Final Report - Data Science Capstone Project

Johnson & Johnson - Prediction of commercial insurance payments for surgical procedures

Rahulraj Singh (rs4211), Prerit Jain (pj2383), Parth Gupta (pg2677), Mahesh Jindal (mj3038), Ayush Baral (ab5247)

December 17, 2022

Abstract

Johnson & Johnson (J&J) forecasts insurance payouts to wisely price and comprehend the costs of the medical devices they produce. In this project, we have implemented novel approaches to predict the payout prices at Surgery Procedure type, Metropolitan Statistical Areas (MSAs) and Year level. There are two main challenges in predicting MSA-level payout. Firstly, only ~10% of Americans having employer-based insurance have the claims level data. Thus, the data might only represent some of the population across the USA. Secondly, the granularity of the prediction is tricky as the number of claims for some procedures within some MSAs is either low or unavailable for inference. The J&J team uses DataRobot, an AutoML tool, to predict payouts. In the project, we have tried to improve the prior results and added a pipeline to obtain the procedure-level performance metrics like Mean Absolute Percentage Error (MAPE) and R2 to remove the dependency on the DataRobot[1].

1. Introduction

Prediction of commercial insurance payments for surgical procedures is a project sponsored by Johnson & Johnson (J&J). It is necessary to quantify customer economics, including the revenue (payment amounts), to comprehend the value that Johnson & Johnson Medical Devices (JJMD) produce. About 10% of Americans participating in employer-sponsored health insurance plans have claim-level data, including payment amounts, in the IBM MarketScan® Commercial Claims and Encounters (CCAE) database. The claim counts for surgeries within some metropolitan statistical areas (MSAs) are either too low (e.g., 50) or unavailable for inference, even though this data is an effective source for assessing national trends in surgical care for this population. Therefore, novel approaches are needed to fill up knowledge gaps in healthcare economics locally. Thus, in this project, we propose a novel machine-learning approach to address payment information gaps at the MSA level. Furthermore, we aim to improve the results given by the AutoML platform, DataRobot, which was previously used in this project.

According to our knowledge, no research was done to examine the potential of using machine learning (ML) to forecast the costs of JJMD surgeries at certain commercial facilities. Our study required a novel strategy to estimate procedure-specific commercial reimbursement inside MSAs where these data are lacking or insufficient. With the help of the AutoML platform DataRobot

(DataRobot, Inc., Massachusetts, USA) [1] and a novel approach, this research combined the features from several datasets (Census, Medicare, and IBM) for prediction.

- Impact and value to the business: To comprehend the value of client compensation, which eliminates the need to buy expensive data sets.
- Application and Scalability: It is simple to add new surgical categories by expanding the input parameters and procedure codes. As required, models can also be used for distributed analysis.

2. Exploratory Data Analysis

A series of preprocessing steps were conducted on the datasets used in this project. To build the final dataset, we combined data from four different sources:

- Hospital Dataset: This dataset contains data about all the hospitals across all regions of the US, holding their MSA codes and the amount paid to each of these hospitals throughout the COVID-19 pandemic.
- MSA Data: This dataset contains information about all MSAs across the US and reports their population estimates for three consecutive years (2010, 2011 and 2012). We further connect this with life expectancy data to combine life expectancy and population information.
- Average Income Data[2]: This dataset contains MSA-level average family incomes for households in that MSA.
- Life Expectancy Data[3]: This county-level dataset gives the average life expectancy for all counties across the US. We further connect this with MSA data to combine life expectancy and population information.

All these data sources were eventually merged into one master data file that was further used for exploratory data analysis and model building. We show the various EDA done below.

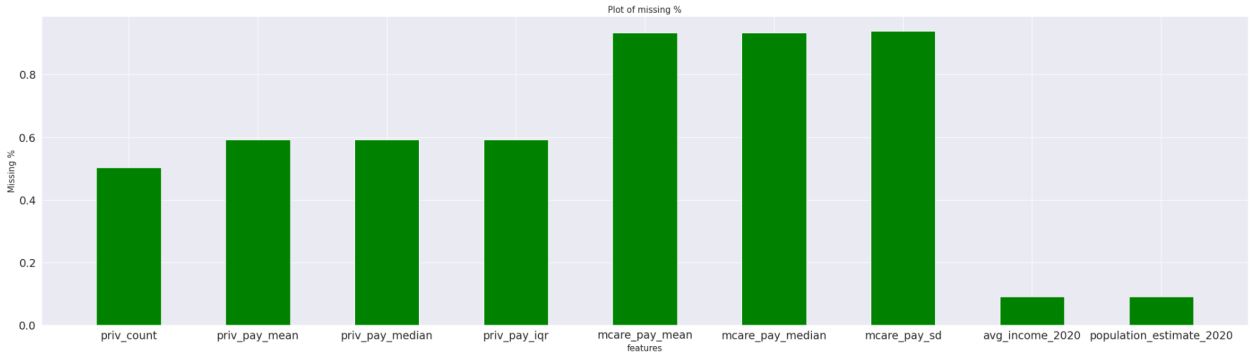

Fig 1. Missing value analysis of variables

Figure 1 displays the columns along the percentages of the missing value. Only columns having >1% missing values are displayed in the above chart. There are >80% missing values in mcare-related features (mean, median and standard deviation), and missing value indicators will be used instead. Avg Income and Population Estimated were imputed by the corresponding MSA-level mean values.

The choropleths below (Figure 2 and Figure 3) show the distribution of hospitals and life expectancy, respectively, spread across all the states of the US. We see that Texas has the highest number of hospitals in the US, while Wyoming has the least. Also, we observe that average life expectancy is higher in New Hampshire, Massachusetts and New York, while it is low in Georgia and Alabama. These analyses help us during the modeling as anticipating the insurance amount spent on surgeries will correlate with the average life expectancy and number of hospitals in regions.

