Stony Brook University Data Science Bootcamp Capstone Project – Predicting Yearly Medical Costs Using Synthetic Patient Data

Diana Kulawiec

December 3rd, 2024

Table of Contents

1.0 Introduction

We all need to visit the doctor at some point in our lifetimes, whether we like it or not. And those medical bills can add up quickly, especially when factoring in medical procedures, medications, and immunizations. Healthcare expenses are on the minds of all, especially in the United States where the health insurance system can seem very confusing at times.

Having an estimate of yearly medical costs would be very beneficial for both individuals and doctors. Individuals would be able to factor these costs into their yearly budget, and patients and providers would be able to work together to determine lifestyle changes that might be able to reduce medical expenses.

However, access to medical data is very limited due to privacy concerns. This makes it difficult to study how medical encounter costs are impacted by patient background, medical test results, and visits to the doctor's office. Luckily, The MITRE Corporation has developed Synthea, an open-source generator of synthetic patient data. It can create realistic healthcare records including demographic information, health insurance claims and coverage, preexisting conditions, medical observations, and many other elements contained in real medical records.

The goal of this project is to develop a machine learning model to predict yearly medical encounter costs based on medical records from 2023. Synthetic patient data representing 5000 unique individuals from all 50 states was generated. A final dataset for modeling was constructed by combining several types of healthcare records including yearly medical encounter costs, observations/measurements, procedures, medications, and patient demographics. The data was cleaned, and exploratory data analysis was performed to discover underlying trends in the data. Additionally, various data imputation techniques were assessed to address missing values.

Once the dataset was ready, the data was split into training and testing sets and numerous machine learning models were developed, including dummy regression, linear regression, ridge regression, lasso regression, random forest, and gradient boosting. The performance of these models was analyzed by comparing R-squared, mean absolute error (MAE), and mean squared error (MSE) values. The random forest model was selected as the final model, and its performance was assessed on the entire dataset resulting in an R-squared value of 0.9495 and a MAE of $2,379.54.

2.0 Data Generation

First, the data was generated from Synthea. Instructions on the Synthea GitHub page were followed to download the patient data using default parameters, except for a slight modification to generate 100 living patients from each of the 50 states.

This resulted in 18 csv files containing different types of medical record data. The csv files could be linked by unique patient identification numbers, encounter numbers, claim transaction numbers, and various other unique identifiers.

In order to create one complete dataframe for modeling, seven dataframes were selected to be included in analysis. These dataframes were titled patients.csv (patient demographic data), encounters.csv (patient encounter data), medications.csv (patient medication data), procedures.csv (patient procedure data including surgeries), immunizations.csv (patient immunization data), allergies.csv (patient allergy data), and observations.csv (patient observations including vital signs and lab reports). The dataframes that were not included in analysis were titled careplans.csv, claims.csv, claims_transactions.csv, conditions.csv, devices.csv, imaging_studies.csv, organizations.csv, payer_transitions.csv, payers.csv, providers.csv, and supplies.csv.

3.0 Data Wrangling

3.1 Patients

The patients.csv file was loaded into the workspace. This file contains 5,871 rows with each row representing a unique patient. The Synthea software occasionally generates dead patients, and whenever this occurs, it will continue to generate additional patients until the criteria for the number of live patients is met. Since the software was prompted to generate 100 live patients from each of the 50 states, the additional 871 entries in the patients dataframe represent dead patients.

This csv file contains each patient's unique identifier, birth date, death date, social security number, drivers license number, passport number, prefix, first name, middle name, last name, suffix, maiden name, marital status, race, ethnicity, gender, birthplace, address, city, state, county, FIPS code, ZIP code, latitude, longitude, total lifetime healthcare expenses, total lifetime healthcare coverage, and annual income.

The number and percentage of missing values were calculated per column. There were numerous missing values for suffix, death date, maiden name, marital status, passport number, prefix, middle name, drivers license number, and FIPS code. Any rows with a value present in the death date column were dropped to only include living patients.

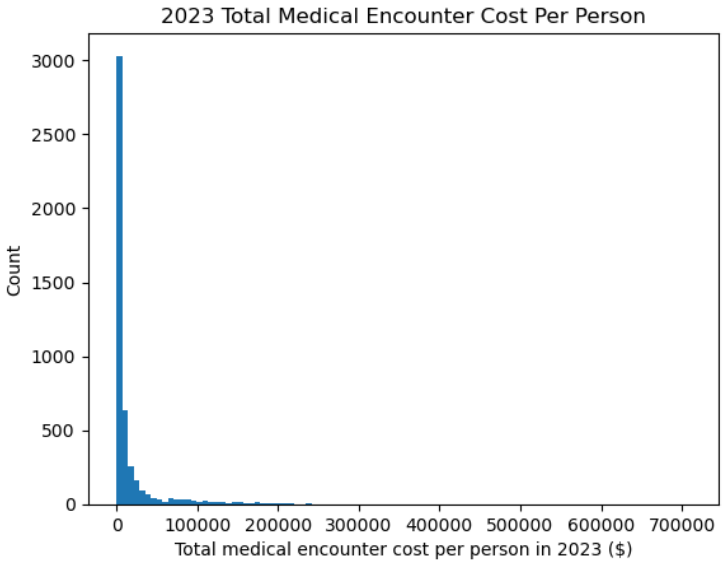
Additionally, the other columns with missing values were dropped since they would not be useful for analysis and would be difficult to impute. Next, each patient's age at the end of 2023 was calculated by subtracting their birth date from December 31st, 2023.

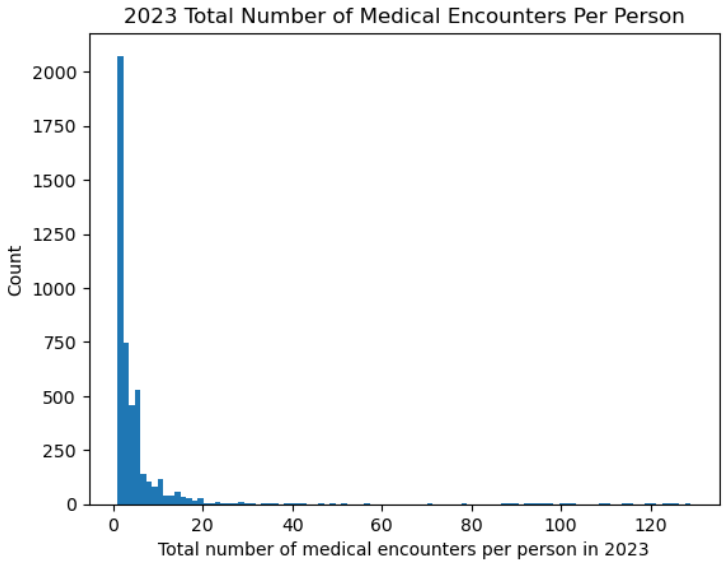
3.2 Encounters

The encounters.csv file was loaded into the workspace. This csv file contains each encounter's unique identifier, the start time of the encounter, the stop time of the encounter, each patient's unique identifier, the organization where the encounter took place, the provider, the payer, the encounter class, the encounter code, the encounter description, the base encounter cost, the total claim cost, the payer coverage, the reason code, and the reason description.

