Patient Characteristics to Predict the Type of Healthcare Service

Rabiya Fatima | Srilakshmi Mallipudi | Gautam Reddy | Barkha Sharma

December 2023
BUAN 5510 01- Capstone Project in Business Analytics

Table of Contents

Abstract…. 4
Introduction. 5
Background Information 5
Contributions 5
Problem Statement. 6
Dataset Description 6
Patient Characteristics Survey (PCS) 2019 Dataset. 6
County population listed by zip code 7
New York Hospitals listed by zip code 7
New York Parks listed by latitude and longitude (converted to zip code) 7
Median Household Income by zip code 7
Data Exploration 8
Exploring Healthcare Service Utilization Patterns 8
***Literatur*e Review 13**
Machine Learning for Developing a Prediction Model of Hospital Admission of Emergency (Melhem et al. 2021) 14
Department Patients: Hype or Hope? (de Hond et al., 2021) 14
Predicting hospital admission at emergency department triage using machine learning (Hong et al., 2018) 15
Predicting hospital admissions to reduce emergency department boarding (Golmohammadi, 2016). 16
Diabetes-Related Inpatient Stays, 2018 (Fingar & Reid, 2018) 16
Hospitalizations Related to Diabetes in Pregnancy (Wier et al., 2008) 17
Reducing Health Care Disparities: Where Are We Now? (Gold, 2014) 18
Implicit Bias and Racial Disparities in Health Care (Bridges) 19
Data Pre-processing. 23
Data Integration 24
Processing Null Values 23
Correlation Analysis 26
Converting Nominal/Ordinal data to Numerical 27
Normalization 27
Feature Selection using Random Forest Model. 28
Primary Component Analysis (PCA) 30
Data Mining Models and Evaluations 31
Hyperparameter Tuning. 32
Models and Evaluations. 32
Clustering 37
Selection of three top performing models 38
Discussion 39
Domain Knowledge. 39
Methodological Contributions 39
Conclusion 41
Summary. 41

Limitations *41*
Future Projects *43*
Recommendations *43*
References 44
Appendices 46
Data Dictionary *41*
Python Code *41*

Abstract

The study aims to assess a patient dataset from the New York region to predict the required healthcare services for individuals. Accurate anticipation of healthcare service needs is vital for aligning hospital resources with the demands of the population. In addition to patient data, various alternative data, including population statistics, park availability, and income distribution, were incorporated. Employing machine learning and data mining techniques, we have identified few patient attributes, insurance details, and socio-economic factors that serve as robust predictors for healthcare service prediction. The paper specifically concentrates on predicting the demand for four services: Inpatient, Outpatient, Emergency Care, and Residential. This predictive analysis facilitates proactive planning and efficient resource allocation within healthcare facilities, significantly influencing patient well-being and associated costs.

We focused on identifying an individual's genuine requirement for healthcare services. Hence, we are considering recall, which identified positive cases of services. Our top models are Random Forest, Gradient Boosting, and Decision Tree models, which use recall numbers to identify the majority of positive cases for outpatient and inpatient service categories. Outcome of feature selection had 20 features out of all which contribute to define the prediction.

Introduction

The US healthcare system is a complex and diverse mix of public and private, for-profit and nonprofit insurers and health care providers. It does not have a system of universal healthcare, and many people lack health insurance or face high costs of care. The US spends more on healthcare than any other country but does not have better health outcomes. Therefore, there is a need for more comprehensive and coordinated reforms to improve the value and quality of health care in the US. By analyzing patient data, their medical history, insurance coverage, household income, education and many more, we are trying to identify trends and patterns of healthcare service choices to be opted for by patients.

Our research uses predictive models to anticipate patients' healthcare services needed based on their profiles, streamlining care plans across inpatient, outpatient, residential, and emergency services. This proactive approach can enhance care quality, efficient resource allocation, and may curtail unnecessary hospitalizations and costs. By engaging patients in personalized care and informed decision-making, we aim for better health outcomes, improved care quality, and reduced expenses, aligning with healthcare's triple aim while ensuring fair access and enriched patient experiences.

Problem Statement

In analyzing the correlation between patient demographics, socio-economic factors, medical conditions, and healthcare service utilization, our study predicts that certain demographic and socio-economic characteristics, along with specific medical conditions, significantly influence the type of healthcare service utilized. Factors such as age, income level, insurance status, chronic illnesses, and severity of medical conditions are pivotal in determining whether patients are more inclined towards inpatient, outpatient, residential, or emergency healthcare services.

Background information

The US healthcare system is a complex and diverse mix of public and private, for-profit, and nonprofit insurers and health care providers. It does not have a system of universal healthcare, and many people lack health insurance or face high costs of care. The US spends more on healthcare than any other country but does not have better health outcomes. The US healthcare system has been the subject of ongoing debate and reform efforts, especially in the areas of cost, coverage, and quality.
The US healthcare system is not very effective in delivering cost-effective and quality service compared to other high-income countries. According to a report by the Commonwealth Fund, the US ranks last among 11 countries on measures of access, equity, quality, efficiency, and health outcomes, despite spending the most on health care. The US also has the highest rate of preventable deaths, the lowest life expectancy, and the lowest patient satisfaction among the countries studied. Some of the factors that contribute to the poor performance of the US healthcare system are the lack of universal coverage, the fragmentation and complexity of the system, the high administrative costs, the low investment in primary care and prevention, and the misalignment of incentives and quality. Therefore, there is a need for more comprehensive and coordinated reforms to improve the value and quality of health care in the US.

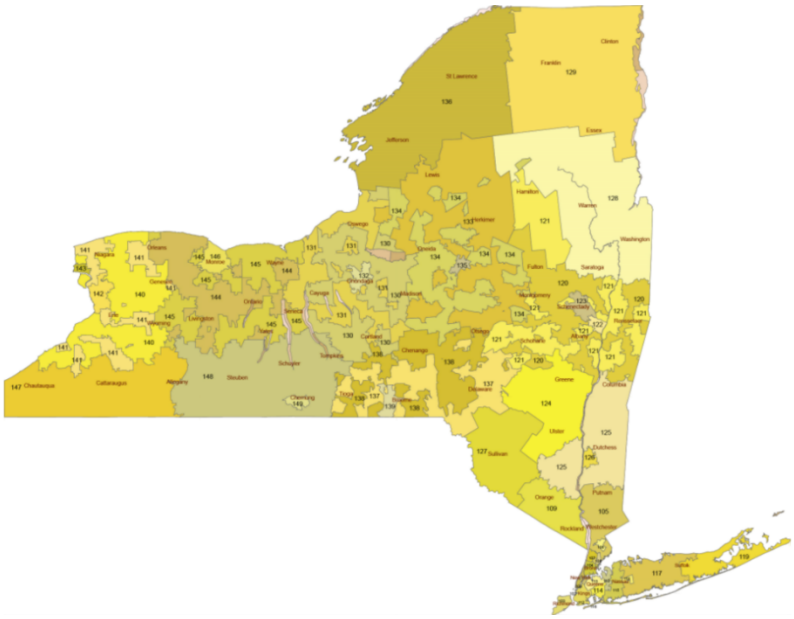
Description of Datasets

Patient Characteristics Survey (PCS) 2019 Dataset
This dataset was chosen for the capstone project because it provides a rich and diverse collection of information about patients and their interactions with the healthcare system. By analyzing this data, we can gain insights into how different factors, such as age, medical conditions, and insurance coverage, influence the type of healthcare services patients use. This knowledge can help improve resource allocation and patient care, making healthcare more efficient and effective.
Data Size: Records: 196K | Attributes: 76 | file size: 102 MB
Data Overview: This data set is a comprehensive collection of healthcare-related information, encompassing a diverse range of attributes related to patients and their interactions with the healthcare system. The data are organized by OMH Region-specific (Region of Provider).
County population listed by zip code
Data Overview: This dataset is a comprehensive collection of population and density in US counties, by zip code
Data Size: Records: 33K | Attributes: 18 | file size: 6 MB

New York Hospitals listed by zip code
Data Overview: This data set lists the Hospital on New York State Department of Health Hospital Profile website and includes demographic, inspection, complaint summary, and enforcement fine data for hospitals in New York State.
Data Size: Records: 225 | Attributes: 10 | file size: 60 KB

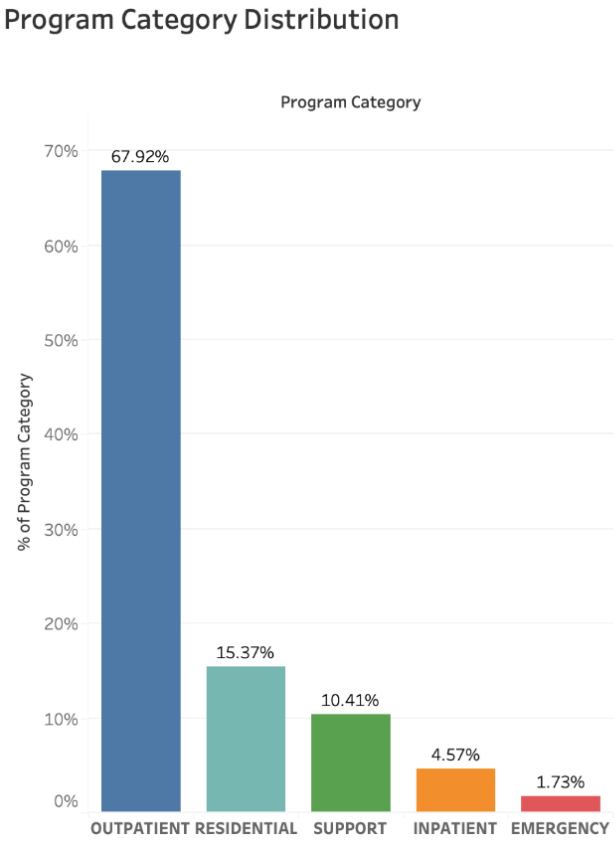
New York Parks listed by latitude and longitude (converted to zipcode) State_Park_Facilities_Points_Map.csv
Data Overview: This dataset lists the Parks in New York State.
Data Size: Parks in NY: 255 | Attributes: 17 | file size: 38 KB

Mean income by Zip code (Income aggregated by zip code) income.csv
Data Overview: The Statistics of Income (SOI) Division's ZIP code data is tabulated using individual income tax returns.
Data Size: AGI: 255 | Attributes: 152 | file size: 199.7MB



Data Exploration

Exploring Healthcare Service Utilization Patterns



Our analysis of 196,102 patient records reveals intriguing trends in healthcare service utilization. Outpatient care emerges as the predominant choice, with approximately 68% of patients favoring this service. Interestingly, adults constitute around 78% of the patient population, with children displaying a higher preference for outpatient services, possibly indicating a leaning towards preventive and routine care. Conversely, adults show a greater inclination towards residential care, suggesting a propensity for self-care.

Gender plays a nuanced role in healthcare utilization, with men exhibiting higher rates across categories except for Outpatient care. Women tend to favor Outpatient services, potentially indicating a preference for routine care over intensive inpatient services.

The analysis also unveils regional disparities; for instance, the Hudson River region displays lower outpatient but higher inpatient proportions, hinting at underlying health, insurance, and demographic factors influencing this pattern.

Transgender individuals, constituting less than 5% of the population, exhibit similar healthcare service patterns to non-transgender individuals. However, understanding and addressing disparities among this group are crucial for equitable healthcare access and outcomes.

Ethnicity appears to influence healthcare utilization, with Hispanic individuals showing higher outpatient service usage, while Black individuals favor residential services, reflecting distinct healthcare preferences.

Insurance type significantly impacts healthcare choices, with patients under private insurance preferring outpatient services, while those with no insurance lean towards inpatient and support services.
Regarding disease distribution, high blood pressure is predominant among outpatients, followed by issues such as obesity and chronic conditions. Mental health disorders, particularly serious mental illness, constitute a significant portion among outpatients. Physical impairments like visual and mobility disorders are prevalent, with outpatient care being a more frequent choice across all categories.
Understanding these utilization patterns across demographics, regions, and disease categories is pivotal for devising strategies to ensure equitable access and improved healthcare outcomes for diverse populations.

Literature Review

In the realm of healthcare management, the decision of whether a patient should receive inpatient, outpatient, residential, or emergency care is a critical one. It not only influences the allocation of valuable resources within healthcare facilities but also profoundly impacts patient well-being and the associated costs. Accurate prediction of the type of healthcare service a patient is likely to require is essential. It enables hospitals to efficiently manage their bed capacity, staffing levels, and medical supplies, ultimately alleviating the financial strain on healthcare systems while ensuring that patients receive precisely the level of care they need when they need it. This predictive capability hinges on the art of predictive modeling, where patient characteristics are harnessed to make informed decisions. While predictive modeling is at the heart of this endeavor, it also explores the critical need to understand the web of factors influencing health care access disparities and mitigate these disparities. In this literature review, we delve into the wealth of research pertaining inpatient and outpatient patterns based on patient characteristics and factors for healthcare access disparities.

Department Patients: Hype or Hope?

In a bid to harness the capability of Machine Learning for predicting models to improvise healthcare systems, the research paper by Hond et al. 2021 contributes to the ongoing discussion about the utility of machine learning in healthcare. It acknowledges the persistent challenges associated with ED overcrowding and its detrimental effects on patient outcomes, emphasizing the importance of accurate prediction models for early identification of patients requiring hospitalization to expedite the admission process and potentially improve patient satisfaction and outcomes. The study highlights the potential advantages of using machine learning (ML) models in predicting hospital admissions, including their ability to handle complex data patterns and large datasets and the growing availability of electronic health records. The study emphasizes the need for prediction models that can provide real-time guidance to medical professionals, thereby facilitating swift decision-making within the ED, which ultimately has the potential to impact patient care positively.

Machine Learning for Developing a Prediction Model of Hospital Admission of Emergency

Another comprehensive study presented in the paper by Melhem et al. 2021 addresses a critical issue in the healthcare sector. It discusses the challenges doctors and specialists face in determining whether patients should receive inpatient or outpatient care, emphasizing the time-consuming nature of this decision-making process and the potential for human errors that can impact patient

safety. To tackle this problem, the study utilizes Electronic Health Record (EHR) data from a private hospital in Indonesia and employs four machine learning models, including Support Vector Machine, Decision Tree, Random Forest, and K-Nearest Neighbors, to predict the appropriate type of care based on patient conditions and laboratory test results. The authors then evaluate these models based on various performance metrics. The results indicate that the Random Forest model achieved the highest accuracy, sensitivity, and precision, making it a promising approach to enhance patient care classification. The paper underscores the potential of machine learning to enhance healthcare services, reduce medical errors, and improve patient outcomes. Additionally, it highlights the potential benefits of integrating machine learning into healthcare decision-making processes and the importance of selecting the appropriate model for specific healthcare applications.

