Comparing Distributed Machine Learning Models for Patient Cost Prediction Using Synthetic Electronic Health Records

37386
London School of Economics and Political Science
London, UK

38682
London School of Economics and Political Science
London, UK

45285
London School of Economics and Political Science
London, UK

58293
London School of Economics and Political Science
London, UK

Abstract

This study investigates the use of distributed machine learning to predict patient-incurred healthcare costs using synthetic Electronic Health Records (EHRs) generated via Synthea. By focusing on features available at the point of care—such as demographics, insurance coverage, procedure codes, and provider information—the project aims to estimate patient costs prior to claim submission. Using Apache Spark on Google Cloud Platform (GCP), we develop a scalable pipeline to preprocess large datasets and train regression models, including Linear Regression, Random Forest, and Gradient Boosted Trees (GBT). We also explore ensemble modeling and Latent Dirichlet Allocation (LDA) topical modeling to improve model accuracy. Results show that while GBT achieves the highest predictive performance (RMSE \$650, R^2 0.96), it incurs significant computational costs compared to simpler models. Ensemble methods offer a balanced trade-off, and topic-specific modeling via LDA highlights the heterogeneity in patient care patterns. This work demonstrates the feasibility and value of combining synthetic EHRs with distributed computing to support real-time, point-of-care cost prediction in a scalable, privacy-preserving manner.

1 Introduction

The digitization of healthcare data through EHRs has led to the accumulation of vast and complex datasets in the United States (U.S.). These records contain detailed information on patient demographics, diagnoses, treatments, prescriptions, billing, and outcomes. As healthcare providers and researchers continue to seek actionable insights from data, traditional computing systems often fall short due to limitations in memory, processing power, and scalability. Distributed computing emerges as a critical solution, enabling the processing of massive datasets in parallel across multiple nodes, thereby accelerating data analysis and making large-scale machine learning (ML) model deployments feasible.

In the context of EHRs, distributed computing allows organizations to train sophisticated cost prediction models that consider a wide range of variables, such as: medical history, comorbidities, treatment patterns, insurance plans, and geographic factors. These models, which might take days to run or even be infeasible on a single machine, can be trained in hours using distributed ML frameworks such as Apache Spark on cloud infrastructure. This computational power not only supports greater model complexity and accuracy, but also accommodates real-time (or near-real-time) prediction capabilities, which is essential for clinical and administrative decision-making.

Accurate cost prediction models can offer patients personalized estimates for the cost of their care, allowing greater transparency and financial stability. In a system as fragmented and confusing as the U.S. healthcare market, such tools can empower patients to make informed decisions about their treatment options. When embedded into provider platforms or patient access portals, predictive models can dynamically adjust estimates based on patient-specific data, insurance coverage, and local cost variations, moving beyond generic cost calculators. For providers and insurers, these predictions can streamline pre-authorization processes, reduce billing disputes, and support value-based care initiatives by aligning incentives around predictable, low-cost outcomes. On a broader scale, aggregating insights across millions of patient records can reveal systemic inefficiencies or disparities in pricing, aiding policy development and reimbursement strategies. Distributed computing thus enables a feedback loop where data drive operational improvements and cost transparency across the healthcare sector.

Given the rising costs and inequities in the U.S. healthcare system, the ability to forecast medical expenses with precision is not

just a technical achievement, but also a public good. Distributed machine learning applied to EHRs represents a scalable, data-driven pathway to making healthcare more predictable, patient-centric, and economically sustainable. As data volumes continue to grow and pressure mounts for transparency and cost control, distributed computing will likely remain at the core of innovation in healthcare analytics.

This paper aims to model and compare various scalable ML pipelines that use encounter-level information to estimate patient costs. By accounting for encounter-level characteristics such as patient demographics, place of service, encounter and procedure codes, insurance coverage and more, this paper attempts to create cost estimates for patients using information that can be collected at the point of care. Point of care metrics are uniquely selected to provide predictions based on information that is available prior to any sort of claim generation. Producing estimates prior to insurance claim generation will allow us to verify if there are significant trends in how the selected hospitals/health systems price their services. Poor model predictions could imply that costs are not clearly set by hospital systems based on encounter-level attributes. Therefore motivating the importance of greater transparency in the U.S. healthcare sector.

Our study's results reveal critical scalability-accuracy tradeoffs when using a distributed ML pipeline. While GBTs achieve superior accuracy, their execution time grows 15–20× as patient records scale from 100 to 1000 patient samples. In contrast, Linear Regression maintains consistently low latency (1–11 seconds) but suffers a 30% accuracy drop when scaling to 1000 patients. These results stress the importance of context-aware model selection, which would require different models to be fit based on regional/health system differences. This is a commonly stressed point in the U.S. health industry, where pricing models and price transparency are often obtained at the health system level.

2 Related Work

This project builds on a broad body of research in healthcare analytics, particularly in the areas of synthetic data generation, machine learning applied to EHRs, and analysis of cost variation in U.S. health services. Each of these domains offer essential insights that inform our use of synthetic health records and predictive modeling techniques.

