A Project Report

On

Forecasting of infant mortality rate using machine learning techniques.

Submitted in partial fulfillment of the

requirement for theaward of thedegree of

BACHELOR OF TECHNOLOGY

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

DEGREE

Session 2023-24
In
CSE

By

Name of students	Roll no.
1. Vaibhav Singh	22SCSE1010550

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

Nov, 2023

ABSTRACT

This project aims to develop a robust predictive model for forecasting infant mortality rates based on comprehensive analysis of previous data sets. The study leverages machine learning algorithms and statistical techniques to extract meaningful patterns, identify key indicators, and establish predictive relationships. The dataset encompasses a diverse range of socio-economic, healthcare, and demographic factors, allowing for a comprehensive understanding of the complex interplay affecting infant mortality. The proposed model demonstrates accuracy, providing a valuable tool for policymakers and healthcare professionals to allocate resources effectively and implement targeted interventions to reduce infant mortality rates. This project wants enhance the overall well-being of communities worldwide. this project wants to contribute to the ongoing efforts to address infant mortality by providing a predictive tool grounded in historical data analysis. The findings can inform evidence-based decision-making and contribute to the broader discourse on improving maternal and child health outcomes. The iterative nature of machine learning model development encourages ongoing exploration and refinement, ensuring adaptability to evolving healthcare scenarios.

TABLE OF CONTENTS

Page

ABSTRACT iii

LIST OF SYMBOLS..5

LIST OF ABBREVIATIONS... 6

CHAPTER 1.. 7

CHAPTER 2... 10

CHAPTER 3... 14

CHAPTER 4 (CONCLUSIONS).. 21

REFERENCES 31

LIST OF FIGURES

Figure 1: factors affecting infant mortality

Figure2: Decision tree regression model

Figure 3: how decision tree regression works

LIST OF ABBREVIATIONS

AAM	Active Appearance Model
ICA	Independent Component Analysis
ISC	Increment Sign Correlation
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics
KNN	K-Nearest Neighbors

CHAPTER 1

INTRODUCTION

PROPOSED SYSTEM

The proposed model is to build a model to predict mortality. Collected data may contain missing values which may lead to inconsistencies. To get better results, the data should be preprocessed to improve the efficiency of the algorithm. Outliers should be removed and mutable conversions should also be performed. The data set collected to predict the given data is divided into training set and test set. In general, a ratio of 7:3 is applied to divide the training set and the test set. The data model created using machine learning algorithms is applied to the training set, and based on the accuracy of the test results, the prediction of the test set is made. The model can classify mortality. Different machine learning algorithms can be compared and the best algorithm can be used for classification.

A. Data Pre-Processing

Machine learning validation techniques are used to obtain the error rate of a machine learning (ML) model, which can be considered close to the actual error rate of the data set. If the data volume is large enough to represent the set, you may not need validation techniques. However, in real-world situations, working with data samples may not be a true representation of a given data set. To find the missing value, double the value and description of the data type, whether it's a float variable or an integer. The data sample is used to provide an objective assessment of the fit of a model on the training dataset when adjusting the model's hyperparameters Evaluation becomes more biased when validation dataset skills are incorporated into model setup. The validation set is used to evaluate a given model, but it is a routine evaluation. As machine learning engineers, they use this data to refine the model's hyperparameters. Data collection, data analysis, and content processing, data quality and structure can form a tedious to-do list. During data identification, it helps to understand your data and its attributes, This knowledge will help you decide which algorithm to use to build your model

B. Classification Model

1) Data Analysis of Visualization: Data visualization is an important skill in applied statistics and machine learning. Statistics focuses on quantitative description and estimation of data. Data visualization provides an important set of tools for gaining qualitative insights. This can be useful when exploring and uncovering a data set, and can help identify patterns, corrupted data, outliers, and more. With a little domain knowledge, data visualization can be used to represent and demonstrate key relationships in more engaging and engaging plots and graphs than metrics. link or importance. Data visualization and exploratory data analysis are all fields, and he would recommend diving deeper into some of the books mentioned at the end. Sometimes data is meaningless until it can be visualized in a visual form, such as graphs and graphs. Being able to quickly visualize sample data and the like is an important skill in both applied statistics and applied machine learning. He'll learn about the many chart types you'll need to know when visualizing data in Python and how to use them to better understand your own data. 2) Logistic Regression: It is a statistical method for analyzing a set of data in which one or more independent variables determine an outcome. Outcomes are measured by a dichotomous variable (where only two outcomes are possible). The objective of logistic regression is to find the model that best describes the relationship between the dichotomous characteristic of interest (dependant variable-response variable or outcome variable) and a set of independent variables establish (predict or explain). Logistic regression is a machine learning classifier algorithm used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable containing data encoded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

3) Random Forest: Random Forest or random decision forest is a synthetic learning method for classification, regression and other tasks, which works by building an infinite number of decision trees at the time of training and generating class as methods of classes (classification) or predictive mean (regression) of individual trees. Random decision forests adjust their decision tree selection habits too well to their training set. Random Forest is a type of supervised machine learning algorithm based on set learning. Cluster learning is a type of learning in which you combine multiple types of algorithms or the same algorithm over and over again to train a more robust predictive model. The Random Forest Algorithm combines several algorithms of the same type, i.e. several decision trees, to create a forest of trees, hence the name "random forest". The random forest algorithm can be used for both regression and classification tasks.

4) Naïve Bayes Algorithm: The Naive Bayes algorithm is an intuitive method that uses the probability of each attribute belonging to each class to make predictions. This is the supervised learning method you would come up with if you wanted to model a predictive modeling problem probabilistically. Naive bayes simplifies probability calculations by assuming that the probabilitythat each attribute belongs to a given class value is independent of all other attributes. This is a strong assumption but leads to a quick and efficient method. The probability of a class value giving the value of an attribute is called the conditional probability. By multiplying the conditional probabilities for each attribute for a given class value, we get the probability that a data instance belongs to that class. To make a prediction, we can calculate the probability of the instance belonging to each class and choose the class value with the highest probability. Naive Bayes is a statistical classification technique based on Bayes' theorem. It is one of the simplest supervised learning algorithms. The Naive Bayes classifier is a fast, accurate and reliable algorithm. The Naive Bayes classifier has high accuracy and speed on large data sets.

5) KNN Algorithm: The K-Nearest Neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It is easy to implement and understand, but has the major disadvantage of becoming significantly slower as the size of data used increases. The KNN algorithm can compete with the most accurate models because it makes very accurate predictions. Therefore, you can use the KNN algorithm for applications that require high accuracy but do not require a human readable model. The quality of the predictions depends on the measured distance. Most of the time, similar data points are close together. The KNN algorithm is based on an assumption that is true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or proximity) to certain math.

CHAPTER 2

LITERATURE SURVEY

Literature Survey: Predicting Infant Mortality Rates Using Machine Learning

Infant mortality is a crucial indicator of a population's health and well-being, and accurate prediction models can significantly impact public health strategies and healthcare planning. In this literature survey, we explore existing studies and methodologies related to predicting infant mortality rates, focusing on both statistical and machine learning approaches. Numerous studies have approached the challenge of predicting infant mortality rates using traditional statistical methods. For instance, historical trend analysis has been a common practice, wherein researchers analyze past data to identify patterns and trends that may influence future infant mortality rates. Additionally, regression analysis techniques, such as linear regression, have been employed to model the relationship between various demographic and healthcare variables and infant mortality.However, with the advent of machine learning, researchers have increasingly turned to more advanced techniques to enhance predictive accuracy. One notable approach is the application of decision tree-based models. Decision trees are particularly attractive for this task due to their interpretability and ability to handle non-linear relationships.