Predicting hospital admission at emergency department triage using machine learning

Inpatient and hospitalization prediction from Emergency Department triage is an actively researched area for predicting inpatient patterns. To this, Hong et al. (2018) focuses on the use of machine learning to predict hospital admission from emergency department data, aimed at improving the triage process and optimizing resource allocation by identifying patients who are likely to be admitted or discharged. Using data from three Emergency Rooms in a single hospital system, and including 972 variables from various categories, the study trained and tested three algorithms (logistic regression, gradient boosting, and deep neural networks) on three types of datasets (triage only, history only, and full). The paper proposes a low-dimensional model with the intent of facilitating implementation into an EHR system. The study also emphasizes the usage of the addition of historical information along with Trigae information resulting in significantly improved predictive performance.

Predicting hospital admissions to reduce emergency department boarding.

Contemporary study by Golmohammadi (2016) reviews the causes and consequences of delay in transferring patients from the emergency department to inpatient units within the hospital, such as lack of inpatient beds, inefficient diagnostic services, and poor communication between units. It proposes a prediction model to estimate the likelihood of admission of each ED patient to the hospital, based on their demographic and clinical information. The paper claims that this model can help improve hospital operational efficiency and reduce ED boarding, by providing better estimation of required resources and preparedness for inpatient care.

Diabetes-Related Inpatient Stays, 2018

Another researched patient characteristic contributing to increase in inpatient patterns was published by Fingar et al. 2018. The study sheds light on the significant contribution of diabetes to inpatient hospitalizations in the United States. The findings indicate that in 2018, there were over 8 million hospital stays related to type 1 or type 2 diabetes, with type 2 diabetes accounting for a substantial 95 percent of these stays. The research also highlights age disparities, with type 1 diabetes being more prevalent among patients aged 18–34 years, while type 2 diabetes predominantly affected those aged 65–84 years. According to study the leading principal diagnosis for stays involving type 1 diabetes was diabetes itself, accounting for half of all stays with this diagnosis. Conversely, septicemia was the leading principal diagnosis for stays involving type 2 diabetes, making up 10 percent of all stays with a type 2 diabetes diagnosis. These findings emphasize the impact of diabetes on hospitalization reasons. This research demonstrated the significant burden of diabetes-related hospitalizations in the United States, with type 2 diabetes

being particularly prevalent among older adults. It also highlights the disparities in in-hospital mortality rates and the leading reasons for hospitalization in patients with diabetes. Diabetes remains a major contributor to inpatient hospital stays and healthcare costs.

Hospitalizations Related to Diabetes in Pregnancy

On the same front, research by Wier et al. 2008 highlights that diabetes-related maternal stays accounted for about 6.5 percent of all maternal stays in 2008, with 5.4 percent involving gestational diabetes and 1.1 percent involving pre-existing diabetes complicating pregnancy. Notably, women with pre-existing diabetes were more prone to hospitalizations for diabetes-related complications during pregnancy, as one-third of hospital stays with pre-existing diabetes complicating pregnancy involved no delivery, primarily aimed at treating maternal complications. The article also mentions that from 1997 to 2007, there was a significant increase in hospitalizations for deliveries involving gestational diabetes and pre-existing diabetes complicating pregnancy, highlighting the increased risk for these patients. Finally, it is noted that the mean length of stay, and mean costs were higher for diabetes-related stays resulting in delivery, indicating the increased burden of hospitalization for diabetic patients during pregnancy.

Reducing Health Care Disparities: Where Are We Now?

Alternative research areas for us would be exploring factors affecting healthcare access disparity. The study by Gold et al. 2014, from Mathematica Policy Research, provides an overview of the evolution and status of efforts to reduce racial and ethnic disparities in healthcare. The article highlights the ongoing disparities in healthcare outcomes despite improvements in overall quality. It references the 2003 Institute of Medicine (IOM) report "Unequal Treatment," which raised awareness of disparities in healthcare quality. The U.S. Department of Health and Human Services (HHS) released its 10th annual report on healthcare disparities in 2013, emphasizing suboptimal quality and access, particularly for minority and low-income groups. The article also discusses Health and Human Services (HHS) goals for achieving health equity, ensuring access to quality care for vulnerable populations, and improving data collection by race, ethnicity, and other demographic factors. It notes the release of HHS's Action Plan to Reduce Racial and Ethnic Health Disparities in 2011, aiming for a nation free of disparities in health and healthcare. The article highlights the development of tools for measuring disparities and cultural competency, as well as efforts to enhance data collection. Furthermore, the article addresses the involvement of various stakeholders, including hospitals, physicians, and health plans, in initiatives to collect race, ethnicity, and language data. It emphasizes the importance of data collection to assess gaps in care and monitor progress in reducing disparities.

Implicit Bias and Racial Disparities in Health Care

Another contemporary research by Bridges in the paper “Implicit Bias and Racial Disparities in Health Care” explores the question of why black individuals tend to experience poorer health outcomes and earlier mortality compared to other racial groups. Bridges discusses the role of healthcare providers and the quality of care received by black patients as a significant factor contributing to these disparities. She cites the Institute of Medicine's report, which found that even when factors like insurance status, income, age, and severity of conditions are comparable, racial and ethnic minorities receive lower-quality healthcare than white individuals. The article delves into studies demonstrating that healthcare providers are less likely to offer effective treatments to people of color, even after controlling for various factors. These disparities are not solely attributed

to explicit racial biases among physicians but are proposed to involve implicit biases—unconscious negative attitudes about racial groups that affect medical decision-making. Bridges argues that implicit biases can explain the observed disparities in healthcare outcomes for racial minorities. The author highlights experiments indicating that physicians with pro-white implicit biases were more likely to prescribe certain treatments to white patients than to black patients. This evidence supports the idea that implicit biases among healthcare providers contribute to racial disparities in health.

Data Pre-Processing

Data Integration

We integrated four alternative datasets with our original patient dataset by a crucial common attribute across all the datasets: Zip code. To ensure seamless integration across diverse sources, we implemented a uniform preprocessing step involving the extraction of the first three digits of zip codes. Furthermore, to maintain consistency, we renamed this attribute in alignment with our original dataset. Subsequently, we leveraged these first three-digit zip codes and standardized the data to the context of New York State; we enhanced our original dataset with additional layers of information on demographic and environmental factors, namely population count, unique hospital count, park count, and median household income data by three-digit zip code.

By tailoring the data to the geographical scope of New York State, we ensured consistency and introduced valuable insights into the dynamics of hospital resources, recreational spaces, and the economic landscape within the region. The overall objective of this data integration process was to create a more comprehensive data set that encompasses a broader spectrum of external factors. By doing so, we aimed to uncover deeper insights and enhance our understanding of the intricate interplay between healthcare dynamics and socio-economic and environmental determinants.

Removal of Obviously Irrelevant Data/Columns

In the initial phase of data preprocessing, we streamlined the dataset by excluding columns that were deemed irrelevant to our analysis. The "Survey Year" column, bearing a constant value ("2019") for all records, was excluded as it did not contribute any discriminatory information to our analysis. Additionally, the "Number of Hours Worked Each Week" column, loaded with approximately 85% "Not Applicable" and "Unknown" entries, was omitted due to its limited utility and potential skewing effects on our analysis. Instead, we considered the "Employment Status" column, which promised more relevance and could offer valuable insights into the relationship between employment status and patient care utilization.

Processing Null Values

Addressing missing or unknown values is critical for reliable analysis. Various strategies were employed:

- ***Removing Rows with Unknown Values***: For attributes like "Age Group," "Sex," and "Hispanic Ethnicity," where unknown values were infrequent, we opted to eliminate rows with such instances. This approach ensures data quality and mitigates potential noise.
- ***Creating a New Label***: Attributes like ‘Transgender’ and ‘Religious Preference’ with unknown or unprovided values were assigned a new label, "Not Shared," to accurately represent the available information while respecting individual preferences.

- ***Replacing Unknown Values***: Attributes such as "Sexual Orientation," "Race," and "Living Situation" with unknown values were replaced with the label "Other" to maintain data integrity while handling the ambiguity introduced by unknown values.
- ***Numerical Imputation:*** Filled the null values in numerical columns (Population, Unique Hospital Count, Unique Park Count, Median Household Income) with zeros.

Dropping Rows, Categorical Standardization by replacing variations of categorical values with standardized labels, Categorical Replacement by replacing unknown values with another category, Numerical Imputation by filling null values in numerical columns with zeros collectively ensure that the dataset is cleansed of null values and is ready for subsequent analysis.

Correlation Analysis

To identify and address multicollinearity, a correlation analysis using Cramer's V values was conducted on the cleaned dataset. Cramer's V, a measure of association for nominal attributes, aided in pinpointing highly correlated variables (correlation > 0.7). Columns exhibiting strong correlations were removed to mitigate redundancy and potential noise:

- Unknown Chronic Med Condition
- No Chronic Med Conditions
- Unknown Insurance Coverage
- Medicare Insurance
- Other Chronic Med Condition
- Veterans Cash Assistance

Converting nominal/ordinal values to numerical values

In the process of preparing our dataset for analysis, we undertook the transformation of nominal and ordinal values into a numerical format. We adopted systematic mapping to convert these ordinal values into a numeric scale for the "Education Status" column, initially containing ordinal values representing various education levels. The transformation involved assigning numerical values to each education level. Consequently, "Pre-K to fifth grade" was mapped to 1, "Middle school to High school" to 2, "Some College" to 3, and "College or Graduate Degree" to 4. Additionally, we introduced a numerical value, 5, to represent the "Unknown Education" category, indicating instances where the education level is unknown or missing.

In the "Hispanic Ethnicity" column, we simplified the representation by replacing the original values "Yes, Hispanic/Latino" and "No, Not Hispanic/Latino" with binary values, where "Yes" now indicates Hispanic ethnicity, while "No" indicates non-Hispanic ethnicity.

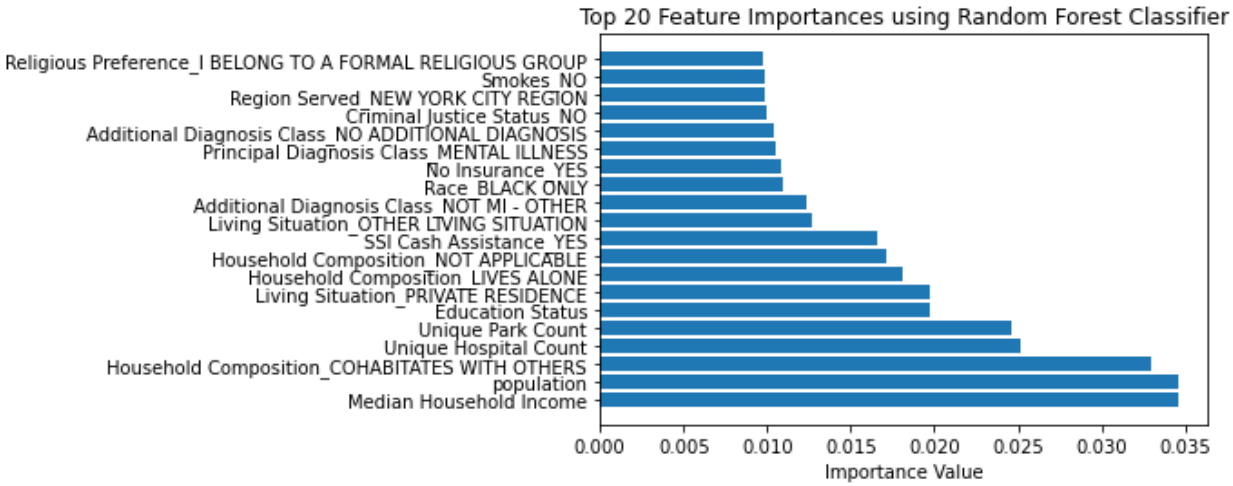
Moreover, to effectively handle nominal attributes, except for "Program Category" (which is our target variable) and "Education Status," we employed a technique known as one-hot encoding, which converts nominal categorical data into a binary format. By incorporating these transformations, our dataset is now structured and compatible with machine learning algorithms, allowing for more meaningful and compelling analysis of the underlying patterns and relationships within the data.

Normalization

Our data preprocessing methodology employed a data transformation technique known as z-score normalization. This process standardizes the scale of numerical features within the dataset by subtracting the mean of each feature and dividing it by its standard deviation. This step is crucial in ensuring that the numerical features are on a comparable scale, preventing any particular feature from influencing the learning process during model training. We applied normalization to the data except for dummy variables. By excluding the dummy variables from this normalization, we preserve their binary nature and the inherent information they provide, striking a balance between standardization and feature preservation in our predictive model for healthcare service utilization.

Feature Selection using Feature Importance's using the Random Forest Model

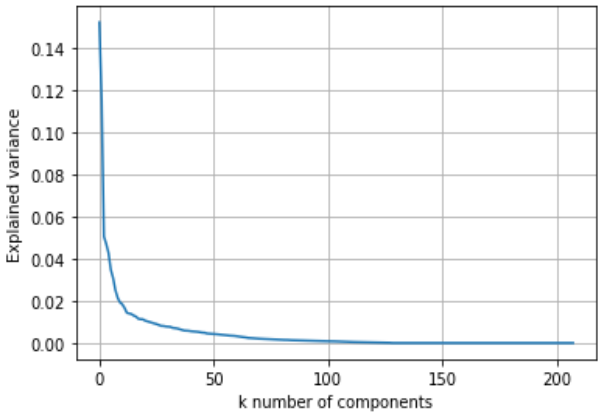
Our analysis employed a feature selection technique based on the importance of features to identify critical factors influencing patient care outcomes. Initially, the original dataset was partitioned into training (70%) and testing (30%) subsets, with a Random Forest classifier being trained on the training data. Subsequent predictions on the test data yielded an overall accuracy of 0.81, a precision of 0.81, and a recall of 0.81. Next, we leveraged the Random Forest classifier to compute their importance values to determine the significance of individual features and eliminate redundant features providing less information. We sorted the importance values to identify the top 20 features. Subsequently, we built a feature selection model by establishing a threshold based on the minimum importance value among these top features. This approach resulted in selecting a reduced set of features that retained predictive efficacy. The resulting Random Forest model, trained on this reduced feature set, exhibited noteworthy performance on the test data with an overall accuracy of 0.77, a precision of 0.74, and a recall of 0.77



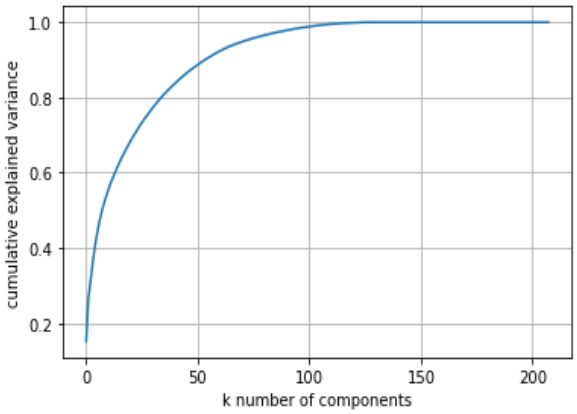
Primary Component Analysis (PCA)

After performing the feature selection technique, we opted for PCA to capture the linear combination of features. Principal Component Analysis (PCA) is a statistical method for dimensionality reduction. It helps simplify the complexity of high-dimensional data while retaining underlying trends and patterns. Our original dataset comprised 208 features, including dummy variables generated through one-hot encoding. In our study, we applied PCA to the original dataset and developed a scree plot to identify an elbow or inflection point to determine the optimal number of components. Following a thorough analysis of the plot, we selected 20 components as the optimal choice. Further analysis, including an alternative plot using cumulative ratios, revealed that these 20 features accounted for approximately 70% of the cumulative explained variance.