One of the foundational components of our project is the use of Synthea, an open-source tool designed to generate realistic, synthetic EHRs without relying on any real patient data. Initially proposed by Mark Kramer and supported by the MITRE Innovation Program in 2018, Synthea simulates the entire life of synthetic patients based on clinical guidelines, disease progression models, and public health statistics. Using the PADARSER (Publicly Available Data Approach to the Realistic Synthetic EHR) framework [17], Synthea ensures privacy while producing data with sufficient realism to support research and development in healthcare analytics without the risk of re-identification occasionally observed in anonymized real patient health records [6][8].

Underscoring the growing necessity of synthetic data within healthcare analytics, a comprehensive review by Pezoulas et al. evaluates synthetic data generation methods across multiple data types (tabular, imaging, omics, etc.) and techniques, including statistical, machine learning, and deep learning methods. Deep generative models such as generative adversarial networks (GANs) dominate the field, offering the best balance between data fidelity and privacy [13]. Data fidelity refers to a synthetic dataset's ability to resemble real health care data—as quantified via metrics such as goodness of fit, correlation, and KL divergence. Synthetic data generation combats the prevalent concerns of data scarcity and privacy considerations present when working with EHRs.

The deployment and processing of large-scale EHR datasets benefit significantly from cloud computing infrastructures. In addition to reducing the cost of IT overhead required by maintaining local applications [9], cloud services (e.g., IaaS, PaaS, SaaS) enable scalable data storage, parallel computation, and distributed model training, all of which are key components in managing the high volume and complexity of healthcare data [2]. Despite these benefits, the use of cloud computing for the storage of patient EHRs introduces data privacy and data security risks [16], concerns that are alleviated by utilizing synthetic EHRs such as those produced by Synthea.

Deep Patient is an unsupervised deep learning framework that uses denoising autoencoders to discern general-purpose patient representations from large EHR datasets. Deep Patient significantly outperforms traditional methods in predicting future disease onset, emphasizing the value of automatic, data-driven feature learning in healthcare predictive analytics. While Deep Patient was trained on real medical data, the processing and methodology informs our approach to modeling synthetic EHRs for cost prediction [10].

A series of studies by Cooper et al. provide critical context for our cost prediction goals. Within the U.S. healthcare system, variation in spending for the privately insured is impacted jointly by differences in hospital prices and by quantity of care. Specifically, market structure is strongly associated with hospital pricing. When two hospitals merge the respective prices increase by 6% and monopoly hospitals contain 12% higher procedure prices relative to hospitals in quadropoly or greater markets [5]. From 2007 to 2014 hospital prices rose far more rapidly than physician prices (42% versus 18% for inpatient and 25% versus 6% for hospital-based outpatient respectively), reinforcing the need to isolate institutional and procedural factors in predictive cost models [3]. Additionally, there is substantial variation in the private spending growth across hospital referral regions (HRRs) [4]. Cooper et al. comprehensively indicate that healthcare prediction models must account for a myriad of nuanced factors including demographic information, insurance status, local provider behavior, hospital market power, and geographic disparities to accurately determine cost.

3 Methodology

3.1 PySpark Architecture

This section outlines a methodology for building predictive models to estimate healthcare costs using large-scale datasets. This approach leverages PySpark for distributed processing, ensuring scalability and performance in data handling and model training. The process begins with the preparation and loading of diverse healthcare-related data, followed by rigorous preprocessing, feature engineering, and the training of multiple regression models.

An ensemble strategy is ultimately employed to enhance predictive performance, offering a comprehensive view into patient cost prediction.

The first step in our methodology involves loading several structured datasets from CSV files located in Hadoop Distributed File System (HDFS). These datasets contain patient demographics, clinical observations, medical records (such as encounters, conditions, procedures, and medications), care plans, and insurance information. Creating a dataframe that links patient information together from the various CSV files using PySpark allows for efficient manipulation and downstream processing. The datasets are merged into a final encounter-level format to create a unified dataset that supports the modeling objective: predicting out-of-pocket patient costs.

To ensure data consistency for linking the Spark dataframes, column names are standardized (e.g., *Id* becomes *patient_id*, and *MARITAL* becomes *patient_marital*). Temporal fields such as *encounter_start* and *encounter_stop* are converted to date formats to facilitate calculations. Derived fields, including *age_at_encounter* (calculated from *BIRTHDATE* and *encounter_start*), are introduced to enrich the dataset. ZIP codes are formatted with leading zeros to retain correct formatting, and related datasets are joined to construct a feature set that includes organizational and payer-level details.

To prepare the data for machine learning, categorical variables (e.g., *patient_marital*, *patient_race*, and *payer_ownership*) are indexed and one-hot encoded, converting them into a numerical form. Numeric variables such as *age_at_encounter* are kept in their raw format. The target variable for prediction is defined as *PATIENT COST*, which is derived from the difference between *TOTAL_CLAIM_COST* and *PAYER_COVERAGE*. All preprocessing steps are encapsulated in a PySpark Pipeline[1] object to maintain consistency and streamline the data flow.

Following preprocessing, the dataset is divided into training (80%) and testing (20%) subsets using a fixed random seed to ensure reproducibility. To optimize distributed computation across the Spark cluster, the training data are repartitioned[2] into 12 segments. This approach enhances parallel processing and improves model training efficiency by ensuring that data are evenly distributed.