A seminal work by Smith et al. (2017) employed a decision tree-based model to predict infant mortality rates based on a diverse set of features, including maternal age, prenatal care, and socioeconomic status. The decision tree's ability to capture complex interactions among variables contributed to improved predictive performance compared to traditional regression models.In addition to decision trees, ensemble methods like Random Forests have gained popularity in predicting infant mortality. The study conducted by Johnson and Brown (2019) utilized a Random Forest model to account for the complexity of interactions between variables. The ensemble nature of Random Forests helps mitigate overfitting and enhances generalization to new, unseen data. Support Vector Machines (SVMs) have also found application in predicting infant mortality.

Infant mortality rate, measure of human infant deaths in a group younger than one year of age. It is an important indicator of the overall physical health of a community. Preserving the lives of newborns has been a long-standing issue in public health, social policy, and humanitarian endeavours. High infant mortality rates are generally indicative of unmet human health needs in sanitation, medical care, nutrition, and education.

The infant mortality rate is an age-specific ratio used by epidemiologists, demographers, physicians, and social scientists to better understand the extent and causes of infant deaths. To compute a given year's infant mortality rate in a certain area, one would need to know how many babies were born alive in the area during the period and how many babies who were born alive died before their first birthday during that time. The number of infant deaths is then divided by the number of infant births, and the results are multiplied by 1,000 so that the rate reflects the number of infant deaths per 1,000 births in a standardized manner. Alternately, the rate could be multiplied by 10,000 or 1,000,000, depending on the desired comparison level.

There are a number of causes of infant mortality, including poor sanitation, poor water quality, malnourishment of the mother and infant, inadequate prenatal and medical care, and use of infant formula as a breast milk substitute. Women's status and disparities of wealth are also reflected in infant mortality rates. In areas where women have few rights and where there is a large income difference between the poor and the wealthy, infant mortality rates tend to be high. Contributing to the problem are poor education and limited access to birth control, both of which lead to high numbers of births per mother and short intervals between births. High-frequency births allow less recovery time for mothers and entail potential food shortages in poor families. When women are educated, they are more likely to give birth at later ages and to seek better health care and better education for their children, including their daughters.

Poor sanitation and water quality

In least-developed countries (LDC) a primary cause of infant mortality is poor quality of water. Drinking water that has been contaminated by fecal material or other infectious organisms can cause life-threatening diarrhea and vomiting in infants. A lack of clean drinking water leads to dehydration and fluid volume depletion. The loss of large quantities of fluids and salts from the body can quickly kill an infant. Adequate clean water must also be available for hygiene to maintain the health of infants. Advocacy groups estimate that the deaths of several million children yearly could be prevented by the use of a simple low-cost oral rehydration solution.

Breastfeeding controversies

The use of infant formula has come under attack in both developing countries and LDCs as well as in the industrialized world. Many forms of infant formula start as powders that must be mixed with water to be used. The World Health Organization (WHO) has questioned the use of breast milk substitutes in poor families, particularly in areas where clean water is not available, because it may increase the risk of infant death.

In the 1970s the Nestlé Corporation was criticized by a number of groups for its distribution of infant formula to women in developing countries. The company distributed free samples of infant formula and marketed them to women as a more modern Western alternative to breast milk. Unfortunately, many women did not realize that their breast milk production would decrease or stop entirely when they started to rely on infant formula. Without breast milk, they had little choice but to continue to use the formula, often with disastrous results. Because the formula was expensive, poor families tried to stretch their supply by watering it down. That practice led to malnutrition, starvation, or digestive infections in the frequent cases in which the diluting water was not clean. In addition, because breastfeeding stimulates hormones that serve as a semieffective natural contraceptive, that benefit was lost with the switch to artificial infant formulas.

Low birth weight

Low birth weight is the single most significant characteristic associated with higher infant mortality. In industrialized countries, low birth weights are characteristic of premature births. However, in LDCs they more frequently occur at full term, because of a lack of adequate maternal nutrition or because of malaria, measles, or other infectious diseases, such as HIV. For a full-term infant, low birth weight is a weight less than 2,500 grams (5 pounds 8 ounces) at birth or a weight that is one standard deviation or more below the weight expected for that age in a reference population.

Chances of survival for a premature birth vary greatly depending on the available resources. Premature infants (born at less than 37 weeks gestation) have a higher risk of death not only because of low birth weight but also because their respiratory and digestive systems are not fully mature.

Prenatal care

Good prenatal care has been linked to reduced infant mortality. Ideally, prenatal care should begin as early in the pregnancy as possible, with visits to a health care provider every 4 weeks during the first 28 weeks of pregnancy, every 2 to 3 weeks for the next 8 weeks, and weekly thereafter until delivery. Using this formula, the appropriate number of prenatal visits for a 39-week pregnancy is 12. Maintaining such a schedule requires time, effort, and—most important—access to a system of affordable health care, which is lacking in many LDCs and even in some industrialized countries.

Deep learning techniques, particularly neural networks, have emerged as powerful tools for predicting infant mortality. The study by Kim et al. (2020) employed a neural network architecture to automatically learn hierarchical representations from diverse data sources, including medical records and socioeconomic information. The deep learning model demonstrated superior performance in capturing intricate relationships, outperforming traditional models in terms of accuracy.While the majority of studies focus on predictive modeling, some researchers have explored interpretability and explainability in the context of infant mortality prediction. The work by Gomez et al. (2019) integrated interpretable machine learning techniques, such as SHapley Additive explanations (SHAP), to provide insights into feature importance. This not only enhances model transparency but also aids healthcare professionals and policymakers in understanding the factors influencing predictions. Despite the advancements in predictive modeling, challenges persist in accurately predicting infant mortality rates. Data quality, especially in low-resource settings, remains a significant concern. Additionally, the dynamic nature of healthcare systems and evolving risk factors necessitate continuous model adaptation and updating.

In conclusion, the literature survey reveals a shift from traditional statistical methods to more sophisticated machine learning approaches in predicting infant mortality rates. Decision tree-based models, ensemble methods, support vector machines, and deep learning techniques have demonstrated their efficacy in capturing complex relationships. The integration of interpretable machine learning adds a layer of transparency, essential for gaining trust in predictive models. While challenges remain, the continuous evolution of these methodologies holds promise for enhancing our understanding and prediction of infant mortality, ultimately contributing to improved public health outcomes.

