Spring 2018

A MEDICAL PRICE PREDICTION SYSTEM

Anuja Tike
San Jose State University

A MEDICAL PRICE PREDICTION SYSTEM

A project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirement for the Degree

Master of Science

by

Anuja Tike

May, 2018

The Designated Project Committee Approves the Project Titled

A Medical Price Prediction System

by

Anuja Tike

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San José State University

May 2018

ABSTRACT

A Medical Price Prediction System

The health care costs constitute a significant fraction of the U.S. economy. Nearly 20% of the Gross Domestic Product (GDP) is spent on health care. The health spending in the US is the highest among all developed nations in absolute numbers as well as a percentage of the economy. The U.S. government bears a large portion of seniors' health expenditure through its Medicare program. The growing health related expenses combined with the fact that the baby-boomer generation is retiring, and hence they will be eligible for Medicare, puts a great burden on the U.S. exchequer. Therefore, it is essential to contain health related payments through all means possible.

In this work, we will develop a medical price prediction system using machine learning algorithms which will aid in steering patients to cost effective providers and thereby curb health spending. The policymakers can also use the tool to better understand which providers are relatively expensive and take punitive actions if necessary. The prediction of the medical price will be done using implementing Random Forest Regression algorithm in machine learning. Additionally, we plan to include the experiments on the same data with other machine learning models such as Gradient Boosted Trees and Linear Regression and compare results. The findings from these experiments will also be included.

***Key terms*- Health care, GDP, medical price prediction system, Random Forest Regression, machine learning, Gradient Boosted Trees, Linear Regression.**

ACKNOWLEDGEMENTS

I want to sincerely thank my project advisor Dr. Sami Khuri, for his continuous support and encouragement throughout this project. I would also like to extend my thanks to my committee members, Dr. Sanket Tavarageri and Dr. Robert Chun for their time and support.

My special thanks to Dr. Sanket Tavarageri for his guidance and valuable suggestions during the course of this project.

Lastly, I would like to thank my parents for their support at all times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The health care costs in the U.S. account for 17.80% of the national output [2], which is the highest among all developed nations. The U.S. government runs Medicare insurance program for seniors and bears nearly half of seniors' total health care spending [3], [4], [5]. The number of seniors is set to expand dramatically with the ongoing and impending retirement of the baby-boomer generation, which is expected to swell the ranks of Medicare beneficiaries a lot more in the coming years. Consequently, the Medicare outlay for the government has to increase and that adds a great strain on the budget. Therefore, ways and means to control health care costs and thereby, slow down the rise of Medicare spending have assumed great significance. One of the key components to restrain the rise in health care costs is access to an accurate medical price prediction system. That is, if patients have accurate information on medical pricing such as, a certain medical procedure costs dollar amount X at hospital A, the same procedure comes with a price tag of Y at hospital B then they have the opportunity to choose the provider that costs them less.

In this project our goal is to predict medical prices based on the data we have in hand. In the first few chapters of this report, we will compare the work of various authors in the area of price prediction and we will also provide the information in detail, about some of the techniques used in health care domain to predict the health care prices. Later, we will propose the design of a new system which will use Medicare payment datasets. The proposed system can be called as a medical price prediction system. Such a system will be useful for patients, and government officials alike. Patients can use the price prediction tool to choose the most cost-efficient providers. It can be used by Medicare administrators to forecast expenditure for future months and years and plan the budget accordingly. Additionally, high cost providers can be identified using the system. Deeper investigations may be subsequently carried out involving high charging providers and punitive measures may be initiated against them when necessary [2]. We will build the proposed system by implementing two machine learning algorithms from the scratch. The first algorithm is, Regression Tree and the second one is, Random Forest Regression. While implementing the Random Forest regression algorithm, we will make use of Regression Trees algorithm to build base trees. In the end, we will also include the results from other two machine learning algorithms which are, Linear Regression and Gradient Boosted Decision Trees. These two algorithms we will not implement from the scratch but we will use in-built libraries of them from python's scikit-learn tool kit [6] to build our machine learning model based on the dataset we have.

Fig. 1 shows the organization of this project. The following are some of the questions which will be answered in this project report: What are the different approaches people have used to predict various types of prices? Which machine learning techniques can be used in this area? What is Classification and Regression in machine learning? How will the new proposed system work?