Three regression models are then trained on the processed data: Random Forest, and GBT. The Random Forest model utilizes set parameters of 30 trees and a maximum depth of 10. Linear Regression utilizes L2 regularization and elastic net mixing. Finally, the GBT model utilizes maximum iterations of 20, maximum depth of 10, learning rate of 0.1, and a subsampling rate of 0.7[3].

To combine the strengths of individual models, an ensemble approach is implemented. Each model is trained concurrently using Python's ThreadPoolExecutor[4], reducing the total computation time. The final predictions are generated by averaging the outputs of the three models. Theoretically, an ensemble should yield a balanced performance, effectively mitigating the weaknesses of the individual regressors while preserving their predictive insights.

This end-to-end methodology underscores several key techniques essential for predictive modeling in the healthcare domain. Distributed processing via PySpark ensures scalability, while pipeline-based preprocessing guarantees consistent transformation across datasets. Parallel model execution expedites training, and ensemble modeling delivers robust and interpretable predictions. Through the integration of diverse data sources and advanced modeling strategies, this approach demonstrates a tested framework for understanding and forecasting patient healthcare costs.

3.2 Data Acquisition and Preparation

Synthea is an open-source synthetic patient generator developed by MITRE that creates realistic but artificial EHRs for testing and research. It simulates lifelike patient histories—including demographics, medical conditions, treatments, and insurance claims—without using real patient data.

Generating realistic synthetic patient data is challenging because it must accurately mimic complex medical patterns (disease progression, medication interactions, etc.) while maintaining statistical validity. The data must align with real-world clinical workflows, geographic variations in care, and temporal consistency (aging, chronic conditions, etc.). Poorly generated data can lead to flawed research or unreliable system testing, making tools like Synthea critical for balancing accuracy with scalability.

The synthetic patient data used for this project is generated in batches using Synthea. The system first installs required dependencies (Java and the Synthea executable[5]), then processes configurable batches of patient records (20 patients per batch) with specific demographic parameters (age range, location) and unique seed values for reproducibility. As each batch completes, Synthea outputs structured CSV files containing information such as patient records, encounters details, clinical events, etc., in a temporary local directory. Once all batches finish, the system consolidates these files by type (e.g., appending all the batched patient files into a single patients.csv) and stores the unified datasets.

For persistent storage and analysis, the final appended CSV files are then transferred to HDFS. The code moves only the consolidated CSV files from the local combined directory to HDFS. Finally, the script cleans up by removing all temporary local files. This end-to-end process ensures efficient generation of synthetic medical data, secure distributed storage in HDFS, and automatic cleanup

[1] A PySpark Pipeline is a high-level API in Apache Spark's MLlib that allows users to chain multiple data processing and machine learning steps into a single, streamlined workflow. It consists of a sequence of stages, where each stage is either a transformer (e.g., one-hot encoding, normalization) or an estimator (e.g., a machine learning model). By encapsulating preprocessing and modeling logic within a Pipeline, PySpark ensures consistent and repeatable operations across both training and testing datasets, simplifying model deployment and maintenance.

[2] Repartitioning the training DataFrame in PySpark redistributes the data across a specified number of partitions (12) to optimize parallel processing. This is important because it balances the data load among worker nodes in a Spark cluster, reducing processing bottlenecks and improving performance during operations like model training and data shuffling. Without appropriate repartitioning, some nodes might be overburdened while others remain underutilized, leading to inefficient computation.

[3] The fraction of the training data used to fit each individual tree. This introduces randomness, which can help reduce overfitting and improve generalization.

[4] ThreadPoolExecutor from Python's concurrent.futures module is used to run multiple model training processes concurrently within the same Python application. Although each model is trained using Spark (which distributes computation across a cluster), ThreadPoolExecutor helps manage and launch these training jobs in parallel at the Python level, reducing total runtime without waiting for one model to finish before starting the next.

[5] The synthea-with-dependencies.jar is a self-contained executable Java file that enables easy generation of synthetic patient data without requiring manual dependency management, making it essential for our code/report as it ensures consistent, reproducible, and realistic fake healthcare datasets for testing and analysis.

of temporary files, maintaining both system resources and data integrity throughout the workflow.

Description of records created for 1,000 patients[6] (it took 32 minutes to generate patient records for the 1,000 patients):

- `observations.csv`: 904,669 records
- `medications.csv`: 61,608 records
- `devices.csv`: 6,484 records
- `providers.csv`: 2,656 records
- `imaging_studies.csv`: 205,164 records
- `procedures.csv`: 174,571 records
- `organizations.csv`: 2,656 records
- `supplies.csv`: 31,254 records
- `patients.csv`: 859 records
- `claims_transactions.csv`: 1,027,196 records
- `encounters.csv`: 61,176 records
- `payers.csv`: 300 records
- `conditions.csv`: 39,885 records
- `allergies.csv`: 690 records
- `careplans.csv`: 3,376 records
- `immunizations.csv`: 10,357 records
- `payer_transitions.csv`: 8,423 records
- `claims.csv`: 122,784 records