Scree Plot



Alternative Plot using Cumulative ratio



Subsequently, we partitioned the original and PCA-transformed data into training (70%) and testing (30%) subsets. The next step involved employing the Random Forest classifier to compare the performance of models built on the original and PCA-transformed datasets. Surprisingly, the PCA-transformed data model exhibited slightly lower performance metrics than the original despite a substantial reduction of almost 90% in features. The overall accuracy of the PCA-transformed model was 0.77, with a precision of 0.76 and a recall of 0.77. In contrast, the original data model demonstrated slightly superior metrics, with an overall accuracy of 0.81, a precision of 0.81, and a recall of 0.81.

Dataset Experimentation for Data Mining Models

In our study, we employed a diverse set of datasets to comprehensively understand and address the challenges present in the original data. These datasets served as crucial inputs for training and evaluating machine learning models. The three primary datasets used in our analysis are:

Original Dataset: This is our baseline dataset, encompassing the original healthcare survey data collected for the analysis. This dataset serves as our starting point, providing a comprehensive view of the data landscape before any modifications or enhancements.

Oversampled Dataset: To address the imbalanced class distribution present in our baseline dataset, we implemented the Naïve Random oversampling technique. This technique involves artificially increasing instances of the minority class, contributing to a more balanced dataset.

Oversampled PCA Dataset: Recognizing the importance of dimensionality reduction, we performed Primary Component Analysis (PCA) and applied the oversampling technique (Naïve Random Over Sampler) on the PCA dataset. This process reduces the number of features while retaining critical information and addresses the class imbalance, potentially enhancing model efficiency and performance.

Rationale for Dataset Experimentation:
Our strategy involves leveraging diverse datasets to address specific challenges in healthcare services data analysis. The oversampled dataset directly mitigates class imbalance, ensuring fair representation of target classes for improved model accuracy. Additionally, the oversampled PCA dataset combines dimensionality reduction through Principal Component Analysis (PCA) with augmented representation of the minority class. This hybrid approach optimizes the feature space, enhancing predictive capabilities while overcoming challenges associated with high-dimensional datasets. Together, these datasets aim to facilitate model generalization to unseen data, making our predictive models more robust and applicable to real-world scenarios.

Data Mining Models and Evaluations

To predict the type of healthcare service a patient might use based on their characteristics and medical history, we have undertaken a thorough analysis employing various data mining models, including Random Forest, Decision tree, Gradient Boosting, Neural networks, Logistic Regression, and Naive Bayesian. Adopting a standardized approach, we trained these models on three distinct datasets.
To establish a fair evaluation framework, we partitioned each dataset into 30% for testing and allocated the remaining 70% for training. Ensuring the equitable distribution of the target variable in both training and testing sets was vital, especially given the presence of imbalanced classes. The use of stratified sampling maintained the ratio of target classes in both subsets, guaranteeing a representative split of the dataset's class distribution. Subsequently, we stored the training and testing subsets for all three datasets—Original, Oversampled, and Oversampled with PCA—ready for deployment in the training and evaluation of machine learning models.
We trained individual models for each dataset, evaluating their performance using test sets without oversampling. This approach allowed us to scrutinize the influence of oversampling and

dimensionality reduction on the models' predictive capabilities within a balanced and high-dimensional feature space.
Based on the use case of predicting the type of patient service, the top three models are selected employ the following criteria:

- ***Accuracy:*** The ratio of correct predictions to the total number of predictions. Accuracy measures how often the model predicts correctly.
- ***Precision:*** The ratio of true positives to the sum of true positives and false positives. Precision measures how accurate the model is when it predicts a positive class.
- ***Recall:*** The ratio of true positives to the sum of true positives and false negatives. It measures how complete the model is when it identifies a positive class.
- ***F1 score:*** The harmonic means of precision and recall. It balances both accuracy and completeness of the model.

We considered Recall as one of our major evaluation metrics for all models, as in healthcare services, identifying all patients who require specific services is critical for delivering appropriate care. Hence, while predicting healthcare services for patients, striking a balance between precision and recall is crucial, as both minimizing unnecessary services and ensuring no service is overlooked are vital considerations in the healthcare domain. While the F1 score is a relevant metric for assessing overall model performance, the specific emphasis varies based on the healthcare service's priorities, including associated costs and consequences for a patient.

Table: Top Performing Models Evaluation metrics

Classification Model	Precision	Recall	Accuracy	Computation Time (in sec)
Random Forest	0.98	0.97	0.97	215.25
Decision Tree	0.96	0.96	0.96	22.84
Gradient Boosting	0.78	0.79	0.79	2322.39

Domain contributions:

Many hospitals are facing economic challenges. With the changing demographics across the country, the demand patterns for their services are changing, while their costs are increasing. Many rural hospitals have shut down in the last few years, despite receiving some COVID-19 funding. Another problem that affects hospital quality is the shortage of nurses and increasing personnel costs.
Our study offers a tool for hospital administrators to plan their services according to the changing population characteristics around their hospital. We use the demographic data and patient characteristics to estimate the number of different types of services that will be needed, such as Outpatient, Residential, Support, Inpatient, and emergency. Each service has different requirements for staff, skills, and costs. Outpatient care is the most utilized and the most cost and

resource-efficient for acute conditions. Inpatient care also addresses acute conditions but is significantly more resourceful and cost intensive. Residential services are most appropriate for chronic conditions that the patient needs long-term help with. Emergency services use a disproportionate number of resources but are critical to saving the lives of patients.
Our model identifies the need for these different services and assists in the appropriate allocation of resources, staffing levels, skills, and cost structures for successful patient care. This can help to optimize the use of resources in the hospital.
Government administrators can also use our tool to identify the service demand and the cost structures for maintaining services. They can support the hospitals that serve the rural populations with more significant needs and reduce the services of hospitals with low utilization. They can also approve new hospitals and services in areas where the population is growing.
By optimizing the patient care services to meet the needs of the population, the right staffing needs can be allocated to the hospitals that serve populations with greater needs.

Methodological Contributions

Throughout the course of our diverse analyses, we made several key observations. We observed that Classification models exhibited optimal performance after normalizing the numeric and non-binary features.
Prior to executing models on the dataset, we tried PCA to reduce dimensionality, capturing the most important features, and employed oversampling to address imbalances in the dataset. We experimented with performing oversampling on PCA data and performing PCA on oversampled data. Oversampling on PCA data was computationally efficient as dimensionality reduction was before oversampling, and then oversampling was focused on principal components. In the case of PCA on the oversampled dataset, PCA was performed on the expanded dataset, which retained all information present in the oversampled data. While evaluating the model performance, we got better results and computational performance for Oversampling on the PCA dataset. Later, we tried model execution on PCA data and applied oversampling post train-validate-test split to preserve the test data integrity. This approach ensured model evaluation on completely unseen data without leakage to test data.
A key step in our model creation and testing involved careful separation of the data into training and a distinct unbiased testing dataset, ensuring that the final model was evaluated with data whose patterns were not utilized to fine-tune model parameters. Additionally, we implemented Train-Validate-Test splits to create three distinct sets of data: one for training the model, a second for evaluating model performance and tuning, and a third for assessing the final generalization error. This methodology guaranteed the separation of test data from both the training process and performance evaluations until the final step. This approach ensured that the model is evaluated against completely unseen data, enhancing the reliability and generalizability of our results.
Stratification played a critical role in maintaining a consistent representation of target classes across test and train subsets for all the datasets. Addressing imbalances in the dataset, particularly concerning the minority class, was achieved through oversampling. Importantly, the oversampling technique was performed post Train-Validate-Test split to preserve the integrity of the test data. This approach prevents knowledge leakage from oversampled test data into the training set, preserving the distinctiveness of the two sets and preventing potential impacts on results with unseen data. Furthermore, to predict the target values using the test data, a deliberate decision was

made to generate predictions using the original test data and PCA test data for the Oversampled model and Oversampled PCA model, respectively. This approach, as opposed to using the respective test datasets, ensured that the test data remained unseen and unaltered by the oversampling technique.
Moreover, cross-validation technique proved valuable for evaluating performance and variance, providing mean and variance scores to estimate variations in the training procedure, and achieving consistency in the model's performance with unseen data. The adjustment of hyperparameters also played a crucial role in fine-tuning our model's behavior for enhanced performance. Simultaneously, the strategic application of regularization techniques played a pivotal role in shaping the learning process of our algorithm. Through careful adjustments to hyperparameters and the incorporation of regularization, we ensured the model's adaptability, effectively preventing overfitting, maintaining the balanced importance of components, and optimizing the learning process.
In this analysis, we applied the k-means clustering algorithm to the preprocessed dataset after scaling the original data and handling outliers. The objective was to identify patterns and group similar data points into clusters for further exploration. Descriptive statistics for each cluster were obtained, considering numerical features such as population, unique hospital count, and unique park count. However, the results showed that the clusters appeared loosely connected, indicating that the algorithm may not effectively capture distinct groupings in the data Despite these efforts, the lack of well-defined clusters led us to explore alternative methodologies for extracting meaningful insights from the dataset.

Conclusions

Summary:

The primary objective of our research was to investigate the feasibility of utilizing machine learning methodologies for accurately predicting the type of healthcare service a patient might require. Our findings suggest that we have successfully demonstrated the potential for achieving this goal, regardless of the availability of a written review from a healthcare professional. Throughout our study, we have shed light on the strengths and limitations of various machine learning models. Notably, Random Forest, Decision Tree, Gradient Boosting and Neural Network models exhibited strong predictive capabilities for outpatient and non-emergency services. However, accurately predicting emergency services remained challenging due to their unpredictable nature. Despite the limitations detailed below, our models hold promise for outpatient services, requiring further refinements for emergency service predictions and broader data validation to enhance real-world applicability.

Limitations:

While our study presents promising results in forecasting healthcare service utilization based on diverse patient attributes, it is vital to acknowledge inherent limitations. Our model encounters challenges in accurately predicting Emergency class, attributed to the unpredictable and urgent nature of such cases, compromising optimal performance in scenarios where precise prediction of emergency healthcare services is essential.
A significant temporal limitation arises from the model's reliance on 2019 data, needing more consideration for post-2019 temporal changes in patient characteristics and healthcare service

utilization trends. Furthermore, our reliance solely on New York healthcare data raises concerns about the model's generalizability to a broader population. Another critical limitation is the need for more access to an external dataset for testing our model's performance on real-world data.
Additionally, the income information in our dataset is aggregated by zip codes, lacking individual patient income details. This constraint hinders the granularity of our analysis, as variations in income at the personal level within a specific zip code are not accounted for. Consequently, our study may not capture the full spectrum of individual income disparities that could influence healthcare service utilization.
Finally, computational challenges are tied to the size of our dataset, particularly with implementing the Support Vector Model. Our SVM model could not execute and generate results on the original and oversampled datasets due to its resource limitations, underscoring the necessity of exploring alternative modeling approaches for large datasets in future research endeavors. Addressing these limitations collectively will refine our predictive model, ensuring its reliability and effectiveness in the dynamic landscape of healthcare service utilization.

Recommendations

Innovative predictive models like Random Forest, Decision Trees, and Gradient Boosting revolutionize healthcare resource allocation. By forecasting patient needs, these models enable precise distribution of staff, facilities, and equipment. This proactive approach ensures resources are optimally available, curbing unnecessary spending and enhancing timely deployment where most critical. Efficient allocation directly translates to heightened operational efficiency and reduced financial strains.
These models also empower proactive healthcare interventions. By foreseeing patient needs, healthcare providers can customize treatment plans and identify at-risk individuals early. This intervention-centric approach prevents complications, curtails chronic conditions, and significantly cuts healthcare costs. It champions a preventive healthcare strategy, prioritizing patient wellness and cost-effective care delivery.
Moreover, leveraging data insights for personalized care plans enriches the patient's experience. Tailoring care based on individual preferences fosters patient engagement, trust, and satisfaction. This engagement encourages active participation in treatment decisions, strengthening adherence and yielding improved health outcomes. Collectively, these strategies reshape healthcare delivery, making it more patient-centric and tailored to individual needs.

Future projects

To ensure the continued relevance and broad applicability of our predictive model, future research endeavors should prioritize updating the dataset to capture the latest trends in patient characteristics and healthcare service utilization. Given that our study is solely based on healthcare data from New York, it becomes imperative to conduct external validation to assess the model's generalizability beyond the limitations of our current dataset. It is advisable to expand the model's testing scope to include data from diverse U.S. states and advanced economies within the European Union (EU), recognizing the significant impact of variations in healthcare systems, cultural factors, and socioeconomic conditions on healthcare service utilization patterns. It aids in exploring healthcare disparities among demographic groups and identifying regions with limited access to healthcare services.

For future studies, a collaborative approach with healthcare institutions across various U.S. states and EU countries should be pursued to facilitate the collection of region-specific data. This collaboration will contribute to a more comprehensive understanding of healthcare service utilization and enhance the model's robustness and applicability on a broader scale.
Additionally, a nuanced approach could involve developing separate models to predict different care needs, allowing for tailored accuracy tuning with each distinct care category. This approach acknowledges the inherent variability in healthcare services and aims to create specialized models that cater to specific care requirements, further refining the predictive capabilities of our model in diverse healthcare settings.