The start time of the encounter was used as a timestamp for the encounter. This column was converted to datetime format and was filtered for encounters just in 2023. Then, the total 2023 encounter cost per patient (encounters_cost) and number of encounters per patient (num_encounters) were calculated from the total claim cost column. These two columns were merged with the patients dataframe based on each patient's unique identifier.

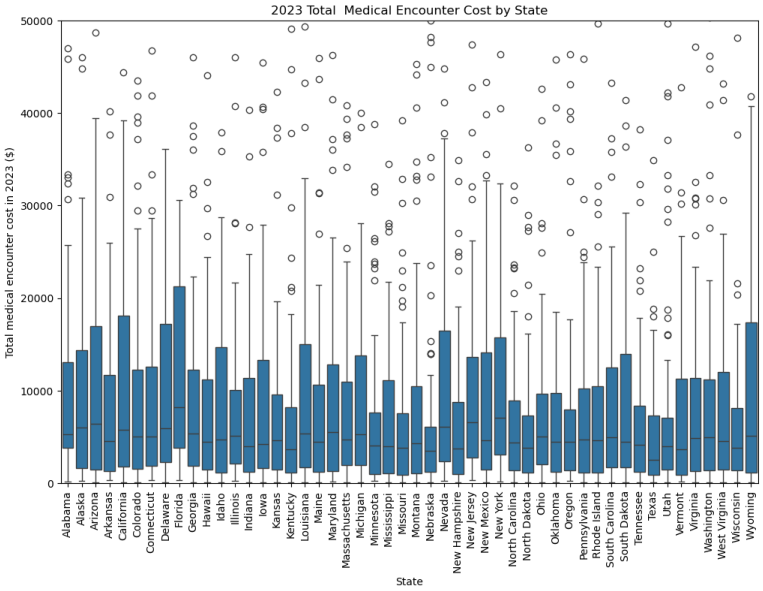
The encounters_cost feature was selected to be the target feature for the predictive model. This feature, along with the num_encounters feature, were plotted as histograms to view their distributions.

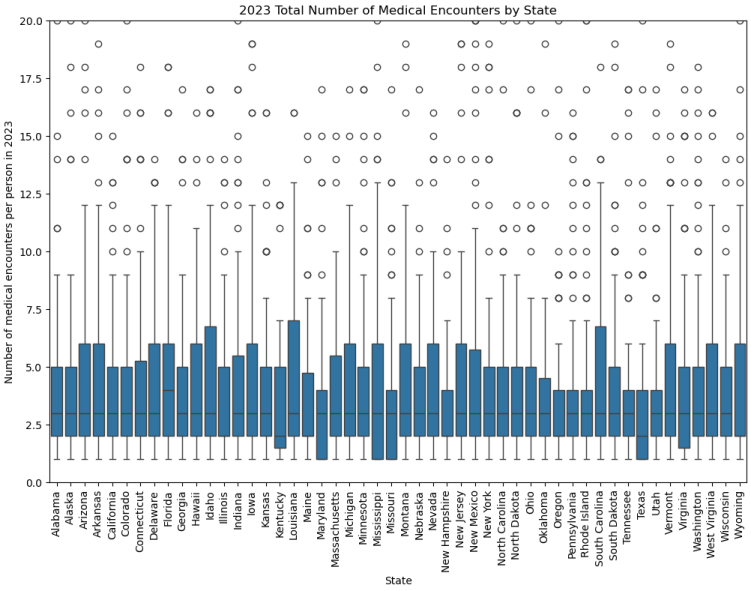




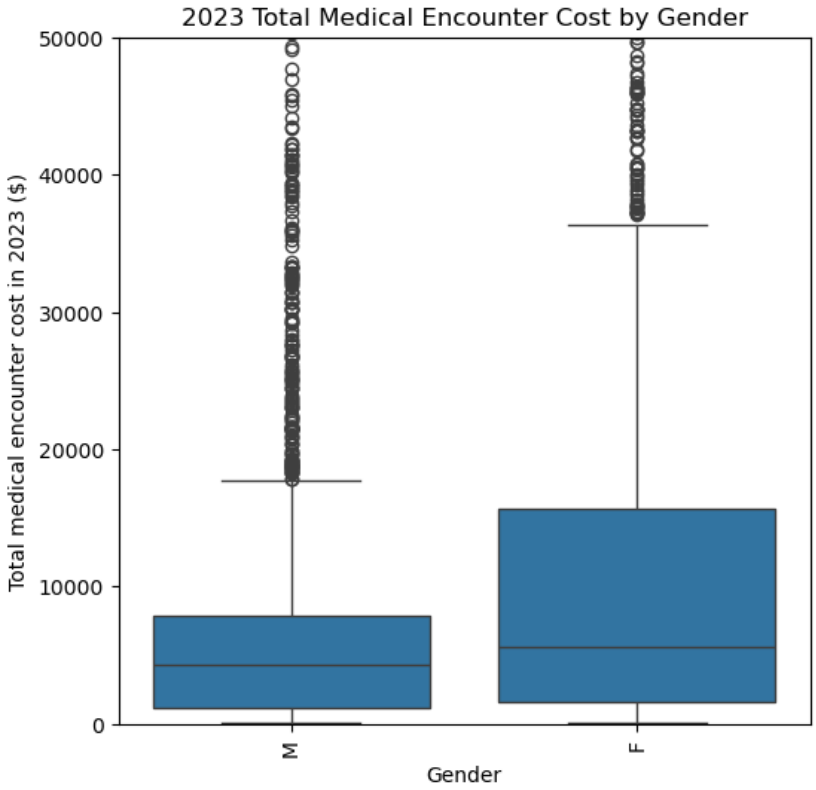
Both distributions are very right skewed, with the majority of the cost values in the $0-50,000 range and the majority of the number of visit values in the 0-20 range. Since this data is very skewed, it would be beneficial to visualize the data based on median rather than mean. Boxplots were created for the encounters_cost and num_encounters features

grouped by state, truncated to the $0-50,000 range for cost and 0-20 range for number of visits.