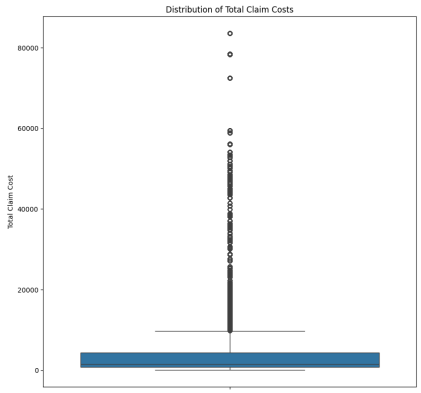
We construct a comprehensive final DataFrame named *encounters* by joining data from several Synthea-generated CSV datasets, each representing a different aspect of synthetic EHRs. Starting with the base encounters DataFrame, we enrich each record by performing a series of left joins. First adding payer-related information, such as payer name and ownership type, from the payers DataFrame using the *payer_id* key. Then, incorporating details about the healthcare organization, such as name and ZIP code, using *organization_id*, followed by provider data, including the provider's specialty, through *provider_id*. Next, procedural details related to each encounter are included using the *encounter_id*, pulling in descriptions and procedure codes from the procedures table. The patient's demographic information, including birth date, race, gender, and ZIP code, is then joined using the *patient_id*. Finally, we calculate the patient's age at the time of the encounter by computing the difference between the encounter start date and the patient's birth date. The resulting DataFrame is a fully integrated, flat table suitable for downstream machine learning, cost prediction, or analytic tasks that require a complete view of each patient encounter.

3.3 Exploratory Data Analysis

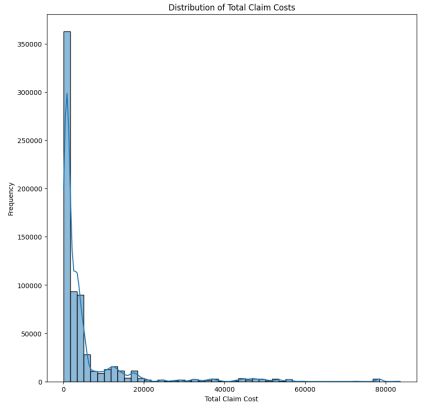
We conduct exploratory data analysis (EDA) to visualize the distribution of the dataset, with a focus on cost patterns. The analysis was performed on a generated sample of 100 patients, selected as representative of the overall population. It first examined cost variations across the entire dataset, then segmented the analysis by encounter class and hospital ZIP code.

[6]Patient records were created in batches to reduce the chances of missing data due to a PySpark kernel crashing. As a result, patient records rarely add up exactly to 1000 due to batch failures during the data generation process. Additionally, Synthea patient generation only aggregates living patients. Therefore, deceased patient records might be included as additional patient records, on top of the 1000/100 patients that we pre-defined.

The plots in Figure 1 confirm the suspected skewness in cost values across the dataset. As shown in the boxplot (Figure 1a), the wide range in total claim costs and the large number of outliers introduce substantial variation. These outliers likely result from significant variation in patient care, including differences in procedure type, encounter class, length of hospital stay, and other clinical or administrative factors. The low median relative to the outliers aligns with the cost distribution in Figure 1b, which shows a high concentration of patients with total claim costs below $2,000. While this indicates clustering around the median, the presence of high-cost outliers suggests significant variability that may impact subsequent analysis. The plot in Figure 2a investigates the impact of encounter class on the percentage of average payer coverage to total claim cost. Finally, the plots in Figure 2b explore the variation in total claim costs across hospital ZIP codes for two encounter classes. A clear disparity between ZIP codes is evident, further confirming the geographic variability in healthcare costs and emphasizing the influence of regional factors on patient expenses.