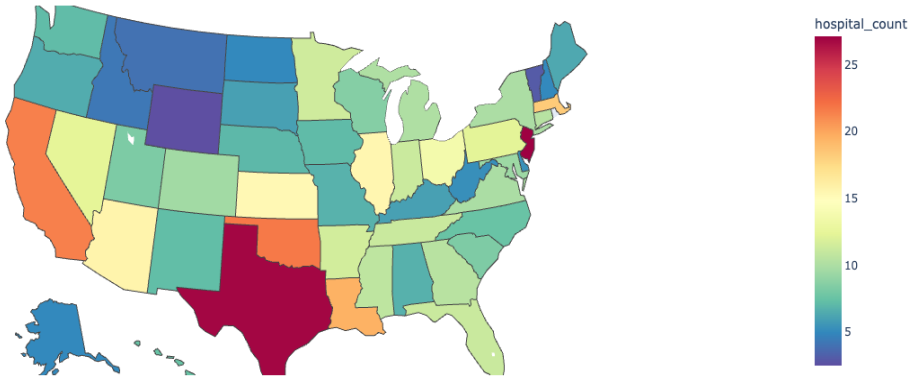

Fig 2. Choropleth plot of count of hospitals in the US

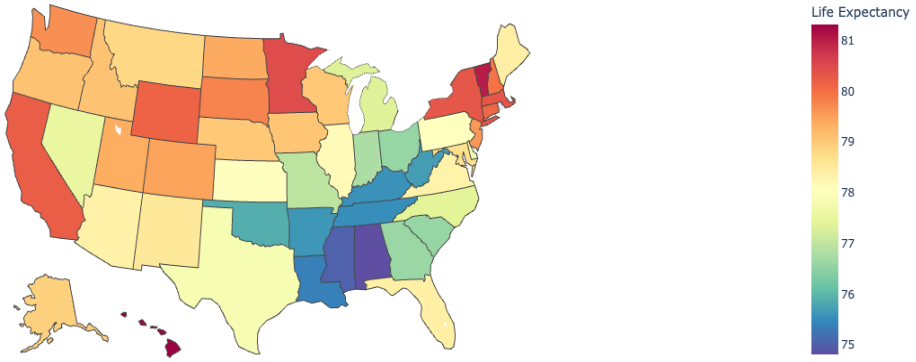

Fig 3. Choropleth plot of average life expectancy across the states of US

The plot below (Figure 4) shows us that our target variable is skewed to the right. It is affected by the mean, which is more towards the left. To make the model function better, we transform our target variable to be normally distributed. We tried two transformations, box-cox and log, to establish this. We can see from the plot below (Figure 5) that our target variable has converged to a normal distribution by using the Box-Cox transformation. We also tried to convert our target variable to be normally distributed by doing a log transformation (Figure 6). But the Box-Cox transformation does a better job than the log transformation.

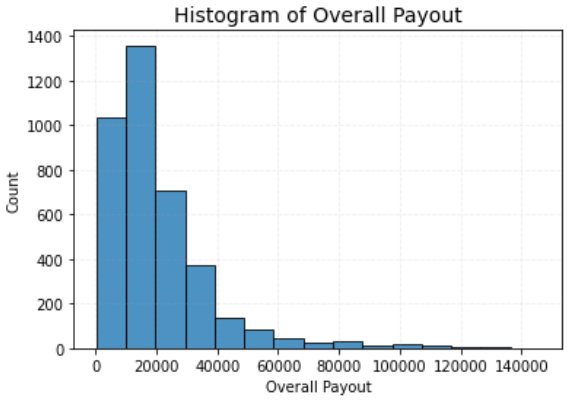

Fig 4. Histogram depicting the target variable

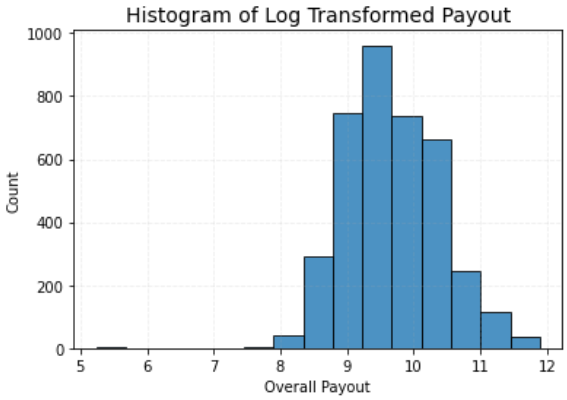

Fig 5. Log Transform of the target variable

We also checked the distribution of various procedures with respect to the mean payout in the box plot below. This plot shows numerous instances where the prices of surgical procedures have resulted in an amount significantly higher than the mean payout amount indicating an uneven distribution of mean prices.