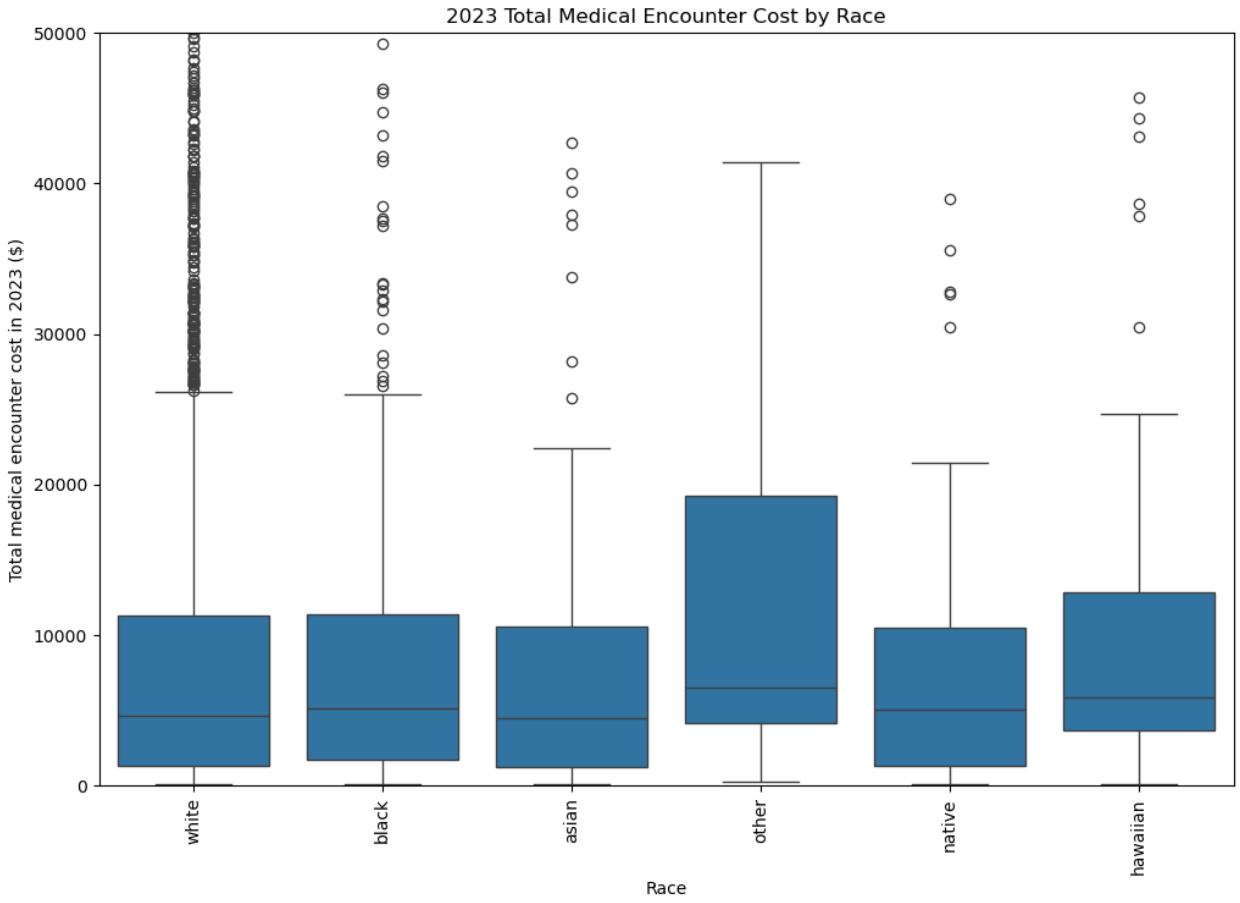




There are many high outliers, which agrees with what was seen in the histograms. The median medical encounter cost and number of medical encounters does not vary much between states. To further understand the spread of the data and identify any differences in medical costs between males and females, boxplots were created for the encounters_cost grouped by gender, truncated to the $0-50,000 range.



As expected, the data is very right skewed with many high outliers. Females have a higher interquartile range and slightly higher median medical encounter cost compared to males. Lastly, boxplots were created for encounters_cost grouped by race, truncated to the $0-50,000 range.



There does not seem to be a large difference in medical encounter costs for each race. “Other” has the greatest interquartile range and the highest median value, which would be interesting to explore, but there is no further information on this category.

3.3 Medications

The medications.csv file was loaded into the workspace. This csv file contains the start time of the medication, the stop time of the medication, each patient’s unique identifier, the payer, the encounter identifier associated with the medication, the medication code, the medication description, the base medication cost, the payer coverage, the number of dispenses, the total medication cost, the reason code, and the reason description.

The start time of the medication was used as a timestamp for the medication. This column was converted to datetime format and was filtered for medications just in 2023.

Then, the total 2023 medication cost per patient (medications_cost) and number of medications per patient (num_medications) were calculated from the total medication cost column. These two columns were merged with the patients dataframe based on each patient's unique identifier.

3.4 Procedures

The procedures.csv file was loaded into the workspace. This csv file contains the start time of the procedure, the stop time of the procedure, each patient's unique identifier, the encounter identifier associated with the procedure, the terminology system for the procedure, the procedure code, the procedure description, the base procedure cost, the reason code, and the reason description.

The start time of the procedure was used as a timestamp for the procedure. This column was converted to datetime format and was filtered for procedures just in 2023. Then, the total 2023 procedure cost per patient (procedures_cost) and number of procedures per patient (num_procedures) were calculated from the base procedure cost column. These two columns were merged with the patients dataframe based on each patient's unique identifier.

3.5 Immunizations

The immunizations.csv file was loaded into the workspace. This csv file contains the date of the immunization, each patient's unique identifier, the encounter identifier associated with the immunization, the immunization code, the immunization description, and the base immunization cost.

The date of the immunization was used as a timestamp for the immunization. This column was converted to datetime format and was filtered for immunizations just in 2023. Then, the total 2023 immunization cost per patient (immunizations_cost) and number of immunizations per patient (num_immunizations) were calculated from the base immunization cost column. These two columns were merged with the patients dataframe based on each patient's unique identifier.

3.6 Allergies

The allergies.csv file was loaded into the workspace. This csv file contains the start time of the allergy, the stop time of the allergy, each patient's unique identifier, the encounter identifier associated with the allergy, the allergy code, the terminology system for the allergy, the allergy description, the allergy type, the allergy category, the allergy reactions, the reaction description, and the reaction severity.

For this file, all recorded allergies for each patient were considered, not just those identified in 2023. The total number of allergies per patient was calculated and this column was merged with the patients dataframe based on each patient's unique identifier.

3.7 Observations

The observations.csv file was loaded into the workspace. This csv file contains the date of the observation, each patient's unique identifier, the encounter identifier associated with the observation, the observation category, the observation code, the observation description, the observation value, the observation units, and the observation type.

The date of the observation was used as a timestamp for the observation. This column was converted to datetime format and was filtered for observations occurring during 2023 and earlier. For this dataframe, the most recent recorded observation for each patient was considered. The dataframe was sorted so that the oldest observations were at the top and the most recent observations were at the bottom. Then, it was checked for any duplicate observations for each patient, and the last (or most recent) observation value was kept. A pivot table was created such that each patient is a row, and each column is an observation, resulting in 305 different observations. Missing values counts and percentages per column were calculated, and columns with more than 70% missing values were removed. Finally, this dataframe was merged with the patients dataframe based on each patient's unique identifier.