Appendix

Data Dictionary

Field Name and Description	Valid Domain Values	Type	Length	Data Type	Null Ratio
Survey Year: The year in which the survey was conducted. Dates are between 10/21/2019 and 10/27/2019.	• 2019	Number	4	Year	0.00
Target: **Program Category:** The category or type of healthcare program the patient is enrolled in.	• Outpatient • Inpatient • Emergency • Residential • Support	Text	11	Nominal	0.00
Region Served: Represents region where the patients received healthcare services.	• New York City Region • Western Region • Hudson River Region • Central NY Region • Long Island Region	Text	20	Nominal	0.00
Age Group: The age group of the patient.	• Adult • Child • Unknown	Text	7	Nominal	0.00
Sex: Gender of the patient.	• Female • Male • Unknown	Text	7	Binary	0.00
Transgender: Indicates whether the patient identifies as transgender.	• No, Not Transgender • Yes, Transgender • Client didn’t answer • Unknown	Text	20	Boolean	0.08

Sexual Orientation: The patient's sexual orientation.	• Straight or Heterosexual • Bisexual • Lesbian or Gay • Other • Client didn't answer • Unknown	Text	24	Nominal	0.18
Hispanic Ethnicity: Indicates whether the patient identifies as Hispanic or Latino.	• Yes, Hispanic/Latino • No, Not Hispanic/Latino • Unknown	Text	23	Boolean	0.03
Race: The patient's racial background.	• White only • Black only • Multi-Racial • Other • Unknown Race	Text	12	Nominal	0.04
Living Situation: The patient's current living situation or housing status.	• Private Residence • Other Living Situation • Institutional Setting • Unknown	Text	22	Nominal	0.05
Household Composition: Describes the patient's household composition.	• Cohabitates with Others • Lives Alone • Not Applicable • Unknown	Text	23	Nominal	0.23
Preferred Language: The patient's preferred language for communication.	• English • Spanish • Indo-European • Asian and Pacific Island • Afro-Asiatic • All other languages • Unknown	Text	24	Nominal	0.02
Religious Preference: The patient's religious preference.	• I belong to a formal religious group • I do not have a formal religion, nor am I a spiritual person • I consider myself spiritual, but not religious • Data not available	Text	60	Nominal	0.29
Veteran Status: Indicates whether the patient is a military veteran.	• Yes • No • Unknown	Text	7	Boolean	0.04

Employment Status: The patient's current employment status.	• Employed • Unemployed, looking for work • Non-paid/Volunteer • Not in Labor Force: Unemployed and not looking for work • Unknown Employment Status	Text	54	Nominal	0.06
Number Of Hours Worked Each Week: The number of hours the patient works each week.	• 01 – 14 Hours • 15 – 34 Hours • 35 Hours or more • Unknown • Not Applicable	Text	24	Ordinal	0.84
Education Status: The patient's education status.	• Pre-K to Fifth grade • Middle School to High School • Some College • College or Graduate Degree • No Formal Education • Other • Unknown		28	Ordinal	0.11
Special Education Services: Indicates if the patient receives special education services.	• Yes • No • Not Applicable	Text	14	Boolean	0.80
Mental Illness: Indicates if the patient has a mental illness.	• Yes • No • Unknown	Text	7	Boolean	0.01
Intellectual Disability: Indicates if the patient has an intellectual disability.	• Yes • No • Unknown	Text	7	Boolean	0.09
Autism Spectrum: Indicates if the patient is on the autism spectrum.	• Yes • No • Unknown	Text	7	Boolean	0.08
Other Developmental Disability: Indicates if the patient has other developmental disabilities.	• Yes • No • Unknown	Text	7	Boolean	0.08

Alcohol Related Disorder: Indicates if the patient has an alcohol-related disorder.	• Yes • No • Unknown	Text	7	Boolean	0.06
Drug Substance Disorder: Indicates if the patient has a drug substance disorder.	• Yes • No • Unknown	Text	7	Boolean	0.06
Opioid Related Disorder: Indicates if the patient has an opioid-related disorder.	• Yes • No • Unknown	Text	7	Boolean	0.08
Mobility Impairment Disorder: Indicates if the patient has a mobility impairment disorder.	• Yes • No • Unknown	Text	7	Boolean	0.08
Hearing Impairment: Indicates if the patient has a hearing impairment.	• Yes • No • Unknown	Text	7	Boolean	0.08
Visual Impairment: Indicates if the patient has a visual impairment.	• Yes • No • Unknown	Text	7	Boolean	0.08
Speech Impairment: Indicates if the patient has a speech impairment.	• Yes • No • Unknown	Text	7	Boolean	0.08
Hyperlipidemia: Indicates if the patient has hyperlipidemia.	• Yes • No • Unknown	Text	7	Boolean	0.07
High Blood Pressure: Indicates if the patient has high blood pressure.	• Yes • No • Unknown	Text	7	Boolean	0.07
Diabetes: Indicates if the patient has diabetes.	• Yes • No • Unknown	Text	7	Boolean	0.07
Obesity: Indicates if the patient has obesity.	• Yes • No • Unknown	Text	7	Boolean	0.07
Heart Attack: Indicates if the patient has had a heart attack.	• Yes • No • Unknown	Text	7	Boolean	0.07

Stroke: Indicates if the patient has had a stroke.	• Yes • No • Unknown	Text	7	Boolean	0.07
Other Cardiac: Indicates if the patient has other cardiac conditions.	• Yes • No • Unknown	Text	7	Boolean	0.07
Pulmonary Asthma: Indicates if the patient has pulmonary asthma.	• Yes • No • Unknown	Text	7	Boolean	0.07
Alzheimer or Dementia: Indicates if the patient has Alzheimer's disease or dementia.	• Yes • No • Unknown	Text	7	Boolean	0.07
Kidney Disease: Indicates if the patient has kidney disease.	• Yes • No • Unknown	Text	7	Boolean	0.07
Liver Disease: Indicates if the patient has liver disease.	• Yes • No • Unknown	Text	7	Boolean	0.07
Endocrine Condition: Indicates if the patient has an endocrine condition.	• Yes • No • Unknown	Text	7	Boolean	0.07
Neurological Condition: Indicates if the patient has a neurological condition.	• Yes • No • Unknown	Text	7	Boolean	0.07
Traumatic Brain Injury: Indicates if the patient has had a traumatic brain injury.	• Yes • No • Unknown	Text	7	Boolean	0.07
Joint Disease: Indicates if the patient has joint disease.	• Yes • No • Unknown	Text	7	Boolean	0.07
Cancer: Indicates if the patient has been diagnosed with cancer.	• Yes • No • Unknown	Text	7	Boolean	0.07
Other Chronic Med Condition: Indicates if the patient has other chronic medical conditions	• Yes • No • Unknown	Text	7	Boolean	0.07

No Chronic Med Condition: Indicates if the patient has no chronic medical conditions	• Yes • No • Unknown	Text	7	Boolean	0.07
Unknown Chronic Med Condition: Indicates if the patient's chronic medical condition is unknown	• False • True	Text	5	Boolean	0.00
Cannabis Recreational Use: Indicates if the patient uses cannabis recreationally	• Yes • No • Unknown	Text	7	Boolean	0.11
Cannabis Medicinal Use: Indicates if the patient uses cannabis for medicinal purposes	• Yes • No • Unknown	Text	7	Boolean	0.12
Smokes: Indicates if the patient smokes	• Yes • No • Unknown	Text	7	Boolean	0.09
Received Smoking Medication: Indicates if the patient has received smoking cessation medication	• Yes • No • Unknown	Text	7	Boolean	0.09
Received Smoking Counseling: Indicates if the patient has received smoking counseling	• Yes • No • Unknown	Text	7	Boolean	0.09
Serious Mental Illness: Indicates if the patient has a serious mental illness	• Yes • No • Unknown	Text	7	Boolean	0.01
Alcohol 12m Service: Indicates if the patient received alcohol-related services in the past 12 months	• Yes • No • Unknown	Text	7	Boolean	0.08
Opioid 12m Service: Indicates if the patient received opioid-related services in the past 12 months	• Yes • No • Unknown	Text	7	Boolean	0.08
Drug/Substance 12m Service: Indicates if the patient received drug/substance-related services in the past 12 months	• Yes • No • Unknown	Text	7	Boolean	0.09

Principal Diagnosis Class: The principal diagnosis class of the patient	• Mental illness • Not MI – Organic Mental Disorder • Not MI – Developmental Disorders • Not MI – Other • Substance-Related and Addictive Disorders • Unknown	Text	41	Nominal	0.04
Additional Diagnosis Class: Additional diagnosis class of the patient	• Mental illness • Not MI – Organic Mental Disorder • Not MI – Developmental Disorders • Not MI – Other • Substance-Related and Addictive Disorders • No Additional Diagnosis • Unknown	Text	41	Nominal	0.19
SSI Cash Assistance: Indicates if the patient receives Supplemental Security Income (SSI)	• Yes • No • Unknown	Text	7	Boolean	0.14
SSDI Cash Assistance: Indicates if the patient receives Social Security Disability Insurance (SSDI)	• Yes • No • Unknown	Text	7	Boolean	0.14
Veterans Disability Benefits: Indicates if the patient receives veterans' disability benefits	• Yes • No • Unknown	Text	7	Boolean	0.11
Veterans Cash Assistance: Indicates if the patient receives veterans' cash assistance	• Yes • No • Unknown	Text	7	Boolean	0.11
Public Assistance Cash Program: Indicates if the patient receives public assistance cash benefits	• Yes • No • Unknown	Text	7	Boolean	0.15
Other Cash Benefits: Indicates if the patient receives other cash benefits	• Yes • No • Unknown	Text	7	Boolean	0.14

Medicaid and Medicare Insurance: Indicates if the patient has both Medicaid and Medicare insurance	• Yes • No • Unknown	Text	7	Boolean	0.07
No Insurance: Indicates if the patient has no insurance	• Yes • No • Unknown	Text	7	Boolean	0.03
Unknown Insurance Coverage: Indicates if the patient's insurance coverage is unknown	• False • True	Text	5	Boolean	0.00
Medicaid Insurance: Indicates if the patient has Medicaid insurance	• Yes • No • Unknown	Text	7	Boolean	0.04
Medicaid Managed Insurance: Indicates if the patient has managed Medicaid insurance	• Yes • No • Not Applicable • Unknown	Text	14	Boolean	0.40
Medicare Insurance: Indicates if the patient has Medicare insurance	• Yes • No • Unknown	Text	7	Boolean	0.07
Private Insurance: Indicates if the patient has private insurance	• Yes • No • Unknown	Text	7	Boolean	0.07
Child Health Plus Insurance: Indicates if the patient has Child Health Plus insurance	• Yes • No • Unknown	Text	7	Boolean	0.09
Other Insurance: Indicates if the patient has other insurance	• Yes • No • Unknown	Text	7	Boolean	0.08
Criminal Justice Status: Indicates the criminal justice status of the patient	• Yes • No • Unknown	Text	7	Boolean	0.09
Three Digit Residence Zip Code: Three-digit residence zip code of the patient	• 100 – 149 • 777 - Indicates the patient lived in another state in US or another country. • 888 - Indicates the patient was homeless at the time of the survey.	Number	3	Nominal	0.00

	• 999 - Indicates the residential zip code is unknown				
Population: Aggregate population in the 3-digit zip **Simplemaps.com:** https://simplemaps.com/data/us-zips	• Numeric	Number	7	Number	0.10
Hospital Count: Number of hospitals in the 3-digit zip **HealthData.gov:** healthdata.gov/State/Hospital-Profile/gw4x-xyhe	• Numeric	Number	2	Number	0.10
Parks Count: Number of parks in the 3-digit zip **Data.ny.gov:** data.ny.gov/Recreation/State-Park-Facilities-Points-Map/97ur-5r4b	• Numeric	Number	2	Number	0.20
Income: Aggregate Mean Income by Zipcodes **IRS.gov :** https://www.irs.gov/statistics/soi-tax-stats-individual-income-tax-statistics-2019-zip-code-data-soi	• Numeric	Number	7	Number	0.10

References:

de Hond, A., Raven, W., Schinkelshoek, L., Gaakeer, M. I., Ter Avest, E., Sir, O., Lameijer, H., Hessels, R. A., Reijnen, R., De Jonge, E., Steyerberg, E. W., Nickel, C. H., & De Groot, B. (2021). Machine Learning for Developing a Prediction Model of Hospital Admission of Emergency Department Patients: Hype or Hope? International Journal of Medical Informatics, 152, 104496. https://doi.org/10.1016/j.ijmedinf.2021.104496

Shatha Melhem, Ahmad Al-Aiad, Muhammad Saleh Al-Ayyad. (2021). Patient care classification using machine learning techniques. In 2021 12th International Conference on Information and Communication Systems (ICICS) (pp. 1-5). https://www.researchgate.net/publication/352806341_Patient_care_classification_using_machine_learning_techniques

Hong, W. S., Haimovich, A. D., & Taylor, R. A. (2018, July 20). Predicting hospital admission at emergency department triage using machine learning. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0201016/

Golmohammadi, D. (2016). Predicting hospital admissions to reduce emergency department boarding. https://www.sciencedirect.com/science/article/abs/pii/S0925527316302523/

Lo-Ciganic, W.-H., Donohue, J. M., Jones, B. L., Perera, S., Thorpe, J. M., Thorpe, C. T., Marcum, Z. A., & Gellad, W. F. (2016). Trajectories of Diabetes Medication Adherence and Hospitalization Risk: A Retrospective Cohort Study in a Large State Medicaid Program. Journal of General Internal Medicine, 31, 1052–1060. https://link.springer.com/article/10.1007/s11606-016-3747-6

Fingar, K.R., & Reid, L.D. (2018). Diabetes-Related Inpatient Stays, 2018. https://hcup-us.ahrq.gov/reports/statbriefs/sb279-Diabetes-Inpatient-Stays-2018.pdf

Wier, L. M., Witt, E., Burgess, J., & Elixhauser, A. (2008). Hospitalizations Related to Diabetes in Pregnancy. https://hcup-us.ahrq.gov/reports/statbriefs/sb102.pdf

