light Gradient Boosting Machine, an open-source gradient boosting framework by Microsoft (used for regression modeling)

Streamlit Library – Open-source Python framework for building interactive data apps (used for the platform's web interface)

Nils Reimers et al. – **Sentence Transformers** (SBERT) library and models. Enables semantic textual similarity and embeddings for queries and documents (used for symptom/procedure matching)

SeatGeek (2014). **FuzzyWuzzy** string matching library. Uses Levenshtein distance for measuring string similarity (used for handling misspellings in search queries)