3.8 Cleaning the Data

At this point, the dataframe contains 5,000 rows (each representing a patient) and 95 columns. All columns were manually checked over to determine which ones would be useful for analysis. For example, some columns may be repeats of one another, and others may not be useful for statistical analysis and modeling since they cannot be turned into a numerical format. All columns were examined, and a list of the most useful columns was created and used to filter the dataframe, resulting in a dataframe with 60 columns.

Certain columns of type 'string' or 'object' with the potential to be converted into numerical values were transformed. For example, 'Stress level' and 'Tobacco smoking status' were of type 'string' and were converted into ordinal numerical variables.

3.9 Summary

In this section of the capstone project, several csv files containing synthetic medical records from Synthea were loaded into the workspace. These data tables were modified

and merged to create one large dataset for modeling purposes. The total medical encounter cost for each patient in 2023 was selected to be the target feature for modeling.

4.0 Exploratory Data Analysis

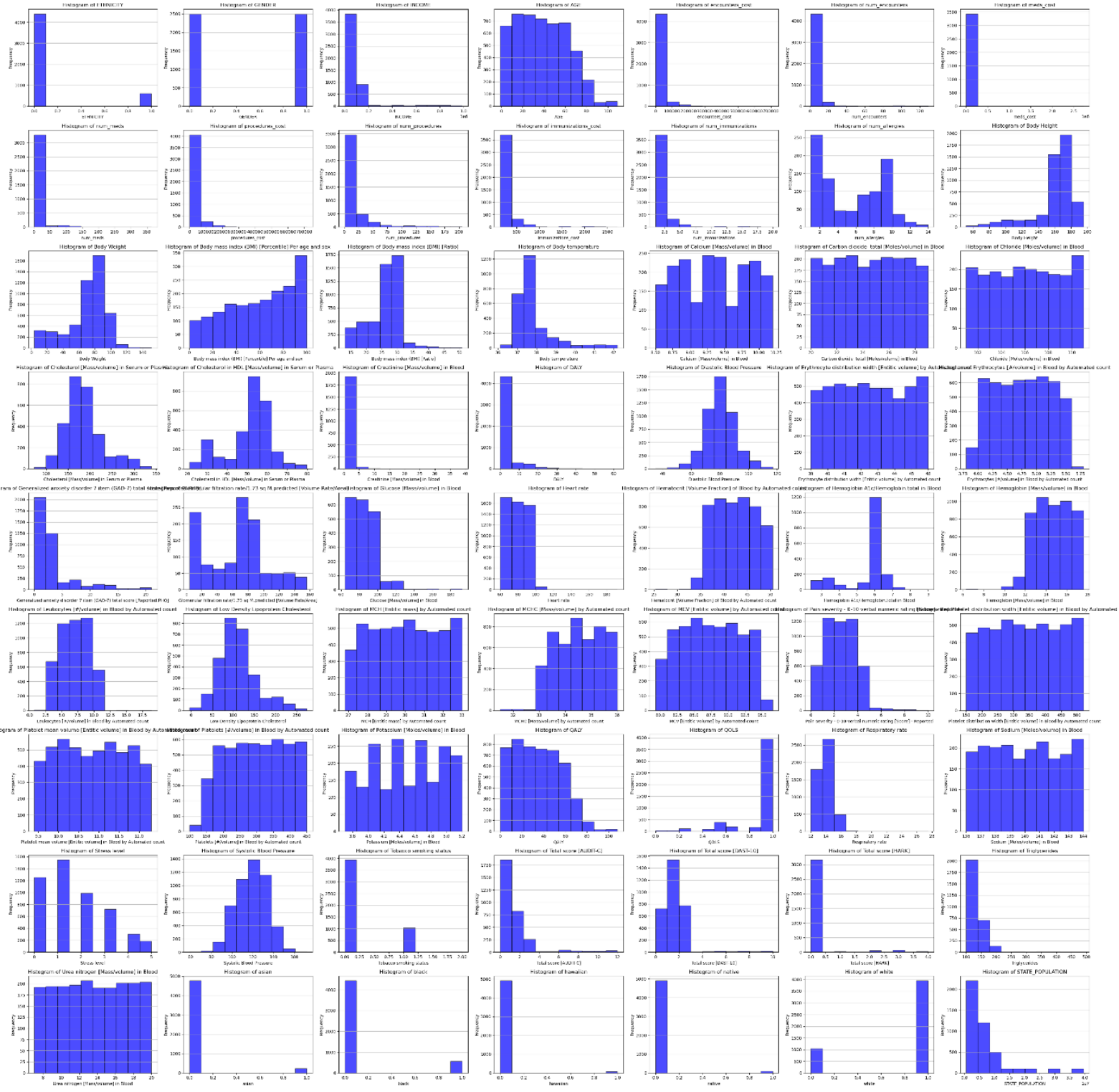
4.1 Converting Data Types

In order to train machine learning models on the data, all data in the dataframe must be of a numerical type. Columns of type 'string' that had a few different values, such as gender, ethnicity, and race, were transformed into numerical columns through one hot encoding.

Next, the states column was transformed into a numeric format. There were multiple ways in which to do this, including one hot encoding, label encoding, and target encoding. Since there are 50 states, one hot encoding would add 49 additional columns, which could cause problems with dimensionality. Label encoding would give each state an integer value from 0 to 49, but since there is no inherent ranking to the states, the behavior of this feature could be disruptive to modeling. To introduce target encoding, the population of each state was used. This gives some information on the availability of healthcare within a state, as a state with a larger population would likely have more cities and thus more healthcare facilities. To do this, state data from Wikipedia was imported and the population column was merged with the patients dataframe. Then, the state and city columns were dropped.

4.2 Data Visualization

To visualize the data, histograms were created for each feature.



There is a lot of variety in the distributions of the features shown above. Some follow very normal distributions, including body weight, BMI, and diastolic and systolic blood pressure. Others follow somewhat uniform distributions, including age, erythrocyte distribution width, erythrocytes in blood, platelet distribution width, platelet mean volume,