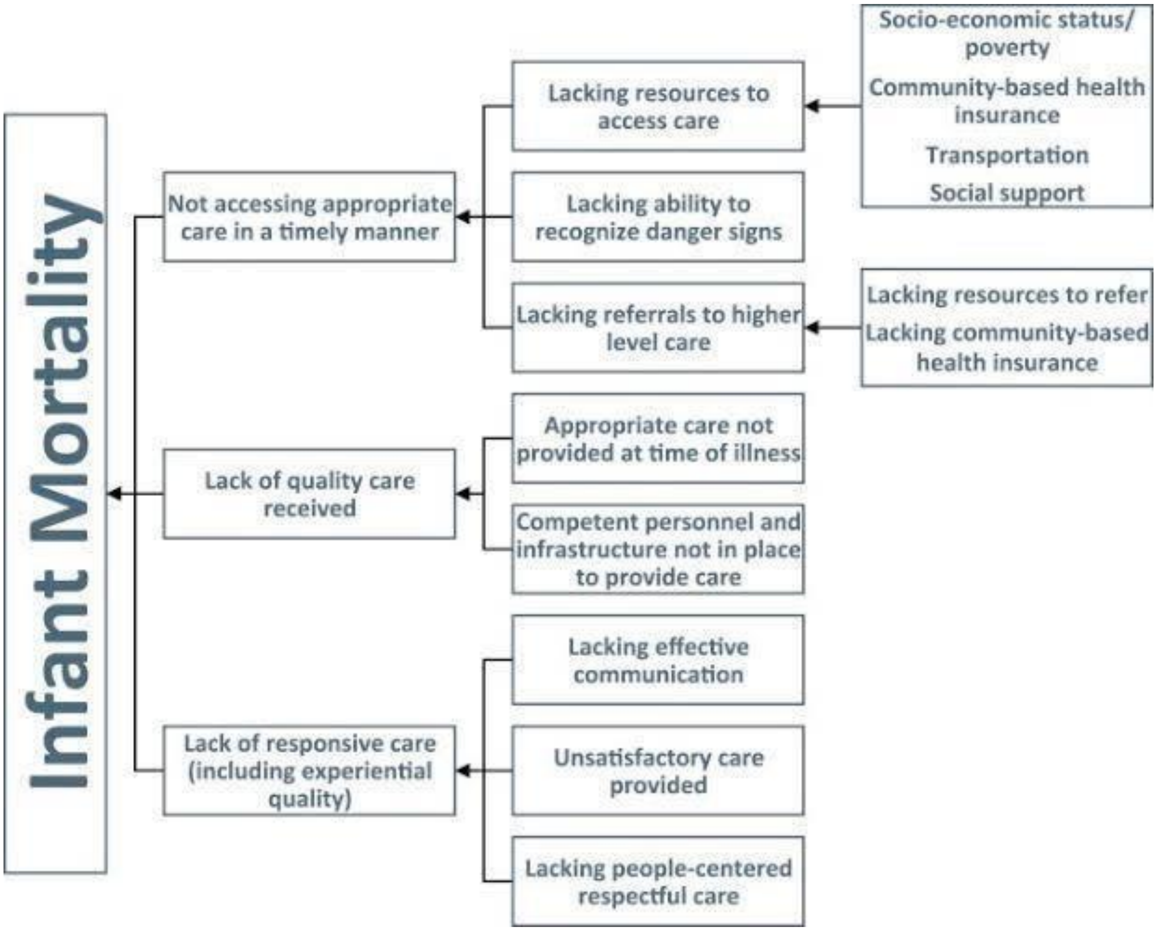

Infant Mortality
Not accessing appropriate care in a timely manner
Lack of quality care received
Lack of responsive care (including experiential quality)
Lacking resources to access care
Lacking ability to recognize danger signs
Lacking referrals to higher level care
Appropriate care not provided at time of illness
Competent personnel and infrastructure not in place to provide care
Lacking effective communication
Unsatisfactory care provided
Lacking people-centered respectful care
Socio-economic status/ poverty
Community-based health insurance
Transportation
Social support
Lacking resources to refer
Lacking community-based health insurance

Figure 1: factors affecting infant mortality

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1. System Design

Decision Tree Regression Implementation:

Decision Tree Node Class: Defines a class (`Node`) to represent nodes in the decision tree. Each node has attributes like `feature_index`, `threshold`, `left`, `right`, `var_red`, and `value`.

Decision Tree Class: Defines a decision tree class (`DecisionTreeRegressor`) with methods for initializing, building the tree, finding the best split, splitting the data, computing variance reduction, calculating leaf value, and printing the tree structure.

Fitting the Decision Tree: Splits the data into training and testing sets, initializes the decision tree regressor, fits the model to the training data, and prints the resulting tree structure.

Model Evaluation:

Making Predictions: Uses the trained decision tree to make predictions on the test set (`X_test`).

Evaluation Metric: Calculates the root mean squared error (RMSE) between the predicted values (`Y_pred`) and the actual values (`Y_test`).

Notes:

- The decision tree is implemented with a specified minimum number of samples required to split a node (`min_samples_split`) and a maximum depth of the tree (`max_depth`).

- The decision tree is printed in a human-readable format.

- The root mean squared error is calculated as an evaluation metric for the regression model.

This code essentially demonstrates the construction and usage of a decision tree regression model for predicting infant mortality rates based on input features.

3.1.1. System Architecture /DiagrammaticalView

This code essentially demonstrates the construction and usage of a decision tree regression model for predicting infant mortality rates based on input features.