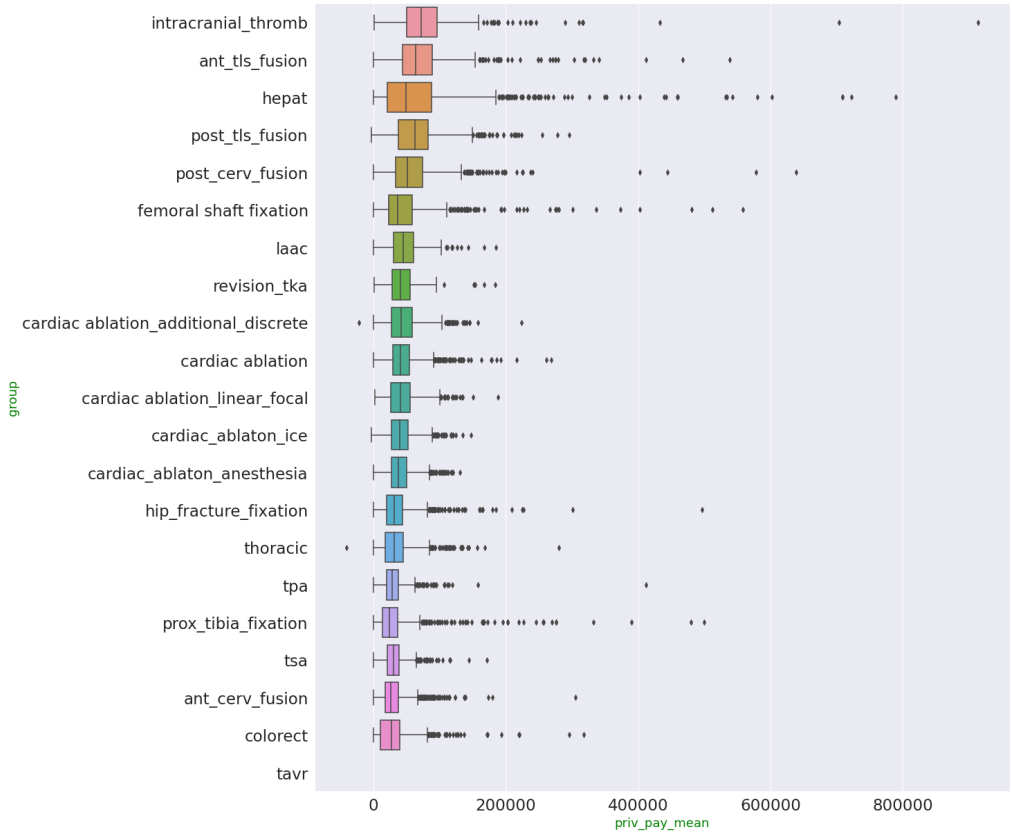

Fig 6. Box plot of payout distribution for different procedure groups

3. Methodology

3.1 Our Approach

After the initial exploratory data analysis, we realized that most of our data set was empty or, in other words, sparse. To correctly predict our target variable 'priv_pay_median' concerning a certain MSA, we needed to build our features. We used data from various census, economic, average income, and life expectancy data to make these features. These data were combined with the initial dataset concerning the MSA column, and additional hospital data was provided by Johnson & Johnson, which was also incorporated. Linear Regression was considered a baseline model, and a Decision Tree Regressor was also implemented. We also created a pipeline for the machine learning models to eradicate the repeated task of cleaning and encoding data. Post the successful deployment of our baseline model; we then sought to improve it by further refining our evaluations and methods. We tried implementing random forest, elastic net and XGBoost into our data and finally found that gradient boosting, specifically XGBoost, was the best-performing algorithm. Once we finalized the algorithm, we used bayesian optimisation techniques to tune the optimal hyperparameters for our model.

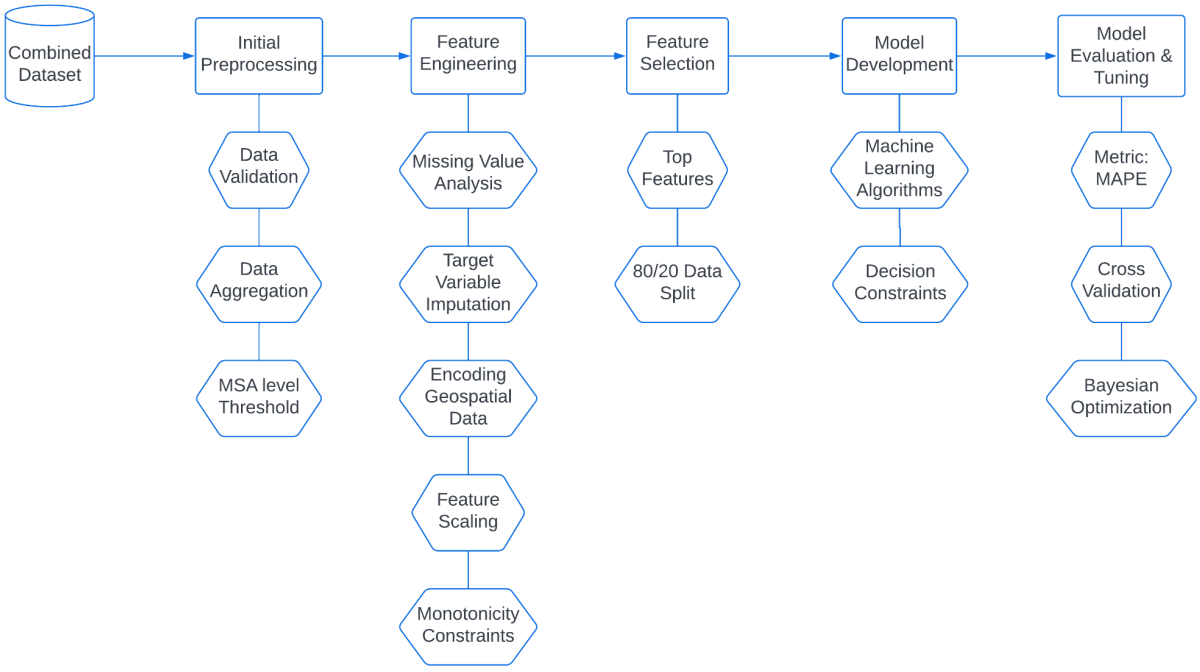

Fig 7. Outline of our approach to solving this problem

3.2 Datasets

We used a combination of data from J&J and public data sources like the US Census. The Hospital Dataset provided to us by J&J contains data about all the hospitals across all regions of the US. It holds their MSA codes and the amount paid to each of these hospitals throughout the COVID-19

pandemic. We also sourced an MSA dataset that contains information about all MSAs across the US and reports their population estimates for three consecutive years (2010, 2011 and 2012). We further connect this with life expectancy data to combine life expectancy and population information. We combined the datasets into one final data file, which was then used for the model building and preparation from this point on. Additional datasets we sourced and used for the project helped construct our custom feature sets during model preparation.

3.3 Initial Data Preprocessing

Initial Preprocessing started by checking how much data was missing for each column in the data given to us by Johnson & Johnson. Once initial analysis for missing data was done, we realized that most of the data was sparse or missing. Also, looking at the target variable and its missingness, we figured out that our target variable, Mean Payout, was missing approximately 60% of the data. It was an interesting problem, given the modeling we had to perform. There was not much we could do, or any of the algorithms would not perform well with the amount of data we had.

Next, we created features to incorporate the missingness in our data. We added Avg. Income, Population, Life Expectancy, Geographical Location and as such in our original dataset. We also added the haversine distance feature in our dataset to extract information from the latitude and longitude information given to us, which merged very well and provided more details with the MSA data.

Another technique to process the MSA and the procedure type was also included in the analysis. We applied the k-medoids method to cluster our data, hoping the accuracy would increase. But even with clustering, the accuracy did not improve as we had hoped, so we did not move forward with the preprocessing through this technique.

3.4 Feature Engineering

To improve the quality of the dataset that we would be using to build and test novel machine learning models, we first performed some feature engineering tasks to clean the data and

- Missing Value Analysis - To start with, we removed all the rows in the dataset that were missing the target variable. This amounted to about 60% of the total dataset. We attempted to use various imputation techniques, including some deep learning approaches, but because the availability of data for training these imputation models was only 40% of the total available data, none of the imputation techniques helped with the model's accuracy. Therefore, we removed all the rows where the target variable was missing.

- Target Variable Imputation - Imputed the target variable (different numbers of rows) using KNN, Decision Tree, Random Forest and deep learning techniques. However, the results regarding MAPE were not convincing enough as a trade-off towards introducing bias in the predictions.