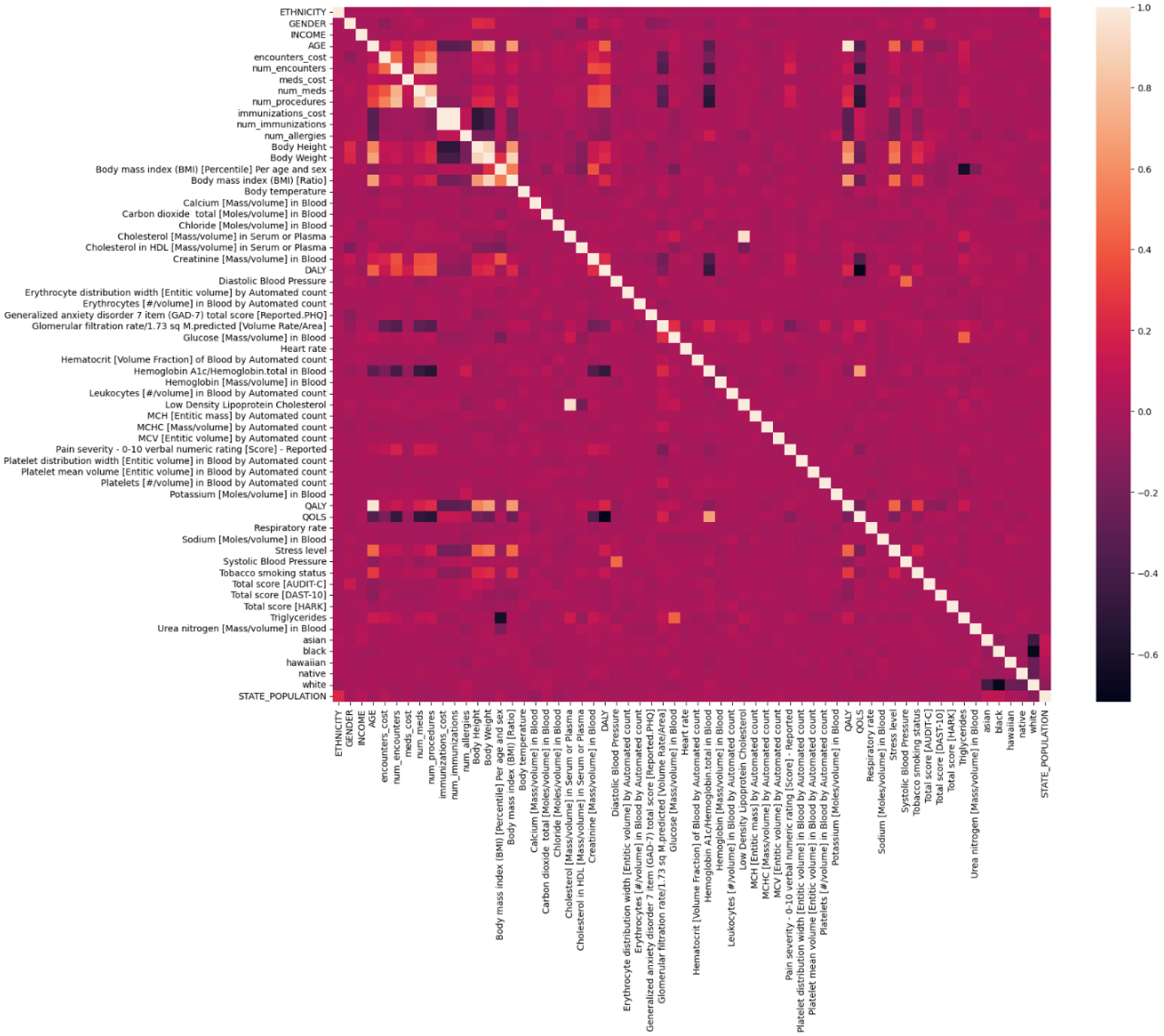
and platelets in blood. Many features have right skewed distributions, such as income, state population, number of encounters, number of procedures, number of immunizations, generalized anxiety score, heart rate, pain severity, respiratory rate, and stress level.

Next, scatterplots are created with each feature on the x axis and the target feature, encounters_cost, on the y axis.

There are not too many clear correlations with encounters_cost. However, there are a couple of key relationships that should be noted. There is a strong positive correlation between procedures_cost and encounters_cost. This makes sense, as procedures_cost refers to the cost of whatever procedure was performed on a patient during an encounter. Since this variable is so strongly related to the target feature, it will be removed from the analysis.

There are also positive correlations with num_encounters and num_procedures, which also makes sense since more medical visits and procedures would lead to a higher total cost of medical encounters. Another visible positive correlation is height, and a few negative correlations include income, immunization cost, and number of immunizations. Other than that, there are not too many clear relationships.

Additionally, a heatmap of correlation coefficients is created to visualize correlations between features.



As identified in the scatterplots, some features that appear to be correlated with encounters_cost include num_procedures, num_encounters, and num_medications.

4.3 Summary

In the exploratory data analysis section of this project, relationships between variables were analyzed to see if there were any potential correlations between encounters_cost and all other features of the dataset. The histograms of the features showed a wide variety of distributions for the variables, which makes sense since there are so many different types of features and medical measurements present. The scatterplots of encounters_cost versus each feature also displayed a variety of relationships, most of which did not appear too strongly positive or negative. The heatmap of correlation coefficients further confirmed that there were only a few features with a strong positive or negative correlation with the target feature.

5.0 Data Imputation

5.1 Manually Filling Missing Values

Before modeling, all missing values must be filled in. Missing value counts and percentages are calculated for each column. Through manual inspection, missing values for encounters_cost, num_encounters, medications_cost, num_medications, num_procedures, number of procedures, immunizations_cost, num immunizations, and num_allergies can be set to 0. Here, it is assumed that if the value is missing, the patient likely did not have any medical encounters, medications, procedures, immunizations, or allergies that year.

5.2 Data Imputation Methods

In order to fill in the rest of the missing values, four different imputation techniques were explored. Mean imputation fills in the missing values of each column with its mean value. Similarly, median imputation fills in the missing values of each column with its median value. K nearest neighbor (KNN) imputation fills in missing values using the values of similar neighboring data points. Lastly, multivariate imputation by chained equations (MICE) fills missing values using regression models.

All four imputation methods were performed and assessed to determine which method was best. First, each imputation technique was evaluated through comparing the R-squared values for a simple ordinary least squares (OLS) model.

Imputation Technique	R-squared
Mean	0.4934
Median	0.4941
KNN	0.5313
MICE	0.5139

The R-squared values are all fairly similar, with KNN slightly higher than the others. Next, histograms of all features were constructed to determine how each imputation technique alters the underlying distribution of the variable (not shown). In general, median, mean, and MICE imputation seemed to create unimodal distributions that did not reflect the distribution of the original dataframe without imputed values. The KNN imputation method seemed to make the distribution more normalized and is about the closest to the original distributions. Therefore, the KNN imputation technique was selected since it produced the highest R-squared value and resulted in somewhat normalized distributions of imputed variables.

6.0 Modeling

6.1 Dummy Regression Model

To prepare for modeling, the data was split into training and testing sets, with 25% of the data reserved for testing. For a baseline model, the mean value of the training data set was used as a predictor. This was performed using a dummy regressor.