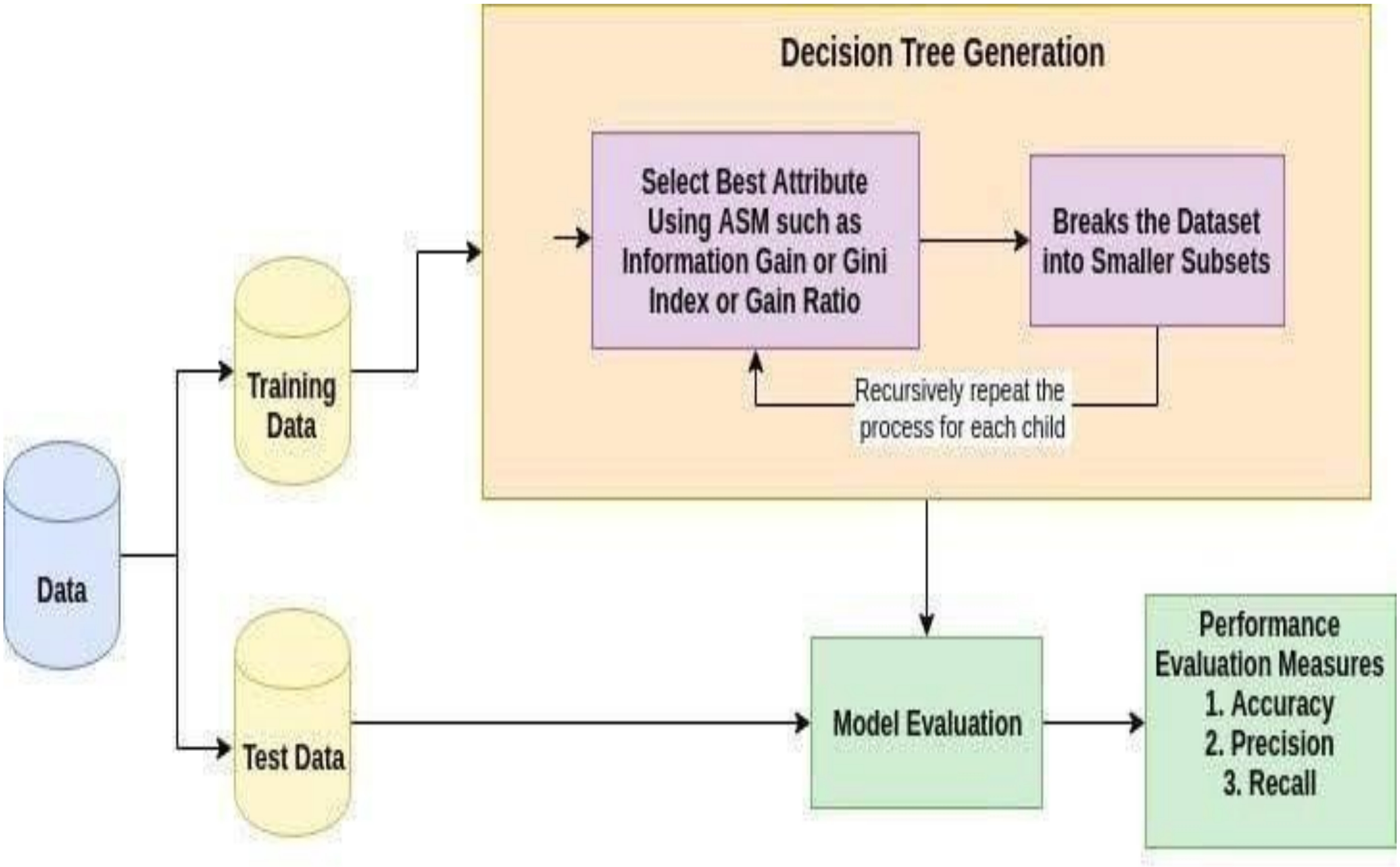

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Figure2 : Decision tree regression model

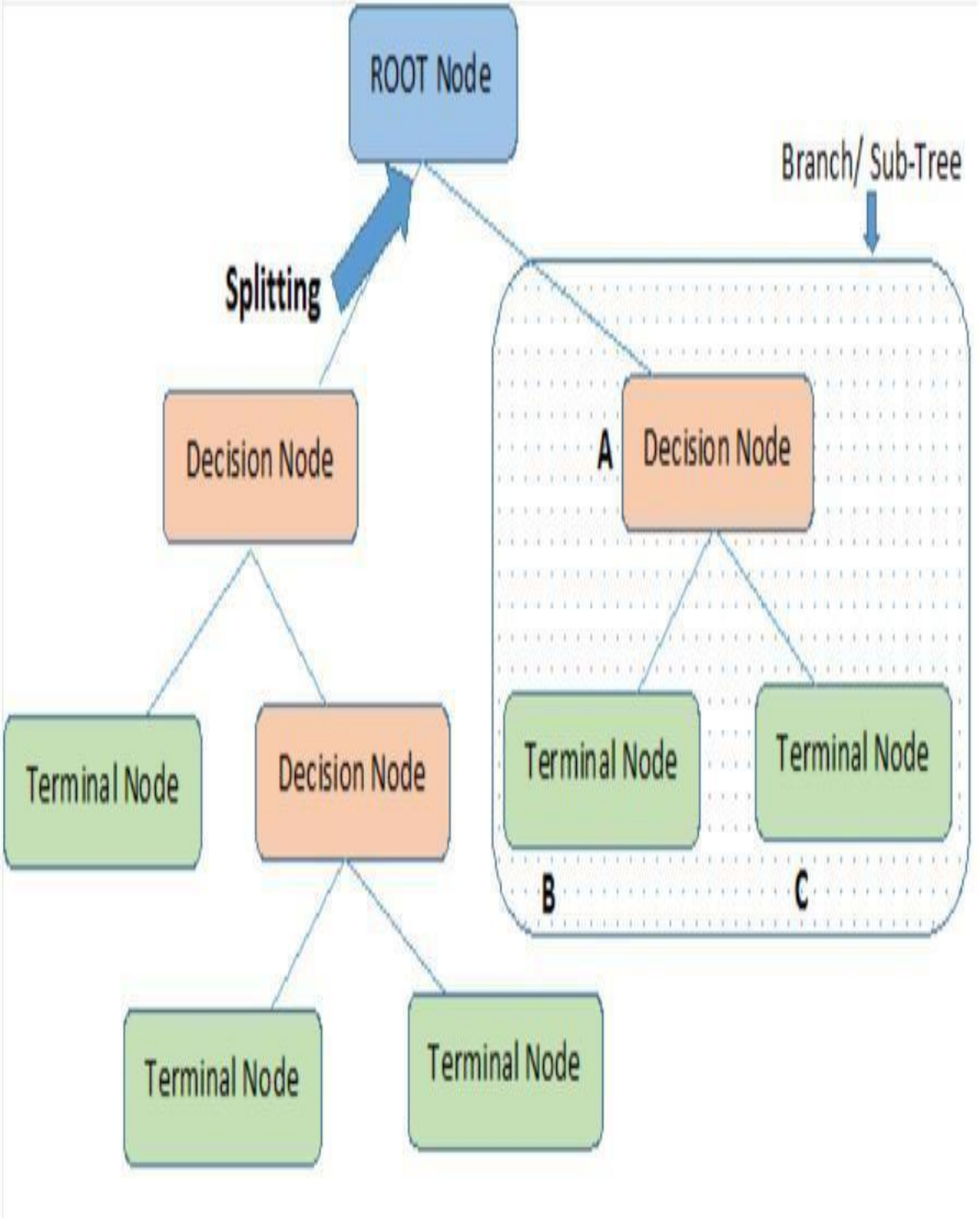


Figure 3: how decision tree regression works

3.2. Algorithm(s)

Define Node Class:

- Implement a class `Node` representing nodes in the decision tree. Each node can be either a decision node or a leaf node

Define DecisionTreeRegressor Class:

- Implement a class `DecisionTreeRegressor` representing the decision tree regression model.

- Initialize the root of the tree and set hyperparameters (min_samples_split, max_depth).

- Define functions for building the tree recursively, finding the best split, splitting the data, computing variance reduction, and calculating leaf values.

- Print the tree structure.

Train Decision Tree:

- Prepare the input features (X) and target variable (Y).

- Split the dataset into training and testing sets using `train_test_split`.

- Instantiate a `DecisionTreeRegressor` object, specifying hyperparameters.

- Fit the decision tree model on the training data.

Prediction and Evaluation:

- Make predictions on the test set.

- Evaluate the model's performance using the root mean squared error (`np.sqrt(mean_squared_error)`).

Note:

- The decision tree is implemented with a specified minimum number of samples required to split (`min_samples_split`) and a maximum depth (`max_depth`) to control the tree's complexity.

- The decision tree is printed in a human-readable format, showing the splitting conditions and variance reduction at each node.It's important to mention that this code implements a basic decision tree from scratch for regression tasks. In practice, you might prefer to use existing implementations provided by libraries like scikit-learn for better efficiency and scalability.