- Encoding Geospatial Data - Geographical location data in our dataset, corresponding to MSA and hospital locations, was in the latitude and longitude format. Using this data in the form of text would not have been beneficial to the overall modeling process and therefore,

to turn them into numeric features, we used haversine distance calculation and encoded these geographical locations to numeric features.

- Scaling Features - The scale of our dependent and independent variables was vastly out of proportion. Hence, we tried using some transformations not limited to box cox and logs to bring the variable to a comparable scale. Based on the best performance, we chose to log transform the target variable. We have applied the following transformations depending on the type of variable.
    - For continuous variables, we have used standardization (standard scaling)
    - For categorical variables, we have used one hot encoding transformation.
    - For categorical variables with many categories, we have applied target encoding.

- Monotonicity Constraints - An important business task from J&J was to impose monotonicity constraints on the model. To do the same, we used the site weight feature to add a custom weight during the model training process.

- Clustering: we also some clustering techniques on the hospital datasets and added cluster numbers to the final dataset as categorical variables
    - Density-based spatial clustering of applications with noise (DBSCAN)
    - K-Medoids
    - Using the Hospital dataset directly instead of cluster numbers.

- Deciding the threshold value for the minimum number of procedures - To have a representative sample, we decided to have a threshold of 30 surgeries for each procedure within an MSA. This was chosen for the following reasons:
    - No. of training samples remaining after filtering - we want a large dataset.
    - No. of unique procedures - ideally want the set to be exhaustive
    - The threshold should be higher enough to represent the actual summary (mean/median/mode) and withstand the effect of outliers.
    - The graphs below, show our approach for calculating the final threshold

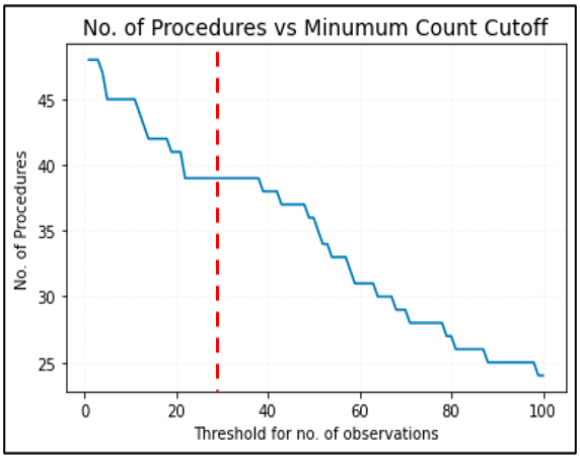

Fig 8. Threshold calculation based on procedures

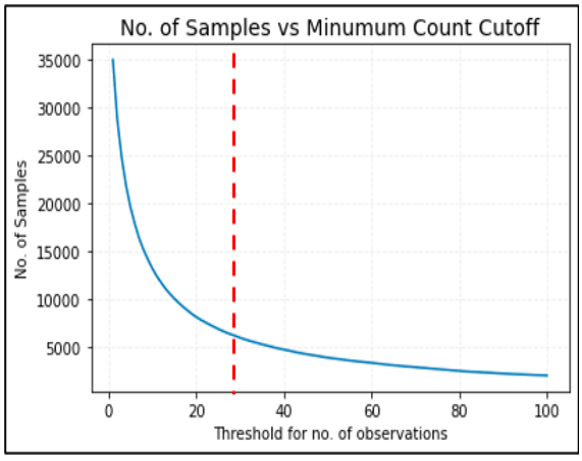

Fig 9. Threshold calculation based on samples

3.5 Clustering

We have used features from the "cost discharge dataset" containing information on discharge cost for a certain group and procedure. As per our discussion with J&J Team, these features can be helpful in terms of improving the model performance for predicting mean payout. We planned to divide the cost discharge information into several clusters. Each cluster will be assigned a cluster number embedded with master dataset features at the procedure and group level.

- **Transforming cost discharge dataset features for clustering:** There are different types of categorical, ordinal, and nominal features within the cost discharge dataset which need to be transformed so that we can further apply clustering techniques. The challenge is, we need to apply appropriate transformations which can incorporate all the information from the features before applying any clustering algorithm. After exploring the possible options, we decided to transform the features into Gower distances.

- **Computing Gower distances:** Gower's Distance can be used to measure the distinction between two records. The records may contain logical, categorical, numerical or text data. The distance is always between 0 (identical) and 1 (maximally dissimilar). Gower's Distance uses different metrics for each data type. The metrics for each data type are described below:

    a. Quantitative (interval): Range-normalized Manhattan distance.
    b. Ordinal: Variable is first ranked, then Manhattan distance is used with a special adjustment for ties.
    c. Nominal: Variables of k categories are first converted into k binary columns and then the Dice coefficient is used.

- **Applying Clustering Techniques:** We have applied different clustering techniques after transforming the input dataset features using Gower distances. A brief overview of different clustering algorithms has been provided below:

    - **DBSCAN Clustering**

      Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm. It is a density-based clustering non-parametric algorithm given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and is also the most cited in scientific literature.

      The DBSCAN algorithm can be abstracted into the following steps:

      I. Find the points in the ε (eps) neighborhood of every point and identify the core points with more than min_samples neighbors.
      II. Find the connected components of core points on the neighbor graph, ignoring all non-core points.
      III. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise, assign it to noise.

Note: Here "ε" (eps) and "min_samples" are the model hyperparameters.

After transforming the dataset features, we applied the DBSCAN algorithm. After algorithm tuning, we found ε = 0.4 and min_samples = 2 to be the optimal hyperparameters. This algorithm has divided the data into eight clusters whereas cluster number "-1" represents the data with noise. The cluster composition after running this algorithm has been provided below:

Cluster Number	Average of pat_cost	Average of discharge_to_home_pct	Average of IP_pct	No. of Groups
0	18273.59	62%	57%	17
1	17508.36	83%	49%	8
2	5670.52	98%	0%	4
3	24438.84	95%	20%	6
4	26883.03	56%	76%	4
5	11416.02	90%	50%	2
6	43448.35	48%	98%	2
-1	16228.02	78%	44%	7