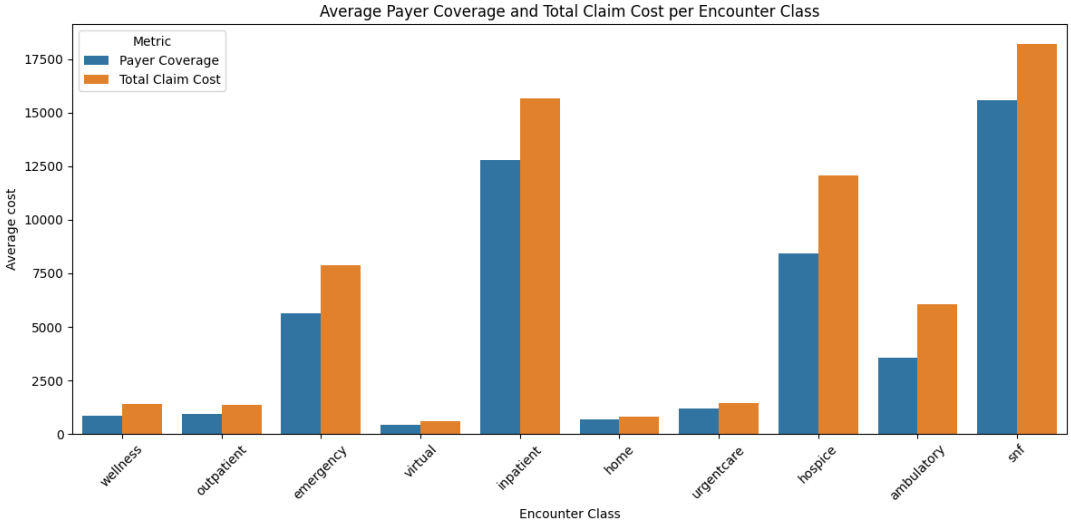


(a) Distribution of Total Claim Cost

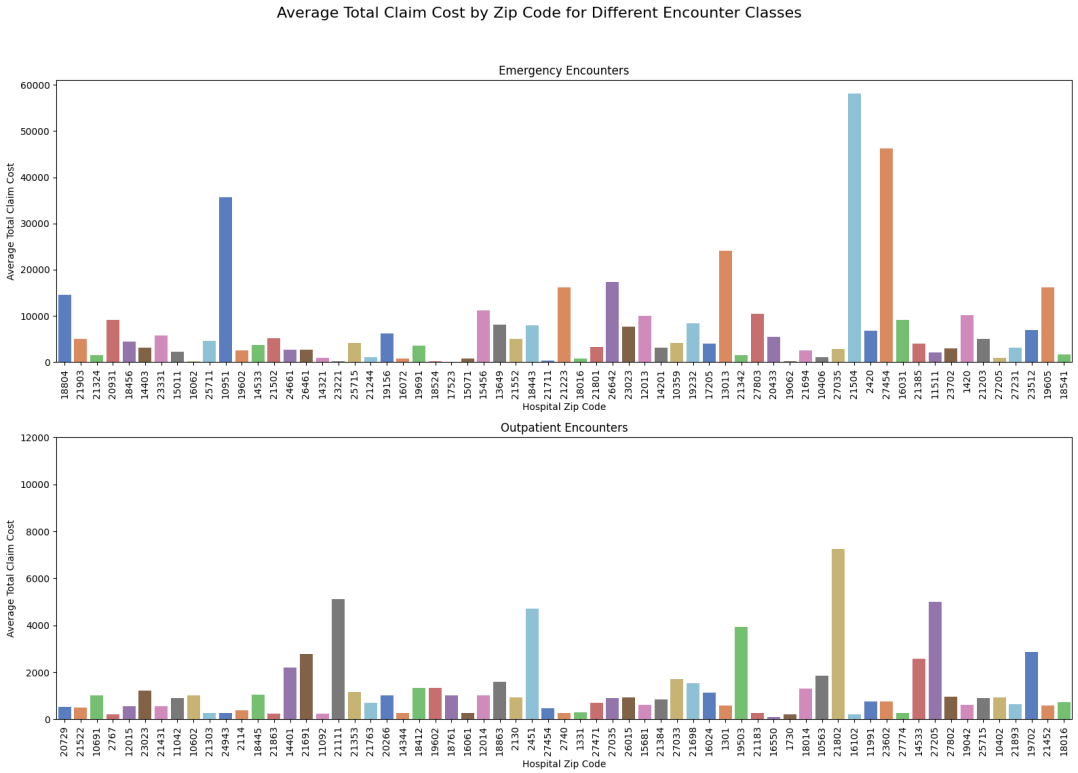


(b) Frequency of Total Claim Cost

Figure 1: Visualizations of cost distribution



(a) Average Cost vs Coverage by Payer Ownership



(b) Variation in Cost Across ZIP Codes

Figure 2: Exploratory data analysis visualizations of cost distribution, payer ownership, and regional variation.

3.4 Model Training and Evaluation

To estimate patient costs based on EHRs, we implement a range of predictive modeling approaches. We begin by applying two widely used models, Linear Regression and Random Forest, to establish baseline cost estimates and assess their individual predictive performance. Linear Regression provides a simple and interpretable benchmark, relying on the assumption of a linear relationship between covariates and the outcome [15]. In contrast, Random Forest, a non-parametric ensemble method, captures complex non-linear interactions by aggregating predictions from multiple decision trees [1].

Furthermore, we implement a GBT model, a method that leverages boosting to handle complex, non-linear datasets by combining the simplicity of decision trees with the robustness of ensemble learning [14]. Although both Random Forest and GBT are ensemble models that use decision trees as base learners, they differ significantly in how these trees are constructed and combined. Random Forest builds trees independently, using a randomly selected subset of both training data and features, and aggregates predictions across all trees. In contrast, GBT builds trees sequentially, where each new tree is trained to correct the errors made by the previous, allowing the model to iteratively minimize prediction error [7].

After training the three models independently, we assess whether using an ensemble approach produces more accurate estimates of patient-incurred costs while also potentially reducing computational time. Ensemble methods combine the predictions of multiple base learners to improve generalisability and robustness compared to relying on a single estimator [12]. Previous research also shows that ensembles frequently outperform any individual model within them [11].

For this analysis, we combine Linear Regression, Random Forest, and GBT into a single ensemble model. Although these models have already been trained individually, we retrain them in parallel for the ensemble, using the same input features. This approach allows us to maintain methodological consistency while improving computational efficiency.