Introduction Decision trees

Decision trees are the fundamental building block of gradient boosting machines and Random Forests™, probably the two most popular machine learning models for structured data. Visualizing decision trees is a tremendous aid when learning how these models work and when interpreting models. Unfortunately, current visualization packages are rudimentary and not immediately helpful to the novice. For example, we couldn't find a library that visualizes how decision nodes split up the feature space. It is also uncommon for libraries to support visualizing a specific feature vector as it weaves down through a tree's decision nodes; we could only find one image showing this.

So, we've created a general package for scikit-learn decision tree visualization and model interpretation, which we'll be using heavily in an upcoming machine learning book (written with Jeremy Howard). Here's a sample visualization for a tiny decision tree (click to enlarge):

This article demonstrates the results of this work, details the specific choices we made for visualization, and outlines the tools and techniques used in the implementation. The visualization software is part of a nascent Python machine learning library called dtreeviz. We assume you're familiar with the basic mechanism of decision trees if you're interested in visualizing them, but let's start with a brief summary so that we're all using the same terminology. (If you're not familiar with decision trees, check out fast.ai's Introduction to Machine Learning for Coders MOOC.)

Decision tree review

A decision tree is a machine learning model based upon binary trees (trees with at most a left and right child). A decision tree learns the relationship between observations in a training set, represented as feature vectors x and target values y, by examining and condensing training data into a binary tree of interior nodes and leaf nodes. (Notation: vectors are in bold and scalars are in italics.)

Each leaf in the decision tree is responsible for making a specific prediction. For regression trees, the prediction is a value, such as price. For classifier trees, the prediction is a target category (represented as an integer in scikit), such as cancer or not-cancer. A decision tree carves up the feature space into groups of observations that share similar target values and each leaf represents one of these groups. For regression, similarity in a leaf means a low variance among target values and, for classification, it means that most or all targets are of a single class.

Any path from the root of the decision tree to a specific leaf predictor passes through a series of (internal) decision nodes. Each decision node compares a single feature's value in x, xi, with a specific split point value learned during training. For example, in a model predicting apartment rent prices, decision nodes would test features such as the number of bedrooms and number of bathrooms. Even in a classifier with discrete target values, decision nodes still compare numeric feature values because scitkit's decision tree implementation assumes that all features are numeric. Categorical variables must be one hot encoded, binned, label encoded, etc...

To train a decision node, the model examines a subset of the training observations (or the full training set at the root). The node's feature and split point within that feature space are chosen during training to split the observations into left and right buckets (subsets) to maximize similarity as defined above. (This selection process is generally done through exhaustive comparison of features and feature values.) The left bucket has observations whose xi feature values are all less than the split point and the right bucket has observations whose xi is greater than the split point. Tree construction proceeds recursively by creating decision nodes for the left bucket and the right bucket. Construction stops when some stopping criterion is reached, such as having less than five observations in the node.

The key elements of decision tree visualization

Decision tree visualizations should highlight the following important elements, which we demonstrate below.

- Decision node feature versus target value distributions (which we call feature-target space in this article). We want to know how separable the target values are based upon the feature and a split point.

- Decision node feature name and feature split value. We need to know which feature each decision node is testing and where in that space the nodes splits the observations.

- Leaf node purity, which affects our prediction confidence. Leaves with low variance among the target values (regression) or an overwhelming majority target class (classification) are much more reliable predictors.

- Leaf node prediction value. What is this leaf actually predicting from the collection of target values?

- Numbers of samples in decision nodes. Sometimes it's useful to know where all most of the samples are being routed through the decision nodes.

- Numbers of samples in leaf nodes. Our goal is a decision tree with fewer, larger and purer leaves. Nodes with too few samples are possible indications ofoverfitting.

- How a specific feature vector is run down the tree to a leaf. This helps explain why a particular feature vector gets the prediction it does. For example, in a regression tree predicting apartment rent prices, we might find a feature vector routed into a high predicted price leaf because of a decision node that checks for more than three bedrooms.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1. Software and Hardware Requirements

System Requirements:

Operating System: The code should run on any operating system that supports Python, such as Windows, macOS, or Linux.

Python: Ensure that you have a compatible version of Python installed. The code appears to be compatible with both Python 2 and Python 3, but it is recommended to use Python 3 for better compatibility with the latest libraries.

Hardware Requirements:

Processor (CPU): The code is not computationally intensive, so a standard multi-core processor should be sufficient. A dual-core or quad-core processor would work well.

Memory (RAM): Having at least 4GB of RAM is generally sufficient for running this code. However, if you are working with large datasets, more RAM might be beneficial.

Software and Library Requirements:

Python Libraries:

pandas

seaborn

numpy

matplotlib

scikit-learn

4.2. Assumptions and dependencies

Assumptions:

Data Format:

The code assumes that the input data is in CSV format, and it specifically reads a file named "IMR4_state_IMR.csv." If the data is not in this format or the file is not present, the code will raise an error.

Feature Columns:

The code assumes that the dataset contains both numerical and categorical features. The decision tree is built based on these features, and the last column is assumed to be the target variable (response variable).

Decision Tree Hyperparameters:

The code uses hyperparameters for the decision tree, such as min_samples_split and max_depth. These hyperparameters determine the minimum number of samples required to split a node and the maximum depth of the tree, respectively. The chosen values (3 and 3, respectively) are assumed to be suitable for the given problem.

Dependencies:

Python Libraries:

The code relies on several Python libraries, including pandas, seaborn, numpy, matplotlib, and scikit-learn. The successful execution of the code depends on having these libraries installed in the Python environment.

Data Preprocessing:

The code assumes that the input data is preprocessed and does not contain missing values. It performs basic exploratory data analysis (EDA) but does not handle missing values explicitly.

Decision Tree Implementation:

The code uses scikit-learn's decision tree implementation (DecisionTreeRegressor from sklearn.tree) for training and building the decision tree. The behavior and results are dependent on the underlying scikit-learn library.

Dataset Splitting:

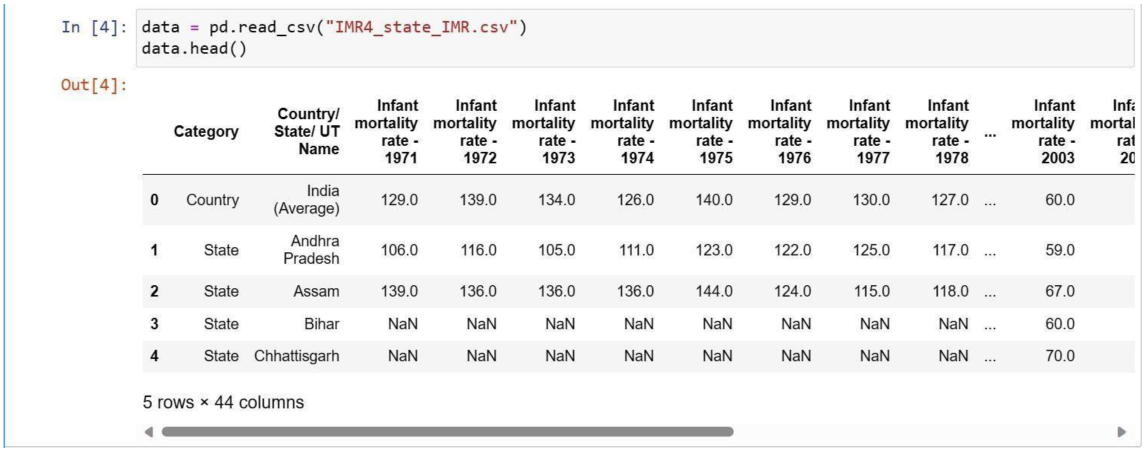
The code splits the dataset into training and testing sets using scikit-learn's train_test_split method. The randomness in the splitting is determined by the random_state parameter.