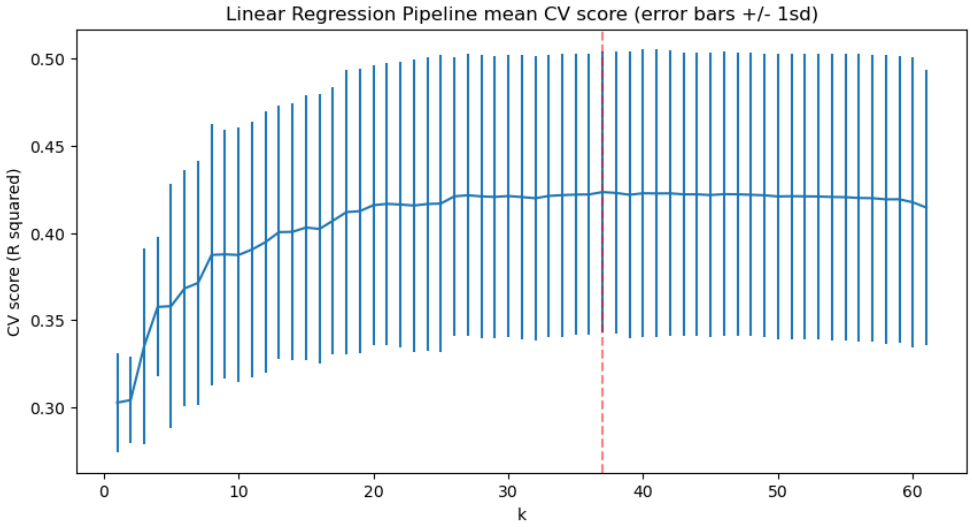
Mean value of training data	15,040.24
Training R-squared	0.0000
Testing R-squared	-0.0006
Training MAE	18,029.71
Testing MAE	17,276.07
Training MSE	1,214,707,483.91
Testing MSE	1,141,617,626.70

The R-squared values are very low and the MAE and MSE values are very high, showing that this is not a great fit for the model, as expected. These values are saved to be compared to the final model.

6.2 Linear Regression Model

For the following models, the data was scaled with a standard scaler to improve performance of the models and ensure that no single feature dominated the calculations.

A standard linear regression model was fit on the training data. The best 10 features were selected to be included in the model using F-regression, and 5-fold cross validation was performed. Next, a grid search was performed to determine the best number of features to include (k), which was determined to be 37. The mean R-squared value versus the number of features was plotted to visualize how the number of features affects the performance of the model.



The graph shows that the R-squared value flattens out around 20 features, so it is not necessary to include 37. The final model is evaluated, using the 20 best features.

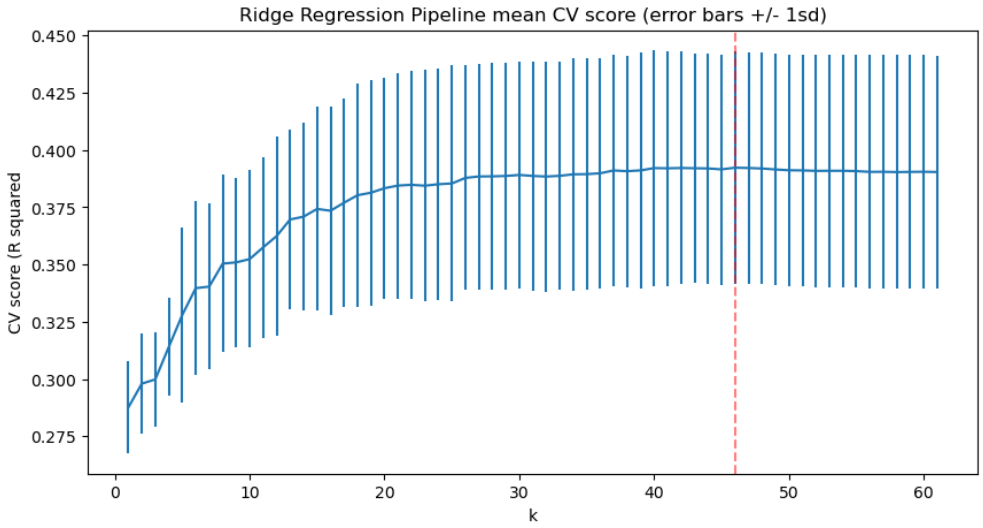
Training R-squared	0.4464
Testing R-squared	-0.1396
Training MAE	13,050.87
Testing MAE	12,799.56
Training MSE	672,463,349.73
Testing MSE	1,300,162,762.60

Since the testing R-squared value is lower than the training R-squared value, it can be concluded that the model is overfitting. The negative R-squared value for the testing data set shows that this model is not a good fit. However, it is an improvement over the dummy regressor prediction when comparing R-squared, MAE, and MSE.

6.3 Ridge Regression Model

Next, a standard ridge regression model was fit on the data. Different values of alpha [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0] were tested to assess various levels of controlling regularization strength. Each alpha value was used to fit the model and R-squared values were calculated. The alpha value of 1000 produced the highest R-squared value, so it was selected.

The best 10 features were selected to be included in the model using F-regression, and 5-fold cross validation was performed. A grid search was performed to determine the best number of features to include, which was determined to be 46. The mean R-squared value versus the number of features was plotted to visualize how the number of features affects the performance of the model.



The graph shows that the R-squared value flattens out around 20 features, similar to the linear regression model. The final model is evaluated, using the 20 best features.

Training R-squared	0.4073
Testing R-squared	0.2567
Training MAE	12,431.39
Testing MAE	12,021.69
Training MSE	719,984,233.37
Testing MSE	848,007,970.02

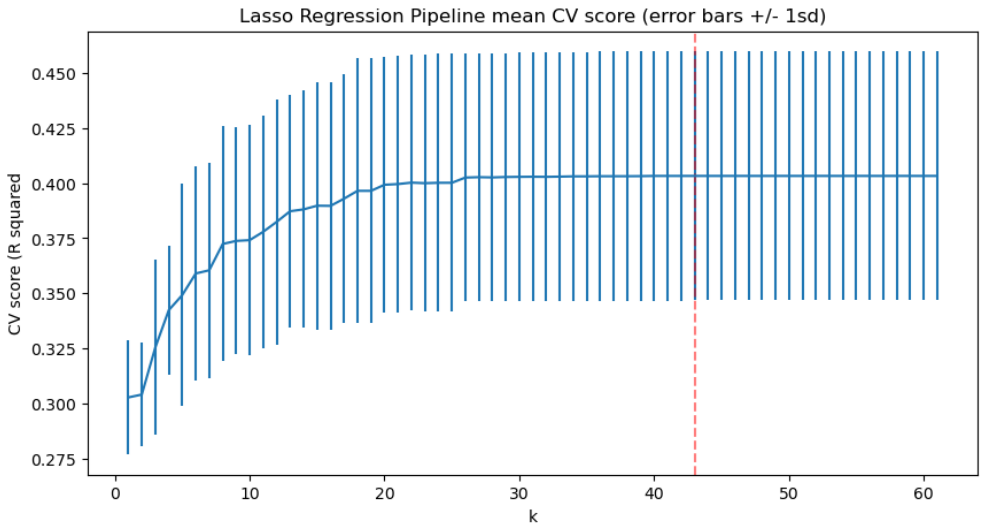
Since the testing R-squared value is lower than the training R-squared value, it can be concluded that the model is overfitting. This model is a slight improvement over the linear regression model when comparing R-squared, MAE, and MSE.