Table 1. Aggregated cluster composition for each cluster group for DBSCAN

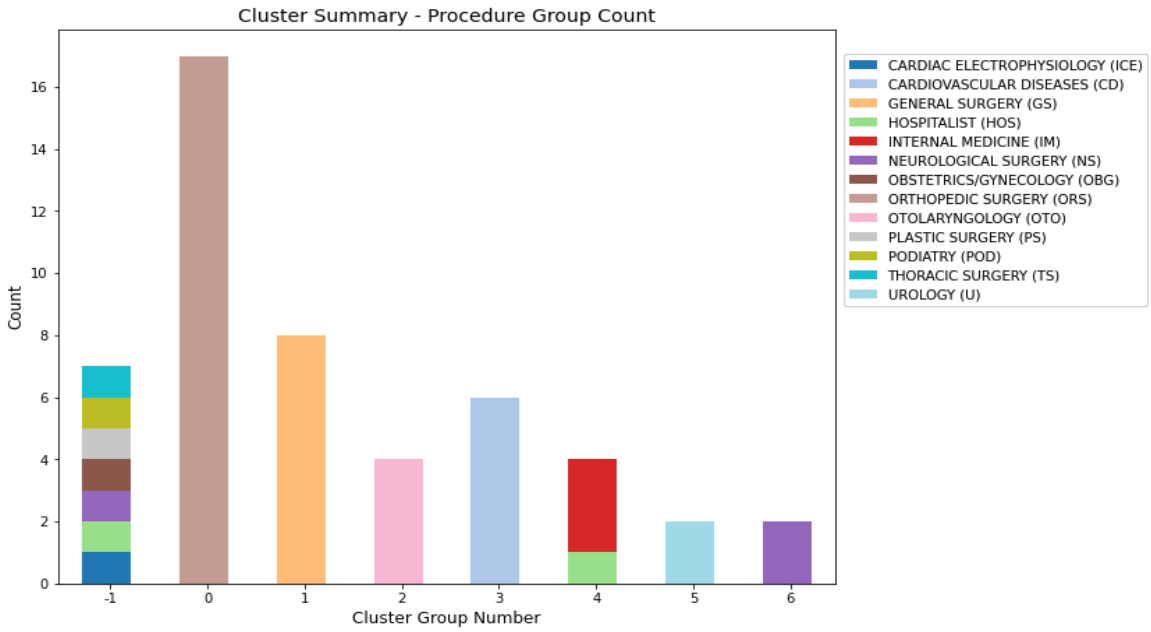

Fig 10. Segregation of different procedure groups within different cluster numbers using DBSCAN

After carefully analyzing the above plot, we can clearly say that the DBSCAN algorithm has correctly clustered the data based on different procedure groups. Moreover, certain procedure groups with insufficient data (only 1 sample) have been moved to the "-1" cluster representing the noise.

- **KMedoids Clustering**

The k-medoids problem is a clustering problem similar to k-means. Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups) and attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses actual data points as centers (medoids or exemplars), and thereby allows for greater interpretability of the cluster centers than in k-means, where the center of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster). Furthermore, k-medoids can be used with arbitrary dissimilarity measures, whereas k-means generally requires Euclidean distance for efficient solutions. Because k-medoids minimize a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, it is more robust to noise and outliers than k-means.

After transforming the dataset features, we applied the KMedoids algorithm. The optimal value of choosing the number of clusters is "5" which has been evaluated using the Silhouette method. The cluster composition after running this algorithm has been provided below.

Cluster Number	Average of pat_cost	Average of discharge_to_home_pct	Average of IP_pct	No. of Groups
1	12,017.72	83%	27%	12
2	23,484.36	96%	18%	7
3	28,729.43	43%	95%	14
4	17,472.97	84%	53%	11
5	5,805.22	98%	0%	5

Table 2. Aggregated cluster composition for each cluster group for K-Medoids

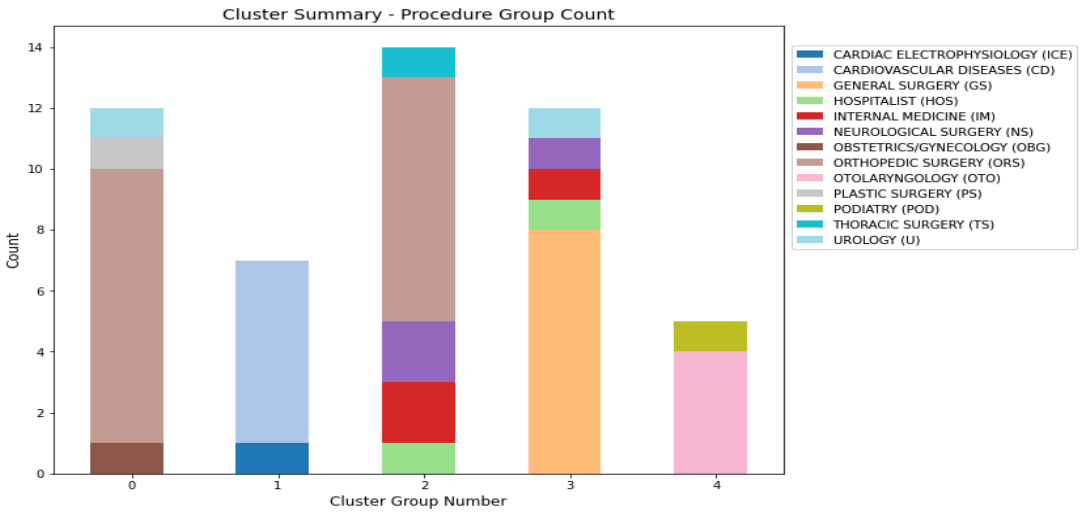

Fig 11. Segregation of different procedure groups within different cluster numbers using KMedoids

After analyzing the different cluster groups shown in the above plot, the K-Medoids algorithm has segregated data based on different procedure group types. The procedure group time with a similar correlation has been placed within a single cluster. For instance, the Cardiac Electrophysiology (ICE) and Cardiovascular Diseases (CD) have been segregated to cluster number "1" from the entire data.

3.6 Feature Selection

No. of observations in the dataset
Initial Rows: 133,089
Joining with Procedure Data: 129,275
Removing missing Target: 54,183
Applying cutoff of 30: 5,821

No. of Columns in the dataset
Initial Columns: 19
Joining with Procedure Data and feature engineering: 113
Removing after business logic and correlation/VIF: 61
Using Feature Importance: 32

Table 3. Feature Selection

On completion of the feature engineering step, all the independent and dependent variables were now scaled and ready for modeling. But, not every column of the data was needed for the machine learning model Therefore, from the data we chose the most important features based on the effect on the target variable and business impacts. Below are the features we selected to be included in the modeling.