Python Code:

```
====================================
Importing the relevant libaries
====================================
import pandas as pd, numpy as np
from scipy.stats import chi2_contingency
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn import metrics
from imblearn.over_sampling import RandomOverSampler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
import time
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, confusion_matrix
from sklearn.svm import LinearSVC


============================================================
Loading the PCS 2019 dataset into a pandas DataFrame
============================================================
df_pcs = pd.read_csv('Patient_Characteristics_Survey__PCS___2019.csv')
print(df_pcs.shape)
print(df_pcs.info())  # Get the null count & data type of each attribute
print(df_pcs.nunique()) # Get the number of unique values in each attribute


======================================================================
1. Identify other datasets and Merge the files into one dataset
======================================================================
# Merging the population variable using zip code
df_pop = pd.read_csv('uszips.csv')
df_pop.info()
```

```
# Extracting first 3 digits of zip code
df_pop['Three Digit Residence Zip Code'] = df_pop['zip'] // 100

# Filtering New York data
df_pop = df_pop[(df_pop['state_id'] == 'NY') & (df_pop['Three Digit Residence Zip Code'] >
99)]

# Selecting only zip code and Population columns from the dataset
df_pop = df_pop[['Three Digit Residence Zip Code', 'population']].groupby('Three Digit
Residence Zip Code').sum()

df1_merged = pd.merge(df_pcs, df_pop, on='Three Digit Residence Zip Code', how='left')

# Merging the hospital count per zip code variable
df_hsptl = pd.read_csv('Entity_Hospitals_Q1_2023.csv')
df_hsptl.info()

# Extracting first 3 digits of zip code
df_hsptl['Three Digit Residence Zip Code'] = df_hsptl['Zipcode'] // 100

# Calculate the unique hospital count per zip code
hsptl_count = df_hsptl.groupby(['Three Digit Residence Zip
Code'])['Name'].nunique().reset_index()

# Renaming the column Name to Unique hospital Count
hsptl_count.rename(columns={'Name': 'Unique Hospital Count'}, inplace=True)

# Selecting only zip code and Hospital count columns from the dataset
hsptl_count = hsptl_count[['Three Digit Residence Zip Code', 'Unique Hospital Count']]

df2_merged = pd.merge(df1_merged, hsptl_count, on='Three Digit Residence Zip Code',
how='left')

# Merging the Park dataset using zipcode
df_park = pd.read_csv('park_zip.csv', encoding='ISO-8859-1')
df_park.info()

# Extracting first 3 digits of zip code
df_park['Three Digit Residence Zip Code'] = df_park['Zipcode'] // 100

# Calculate the unique park count per zip code
park_count = df_park.groupby(['Three Digit Residence Zip
Code'])['Name'].nunique().reset_index()

# Renaming the column Name to Unique Park Count
park_count.rename(columns={'Name': 'Unique Park Count'}, inplace=True)
```

```
# Selecting only zip code and Park count columns from the dataset
park_count = park_count[['Three Digit Residence Zip Code', 'Unique Park Count']]

df3_merged = pd.merge(df2_merged, park_count, on='Three Digit Residence Zip Code',
how='left')

# Merging the Income dataset using Zip code
df_income = pd.read_csv('Income.csv')
df_income.info()

# Extracting first 3 digits of zip code
df_income['Three Digit Residence Zip Code'] = df_income['ZIPCODE'] // 100

# Filtering New York data
df_income = df_income[(df_income['STATE'] == 'NY') & (df_income['Three Digit Residence
Zip Code'] != 999)]

# Calculate the median household income per zip code
median_income = df_income.groupby(['Three Digit Residence Zip
Code'])['A00100'].median().reset_index()

# Renaming the column Name to Median Household Income
median_income.rename(columns={'A00100': 'Median Household Income'}, inplace=True)

# Selecting only zip code and Median Household Income columns from the dataset
median_income = median_income[['Three Digit Residence Zip Code', 'Median Household
Income']]

df_merged = pd.merge(df3_merged, median_income, on='Three Digit Residence Zip Code',
how='left')

# df_merged.to_csv('PCSdata_merged.csv', index=False)
```

==========

2. Show the number of distinct values and frequency of each nominal and ordinal values

==========

```
print('Distinct values and their frequencies of each nominal and ordinal value\n' )

for col in df_merged.columns:
    print(f"Column Name: {col}")

    max_length = df_merged[col].astype(str).str.len().max()
    print(f"Max Length: {max_length}")
```

```
    distinct_count = df_merged[col].nunique()
    print(f"Distinct Count: {distinct_count}")

    value_counts = df_merged[col].value_counts()
    print(f"Frequencies:\n{value_counts}")

    print("\n")


============================================================
3. Removal of obviously irrelevant data/columns
============================================================
print('Irrelevant Columns:\n 1) Survey Year\n 2) Three Digit Residence Zip Code\n 3) Number
Of Hours Worked Each Week \n')

df_merged.drop(columns=['Survey Year', 'Number Of Hours Worked Each Week','Three Digit
Residence Zip Code'], inplace=True)


==========================
4. Processing Null Values
==========================
print('Processing null values \n')
df_merged = df_merged[df_merged['Age Group'] != "UNKNOWN"]  #dropping 80 rows with
that are neither Adult nor child
df_merged = df_merged[df_merged.Sex != "UNKNOWN"]  #droping 395 rows with that are
neither Male nor Female
df_merged = df_merged[df_merged['Hispanic Ethnicity'] != "UNKNOWN"]  #dropping 5,965
rows with unknown hispanic ethnicity

#Replacing the text with yes or no
df_merged['Transgender'] = df_merged['Transgender'].replace({'YES, TRANSGENDER':
                                    'YES', 'NO, NOT TRANSGENDER': 'NO',\
                                        "CLIENT DIDN'T ANSWER": 'NOT SHARED',\
                                        'UNKNOWN':'NOT SHARED'})

#Replacing Unknown and Client did not answer with OTHER
df_merged['Sexual Orientation'] = df_merged['Sexual Orientation'].replace({'UNKNOWN':
                                    'OTHER',"CLIENT DID NOT ANSWER": 'OTHER'})

#Replacing the Unknown Race with OTHER
df_merged['Race'] = df_merged['Race'].replace({'UNKNOWN RACE':'OTHER'})

#Replacing Unknown living condition  with OTHER
df_merged['Living Situation'] = df_merged['Living Situation'].replace({'UNKNOWN': 'OTHER
                                                                    LIVING SITUATION'})
```

```
# Replace "unknown" with "cohabitates with other" where "Living situation" is "private
#residence"
df_merged.loc[(df_merged['Living Situation'] == 'PRIVATE RESIDENCE') & \
                                    (df_merged['Household Composition'] == 'UNKNOWN'),\
                                    'Household Composition'] = 'COHABITATES WITH OTHERS'

# Replace the remaining "unknown" with "not applicable"
df_merged.loc[df_pcs['Household Composition'] == 'UNKNOWN', 'Household Composition'] =
'NOT APPLICABLE'

#Replacing Unknown Prefered language with OTHER
df_merged['Preferred Language'] = df_merged['Preferred Language'].replace({'UNKNOWN':
'ALL OTHER LANGUAGES'})

#Replacing Data not available with Religion nt shared
df_merged['Religious Preference'] = df_merged['Religious Preference'].replace({'DATA NOT
AVAILABLE': 'RELIGION NOT SHARED'})

#Replacing Unknown veteran satus with NO
df_merged['Veteran Status'] = df_merged['Veteran Status'].replace({'UNKNOWN': 'NO'})

#Replacing Unemployed-looking for work and not looking for work with Unemployed & Non-
#paid/Volunteer with unknown employment status
df_merged['Employment Status'] = df_merged['Employment Status'].replace({'UNEMPLOYED,
                                    LOOKING FOR WORK':'UNEMPLOYED',\
                                    'NOT IN LABOR FORCE:UNEMPLOYED AND NOT
                                LOOKING FOR WORK':'UNEMPLOYED',\
                                    'NON-PAID/VOLUNTEER': 'UNKNOWN
EMPLOYMENT STATUS'})

#Replacing No formal education, other & unknown status with Unknown education
df_merged['Education Status'] = df_merged['Education Status'].replace({'NO FORMAL
                                    EDUCATION': 'UNKNOWN EDUCATION',\
                                    'OTHER': 'UNKNOWN EDUCATION',\
                                    'UNKNOWN': 'UNKNOWN EDUCATION'})

#Replacing Unknown status with NO
df_merged['Special Education Services'] = df_merged['Special Education
Services'].replace({'NOT APPLICABLE': 'NO', 'UNKNOWN': 'NO'})

#Replacing Unknown status with NOT MI - Other
df_merged['Principal Diagnosis Class'] = df_merged['Principal Diagnosis
Class'].replace({'UNKNOWN': 'NOT MI - OTHER'})

#Replacing Unknown status with NOT MI - Other
```

```
df_merged['Additional Diagnosis Class'] = df_merged['Additional Diagnosis
Class'].replace({'UNKNOWN': 'NOT MI - OTHER'})

#Replacing Not Applicable status with NO
df_merged['Medicaid Managed Insurance'] = df_merged['Medicaid Managed
Insurance'].replace({'NOT APPLICABLE': 'NO'})

#Replacing Null values with 0 in 'Population', 'Hospital Count', 'Park Count', and 'Median
Income' columns
df_merged['population'] = df_merged['population'].fillna(0)
df_merged['Unique Hospital Count'] = df_merged['Unique Hospital Count'].fillna(0)
df_merged['Unique Park Count'] = df_merged['Unique Park Count'].fillna(0)
df_merged['Median Household Income'] = df_merged['Median Household Income'].fillna(0)
```

==========

5. Correlation Analysis for nominal data using Cramer's V and chi-square values

==========

```
print('Correlation Analysis \n')

# This function generates the Cramer's V value
def cramer_v(x, y):
    n = len(x)
    ct = pd.crosstab(x, y) # crosstab
    chi2 = chi2_contingency(ct)[0]
    v = np.sqrt(chi2 / (n * (np.min(ct.shape) - 1)))
    return v

# This function returns a dataframe with Cramer's V values.
def cramer_values (df):
    '''Parameters:DataFrame; Returns: DataFrame
    Takes a DataFrame with nominal attributes and returns a DataFrame with
    Cramer's V values between all pairs of those attributes
    Required libraries:
        import pandas as pd, numpy as np
        from scipy.stats import chi2_contingency'''

    cramer_table = pd.DataFrame(columns=['col1','col2','Cramers V'])
    for i in df.columns:
        for j in df.columns:
            if i != j:
                v = cramer_v(df[i],df[j])
                row = pd.DataFrame({'col1':[i],'col2':[j],'Cramers V':[v]})
                cramer_table = pd.concat([cramer_table, row], ignore_index=True)
    return cramer_table.sort_values(by=['Cramers V'],ascending=False)
```

```
# cramer_values
pd.options.display.float_format = '{:.2f}'.format
c_results = cramer_values(df_merged)
c_results.to_csv('Cramer_Values.csv', index=False)

# Dropping highly correlated variables considering Cramer's value
df_merged.drop(columns=['Unknown Chronic Med Condition','No Chronic Med Condition',\
                'Unknown Insurance Coverage','Medicare Insurance',\
                'Other Chronic Med Condition', 'Veterans Cash Assistance'], inplace=True)


============================================================
6. Converting nominal/ordinal values to numerical values
============================================================
# Ordinal to numeric values
df_merged = df_merged.replace({'Education Status':{'PRE-K TO FIFTH GRADE':1,'MIDDLE
SCHOOL TO HIGH SCHOOL':2,\
                        'SOME COLLEGE':3,'COLLEGE OR GRADUATE
DEGREE':4,'UNKNOWN EDUCATION':5}})

# Replacing the text with Yes or No
df_merged['Hispanic Ethnicity'] = df_merged['Hispanic Ethnicity'].replace({'YES,
HISPANIC/LATINO': 'YES',\
                                        'NO, NOT HISPANIC/LATINO': 'NO'})

# Converting Nominal attribute to multiple binary attributes
cols_to_exclude = ['Program Category','Education Status','population','Unique Hospital
Count','Unique Park Count','Median Household Income']
df_merged = pd.get_dummies(df_merged, columns=[col for col in df_merged.columns if col not
in cols_to_exclude])

# Coverting categorical target variable to numerical values
le = LabelEncoder()
df_merged['Program Category'] = le.fit_transform(df_merged['Program Category'])
# df_merged.to_csv('PCS_converted.csv', index=False)

# Get the mapping of class names to numerical values
class_names = list(le.classes_)

# Generating correlation matrix for numerical data
corr_matrix = df_merged.corr()


================================================================
7. Feature Selection using Feature Importances using the Random Forest Model
================================================================
```

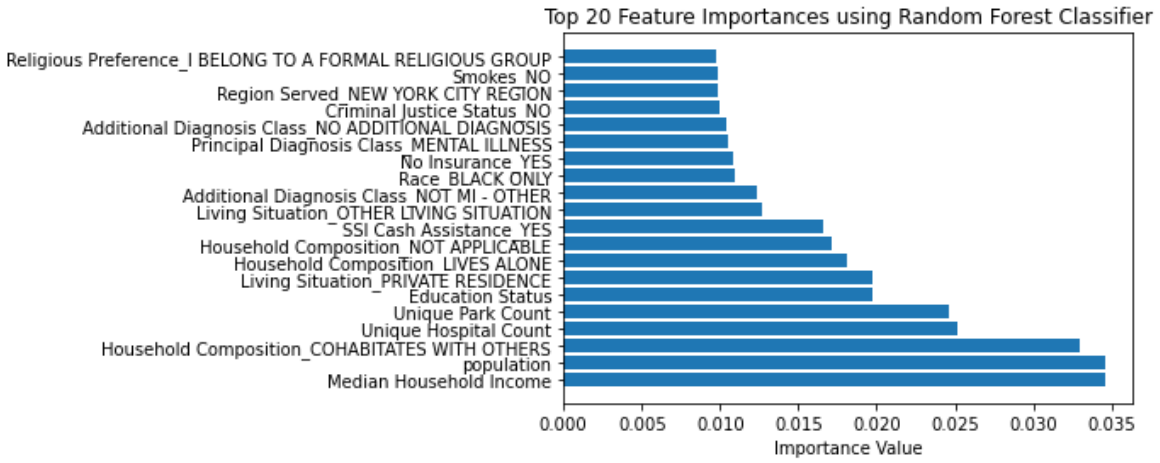
```
# Assigning target and feature variables
X = df_merged.iloc[:, 1: ]
y = df_merged.iloc[:, 0]
print(f'\nShape of the original feature data: {X.shape}')

fn = X.columns[0:]
print(f'Originally, we have {len(fn)} features.')

# Split the data into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =.3,stratify=y)

# Create an instance (object) for classification and build a model.
rfcm = RandomForestClassifier().fit(X_train, y_train)

# Make predictions using the test data
y_pred = rfcm.predict(X_test)

# Show the Classification Report.
print('***Random Forest Model***')
print('\nClassification Report - Original data\n')
print(metrics.classification_report(y_test,y_pred, target_names=class_names))

# Find out the significant features for determining the Patient Care
importances = rfcm.feature_importances_
np.sum(importances)

# Draw a bar chart to see the sorted importance values with feature names.
df_importances = pd.DataFrame(data=importances, index=fn,
                              columns=['importance_value'])
df_importances.sort_values(by = 'importance_value', ascending=False,
                           inplace=True)
top_20_features = df_importances.head(20)

plt.barh(top_20_features.index,top_20_features.importance_value)
plt.xlabel('Importance Value')
plt.title('Top 20 Feature Importances using Random Forest Classifier')
plt.show()

# Set the threshold to the min importance value among the top 20 features
threshold = top_20_features['importance_value'].min()

# Build a model with a threshold based on the importance values of the top 20 features
selector = SelectFromModel(estimator=RandomForestClassifier(),threshold=threshold)
X_reduced = selector.fit_transform(X,y)
selected_TF = selector.get_support()
```

```
print(f'\nBy setting the threshold to be the min imporatnce of the top 20 features,
          {selected_TF.sum()} features are selected from the original feature data.')

# Show the first five names of those selected features.
selected_features = []
for i,j in zip(selected_TF, fn):
    if i:
        selected_features.append(j)
print(f'The first five names of selected features are: \n{selected_features[:5]}')

# Build a model using those reduced number of features.
X_reduced_train, X_reduced_test, y_reduced_train, y_reduced_test \
      = train_test_split(X_reduced,y,test_size =.3, stratify=y)

rfcm2 = RandomForestClassifier().fit(X_reduced_train, y_reduced_train)
y_reduced_pred = rfcm2.predict(X_reduced_test)
print('\nClassification Report - Reduced set of data\n')
print(metrics.classification_report(y_reduced_test,y_reduced_pred,target_names=class_names))
```

Output:

Shape of the original feature data: (189870, 208)

Originally, we have 208 features.