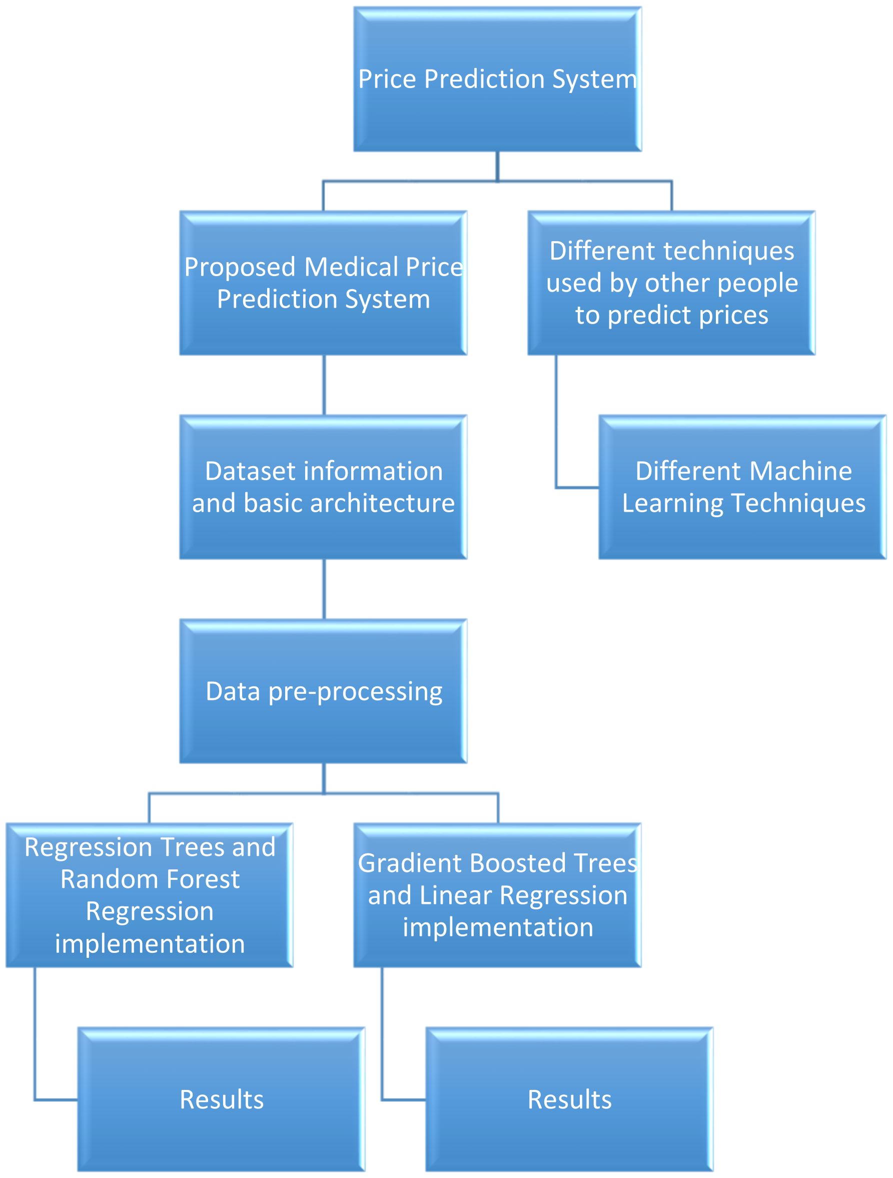

Price Prediction System
Proposed Medical Price Prediction System
Different techniques used by other people to predict prices
Dataset information and basic architecture
Different Machine Learning Techniques
Data pre-processing
Regression Trees and Random Forest Regression implementation
Gradient Boosted Trees and Linear Regression implementation
Results
Results

Fig. 1. Organization of this project

Some of the work done in this project is done in the paper Tike et al. [1]. The rest of the work in this report is organized in the following way. Chapter 2 contains a history of price prediction system. Chapter 3 describes different machine learning approaches used in price prediction, for example, supervised and unsupervised learning techniques in machine learning. In Chapter 4, we propose a new medical price prediction system. Chapter 5 specifies data pre-processing techniques performed in order to get the cleaned input data for processing. In Chapter 6, we give implementation details of two algorithms which are Decision trees and Random Forest regression. Chapter 7 describes Gradient boosted decision trees and linear regression in detail. In Chapter 8, we provide results from all the experiments done in this project. Finally, we conclude the work done and the future work for this project in Chapter 9. Additionally, we are giving references used in this project and the source code in the end.