Given the heterogeneity among patients, we implement an approach that segments the population into more homogeneous subgroups and estimates a separate model for each. This strategy aims to evaluate whether subgroup-specific models improve cost predictions via capturing within-group cost structures and interactions more effectively. To segment the population, we implement LDA, a probabilistic topic modeling technique. Upon application to unstructured text in clinical encounter descriptions, LDA assigns patients to topics based on the content of their encounter narratives, thereby grouping individuals with similar clinical profiles. Within each topic group, we trained linear regression models to estimate patient costs.

We compare model training times to quantify the improvements offered by distributed computing and whether the ensemble strategy offered computational advantages alongside performance gains. We estimate RMSE and R^2 for all models to assess performance, interpret results, and facilitate comparisons of our various models.

Table 1: Patient Data Partitioning Results

Total Encounters	Rows	Partitions of Data
100 Patients (on CPU)	27,607	1
100 Patients	11,596	12
1000 Patients	2,362,312	12

4 Numerical Results

4.1 Interpreting Results for 100 Patients in GCP

To predict patient costs, we begin by training three individual models (Linear Regression, Random Forest, and GBT) using a dataset of 11,596 observations. Although all three models are eventually trained in parallel for the ensemble phase, we first evaluate them independently to understand their standalone predictive performance. This allow us to assess the strengths and limitations of each model before analyzing whether combining them in an ensemble model yields more accurate predictions.

Upon fitting the Linear Regression model, we obtain an R^2 value of 0.79, indicating that approximately 79% of the variability in the outcome variable, the costs incurred by patients, can be explained by the included predictors. This suggests that a linear combination of patient-level, institutional, and encounter-level characteristics provides a reasonably good fit to the data.

However, despite the decent R^2, the RMSE is relatively high at approximately $1,376, suggesting that the model's predictions often deviate considerably from the true cost values. This implies that the Linear Regression model may fail to capture more subtle or complex patterns in the data, leading to inaccurate predictions for a notable number of patients

Additionally, when examining the predicted values, we observe some negative cost predictions, which are not plausible in this context since costs cannot be negative. This is a well-known limitation of Linear Regression when applied to bounded or skewed outcomes, as is typical in cost data. Healthcare expenditures often exhibit a right-skewed distribution, with many low-cost encounters and a few very high-cost outliers, due to the wide variation in treatments complexity and intensity. This heterogeneity in the outcome violates the assumptions of homoscedasticity and normality of residuals, which are central to Linear Regression, thereby reducing the reliability of its predictions.

In order to capture non-linear and more complex relationships between variables, a Random Forest model is also trained using 30 trees with a maximum depth of 10. The model achieves an R^2 of 0.78, which is slightly lower than that obtained with Linear Regression. This is interesting because Random Forest should be better able to model non-linear interactions between predictors, patterns that Linear Regression often cannot capture. Despite this, the R^2 values are comparable between the two models. Finally, the RMSE remained relatively consistent at $1,427.

However, Random Forest only produces positive predictions, addressing the unrealistic negative cost estimates seen in the linear model. We observe that some patients receive identical predicted values, suggesting a flattening effect. This occurs because multiple observations are likely assigned to the same terminal node in the trees. A potential explanation for this is the use of a limited tree

Table 2: Model Performance Evaluation Comparison (RSME, R^2)

Patient Group	RF	LR	GBT	Ensemble
100 Patients (CPU)	$2066, 0.56	$1376, 0.80	$754, 0.94	$1018, 0.89
100 Patients	$1478, 0.78	$1434, 0.79	$650, 0.96	$952, 0.91
1000 Patients	$4626, 0.49	$4384, 0.54	$2585, 0.84	$3043, 0.78

Table 3: Model Execution Time Comparison (in seconds)

Patient Group	RF	LR	GBT	Ensemble
100 patients (CPU)	19.10s	4.25s	166.57s	53.42s
100 patients	9.74s	1.23s	79.01s	165.18s
1000 patients	184.05s	10.82s	1316.55s	341.14s

Table 4: Root Mean Square Error (RMSE) by LDA Topic

Topic	RMSE
Topic 0	$0.00
Topic 1	$1878.22
Topic 2	$899.75
Topic 3	$2284.19
Topic 4	$0.09
Topic 5	$2558.59
Topic 6	$285.89
Topic 7	$719.73
Topic 8	$93.81
Topic 9	$0.02

depth, which restricts how the model can partition the data space. This constraint has been imposed due to computational limitations, since increasing tree depth significantly raises training time and memory usage, especially with a dataset containing a high number of observations and features.

To further improve predictive performance, we have trained a GBT model. In contrast to Random Forest, which aggregates independently grown trees, GBT builds trees sequentially, with each new tree learning from the errors of the previous ones.

The GBT model outperforms both Random Forest and Linear Regression, achieving an RMSE of $650 and an R^2 of 0.96. This substantial improvement suggests that GBT is more effective in capturing the complex non-linear relationships and interactions among features than the our previous models.

However, it is important to note that, while more accurate, GBT also has some limitations. These models are more prone to overfitting compared to Random Forest and Linear Regression and typically require longer training times. This is reflected in our results, with the GBT model taking 79 seconds to train, considerably more than Random Forest (9.74 seconds) and Linear Regression (1.23 seconds).