Random Forest Model

Classification Report - Original data

	precision	recall	f1-score	support
EMERGENCY	0.74	0.09	0.16	941
INPATIENT	0.80	0.30	0.43	2643
OUTPATIENT	0.83	0.95	0.88	38794
RESIDENTIAL	0.71	0.70	0.70	8851
SUPPORT	0.87	0.41	0.56	5732
accuracy			0.81	56961
macro avg	0.79	0.49	0.55	56961
weighted avg	0.81	0.81	0.79	56961

By setting the threshold to be the min importance of the top 20 features, 20 features are selected from the original feature data.

The first five names of selected features are:
['Education Status', 'population', 'Unique Hospital Count', 'Unique Park Count', 'Median Household Income']

Classification Report - Reduced set of data

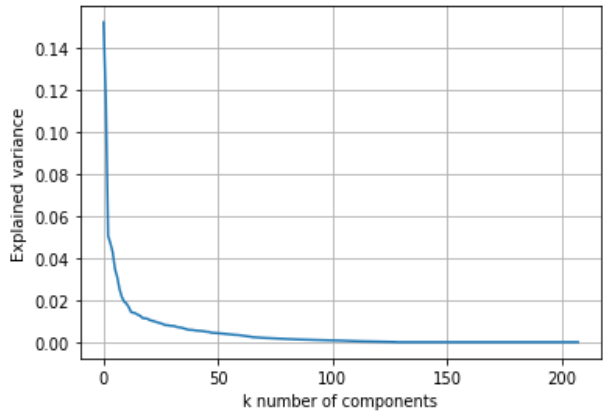
	precision	recall	f1-score	support
EMERGENCY	0.29	0.05	0.09	941
INPATIENT	0.57	0.21	0.31	2643
OUTPATIENT	0.81	0.92	0.86	38794
RESIDENTIAL	0.61	0.59	0.60	8851
SUPPORT	0.67	0.39	0.49	5732
accuracy			0.77	56961
macro avg	0.59	0.43	0.47	56961
weighted avg	0.74	0.77	0.74	56961

```
=====================================
8. PCA (Primary Component Analysis)
=====================================
# z_score normalize the data except the dummy variables
scaler = StandardScaler()
Xn = np.c_[scaler.fit_transform(X.iloc[:,:5].values), X.iloc[:, 5:].values]

# Create an instance PCA and build the model using Xn
pca_prep = PCA().fit(Xn)

pca_prep.explained_variance_  #Eigen Values
pca_prep.explained_variance_ratio_

# Generating a scree plot to find an elbow or an inflection point on the plot
plt.plot(pca_prep.explained_variance_ratio_)
plt.xlabel('k number of components')
plt.ylabel('Explained variance')
plt.grid(True)
plt.show()

# Alternative plot using cumulative ratios
plt.plot(np.cumsum(pca_prep.explained_variance_ratio_))
plt.xlabel('k number of components')
plt.ylabel('cumulative explained variance')
plt.grid(True)
plt.show()

# From scree plot, we choose 30 components
n_pc = 20
pca = PCA(n_components= n_pc).fit(Xn)

# X_pca has now 30 columns of primary components.
Xp = pca.transform(Xn)
print(f'After PCA, we use {pca.n_components_} components.\n')

# Split the data into training and testing subsets for PCA data
Xp_train, Xp_test, yp_train, yp_test = train_test_split(Xp,y,test_size =.3,
                                random_state=1234,stratify=y)

# Create random forest model using the transformed data.
rfcm_pca = RandomForestClassifier().fit(Xp_train, yp_train)

# Predict the target values using the test data.
y_pred_pca = rfcm_pca.predict(Xp_test)

# Generate the Classification report
```

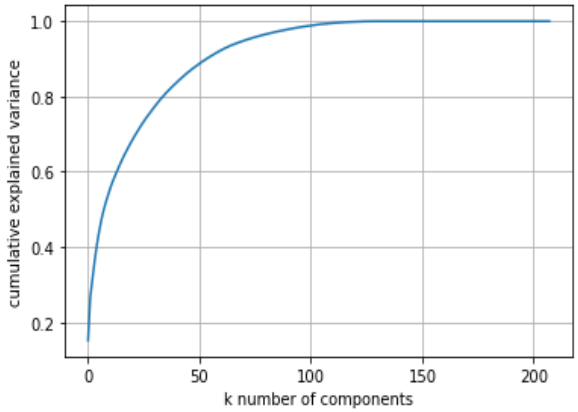
```
print('Classification Report - PCA\n')
print(metrics.classification_report(yp_test,y_pred_pca, target_names=class_names))
```

Output:

Scree Plot



Alternative Plot using Cumulative ratios



After PCA, we use 20 components.

Random Forest Model

Classification Report - PCA

	precision	recall	f1-score	support
EMERGENCY	0.58	0.04	0.07	941
INPATIENT	0.64	0.13	0.22	2643
OUTPATIENT	0.78	0.95	0.86	38794
RESIDENTIAL	0.66	0.55	0.60	8851
SUPPORT	0.82	0.30	0.44	5732
accuracy			0.77	56961
macro avg	0.70	0.39	0.44	56961
weighted avg	0.76	0.77	0.73	56961

==========
9. Naive Random Over Sampling
==========

```
# Create an instance of RandomOverSampler
ros = RandomOverSampler(random_state=1234)
X_rs, y_rs = ros.fit_resample(Xn, y)
Xp_rs, yp_rs = ros.fit_resample(Xp, y)

X_rs.shape
y_rs.shape
Xp_rs.shape
yp_rs.shape
print(f'Over-sampled data: {np.unique(y_rs, return_counts=1)}')
```

Output:

Over-sampled data: (array([0, 1, 2, 3, 4]), array([129314, 129314, 129314, 129314, 129314]))

==========
Split the data into training and testing subsets for Original, Oversampled and PCA with Oversampled data
==========

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =.3,
                                    random_state=1234, stratify=y)

X_rs_train, X_rs_test, y_rs_train, y_rs_test = train_test_split(X_rs,y_rs,test_size =.3,
                                    random_state=1234,stratify=y_rs)

Xp_rs_train, Xp_rs_test, yp_rs_train, yp_rs_test = train_test_split(Xp_rs,yp_rs,test_size =.3,
                                    random_state=1234,stratify=yp_rs)
```

==========
10. Random Forest Classification Model
==========

```
'''Create three models: first using the Original data, second using the Oversampled data
and the third using the Oversampled transformed data'''

rfc_start = time.time()

rfc = RandomForestClassifier().fit(X_train, y_train)
rfc_rs = RandomForestClassifier().fit(X_rs_train, y_rs_train)
rfc_rs_pca = RandomForestClassifier().fit(Xp_rs_train, yp_rs_train)
```

```
# Predict the target values using the test data.
y_pred_rf = rfc.predict(X_test)
y_rs_pred_rf = rfc_rs.predict(X_test)
yp_rs_pred_rf = rfc_rs_pca.predict(Xp_test)

# Generate the Classification report
print('*** Random Forest Classification Model ***\n')
print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_rf, target_names=class_names))

print('\nClassification Report - Oversampled Data\n')
print(metrics.classification_report(y_test,y_rs_pred_rf, target_names=class_names))

print('\nClassification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_rf, target_names=class_names))

rfc_end = time.time()
print(f'Random Forest Classification Model Execution time is {(rfc_end - rfc_start):.2f}
seconds')
```

Output:

*** Random Forest Classification Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.71	0.09	0.16	941
INPATIENT	0.82	0.29	0.43	2643
OUTPATIENT	0.83	0.95	0.88	38794
RESIDENTIAL	0.71	0.70	0.71	8851
SUPPORT	0.85	0.41	0.56	5732
accuracy			0.81	56961
macro avg	0.79	0.49	0.55	56961
weighted avg	0.81	0.81	0.79	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.87	0.52	0.65	941
INPATIENT	0.44	0.69	0.54	2643

OUTPATIENT	0.90	0.89	0.90	38794
RESIDENTIAL	0.64	0.85	0.73	8851
SUPPORT	0.92	0.37	0.52	5732
accuracy			0.82	56961
macro avg	0.75	0.66	0.67	56961
weighted avg	0.84	0.82	0.81	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support
EMERGENCY	0.92	1.00	0.96	941
INPATIENT	0.95	1.00	0.98	2643
OUTPATIENT	1.00	0.96	0.98	38794
RESIDENTIAL	0.90	0.99	0.94	8851
SUPPORT	0.97	1.00	0.98	5732
accuracy			0.97	56961
macro avg	0.95	0.99	0.97	56961
weighted avg	0.98	0.97	0.97	56961

Random Forest Classification Model Execution time is **215.25 seconds**

```
# Regularization and hyperparameters
rfc_clf = RandomForestClassifier(n_estimators = 100,
                         bootstrap = True,
                         max_samples = None, # int, float
                         oob_score = True,
                         criterion = "gini",
                         #splitter = "best", not available here, performs best
                         max_depth = 1,#decision stumps
                         min_samples_split = 2,
                         min_samples_leaf = 1,
                         min_weight_fraction_leaf = 0,
                         max_features = None,
                         max_leaf_nodes = None,
                         random_state = None,
                         min_impurity_decrease = 0.0,
                         class_weight = None,
                         ccp_alpha = 0.0,
                         n_jobs = -1,
                         verbose=1
                        )
rfc_clf.fit(Xp_rs_train, yp_rs_train)
```

```
print("Test score:",rfc_clf.score(Xp_rs_train, yp_rs_train))
print("OOB score:",rfc_clf.oob_score_)
```

Output:

```
Test score: 0.29713099963911943
OOB score: 0.2970433572201887
```

```
==========================================
11. Decision Tree Classification Model
==========================================
'''Create three models: first using the Original data, second using the Oversampled data
and the third using the Oversampled transformed data'''

dtc_start = time.time()

dtc = DecisionTreeClassifier().fit(X_train, y_train)
dtc_rs = DecisionTreeClassifier().fit(X_rs_train, y_rs_train)
dtc_rs_pca = DecisionTreeClassifier().fit(Xp_rs_train, yp_rs_train)

# Predict the target values using the test data
y_pred_dt = dtc.predict(X_test)
y_rs_pred_dt = dtc_rs.predict(X_test)
yp_rs_pred_dt = dtc_rs_pca.predict(Xp_test)

# Generate the Classification report
print('*** Decision Tree Classification Model ***\n')
print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_dt,target_names=class_names))

print('\nClassification Report - Oversampled Data\n')
print(metrics.classification_report(y_test,y_rs_pred_dt,target_names=class_names))

print('\nClassification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_dt, target_names=class_names))

dtc_end = time.time()
print(f'Decision Tree Classification Model Execution time is {(dtc_end - dtc_start):.2f} seconds')
```

Output:

*** Decision Tree Classification Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.14	0.15	0.14	941
INPATIENT	0.33	0.36	0.34	2643
OUTPATIENT	0.83	0.82	0.82	38794
RESIDENTIAL	0.58	0.59	0.58	8851
SUPPORT	0.45	0.48	0.47	5732
accuracy			0.71	56961
macro avg	0.47	0.48	0.47	56961
weighted avg	0.72	0.71	0.72	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.80	1.00	0.89	941
INPATIENT	0.85	1.00	0.92	2643
OUTPATIENT	1.00	0.94	0.97	38794
RESIDENTIAL	0.92	0.99	0.95	8851
SUPPORT	0.89	1.00	0.94	5732
accuracy			0.96	56961
macro avg	0.89	0.99	0.93	56961
weighted avg	0.96	0.96	0.96	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support
EMERGENCY	0.74	1.00	0.85	941
INPATIENT	0.81	1.00	0.90	2643
OUTPATIENT	1.00	0.92	0.95	38794
RESIDENTIAL	0.87	0.99	0.92	8851
SUPPORT	0.84	1.00	0.91	5732
accuracy			0.94	56961
macro avg	0.85	0.98	0.91	56961
weighted avg	0.95	0.94	0.94	56961

Decision Tree Classification Model Execution time is **22.84 seconds**

```
# Hyperparameter Tuning
param_grid = {
```

```
    'criterion': ['gini', 'entropy'],
    'max_depth': [25, 30, 35],
    'min_samples_split': [2,3],
    'min_samples_leaf': [2,3],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'class_weight': ['balanced', None]
}