CHAPTER 2

Price Prediction History

Price prediction is a popular problem. There are several price prediction systems which are used to predict different kinds of prices. Some of them include stock prices [7], [8], [9], [10], [11], home prices [12], [13], electricity prices [14], [15], [16] etc. The techniques used in above papers have been varied such as fuzzy logic, neural networks, genetic algorithms, Naive Bayesian method and others.

In the health domain, there have been numerous works on predicting medical prices in different contexts [17], [18], [19], [20], [21]. For example, Moran et al. [17] use generalized linear regression methods to predict Intensive Care Unit (ICU) costs and use patient demographics, DRG (Diagnostic Related Group), length of stay in the hospital and a few others as features. Sushmita et al. [18] attempt to predict future health-care costs of individuals based on their medical and cost history. The expected costs for the future 3-, 6-, 9-, and 12-months are projected. They apply linear regression and Random Forest regression analysis for cost prediction. Lahiri et al. [19] employ classification algorithms to predict whether an individual's health care costs will increase in the next year given the health care costs for the previous year. Researchers have also used hierarchical regression analysis to tackle price prediction problem. Multilevel linear regression is used to determine effects of patient and physician characteristics on diagnostic testing [22]. Hierarchical decision trees are used for classification tasks where the class labels are hierarchical in nature [23].

The problem which will be addressed in this project is distinct from the problems addressed by other researchers. We will tackle the problem of predicting costs of treating DRGs at a hospital located anywhere in the U.S. using Medicare payment datasets and picking only relevant attributes from the dataset which are useful for our problem.

In the next chapter, we provide information regarding different machine learning techniques used in the price prediction systems.

CHAPTER 3

Machine Learning Approaches

Price prediction history shows that authors have used machine learning techniques in this domain extensively. Health domain is no exception for this where medical prices are being predicted using health related data. Broadly machine learning techniques are categorized into two types of learnings, supervised and unsupervised learning.

3. 1 Supervised Learning

Supervised learning is more widely used among the mentioned types. It is called supervised learning because in this approach, the algorithm is trained on the input data to get the desired result. Another way to define it is an algorithm is trained under a supervision of training data. In this case, the data used to train the algorithm is a labeled data. When the target or the result is known, the data is called labeled data. In a typical supervised machine learning problem, the data has two parts. The first part consists of input variables which are called features. The second part is actual target variable or label. Features helps to find out the target. The mapping of features and the label would look like,

$$Y= f(X)$$

where Y is the label and X is the input variable.

There are two phases in supervised learning. The first is the training phase and the second is the testing phase. In the training phase, data consists of input variables and a label associated with those. When the algorithm is sufficiently trained on this data, new data is given for testing. This new data is not labeled. Therefore, based on the training algorithm has gained in the training phase, it has to predict the label for new set of data. Examples of supervised learning are classification and regression techniques. These techniques train your model/machine with the chosen dataset and when the new input comes classifies or predicts the output.

3.1.1 Classification

The target variable or the label in a classification problem holds categorical values. That means the target variable is discrete. The categorical values of this target variable are usually finite. The mapping function given in the previous section then would try to predict one of the categorical values, the target variable has with the help of input variables.

For example, given a set of input variables an algorithm would try to predict whether a person will buy a house or not. In this example, Buying a house is a label and it has two values Yes or No. When the label has only two values such as either "Yes" or "No", or "1" or "0" then the problem is called as a binary classification problem. The label can have multiple class values as well then it is called as multi-class classification problem. For example, given a set of input variables an algorithm would try to predict whether a given fruit is "Mango" or "Apple" or "Banana". There are many algorithms used to solve classification problems. Some of the algorithms are Decision Trees, Random Forest, Naïve Bayes, Logistic regression etc.

3.1.2 Regression

The target variable or the label in a regression problem holds continuous values. That means the target variable is continuous e.g. 1240, 1256, 4800.89 etc. These continuous numerical values of this target variable are not finite. These values can range from any real value to another real value. The mapping function given in the previous section then would try to predict the continuous value in the given range of the target variable with the help of input variables.