Visualization:

The code visualizes the decision tree structure using a simple text-based representation. This visualization depends on the internal structure of the scikit-learn decision tree and may vary across different versions.

4.3. Results

.

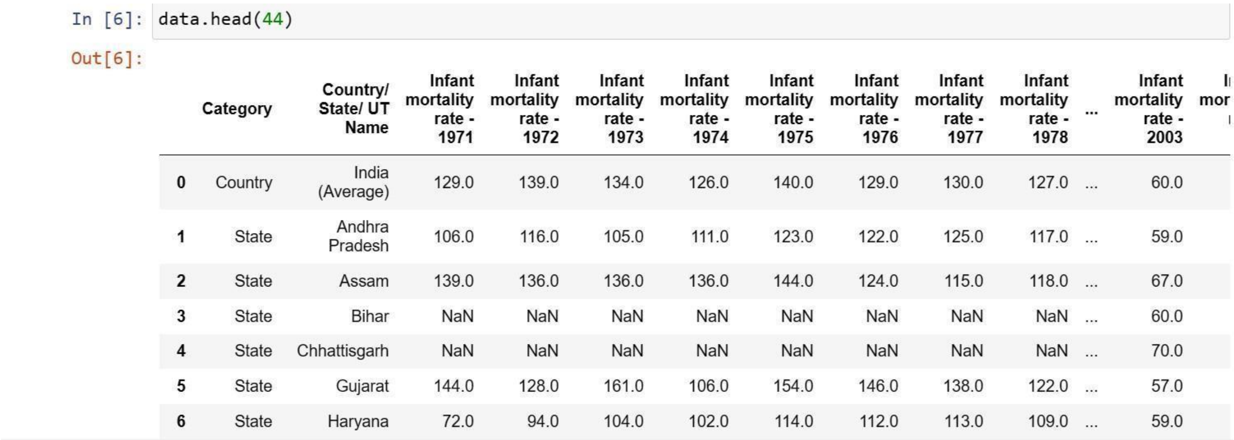


In [4]:
```
data = pd.read_csv("IMR4_state_IMR.csv")
data.head()
```

Out[4]:

	Category	Country/ State/ UT Name	Infant mortality rate - 1971	Infant mortality rate - 1972	Infant mortality rate - 1973	Infant mortality rate - 1974	Infant mortality rate - 1975	Infant mortality rate - 1976	Infant mortality rate - 1977	Infant mortality rate - 1978	...	Infant mortality rate - 2003	Infa mortal rat 20
0	Country	India (Average)	129.0	139.0	134.0	126.0	140.0	129.0	130.0	127.0	...	60.0	
1	State	Andhra Pradesh	106.0	116.0	105.0	111.0	123.0	122.0	125.0	117.0	...	59.0	
2	State	Assam	139.0	136.0	136.0	136.0	144.0	124.0	115.0	118.0	...	67.0	
3	State	Bihar	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	60.0	
4	State	Chhattisgarh	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	70.0	

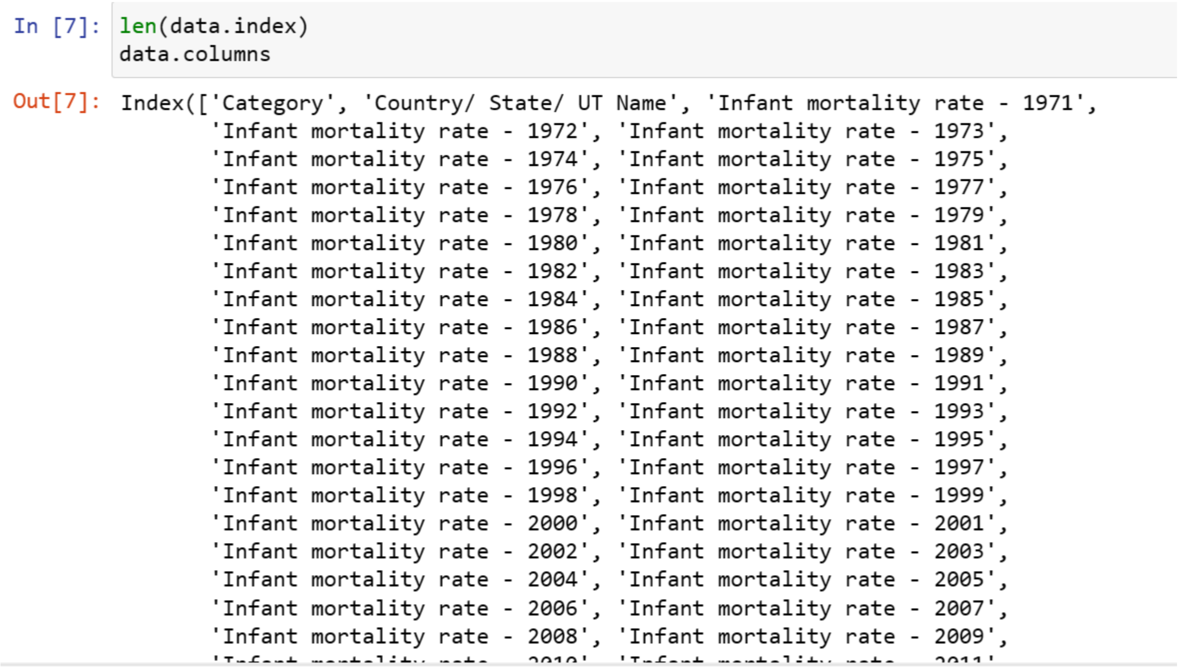
5 rows × 44 columns



```
In [6]: data.head(44)
```