6.4 Lasso Regression Model

Next, a standard lasso regression model was fit on the data. Different values of alpha [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0] were tested to assess various levels of controlling regularization strength. Each alpha value was used to fit the model and R-squared values were calculated. The alpha value of 1000 produced the highest R-squared value, so it was selected.

The best 10 features were selected to be included in the model using F-regression, and 5-fold cross validation was performed. A grid search was performed to determine the

best number of features to include, which was determined to be 43. The mean R-squared value versus the number of features was plotted to visualize how the number of features affects the performance of the model.



Once again, the graph shows that the R-squared value flattens out around 20 features, so it is not necessary to include 43. The final model is evaluated, using the 20 best features.

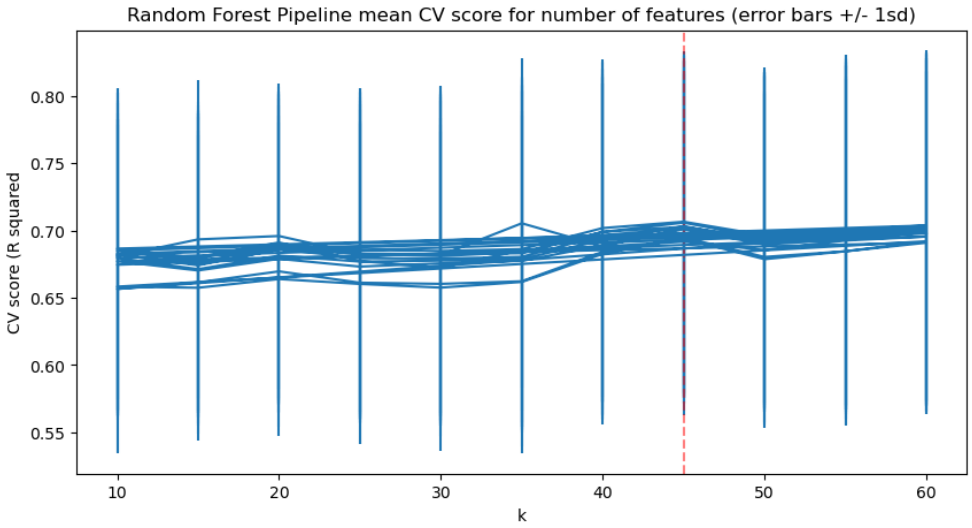
Training R-squared	0.4223
Testing R-squared	0.4184
Training MAE	12,130.95
Testing MAE	11,354.02
Training MSE	701,754,831.80
Testing MSE	663,539,670.31

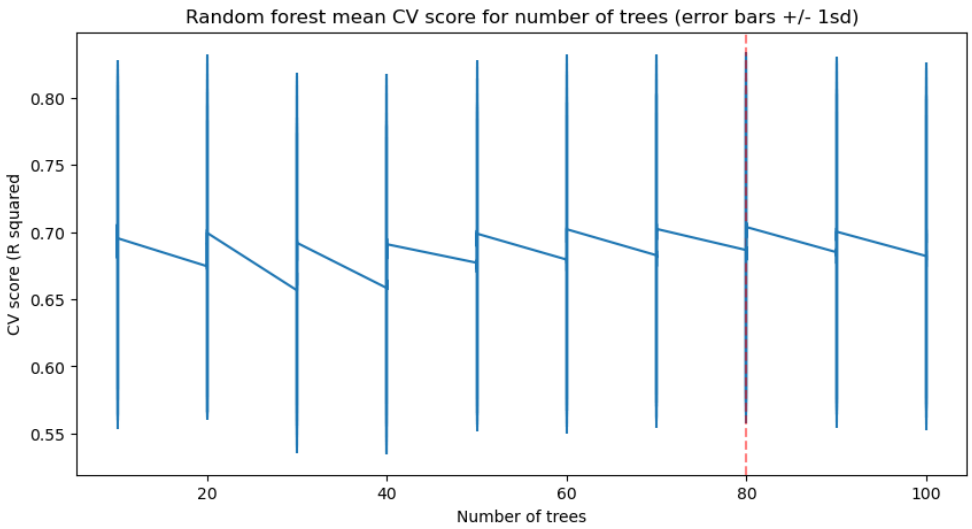
The testing R-squared value is closer to the training R-squared model, suggesting this model performed well with the testing data and is not overfitting to the training data. However, these R-squared scores are quite low. This model is a slight improvement over the ridge regression model when comparing R-squared, MAE, and MSE.

6.5 Random Forest Model

A default random forest model was fit on the data using F-regression to select the best 10 features to include. 5-fold cross validation was performed. A grid search was used to find the best number of features to include (k = [10, 15, 20, 25, 30, 35, 40, 45, 50 55, 60]) and the optimal number of trees in the random forest (n_est = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]). The best number of features to include was determined to be 45, and the optimal number of trees in the random forest was determined to be 80. The mean R-squared value

versus the number of features and the number of trees were plotted to visualize how these parameters affect the performance of the model.





The final model is evaluated, using the 45 best features and 80 trees.

Training R-squared	0.9426
Testing R-squared	0.6350
Training MAE	2,519.64
Testing MAE	6,167.80
Training MSE	69,728,646.81
Testing MSE	416,450,743.75

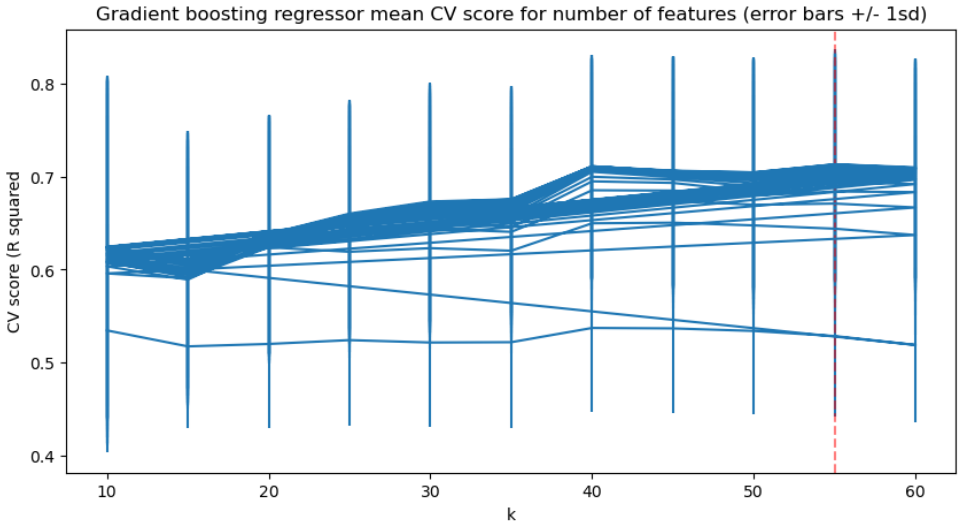
Cross validation was performed on the final model to determine its accuracy.