Features	
Year of the claim/procedure	Site Weight (Monotonicity Constraint Feature)
Metropolitan statistical areas	Ownership of the site
Average Income (MSA-level)	Location of the Site (Urban/Rural)
Population (MSA-level)	Teaching Site
Average Life Expectancy (MSA-level)	Median Covid Payment
Haversine Distance using Latitude and Longitude	Cluster Numbers (not included in the final model)
Hospital categories based on numbers of beds	Category of Procedure (ex. Ortho, Cardio, etc.)
Type of Site - Inpatient, Outpatient, Ambulatory	Average Procedure Cost

surgery centers (ASC)	
Type of surgery (51 unique groups)	% Inpatient Claims
Mean Claim Payout from Insurance Company (Target)	MSA

Table 4. Features used for modeling

Finally, upon selecting the features, we used an 80-20 data split for training and testing purposes. Within the 80% training data, we further split it into 60% for pure model training and 20% for bayesian optimization and hyperparameter tuning.

3.7 Model Development and Training

For the implementation of our machine learning models, we tried five different approaches for solving the problem. The baseline model we used was Linear Regression, but then we moved to tree-based methods. The best-performing method turned out to be gradient boosting, specifically XGBoost. We have explained the implementation and results in further detail in the sections below.

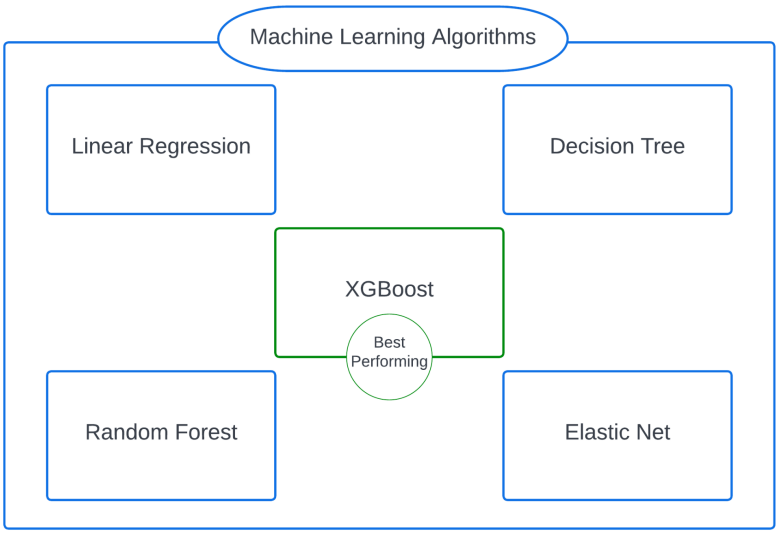

Fig 12. Machine Learning algorithms implemented in the project

3.8 Decision Constraints

According to the business requirement from J&J, we must enforce the constraint that– the payout for the ASC site should be lower than the Inpatient care facility at MSA and procedure level. A common type of constraint in this situation is that certain features bear a monotonic relationship to the predicted response:

- $f(x_1, x_2, \ldots, x, \ldots, x_{n-1}, x_n) \leq f(x_1, x_2, \ldots, x', \ldots, x_{n-1}, x_n)$

whenever $x \leq x'$ is an increasing constraint; or

- $f(x_1, x_2, \ldots, x, \ldots, x_{n-1}, x_n) \geq f(x_1, x_2, \ldots, x', \ldots, x_{n-1}, x_n)$

whenever $x \leq x'$ is a decreasing constraint.

XGBoost can enforce monotonicity constraints on any of the features using the monotone parameter. In our case, we have created a feature site_weight = 0 if the site is ASC, otherwise 1. We imposed increasing monotonic constraints on site_weight while training the XGBoost model. We created an array of length 29, with all 0s and 1 on the starting index. We imply that the model should be monotonically increasing for variable 1 and not enforce constraints on the rest of the variables.

After enforcing this constraint in our model and comparing for (msa, group, year) for ASC and inpatient in the training + test dataset, we are predicting correctly for all 81 cases (unique combination of MSA, Group and Year).

3.9 Model Evaluation and Tuning

Bayesian optimization is a method for the global optimization of black-box functions. It is particularly useful for optimizing expensive functions, such as those that take a long time to evaluate or require physical experiments to be run. The basic idea behind Bayesian optimization is to construct a probabilistic model of the function being optimized, based on observations of the function, and then use this model to choose the next point to evaluate to maximize the expected improvement in the function's value.

One of the key equations used in Bayesian optimization is the acquisition function, which determines the next point to evaluate. The acquisition function considers the uncertainty in the probabilistic model of the function and the expected improvement in the function's value at each point. The exact form of the acquisition function depends on the specific method being used, but a common form is the expected improvement (EI) function, which is defined as

$$EI(x) = p(x) * (f^* - f(x))$$

where f^* is the current best value of the function, $f(x)$ is the predicted value of the function at point x, and $p(x)$ is the probability that the function has a maximum at x.

Another important equation in Bayesian optimization is the probability of improvement (PI) function, which is defined as

$$PI(x) = P(f(x) > f^*)$$

where $P(X)$ is the probability of the given event. The probability of improvement function gives the probability that the function will improve if it is evaluated at point x. This can be useful for deciding how many points to evaluate, for example, by choosing a set of points with the highest probability of improvement. Bayesian optimization can be applied to a wide range of optimization problems, from hyperparameter tuning in machine learning to design optimization in engineering. It has been shown to be effective in many scenarios and can provide significant improvements in the speed and accuracy of global optimization.

We tried to implement bayesian optimization for hyperparameter tuning of the XGBoost models. We have considered the following parameters while training the model:

1) n_estimators: Number of gradient-boosted trees. We considered a range from 400 to 1000 trees
2) gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. We considered the values in the range from 0 to 5.
3) reg_alpha: L1 regularization term on weights. Increasing this value will make the model more conservative. We considered the range from 0 to 20
4) max_depth: Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. We considered the range from 3 to 10.