# Create a decision tree classifier object
dt = DecisionTreeClassifier(random_state=42)

# Create a GridSearchCV object with 10-fold cross-validation
gscv = GridSearchCV(dt, param_grid, cv=2, scoring='accuracy', verbose=1)
gscv_precision = GridSearchCV(dt, param_grid, cv=2, scoring='precision_macro', verbose=1)
gscv_recall = GridSearchCV(dt, param_grid, cv=2, scoring='recall_macro', verbose=1)

# Fit the GridSearchCV object on the training data
gscv.fit(Xp_rs_train, yp_rs_train)

# Print the best parameters and the best score
print('Best parameters:', gscv.best_params_)
print('Best score:', gscv.best_score_)
```

Output:

```
Best parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 30,
'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}
Best score: 0.9070603701603341
```

==========

12. Gradient Boosting Classification Model

==========

```
'''Create three models: first using the Original data, second using the Oversampled data
and the third using the Oversampled transformed data'''

gbc_start = time.time()

gbc = GradientBoostingClassifier().fit(X_train, y_train)
gbc_rs = GradientBoostingClassifier().fit(X_rs_train, y_rs_train)
gbc_rs_pca = GradientBoostingClassifier().fit(Xp_rs_train, yp_rs_train)

# Predict the target values using the test data
y_pred_gb = gbc.predict(X_test)
y_rs_pred_gb = gbc_rs.predict(X_test)
yp_rs_pred_gb = gbc_rs_pca.predict(Xp_test)
```

```
# Show the Classification Report
print('*** Gradient Boosting Classification Model ***\n')

print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_gb, target_names=class_names))

print('\nClassification Report - Oversampled Data\n')
print(metrics.classification_report(y_test,y_rs_pred_gb, target_names=class_names))

print('\nClassification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_gb, target_names=class_names))

gbc_end = time.time()
print(f'Gradient Boosting Classification Model Execution time is {(gbc_end - gbc_start):.2f}
seconds')
```

Output:

*** Gradient Boosting Classification Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.63	0.06	0.11	941
INPATIENT	0.76	0.24	0.37	2643
OUTPATIENT	0.81	0.94	0.87	38794
RESIDENTIAL	0.67	0.62	0.64	8851
SUPPORT	0.77	0.37	0.50	5732
accuracy			0.79	56961
macro avg	0.73	0.45	0.50	56961
weighted avg	0.78	0.79	0.76	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.08	0.50	0.14	941
INPATIENT	0.16	0.64	0.25	2643
OUTPATIENT	0.91	0.58	0.71	38794
RESIDENTIAL	0.51	0.69	0.59	8851
SUPOORT	0.59	0.33	0.42	5732

accuracy			0.57	56961
macro avg	0.45	0.55	0.42	56961
weighted avg	0.77	0.57	0.63	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support
EMERGENCY	0.07	0.49	0.13	941
INPATIENT	0.19	0.46	0.27	2643
OUTPATIENT	0.90	0.57	0.70	38794
RESIDENTIAL	0.48	0.76	0.59	8851
SUPPORT	0.38	0.40	0.39	5732
accuracy			0.57	56961
macro avg	0.41	0.54	0.42	56961
weighted avg	0.74	0.57	0.62	56961

Gradient Boosting Classification Model Execution time is **2322.39** seconds

==========
13. Logistic Regression Model
==========

'''Create three models: first using the Original data, second using the Oversampled data and the third using the Oversampled transformed data'''

```
clr_start = time.time()

clr = LogisticRegression().fit(X_train, y_train)
clr_rs = LogisticRegression().fit(X_rs_train, y_rs_train)
clr_rs_pca = LogisticRegression().fit(Xp_rs_train, yp_rs_train)

# Predict the target values using the test data
y_pred_lr = clr.predict(X_test)
y_rs_pred_lr = clr_rs.predict(X_test)
yp_rs_pred_lr = clr_rs_pca.predict(Xp_test)

# Show the Classification Report
print('*** Logistic Regression Model ***\n')

print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_lr, target_names=class_names,
zero_division=0))

print('\nClassification Report - Oversampled Data\n')
```

```
print(metrics.classification_report(y_test,y_rs_pred_lr, target_names=class_names,
zero_division=0))

print('\nClassification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_lr, target_names=class_names,
zero_division=0))

clr_end = time.time()
print(f'Logistic Regression Model Execution time is {(clr_end - clr_start):.2f} seconds')
```

Output:

*** Logistic Regression Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.00	0.00	0.00	941
INPATIENT	0.00	0.00	0.00	2643
OUTPATIENT	0.68	1.00	0.81	38794
RESIDENTIAL	0.00	0.00	0.00	8851
SUPPORT	0.00	0.00	0.00	5732
accuracy			0.68	56961
macro avg	0.14	0.20	0.16	56961
weighted avg	0.46	0.68	0.55	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.07	0.17	0.10	941
INPATIENT	0.31	0.12	0.17	2643
OUTPATIENT	0.91	0.02	0.03	38794
RESIDENTIAL	0.17	0.99	0.29	8851
SUPPORT	0.65	0.04	0.07	5732
accuracy			0.18	56961
macro avg	0.42	0.27	0.13	56961
weighted avg	0.73	0.18	0.08	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support

EMERGENCY	0.05	0.39	0.09	941
INPATIENT	0.16	0.35	0.22	2643
OUTPATIENT	0.88	0.53	0.66	38794
RESIDENTIAL	0.46	0.69	0.55	8851
SUPPORT	0.27	0.33	0.30	5732
accuracy			0.52	56961
macro avg	0.36	0.46	0.36	56961
weighted avg	0.71	0.52	0.58	56961

Logistic Regression Model Execution time is **126.15** seconds

```
# Hyperparameter tuning using GridSearchCV
clr_grid_start = time.time()

params_clr = {"penalty":["l2"],"C":[0.1,1,10]},
clr_grid =
GridSearchCV(estimator=LogisticRegression(max_iter=1000),param_grid=params_clr,
                scoring = ["accuracy","roc_auc_ovr_weighted","f1_macro"],
                refit="roc_auc_ovr_weighted", #True
                cv = 3,  #If our estimator is classifier automatically do stratified CV
                n_jobs=-1,  #Num CPUs to use for calculation, -1 means all
                verbose = 1,  #Output status updates, higher number-> more messages
                return_train_score=True    #if false our results won't contain training scores
                )
clr_grid.fit(Xn, y)

clr_grid_end = time.time()

print('\n\n **Report**')
print(f'The best estimator: {clr_grid.best_estimator_}')
print(f'The best parameters:\n {clr_grid.best_params_}')
print(f'The best score: {clr_grid.best_score_:.4f}')
print(f'Total run time for GridSearchCV: {(clr_grid_end - clr_grid_start):.2f} seconds')
```

Output:

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
**Report**
The best estimator: LogisticRegression(C=0.1, max_iter=1000)
The best parameters: {'C': 0.1, 'penalty': 'l2'}
The best score: 0.8271
Total run time for GridSearchCV: 200.87 seconds
```

==========

14. Naive Bayesian Classification Model

==========

'''Create three models: first using the Original data, second using the Oversampled data and the third using the Oversampled transformed data'''

```
gnb_start = time.time()

gnb = GaussianNB().fit(X_train, y_train)
gnb_rs = GaussianNB().fit(X_rs_train, y_rs_train)
gnb_rs_pca = GaussianNB().fit(Xp_rs_train, yp_rs_train)

# Calculate the posteriori probabilities
p = gnb.predict_proba(X_test)
p_rs = gnb_rs.predict_proba(X_test)
p_rs_pca = gnb_rs_pca.predict_proba(Xp_test)

# Predict the target values using the test data
y_pred_gnb = gnb.predict(X_test)
y_rs_pred_gnb = gnb_rs.predict(X_test)
yp_rs_pred_gnb = gnb_rs_pca.predict(Xp_test)

# Show the Classification Report
print('*** Naive Bayesian Classification Model ***\n')

print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_gnb,zero_division=0))

print('\nClassification Report - Oversampled Data\n')
print(metrics.classification_report(y_test,y_rs_pred_gnb,zero_division=0))

print('\nClassification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_gnb,zero_division=0))

gnb_end = time.time()
print(f'Gaussian Naive Bayesian Model Execution time is {(gnb_end - gnb_start):.2f} seconds')
```

Output:

*** Naive Bayesian Classification Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.00	0.00	0.00	941
INPATIENT	0.00	0.00	0.00	2643
OUTPATIENT	0.68	1.00	0.81	38794
RESIDENTIAL	0.00	0.00	0.00	8851

SUPPORT	0.00	0.00	0.00	5732
accuracy			0.68	56961
macro avg	0.14	0.20	0.16	56961
weighted avg	0.46	0.68	0.55	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.02	0.29	0.03	941
INPATIENT	0.20	0.11	0.15	2643
OUTPATIENT	0.69	0.68	0.68	38794
RESIDENTIAL	0.07	0.01	0.01	8851
SUPPORT	0.67	0.03	0.06	5732
accuracy			0.48	56961
macro avg	0.33	0.23	0.19	56961
weighted avg	0.55	0.48	0.48	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support
EMERGENCY	0.04	0.33	0.07	941
INPATIENT	0.13	0.45	0.21	2643
OUTPATIENT	0.88	0.52	0.66	38794
RESIDENTIAL	0.41	0.64	0.50	8851
SUPPORT	0.35	0.22	0.27	5732
accuracy			0.50	56961
macro avg	0.36	0.43	0.34	56961
weighted avg	0.71	0.50	0.56	56961

Gaussian Naive Bayesian Model Execution time is **2.43** seconds

```
# Hyperparameter tuning using GridSearchCV
gnb_grid_start = time.time()

param_gnb = {'priors': [[0.2, 0.2, 0.2, 0.2, 0.2], [0.1, 0.15, 0.25, 0.3, 0.2]],
          'var_smoothing': [1e-50, 1e-40]}


# Create a Naive Bayesian classifier object
```

```
gnb_grid = GaussianNB(priors = None,  # class priors, if defined priors won't be set from data
                      var_smoothing = 1e-9)  # added this time maximum feature variance
                                             onto variance for smoothing

# Create a GridSearchCV object with 10-fold cross-validation
gscv = GridSearchCV(gnb_grid, param_gnb, cv=10, scoring='accuracy', verbose=1)
gscv_precision = GridSearchCV(gnb_grid, param_gnb, cv=10, scoring='precision_macro',
verbose=1)
gscv_recall = GridSearchCV(gnb_grid, param_gnb, cv=10, scoring='recall_macro', verbose=1)

# Fit the GridSearchCV object on the training data
gscv.fit(Xp_rs_train, yp_rs_train)

gnb_grid_end = time.time()

print('\n\n **Report**')
print(f'The best estimator: {gscv.best_estimator_}')
print(f'The best parameters:\n {gscv.best_params_}')
print(f'The best score: {gscv.best_score_:.4f}')
print(f'Total run time for GridSearchCV: {(gnb_grid_end - gnb_grid_start):.2f} seconds')
```

Output:

Fitting 10 folds for each of 4 candidates, totalling 40 fits

Report

The best estimator: GaussianNB(priors=[0.2, 0.2, 0.2, 0.2, 0.2], var_smoothing=1e-50)

The best parameters:{'priors': [0.2, 0.2, 0.2, 0.2, 0.2], 'var_smoothing': 1e-50}

The best score: 0.4340

Total run time for GridSearchCV: **4.51** seconds

==========
15. Neural Network Classification Model
==========

 # Hyperparameter tuning using RandomizedSearchCV

```
nnm_rand = MLPClassifier()

params_rand = {
   'hidden_layer_sizes': [(160,), (180,), (200,), (220,), (160, 160), (180, 180), (200, 200), (220,
220)],
   'activation': ['logistic', 'tanh', 'relu'],
   'learning_rate': ['adaptive', 'constant'],
   'learning_rate_init': [0.001, 0.01, 0.1],
   'max_iter': [200, 500, 1000, 2000]
   }

start_rand = time.time()

rand_src = RandomizedSearchCV(estimator= nnm_rand, param_distributions =
params_rand,n_iter=10, random_state=1234, scoring='accuracy')
rand_src.fit(Xn,y)

end_rand = time.time()
print('\n\n **Report**')
print(f'The best estimator: {rand_src.best_estimator_}')
print(f'The best parameters:\n {rand_src.best_params_}')
print(f'The best score: {rand_src.best_score_:.4f}')
print(f'Total run time for RandomizedSearchCV: {(end_rand - start_rand):.2f} seconds')
```

Output:

The best estimator: MLPClassifier(hidden_layer_sizes=(200, 200), learning_rate_init=0.01, max_iter=1000)

The best parameters: {'max_iter': 1000, 'learning_rate_init': 0.01, 'learning_rate': 'constant', 'hidden_layer_sizes': (200, 200), 'activation': 'relu'}

The best score: 0.7576

Total run time for RandomizedSearchCV: **36189.82** seconds

 # Hyperparameter tuning using GridSearchCV

```
nnm = MLPClassifier()
params_grid = {'hidden_layer_sizes':[(20), (30)], 'activation':['logistic','relu','tanh'],
                                                    'max_iter': [4000,5000]}

start_grid = time.time()

grid_src = GridSearchCV(estimator= nnm, param_grid= params_grid)
grid_src.fit(Xn, y)

end_grid = time.time()
print('\n\n **Report**')
print(f'The best estimator: {grid_src.best_estimator_}')
print(f'The best parameters:\n {grid_src.best_params_}')
print(f'The best score: {grid_src.best_score_:.4f}')
print(f'Total run time for GridSearchCV: {(end_grid - start_grid):.2f} seconds')
```

Output:

The best estimator: MLPClassifier(hidden_layer_sizes=30, max_iter=5000)

The best parameters: {'activation': 'relu', 'hidden_layer_sizes': 30, 'max_iter': 5000}