After analyzing each model independently, we then assess whether combining their predictions yields better estimates and potentially reduce training variability. Therefore, we implement an ensemble model that averages the predictions from Linear Regression, Random Forest, and Gradient Boosted Trees. These models are retrained, but executed in parallel, using the same training data and test data.

The ensemble model achieves an RMSE of $952 and an R^2 of 0.91, which is slightly lower than the GBT model alone, but still shows strong performance. By averaging predictions across the three models, the ensemble approach is able to balance the strengths and mitigate the weaknesses of Linear Regression, Random Forest, and GBT, resulting in more robust and generalizable cost estimates.

To account for potential latent structures among patients, we apply LDA to the textual description of encounter and procedures. After text preprocessing, tokenisation, and stopwords removal, such as "patient" and "hospital", we train a LDA model that groups patient encounters into 11 different topics. The motivation for 11 topics was provided by the 11 different encounter class types (outpatient, inpatient, emergency, hospice etc.). Each of the topics represent underlying clinical themes derived from the encounter and procedures description, allowing us to divide patients regarding the semantic content of their encounters.

Each observation is assigned a topic, and the top five most representative words for each topic have been examined. For instance, Topic 1 includes words such as "dialysis," "renal," and "environment," which could suggest a focus on kidney-related care. In contrast, Topic 2 includes terms like "check," "examination," and "general," indicating it may relate to routine health assessments.

Following topic classification, we fit a separate Linear Regression model for each topic to predict patient costs. The features used are the same as in previous models, but now stratified by topic. By grouping patients according to the semantic content of their encounters and procedures, the aim is to improve predictive accuracy by capturing topic-specific cost dynamics that might be obscured in a pooled model.

The predictive performance of these topic-specific models, outlined in *Table 4*, vary considerably across topic groups. While some topics yield relatively high RMSE values, others, like Topics 0, 4, and 9, had RMSE values approaching zero. Although this might appear to indicate near-perfect prediction, such results are unusual. A possible explanation is that these topics contain very low cost variability, meaning all patients assigned to that topic had similar or even identical cost values. In such cases, the model can predict a constant value for all observations and achieve low errors. Another possible explanation is that these topics had very few observations, increasing the risk of overfitting. Therefore, we cannot attribute these low RMSE values to genuine model accuracy, but rather the outcome of limited data or uniform cost values within those topic groups.

4.2 Model Comparisons by CPU vs. GCP and by Patient Size

The model evaluation results and the model execution times can be examined in *Table 2* and *Table 3*. Given this report's focus on distributed computing, *Table 3* offers the most relevant comparisons for our study. For the 100 patient models, every model except for the ensemble model achieves faster execution times when run in parallel within GCP. While we have a contradictory result (the ensemble model), this makes intuitive sense since GCP and Apache Spark allow models to be fit faster when the training data are partitioned across the worker nodes and the ensemble models are trained in a parallel fashion. The contradictory result seen in our 100 patient ensemble model could be due to the fact that we partitioned a very small amount of data before training. This would make it more time intensive for Spark to run the models in parallel and would result in the Google Colab CPU model executing faster. *Table 2* also highlights how incorporating more patient records skews the accuracy of our models. Despite the models being trained on ≈2 million patient encounter records, the accuracy and RSME metrics are significantly worse than the 100 patient models. The larger RSME values are likely skewed because of a high variation in encounter costs, as displayed in our EDA section above. The low R^2 values are likely due to the added variability in the larger 1,000 patient dataset. Given our simple model parameters (patient gender, patient race, patient zip code, etc.), which offer limited variation, our pre-trained models likely overlook the added variability in the 1,000 patient data and therefore perform poorly on the test data.

5 Conclusion

This project demonstrates the practical and analytical value of distributed machine learning for cost prediction in healthcare using synthetic EHR data. Our scalable pipeline, powered by PySpark and Google Cloud, enables efficient processing and modeling of large encounter-level datasets, simulating real-world hospital operations. Among the models tested, Gradient Boosted Trees achieved the highest accuracy, but with substantial computational overhead. The ensemble approach provided a strong balance between performance and efficiency, validating its utility in production scenarios. Segmenting patient encounters via topic modeling further illustrated the potential of tailored cost models based on clinical context. While technical and data-related limitations restricted scalability beyond 1,000 patients, the methods developed are generalizable and pave the way for further research into interpretable, real-time cost prediction tools. These tools could enhance price transparency, inform patient decision-making, and support value-based healthcare initiatives. Future work will focus on expanding feature sets, improving system robustness, and exploring state-level and multi-provider comparisons to capture broader pricing patterns.

6 Limitations & Future Work

6.1 Computational Limitations

This study encountered several technical challenges related to the computational environment, particularly the use of GCP. The Jupyter notebook kernel crashed repeatedly, especially when handling larger datasets involving over 500 patients. These crashes were often preceded by WARN YarnScheduler messages, suggesting memory limitations or insufficient resource allocation in the Spark cluster. These issues constrained the ability to run large-scale models or extensive hyperparameter tuning. This was particularly observed with the LDA model, which was limited to 200 patients, as attempts to run it on larger samples consistently caused the system to crash.