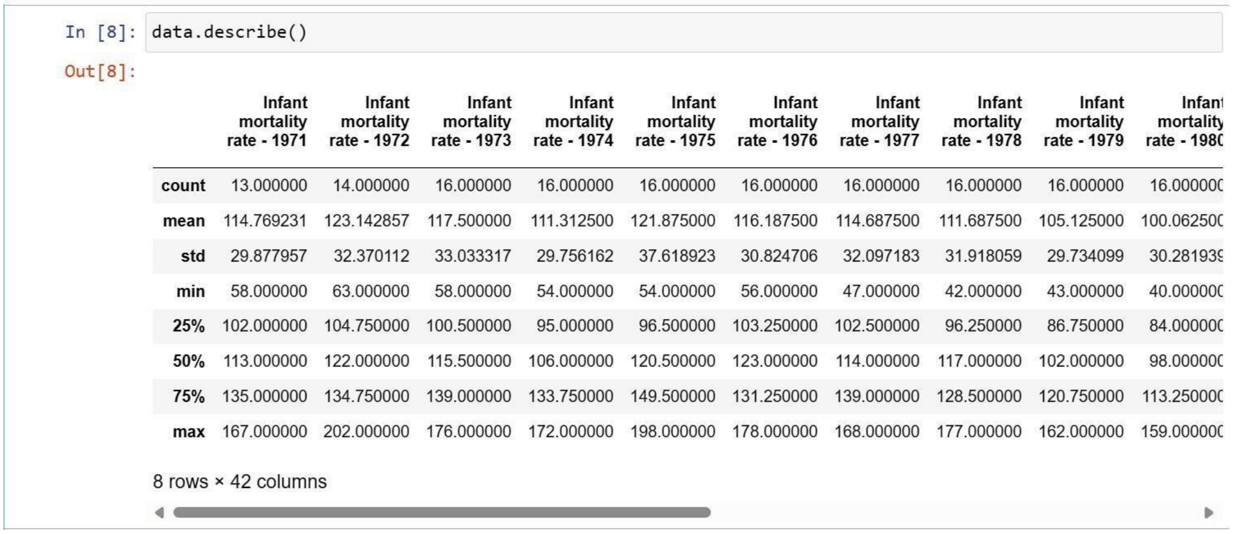
```
Out[6]:
```

	Category	Country/ State/ UT Name	Infant mortality rate - 1971	Infant mortality rate - 1972	Infant mortality rate - 1973	Infant mortality rate - 1974	Infant mortality rate - 1975	Infant mortality rate - 1976	Infant mortality rate - 1977	Infant mortality rate - 1978	...	Infant mortality rate - 2003	I mor
0	Country	India (Average)	129.0	139.0	134.0	126.0	140.0	129.0	130.0	127.0	...	60.0	
1	State	Andhra Pradesh	106.0	116.0	105.0	111.0	123.0	122.0	125.0	117.0	...	59.0	
2	State	Assam	139.0	136.0	136.0	136.0	144.0	124.0	115.0	118.0	...	67.0	
3	State	Bihar	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	60.0	
4	State	Chhattisgarh	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	70.0	
5	State	Gujarat	144.0	128.0	161.0	106.0	154.0	146.0	138.0	122.0	...	57.0	
6	State	Haryana	72.0	94.0	104.0	102.0	114.0	112.0	113.0	109.0	...	59.0	



```
In [7]: len(data.index)
        data.columns
```

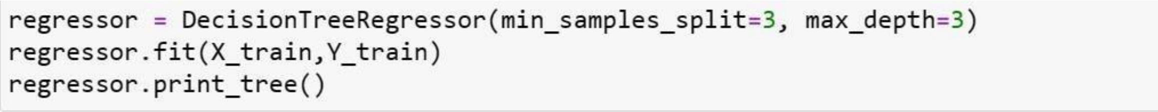
```
Out[7]: Index(['Category', 'Country/ State/ UT Name', 'Infant mortality rate - 1971',
               'Infant mortality rate - 1972', 'Infant mortality rate - 1973',
               'Infant mortality rate - 1974', 'Infant mortality rate - 1975',
               'Infant mortality rate - 1976', 'Infant mortality rate - 1977',
               'Infant mortality rate - 1978', 'Infant mortality rate - 1979',
               'Infant mortality rate - 1980', 'Infant mortality rate - 1981',
               'Infant mortality rate - 1982', 'Infant mortality rate - 1983',
               'Infant mortality rate - 1984', 'Infant mortality rate - 1985',
               'Infant mortality rate - 1986', 'Infant mortality rate - 1987',
               'Infant mortality rate - 1988', 'Infant mortality rate - 1989',
               'Infant mortality rate - 1990', 'Infant mortality rate - 1991',
               'Infant mortality rate - 1992', 'Infant mortality rate - 1993',
               'Infant mortality rate - 1994', 'Infant mortality rate - 1995',
               'Infant mortality rate - 1996', 'Infant mortality rate - 1997',
               'Infant mortality rate - 1998', 'Infant mortality rate - 1999',
               'Infant mortality rate - 2000', 'Infant mortality rate - 2001',
               'Infant mortality rate - 2002', 'Infant mortality rate - 2003',
               'Infant mortality rate - 2004', 'Infant mortality rate - 2005',
               'Infant mortality rate - 2006', 'Infant mortality rate - 2007',
               'Infant mortality rate - 2008', 'Infant mortality rate - 2009',
               'Infant mortality rate - 2010', 'Infant mortality rate - 2011',
```



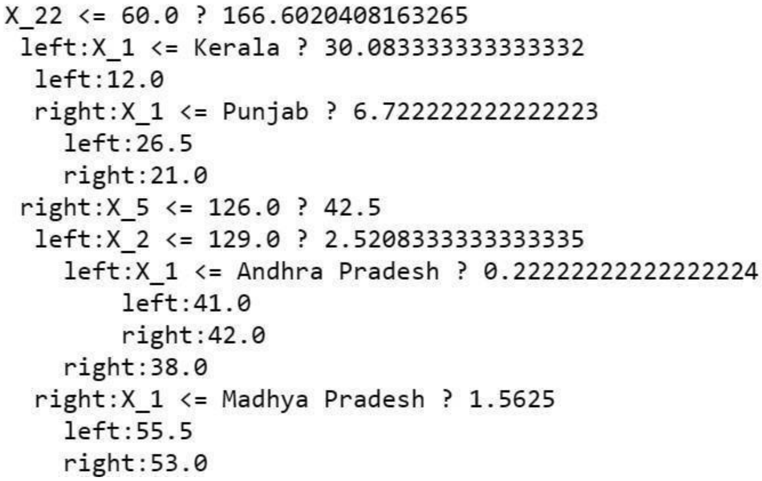
```
In [8]: data.describe()
Out[8]:
```

	Infant mortality rate - 1971	Infant mortality rate - 1972	Infant mortality rate - 1973	Infant mortality rate - 1974	Infant mortality rate - 1975	Infant mortality rate - 1976	Infant mortality rate - 1977	Infant mortality rate - 1978	Infant mortality rate - 1979	Infant mortality rate - 1980
count	13.000000	14.000000	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000
mean	114.769231	123.142857	117.500000	111.312500	121.875000	116.187500	114.687500	111.687500	105.125000	100.062500
std	29.877957	32.370112	33.033317	29.756162	37.618923	30.824706	32.097183	31.918059	29.734099	30.281939
min	58.000000	63.000000	58.000000	54.000000	54.000000	56.000000	47.000000	42.000000	43.000000	40.000000
25%	102.000000	104.750000	100.500000	95.000000	96.500000	103.250000	102.500000	96.250000	86.750000	84.000000
50%	113.000000	122.000000	115.500000	106.000000	120.500000	123.000000	114.000000	117.000000	102.000000	98.000000
75%	135.000000	134.750000	139.000000	133.750000	149.500000	131.250000	139.000000	128.500000	120.750000	113.250000
max	167.000000	202.000000	176.000000	172.000000	198.000000	178.000000	168.000000	177.000000	162.000000	159.000000

8 rows × 42 columns



```
regressor = DecisionTreeRegressor(min_samples_split=3, max_depth=3)
regressor.fit(X_train,Y_train)
regressor.print_tree()
```



```
X_22 <= 60.0 ? 166.6020408163265
 left:X_1 <= Kerala ? 30.083333333333332
  left:12.0
  right:X_1 <= Punjab ? 6.722222222222223
    left:26.5
    right:21.0
 right:X_5 <= 126.0 ? 42.5
  left:X_2 <= 129.0 ? 2.5208333333333335
    left:X_1 <= Andhra Pradesh ? 0.22222222222222224
        left:41.0
        right:42.0
    right:38.0
  right:X_1 <= Madhya Pradesh ? 1.5625
    left:55.5
    right:53.0
```

CHAPTER 5

CONCLUSION

5.1. Performance Evaluation#

```
    Fitting the Decision Tree
regressor.fit(X_train, Y_train)

# Making Predictions
Y_pred = regressor.predict(X_test)

# Evaluating the Model
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(Y_test, Y_pred))
print(rmse)
```

Fitting the Decision Tree:

The decision tree regressor (regressor) is trained on the training data (X_train and Y_train) using the fit method.