Below we have created two scatter plots of train and test MAPE for 100 trials of Bayesian Optimization for hyperparameter tuning. The best Test MAPE is 14.02%

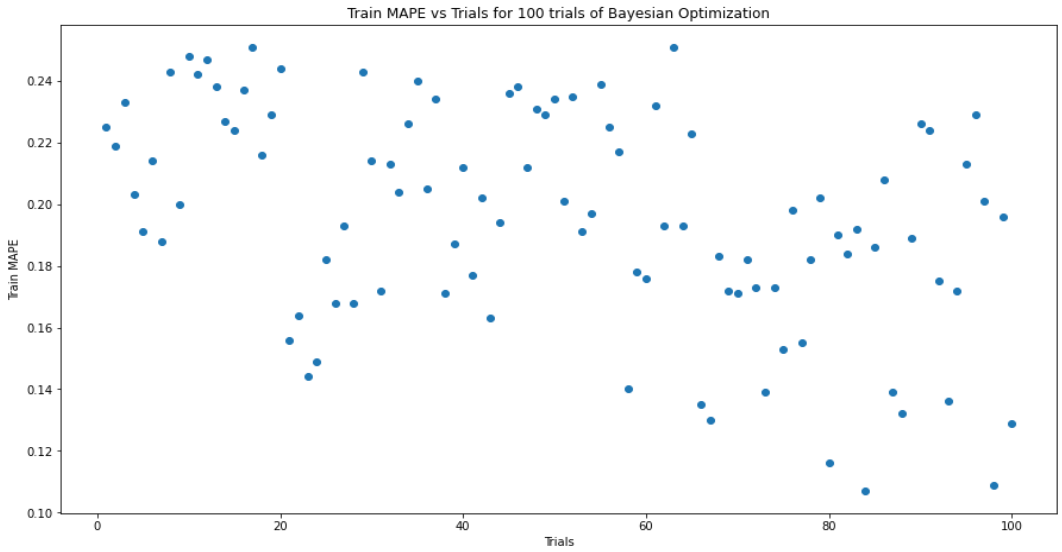

Fig 13. Plot of Train MAPE for 100 trails of Bayesian Optimization

Also, we used an objective as 'reg: squared error' and enforced monotonicity constraints using the monotone parameter. One Caveat of Bayesian Optimization is that since it builds surrogate functions based on Gaussian Processes, we can't ensure getting the same combination of hyperparameters after 100 trials every time.

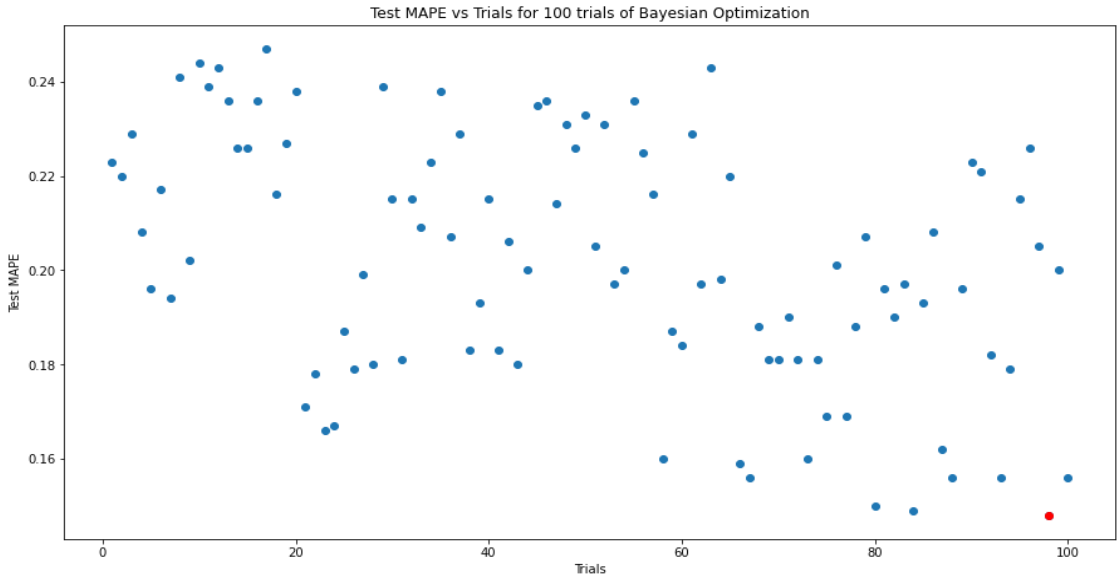

Fig 14. Plot of Test MAPE for 100 trails of Bayesian Optimization

4. Results

4.1 Performance of Models

As recorded earlier, we implemented five different machine learning approaches in the search of the optimal best-performing algorithm. Every experiment helped us gain more insight into the problem and ultimately helped us come to a final approach which was fast and accurate. Below, we list the performances (measured in MAPE) for all the models we worked with throughout this project using different experiments.

MAPE is defined as

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

where A_t is the actual value and F_t is the predicted value.

Model Performance results with Clustering Techniques

We have tried embedding the cluster numbers with our existing master dataset and retrained the models. These cluster numbers have been generated through different approaches by dividing the

cost discharge data into different clusters mentioned in the Clustering section above. Among the two different clustering approaches mentioned, model evaluation results after clustering with K-Medoids outperformed in comparison with the DBScan clustering algorithm. The results for model evaluation with K-Medoids clustering are presented below:

Model	Test MAPE
Elastic Net	27.29%
Decision Tree	23.25%
RandomForest	21.50%
Gradient Boosting	14.87%
XGBoost	14.68%

Table 5. Model Comparison with MAPE results after adding cluster numbers as features

We can clearly see that the addition of new information in the form of cluster numbers didn't help much in terms of improving the model evaluation results. Due to this, we discarded the approach of using cluster numbers with our master dataset for the final model version.

Final Model Performances

The following results have been generated upon performing model training and evaluation on the final set of features:

Model	Test MAPE
Linear Regression	30.86%
Decision Trees	28.94%
Random Forest	24.04%
Elastic Net	21.34%
XGBoost	14.02%

Table 6. Model Comparison with MAPE results

4.2 Feature Importances

From the XGBoost model, which we ultimately chose as the final algorithmic approach for solving this problem of predicting insurance payouts, we calculated the features which were most important

to the model's training and accuracy. Below we list out all the features in order of their importance along with the resulting numerical factor.