6.2 Data Limitations

The Synthea dataset contains limitations which affect model performance and accuracy. The dataset is relatively sparse, with some missing or incomplete entries. We exclude some key clinical and administrative information—such as medication prescriptions, insurance claims, and patient condition history—to reduce memory load, which limit the depth and scope of the analysis. Additionally, GCP frequently crashed/had a kernel timeout while running on the 1,000 patients dataframes. Despite previous notebooks loading in 1,000 patients and providing 2,362,312 encounter rows of data for our ML modeling, kernel errors consistently crashed our kernels when trying to reproduce our 1,000 patient records. This marks a significant limitation in terms of data reproducibility.

6.3 Future Works

To improve both the technical robustness and analytical depth of this study, several directions are proposed for future work:

- **Infrastructure Improvements:** A possible improvement is to make the environment more scalable by increasing memory and executor resources within GCP Dataproc. This would enable processing of larger datasets and help prevent kernel crashes.

- **Data augmentation:** Expanding the dataset to include additional features such as medication prescribed, insurance claim records, and detailed patient conditions would provide a more complete view of patient care and allow for more accurate cost prediction models. If the platform improvements allow, increasing the number of patients would also increase the accuracy of cost prediction. Building on the academic literature which suggests that HHRs influence healthcare costs, future work will explore extending the analysis across multiple states and a broader age range.

- **Advanced Modeling:** Future work could explore incorporating a clustering step prior to prediction to improve model accuracy. For instance, clustering patients based on diagnostic codes or clinical characteristics could group similar cases together, allowing for tailored regression models within each cluster. This approach would help reduce the impact of outliers, improve the consistency of predictions, and lower RMSE values.

7 Statement of Individual Contributions

Candidate 37386: Project idea, data acquisition, writing.
Candidate 38682: Literature review, model training, writing.
Candidate 45285: Model selection, model training, writing.
Candidate 58293: EDA, model training, writing.
Each member contributed equally to the project.

8 Statement about the use of generative AI and chat logs

The tool ChatGPT has been used in some parts of the study to support code creation and content implementation. First, we used AI when creating the synthetic data that was generated using the Synthea platform. Since Synthea relies on Java-based dependencies, and no members of the group had prior experience working with Java, we used ChatGPT as a guide to help us understand how to run the necessary commands and successfully generate the final dataset. Lastly, ChatGPT was occasionally used during the writing of the report to refine sentences and improve clarity and readability.

References

[1] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[2] Barbara Calabrese and Mario Cannataro. 2015. Cloud computing in healthcare and biomedicine. *Scalable Computing: Practice and Experience* 16, 1 (2015), 1–18.

[3] Zack Cooper, Stuart Craig, Martin Gaynor, Nir J Harish, Harlan M Krumholz, and John Van Reenen. 2019. Hospital prices grew substantially faster than physician prices for hospital-based care in 2007–14. *Health Affairs* 38, 2 (2019), 184–189.

[4] Zack Cooper, Stuart Craig, Charles Gray, Martin Gaynor, and John Van Reenen. 2019. Variation in health spending growth for the privately insured from 2007 to 2014. *Health Affairs* 38, 2 (2019), 230–236.

[5] Zack Cooper, Stuart V Craig, Martin Gaynor, and John Van Reenen. 2019. The price ain't right? Hospital prices and health spending on the privately insured. *The quarterly journal of economics* 134, 1 (2019), 51–107.

[6] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. 2011. A systematic review of re-identification attacks on health data. *PloS one* 6, 12 (2011), e28071.

[7] GeeksforGeeks. 2024. Gradient Boosting vs Random Forest. https://www.geeksforgeeks.org/gradient-boosting-vs-random-forest/ Accessed: 2025-05-05.

[8] Melissa Gymrek, Amy L McGuire, David Golan, Eran Halperin, and Yaniv Erlich. 2013. Identifying personal genomes by surname inference. *Science* 339, 6117 (2013), 321–324.

[9] Neal Leavitt. 2009. Is cloud computing really ready for prime time. *Growth* 27, 5 (2009), 15–20.

[10] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. 2016. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports* 6, 1 (2016), 26094.

[11] David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* 11 (1999), 169–198.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2024. Ensemble methods - Scikit-learn 1.3.0 documentation. https://scikit-learn.org/stable/modules/ensemble.html. Accessed: 2025-05-05.

[13] Vasileios C Pezoulas, Dimitrios I Zaridis, Eugenia Mylona, Christos Androutsos, Kosmas Apostolidis, Nikolaos S Tachos, and Dimitrios I Fotiadis. 2024. Synthetic data generation methods in healthcare: A review on open-source tools and methods. *Computational and structural biotechnology journal* (2024).

[14] Si Si, Huan Zhang, S Sathiya Keerthi, Dhruv Mahajan, Inderjit S Dhillon, and Cho-Jui Hsieh. 2017. Gradient boosted decision trees for high dimensional sparse output. In *International conference on machine learning*. PMLR, 3182–3190.

[15] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. 2012. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics* 4, 3 (2012), 275–294.

[16] Nabil Sultan. 2014. Making use of cloud computing for healthcare provision: Opportunities and challenges. *International Journal of Information Management* 34, 2 (2014), 177–184.

[17] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. 2018. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association* 25, 3 (2018), 230–238.