The best score: 0.7665

Total run time for GridSearchCV: **13483.00** seconds

```
nnm_start = time.time()

'''Create three models: first using the Original data, second using the Oversampled data
and the third using the Oversampled transformed data'''

nnm = MLPClassifier(hidden_layer_sizes=(200, 200), activation='relu',
                    max_iter=1000, learning_rate_init = 0.01, learning_rate = 'constant',
                    random_state=1234)
nnm_rs = MLPClassifier(hidden_layer_sizes=(200, 200), activation='relu',
                    max_iter=1000, learning_rate_init = 0.01, learning_rate = 'constant',
                    random_state=1234)
nnm_rs_pca = MLPClassifier(hidden_layer_sizes=(200, 200), activation='relu',
                    max_iter=1000, learning_rate_init = 0.01, learning_rate = 'constant',
                    random_state=1234)

nnm.fit(X_train, y_train)
nnm_rs.fit(X_rs_train, y_rs_train)
nnm_rs_pca.fit(Xp_rs_train, yp_rs_train)
```

```
# Predict the target values using the test data
y_pred_nnm = nnm.predict(X_test)
y_rs_pred_nnm = nnm_rs.predict(X_test)
yp_rs_pred_nnm = nnm_rs_pca.predict(Xp_test)

# Show the Classification Report
print('*** Neural Networks Classification Model ***\n')

print('Classification Report - Original Data\n')
print(metrics.classification_report(y_test,y_pred_nnm,target_names=class_names))

print('Classification Report - Oversampled Data\n')
print(metrics.classification_report(y_test,y_rs_pred_nnm,target_names=class_names))

print('Classification Report - Oversampled PCA Data\n')
print(metrics.classification_report(yp_test,yp_rs_pred_nnm,target_names=class_names))

nnm_end = time.time()
print(f'Neural Networks Classification Model Execution time is {(nnm_end - nnm_start):.2f}
seconds')
```

Output:

*** Neural Networks Classification Model ***

Classification Report - Original Data

	precision	recall	f1-score	support
EMERGENCY	0.49	0.04	0.07	941
INPATIENT	0.52	0.08	0.14	2643
OUTPATIENT	0.69	0.99	0.82	38794
RESIDENTIAL	0.59	0.01	0.02	8851
SUPPORT	0.68	0.08	0.15	5732
accuracy			0.69	56961
macro avg	0.60	0.24	0.24	56961
weighted avg	0.66	0.69	0.58	56961

Classification Report - Oversampled Data

	precision	recall	f1-score	support
EMERGENCY	0.13	0.12	0.13	941
INPATIENT	0.32	0.11	0.16	2643
OUTPATIENT	0.69	0.95	0.80	38794
RESIDENTIAL	0.11	0.02	0.03	8851
SUPPORT	0.54	0.06	0.11	5732
accuracy			0.66	56961
macro avg	0.36	0.25	0.24	56961
weighted avg	0.56	0.66	0.57	56961

Classification Report - Oversampled PCA Data

	precision	recall	f1-score	support
EMERGENCY	0.36	0.96	0.53	941
INPATIENT	0.40	0.82	0.54	2643
OUTPATIENT	0.93	0.62	0.74	38794
RESIDENTIAL	0.52	0.85	0.64	8851
SUPPORT	0.43	0.65	0.52	5732
accuracy			0.67	56961
macro avg	0.53	0.78	0.59	56961
weighted avg	0.78	0.67	0.69	56961

Neural Networks Classification Model Execution time is **1485.34** seconds.

```
================================================
16. K-Fold Cross Validation for Classification – Neural Network
================================================
nnm_kf_start = time.time()

# Use Cross_val_score
nnm_mean_score = np.mean(cross_val_score(nnm,Xn,y,cv=5))
nnm_rs_mean_score = np.mean(cross_val_score(nnm_rs,X_rs,y_rs,cv=5))
nnm_rs_pca_mean_score = np.mean(cross_val_score(nnm_rs_pca,Xp_rs,yp_rs,cv=5))

# Print the scores
print('** Mean Scores (Accuracies) **')
print(f'Mean Score for Neural Network - Original Data: {nnm_mean_score:.4f}')
print(f'Mean Score for Neural Network - Oversampled Data: {nnm_rs_mean_score:.4f}')
print(f'Mean Score for Neural Network - Oversampled PCA Data:
{nnm_rs_pca_mean_score:.4f}')

nnm_kf_end = time.time()
print(f'Neural Network Model with K-Fold Cross Validation Execution time is {(nnm_kf_end -
nnm_kf_start):.2f} seconds')
```

Output:

```
** Mean Scores (Accuracies) **
Mean Score for Neural Network - Original Data: 0.7574
Mean Score for Neural Network - Oversampled Data: 0.7100
Mean Score for Neural Network - Oversampled PCA Data: 0.7612
```

Neural Network Model with K-Fold Cross Validation Execution time is **10170.29** seconds

```
================================
17.Support Vector Machines
================================

# Hyperparameter tuning using RandomizedSearchCV

params_SVM = {'C': (5, 10, 50),  # Regularization parameter
   'kernel': ['linear', 'rbf', 'poly'],  # Kernel type
   'penalty': ["l1", "l2"],  # The norm used in the penalization
   'dual': [True, False],  # Whether to use the dual or primal formulation
   'max_iter': [100000],  # The maximum number of iterations
   'tol': (1e-8, 1e-10, 1e-15)}

# Create an instance of LinearSVC
```

```
svm = LinearSVC()
rand_svm = RandomizedSearchCV(estimator= svm, param_distributions = params_SVM,
                              n_iter=10,cv=2, verbose=0, n_jobs=-1,random_state=42)
rand_svm.fit(Xn,y)

print('\n\n **Report**')
print(f'The best estimator: {rand_svm.best_estimator_}')
print(f'The best parameters:\n {rand_svm.best_params_}')
print(f'The best score: {rand_svm.best_score_:.4f}')
```

Output:

The best estimator: LinearSVC(C=50, dual=False, max_iter=100000, tol=1e-10)

The best parameters:{'tol': 1e-10, 'max_iter': 100000, 'dual': False, 'C': 50}

The best score: 0.7478

```
linear_svm_start = time.time()

# Specific linear SVC implementation that scales better to larger number of data
lin_svc_clf = LinearSVC(penalty="l2",
                        loss="squared_hinge",
                        tol=1e-4,
                        C = 1,
                        max_iter=10000,
                        verbose=True)  #ovr multiclass implementation

lin_svc_clf.fit(Xp_train, yp_train)
lin_svc_clf.coef_, lin_svc_clf.classes_   #weights for each feature
y_pred_svc = lin_svc_clf.predict(Xp_test)

print('*** Support Vector Machines Classification Model ***\n')

lin_svc_cm = confusion_matrix(yp_test, y_pred_svc)
print('Confusion Matrix\n')
print(lin_svc_cm)
print(f'Accuracy from training data: {lin_svc_clf.score(Xp_train, yp_train):.2f}')
print(f'Accuracy from testing data: {lin_svc_clf.score(Xp_test, yp_test):.2f}')

linear_svm_end = time.time()
print(f"Linear SVM RUN time: {(linear_svm_end - linear_svm_start):.2f} seconds")
```

Output:

*** Support Vector Machines Classification Model ***

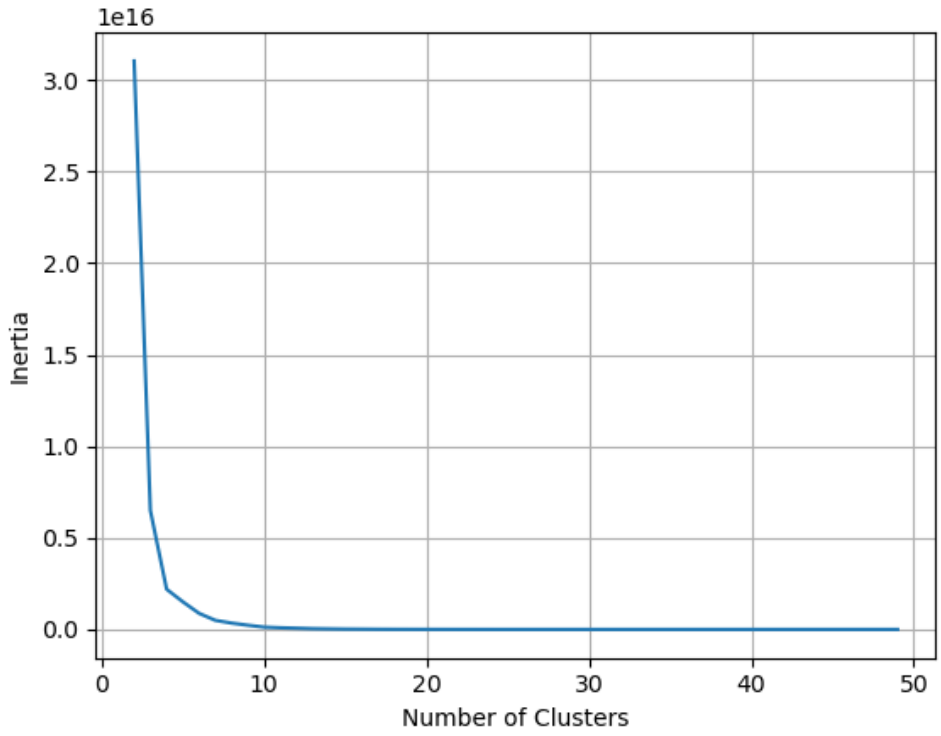
```
Confusion Matrix
[[   0    0   856     75     10    ]
 [   0    0   2070    317    256   ]
 [   0    0   36663   2028   103   ]
 [   0    0   4875    3931   45    ]
 [   0    0   3801    642    1289 ]]

Accuracy from training data: 0.73
Accuracy from testing data: 0.74
Linear SVM RUN time: 228.10 seconds
```

==========

18. K-means

==========

```
#scaled and handled outliers
clf=ECOD()
clf.fit(X)
outliers=clf.predict(X)
X["outliers"]=outliers
#Data without outliers
X_org_no_outlier=X[X"outliers"]==0]
X_org_no_outlier=X_org_no_outlier.drop(["outliers"],axis=1)
#Data with outliers
X_org_with_outlier=X.copy()
X_org_with_outlier=X_org_with_outlier.drop(["outliers"],axis=1)


# Initialize the list for inertia values (sum of squared distances)
inertia_list=[]
#Calculate the inertia for the number of clusters
for i in range(2,50):
    km=KMeans(n_clusters=i,random_state=1234)
    km.fit(X_org_no_outlier)
    inertia_list.append(km.inertia_)
 #draw plot to find elbow
plt.plot(range(2,50),inertia_list)
plt.grid(True)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```




```
#Silhoutte for each cluster
def make_Silhouette_plot(X, n_clusters):
    plt.xlim([-0.1, 1])
    plt.ylim([0, len(X) + (n_clusters + 1) * 10])
    clusterer = KMeans(n_clusters=n_clusters, max_iter = 1000, n_init = 10, init = 'k-means++',
random_state=10)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print(
        "For n_clusters =", n_clusters,
        "The average silhouette_score is :", silhouette_avg,
    )

# Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i
        color = cm.nipy_spectral(float(i) / n_clusters)
        plt.fill_betweenx(
            np.arange(y_lower, y_upper),
            0,
```

```
            ith_cluster_silhouette_values,
            facecolor=color,
            edgecolor=color,
            alpha=0.7,
        )
        plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10
        plt.title(f"The Silhouette Plot for n_cluster = {n_clusters}", fontsize=26)
        plt.xlabel("The silhouette coefficient values", fontsize=24)
        plt.ylabel("Cluster label", fontsize=24)
        plt.axvline(x=silhouette_avg, color="red", linestyle="--")
        plt.yticks([])
        plt.xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

range_n_clusters = list(range(2,8))

for n_clusters in range_n_clusters:
    print(f"N cluster: {n_clusters}")
    make_Silhouette_plot(X_org_no_outlier, n_clusters)
    plt.savefig('Silhouette_plot_{}.png'.format(n_clusters))
    plt.close()
```

```
N cluster: 2
For n_clusters = 2 The average silhouette_score is : 0.7169587649563056
N cluster: 3
For n_clusters = 3 The average silhouette_score is : 0.8131721901490682
N cluster: 4
For n_clusters = 4 The average silhouette_score is : 0.7565695122730917
N cluster: 5
For n_clusters = 5 The average silhouette_score is : 0.7070526338709167
N cluster: 6
For n_clusters = 6 The average silhouette_score is : 0.7134284713947161
N cluster: 7
For n_clusters = 7 The average silhouette_score is : 0.7271153883236817
```

```
km=KMeans(n_clusters=5,
        init='k-means++',
        n_init=10,
        max_iter=100,
        random_state=42)

cluster_predict=km.fit_predict(X_org_no_outlier) #build a model
kmlabels=km.labels_  #Show the cluster number and assign them to a variable
```

```
km.inertia_
km.n_clusters
X_org_no_outlier_cluster=np.column_stack((kmlabels,X_org_no_outlier))   #Add the cluster numbers to the original data

#For further analysis, creating pandas dataframes
colnames=['cluster_no']+X_org_no_outlier.columns.tolist()
df=pd.DataFrame(data=X_org_no_outlier, columns=colnames)
df.info()

#Create dataframes for each cluster
df_org_outlier_cluster_0= df.loc[df.cluster_no==0]
df_org_outlier_cluster_1= df.loc[df.cluster_no==1]
df_org_outlier_cluster_2= df.loc[df.cluster_no==2]
df_org_outlier_cluster_3= df.loc[df.cluster_no==3]
df_org_outlier_cluster_4= df.loc[df.cluster_no==4]
print(f"Davies bouldin score: {davies_bouldin_score(X_org_no_outlier,cluster_predict)}")
print(f"Calinski Score: {calinski_harabasz_score(X_org_no_outlier,cluster_predict)}")
print(f"Silhouette Score: {silhouette_score(X_org_no_outlier,cluster_predict)}")
```

```
Davies bouldin score: 0.3957692977093926
Calinski Score: 3708851.9147377876
Silhouette Score: 0.7070526338709167
```

```
#Finding summary analysis for the k means cluster for Numerical cols
cols_num=['cluster_no','population','Unique Hospital Count','Unique Park Count']
df[cols_num].groupby('cluster_no').agg([min,max,q1,pd.Series.median,q3])
```

	population					Unique Hospital Count					Unique Park Count				
	min	max	q1	median	q3	min	max	q1	median	q3	min	max	q1	median	q3
cluster_no															
0.00	0.00	240814.00	45866.00	138781.00	178513.00	0.00	8.00	1.00	2.00	4.00	0.00	30.00	0.00	1.00	5.00
1.00	2712217.00	2712217.00	2712217.00	2712217.00	2712217.00	17.00	17.00	17.00	17.00	17.00	1.00	1.00	1.00	1.00	1.00
2.00	1205612.00	1581827.00	1473354.00	1473354.00	1581827.00	4.00	22.00	10.00	10.00	22.00	0.00	17.00	1.00	1.00	4.00
3.00	583061.00	823094.00	585663.00	673516.00	823094.00	2.00	10.00	2.00	8.00	10.00	0.00	16.00	0.00	6.00	11.00
4.00	262417.00	493194.00	303120.00	387416.00	479537.00	2.00	6.00	3.00	5.00	5.00	0.00	12.00	2.00	9.00	11.00