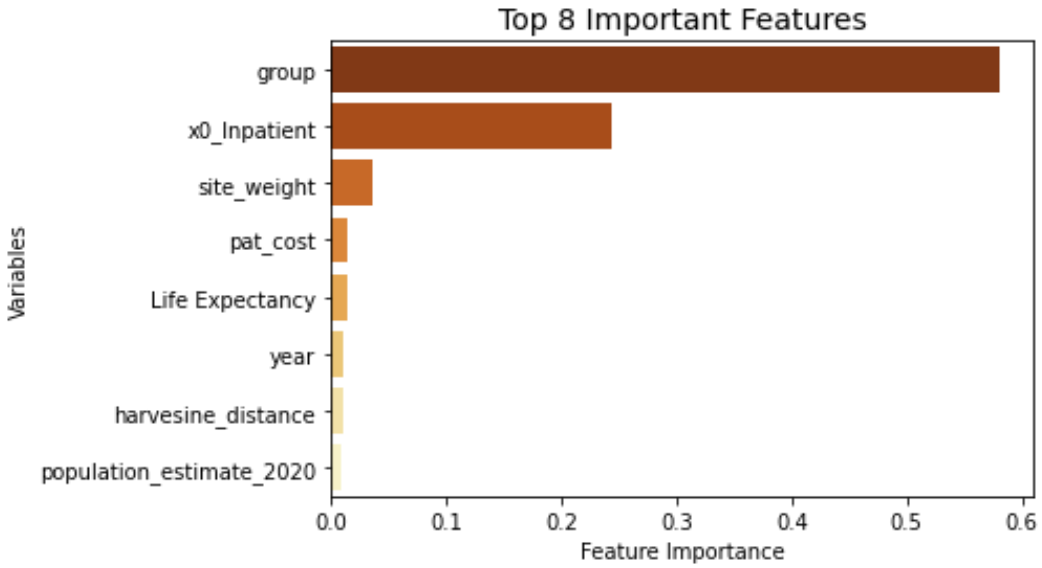

Fig 15. Plot of top 8 most important features for XGBoost Model

Feature	Description	Importance Factor
Procedure	Type of surgery	0.58
Inpatient	One Hot Encoded - Inpatient Site of Care	0.24
Site Weight	Monotonic Constraint Flag - 1/0	0.036
Pat Cost	Average procedure cost	0.014
Life Expectancy	Average Life Expectancy (MSA-level)	0.013
Haversine Distance	Haversine Distance using Latitude and Longitude	0.011
Year	Year of the claim/procedure	0.011
Population	Population (MSA-level)	0.008

Table 7. Feature Importance

4.3 Performance Comparison with DataRobot Model

The goal of this project was to not only build a robust machine learning model for predicting insurance payouts but also to eradicate the dependency on DataRobot, which is a tool J&J was currently relying on for this task. With our optimized and best-performing XGBoost model, we were able to outperform the DataRobot algorithm in both its accuracy and training time. The XGBoost model turned out to be ~15% better in terms of performance and about ~2.5 times faster to train and predict. The comparisons between our model and the DataRobot model are highlighted below.

Model	MAPE	Training Time
DataRobot	16.11%	~300 seconds
XGBoost	14.02%	~120 seconds

Table 8. Model Comparisons with DataRobot

Following the overall improvement in prediction accuracy and training time, we completed the modeling, evaluation, and prediction phases. With that said, there still could be potential improvements to the modeling approach in future iterations and we talk about them in the next section.

4.4 Procedure-wise Performance

One of the important goals of the project was to create a pipeline step for finding the surgical procedure level performance (MAPE). The results along with those of the DataRobot Model are in the below Table 9.

Procedure	Model MAPE	DataRobot MAPE
ankle_fix	19.79%	25.57%
ant_cerv_fusion	16.85%	21.31%
bariatric	9.70%	11.71%
breast reconstruction	12.69%	15.60%
bsp	21.18%	12.16%
bunionectomy	13.25%	27.99%
cardiac ablation	14.76%	16.84%
cardiac_ablaton_ice	11.45%	22.32%
colorect	15.85%	10.02%
fess	15.56%	24.07%

hepat	29.94%	32.23%
hernia	20.88%	11.73%
hysterect	12.82%	11.96%
lap appendectomy	9.20%	13.51%
mastectomy	15.04%	23.19%
post_tls_fusion	11.89%	8.96%
prox_tibia_fixation	4.53%	16.18%
radius/ulna internal fixation	11.83%	13.12%
robotic_assisted_surgery	21.66%	18.15%
rtc_slap_bank	18.05%	15.93%
septoplasty	13.70%	14.29%
tha	8.78%	6.81%
tka	13.03%	8.83%
tpa	27.33%	7.07%

Table 9. Procedure-level Performance Comparison with DataRobot

5. Goals and Next Steps

The project solved two key problems, one being the elimination of dependencies on the DataRobot which was important to the business team since maintaining DataRobot adds additional external costs. In addition to that, the model that we have implemented in this project also improves in terms of performance. This saves licensing costs and improves the prediction power of the model. With these predicted results, the J&J team can use it for revenue from sales of their devices. The prediction of insurance payouts can help JJMD price its devices in a way which yields more profit and sales. The complete end-to-end pipeline is highly scalable and can be easily adapted across multiple verticals within J&J. The project is also expected to impact J&J Data Scientists, the medical devices pricing team, patients, and the broader J&J group.

There are some next steps that we have highlighted for the future below:

- There is potential in exploring options for the explainability of the model, and it can be established using LIME and SHAP.
- Once the model is deployed to production it should be monitored for drifts.
- Developing optimal pricing and product penetration strategies to boost sales.

6. Contributions

- Mahesh Jindal: Contributed to data preprocessing and combining raw data from different sources. Built the master data file that is being used for further model building. Also worked on designing clustering techniques and implementing Missing Data Imputation Techniques.

- Prerit Jain: Designed and experimented with various machine learning algorithms and varied approaches to determine the best-performing algorithm. Worked on interpreting the clustering results and testing with different iterations of the test-train split.

- Parth Gupta: Performed extensive model evaluation post implementing monotonicity constraints to select the final model. Implemented bayesian optimization to find optimal hyperparameters for the XGBoost model.

- Ayush Baral: Performed exploratory data analysis to understand the data. Contributed to feature selection and the creation of new features for the master dataset.

- Rahulraj Singh (Team Captain): Set up meetings and milestones, and managed progress. Sourced and prepared life expectancy and income data to be used for modeling. Contributed to model validation and testing.

7. References

[1] DataRobot. [n. d.]. Data Robot: Automated Machine Learning for Predictive Modeling. Retrieved 05-Aug-2021 from https://datarobot.com

[2] Bureau of Economic Analysis https://apps.bea.gov/regional/downloadzip.cfm

[3] U.S. Department of Health & Human Services, Centers for Disease Control and Prevention https://www.cdc.gov/nchs/data-visualization/life-expectancy/index.html