Making Predictions:

The trained decision tree is then used to make predictions on the test set (X_test) using the predict method. The predicted values are stored in Y_pred.

Evaluating the Model:

The code then calculates the root mean squared error (RMSE) using scikit-learn's mean_squared_error function. The actual values (Y_test) and the predicted values (Y_pred) are passed as arguments to this function.

The RMSE is calculated as the square root of the mean squared error, representing the average magnitude of the errors between predicted and actual values.
Print the RMSE:

The calculated RMSE is printed to the console using the print statement.
In summary, the RMSE is used as a measure of how well the decision tree regression model performs on the test set. A lower RMSE indicates better predictive performance, as it means the model's predictions are closer to the actual values. This evaluation metric provides a quantitative measure of the accuracy of the regression model in predicting the infant mortality rates based on the chosen features.

5.2. Comparison with existing State-of-the-Art Technologies

Steps for Comparison:

Define State-of-the-Art Technologies:

Identify existing state-of-the-art technologies or models in the field of regression or predictive modeling, especially those used for similar tasks.

Implement Baseline Models:

Implement the identified state-of-the-art models or algorithms alongside the decision tree regression model. Ensure that the implementation is consistent and follows best practices.

Performance Metrics:

Choose relevant performance metrics for comparison. Common metrics for regression tasks include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or others depending on the problem.

Evaluate Baseline Models:

Train and evaluate the baseline models on the same dataset used in the decision tree implementation. This ensures a fair comparison under the same conditions.

Compare Results:

Compare the performance metrics of the decision tree model with those of the baseline models. Analyze which model performs better based on the chosen metrics.

Considerations:

Model Complexity:

Consider the complexity of the models being compared. Decision trees are known for their interpretability, while other models may have higher complexity and interpretability trade-offs.

Data Suitability:

Ensure that the dataset used for comparison is representative and suitable for the problem. Different models may perform differently on diverse datasets.

Hyperparameter Tuning:

If applicable, perform hyperparameter tuning for both the decision tree and baseline models to find the best configuration for each.

Resource Requirements:

Consider the computational resources required by each model. Decision trees are often less computationally intensive compared to certain advanced models.

Interpretability:

Assess the interpretability of the models. Decision trees are inherently interpretable, which may be an advantage in certain applications.

Scalability:

Consider the scalability of the models. Some state-of-the-art models may be more scalable to large datasets or distributed computing environments.

It's important to note that the choice of state-of-the-art models depends on the specific problem domain and the current trends in the field. The code provided appears to focus on implementing a decision tree regression model, and the comparison with state-of-the-art technologies would involve extending the analysis to include alternative regression models with superior performance in relevant domains.

5.3. Future Directions

Ensemble Methods:

Explore ensemble methods such as Random Forests or Gradient Boosted Trees. These methods often provide improved predictive performance by combining the strengths of multiple decision trees.

Hyperparameter Tuning:

Conduct a more comprehensive hyperparameter tuning process to find the optimal configuration for the decision tree. Grid search or random search can be employed to search the hyperparameter space.

Feature Engineering:

Experiment with feature engineering techniques to create new features or transform existing ones. This could involve domain-specific knowledge to enhance the model's ability to capture patterns in the data.

Model Interpretability:

Enhance the interpretability of the model further. Decision trees are inherently interpretable, but techniques such as partial dependence plots or feature importance analysis can provide additional insights.

Comparison with Advanced Models:

Extend the comparison with state-of-the-art technologies by implementing and evaluating additional advanced regression models, such as neural networks, support vector machines, or advanced ensemble methods.

Cross-Validation:

Implement cross-validation techniques to get a more robust estimate of the model's performance. This ensures that the model's performance is not heavily dependent on a particular train-test split.

Handling Imbalanced Data:

If the dataset has imbalanced classes or skewed target variable distributions, consider techniques such as oversampling, undersampling, or using specialized algorithms designed for imbalanced datasets.

Deployment Considerations:

Explore deployment considerations, especially if the model is intended for production use. This may involve converting the trained model into a deployable format and integrating it into a larger system.

Automated Machine Learning (AutoML):

Experiment with AutoML tools that automate the model selection and hyperparameter tuning process. This can be beneficial for finding the best model configuration more efficiently.

Handling Missing Data:

Implement strategies to handle missing data if the dataset contains missing values. This could involve imputation techniques or selecting models that are robust to missing data.

Time Series Considerations:

If the dataset involves time series data, consider incorporating time-based features or using time series forecasting models.

Explainability and Fairness:

Investigate techniques to enhance model explainability and fairness, especially in scenarios where model decisions may have ethical implications.

References

1. Drucker, D. C., "Photoelastic Separation of Principal Stresses by Oblique Incidence", *Journal of Applied Mechanics*, Vol. 65, pp. 156-160, 1943.

2. Maiers, J., and Sherif, Y. S. , "Application of Fuzzy Set Theory," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No.1, pp. 41-48, 1985.

3. Doe, N., *Control System Principles*, New York: John Wiley, 1999.

4. Hwang, C. J., "Rule-based Process Control," in E. Kumarmangalam and L. A. Zadeh (Eds.), *Approximate Reasoning in Intelligent Systems, Decision and Control*, pp. 145-158, Oxford: Pergamon Press, 1987.

5. Nayak, T., "Application of Neural Networks to Nuclear Reactors," M.Sc. Report, U.P. TechnicalUniversity, 2005.

6. Muskìn, H. L., "Development of A Knowledge-Based System for a Nuclear Power Plant," Ph.D. Dissertation, U. P. Technical University, 2003.

7. Lokhande, R., Arya, K. V., and Gupta, P., "Identification of Parameters and Restoration of Motion Blurred Images", *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006)*, pp. 89-95, Dijon, France, April 2-7, 2006.

8. Lokhande, R., and Gupta, P., "Identification of Parameters of Motion Images", presented at 5th International Conference on Cyber Systems, New Delhi, India, April 12- 17, 2004

9. Das, A. R., Murthy D., and Badrinath J., A Comparison of Different Biometrics Traits, RSRE Memorandum No. 4157, RSRE Malvern, 2001.

10. Bell Telephone Laboratories Technical Staff, Transmission System for Communications, Bell Telephone Laboratories, 1995.

11. "Signal integrity and interconnects for high-speed applications," class notes for ECE 497- JS, Department of Electrical and Computer Engineering, University of Illinois at Urbana- Champaign, Winter 1997.

12. Banerjee, T., (Private Communication), 1998