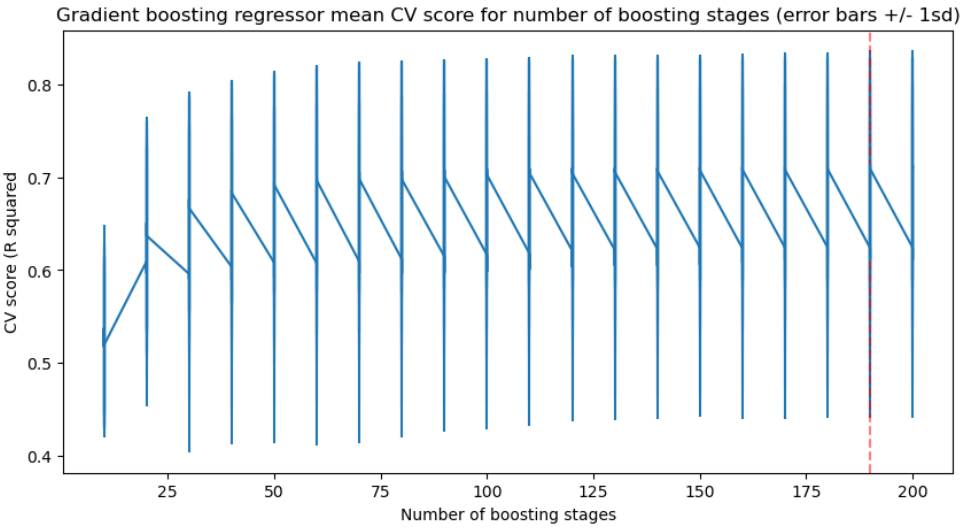
Mean R-squared	0.7065
Standard deviation R-squared	0.1263

Since the testing R-squared value is lower than the training R-squared value, it can be concluded that the model is overfitting. However, the R-squared, MAE, and MSE values are much better than those of any other model tested thus far.

6.6 Gradient Boosting Model

A default gradient boosting model was fit on the data using F-regression to select the best 10 features to include. 5-fold cross validation was performed. A grid search was used to find the best number of features to include (k = [10, 15, 20, 25, 30, 35, 40, 45, 50 55, 60]) and the optimal number of boosting stages (n_est = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]). The best number of features to include was determined to be 55, and the optimal number of boosting stages was determined to be 190. The mean R-squared value versus the number of features and the number of boosting stages were plotted to visualize how these parameters affect the performance of the model.





The final model is evaluated, using the 55 best features and 190 boosting stages.

Training R-squared	0.9475
Testing R-squared	0.6404
Training MAE	4,236.62
Testing MAE	6,439.04
Training MSE	63,764,250.97
Testing MSE	410,288,724.39

Cross validation was performed on the final model to determine its accuracy.

Mean R-squared	0.7129
Standard deviation R-squared	0.1234

Since the testing R-squared value is lower than the training R-squared value, it can be concluded that the model is overfitting. The R-squared, MAE, and MSE values as well as the cross validation results are very similar to the random forest model.

6.7 Final Model Selection and Assessment

Both the random forest model and the gradient boosting model performed much better than any of the other models tested. The difference between these two is so minimal and is likely due to how the data was split into training and testing sets. Since both models are similar, the random forest model will be selected for this dataset. This model is saved and exported.

The random forest model scores are compared to the dummy regressor model scores to show improvement over simply using the average as a predictor.

Percent change training R-squared	100.00%
Percent change testing R-squared	100.10%
Percent change training MAE	615.57%
Percent change testing MAE	180.10%
Percent change training MSE	1642.05%
Percent change testing MSE	174.13%

The final model is fit on the entire dataset, and its R-squared, MAE, and MSE values are calculated.

R-squared	0.9495
MAE	2,379.54
MSE	60,360,773.59

The R-squared value is very close to 1, suggesting that this model is a good fit for the data. The MAE and MSE values are also quite low, with the MAE of 2,379.54 suggesting that on average, this model would be expected to estimate a patient's yearly medical encounters cost within about $2,500.

Finally, the top 20 features are listed in order of importance.

1. num_encounters
2. num_procedures
3. DALY
4. Glomerular filtration rate/1.73 sq M.predicted [Volume Rate/Area]
5. Leukocytes [#/volume] in Blood by Automated count
6. Hematocrit [Volume Fraction] of Blood by Automated count
7. Body mass index (BMI) [Ratio]
8. Pain severity - 0-10 verbal numeric rating [Score] – Reported
9. AGE
10. meds_cost
11. Urea nitrogen [Mass/volume] in Blood
12. Chloride [Moles/volume] in Blood
13. Cholesterol in HDL [Mass/volume] in Serum or Plasma
14. Potassium [Moles/volume] in Blood
15. num_meds
16. Triglycerides
17. Carbon dioxide total [Moles/volume] in Blood
18. QALY
19. Creatinine [Mass/volume] in Blood
20. STATE_POPULATION

Unsurprisingly, the number of medical encounters and number of procedures are the top two features, as these are expected to be correlated with the cost of medical encounters. However, some other interesting factors include DALY (disability-adjusted life year, a metric used to measure overall disease burden of a population), BMI (body mass index), pain severity, age, and number/cost of medications. Some of these factors are out of a patient's control, such as age, but it would be beneficial to further investigate which factors could be altered through lifestyle interventions to reduce a patient's medical encounters costs.

7.0 Conclusion

In this project, a predictive model of yearly healthcare costs was developed using synthetic patient data from Synthea. The final dataframe was established by combining several different types of healthcare records. The data was cleaned and visualized, and missing values were imputed. Several machine learning models were trained and assessed to select the optimal model for predicting healthcare costs. The final model, a random

forest model, resulted in an R-squared value of 0.9495 and a MAE of 2,379.54 suggesting that on average, this model would be expected to estimate a patient's yearly medical encounters cost within about $2,500.

There remains much room for future work with this dataset. There was a large amount of data that was left out since it would be difficult to transform into numerical data. For example, taking into account the different types of encounters, medications, procedures, immunizations, and allergies would provide much more detail to the model rather than summing up the number of these variables and their associated costs. Additionally, considering health insurance coverage would reveal added information on how much patients are expected to pay for their medical encounters. Furthermore, there were 11 additional csv files not included in this analysis that could provide substantial data to strengthen this model. To further evaluate the robustness of this model, previous years of health data could be extracted from the datasets and compared to the following year's actual results. For example, a dataframe of 2022 healthcare data could be used to predict costs for 2023 and compared to the actual data of 2023.

This work provides a starting point for further development of machine learning models to help patients and providers work together to reduce the burden of healthcare expenses.