For example, given a set of input variables, an algorithm would try to predict the house price. In this example the house price is a label and it can have any value within the given range of values of training data. There are many algorithms used to solve a regression problem. Some of the algorithms are Decision Trees usually called as Regression Trees, Random Forest Regression, Linear Regression etc.

Tree Approaches and Regression Algorithms

A price prediction is a regression task because any kind of a price is a continuous value and as mentioned in the previous section, regression problem tries to predict a continuous value [24]. In a regression task the dependent variables may or may not be continuous valued but the final outcome being predicted should be continuous.

A typical price prediction system will take the set of input variables, apply the technique chosen to get the final outcome and based on that predicts the price. Some of these techniques are machine learning algorithms which are related to trees approaches. These algorithms are decision trees and ensemble methods like random forest regression, gradient boosted regression trees etc. These algorithms are popular because of their simplicity and their efficiency over other complicated techniques. We will discuss these algorithms as we are going to use them to solve our medical payment price prediction problem.

(a) Decision Trees

Decision trees [25] are a class of popular machine learning models that are used for classification and regression. Fig. 2 shows a simple decision tree used for classification that uses features to classify a person into two categories fit or unfit. The feature set contains age, whether a person eats a lot of pizzas and whether a person exercise in the morning.

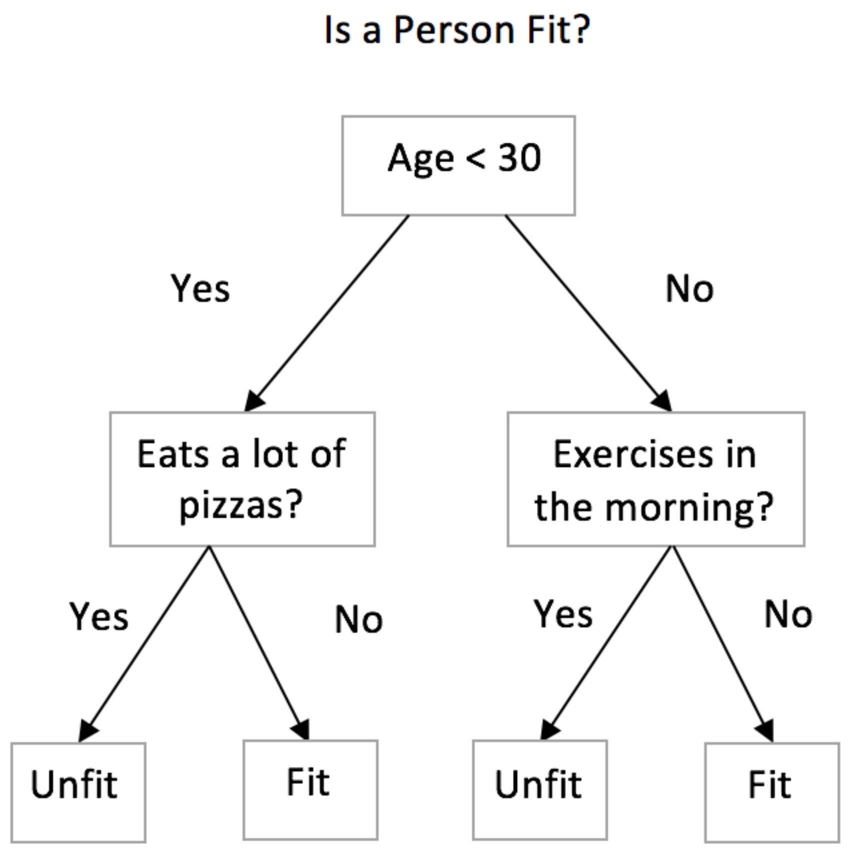


Fig. 2. An example of Decision Tree classification

The decision trees are constructed in a top-down fashion using the training data. At each step, the goal is to split the elements of the training set such that the subsets are as homogeneous as possible. Towards this end, the common metrics used while constructing a decision tree classifier are gini impurity and information gain [26]. For decision tree regressor, the criterion employed for splitting is variance [27].

While using the decision trees one of the best practice is to calculate the importance of each feature [28]. The basic idea is to capture the relative importance of features in a particular dataset. Features are the different variables in a dataset whose values differentiate each row in the data. The feature having the highest importance gets the priority and data is split according to that and so on. The constructed decision tree is subsequently used to classify or predict a new instance. If the decision tree is being used as a classifier, then the new instance is classified into one of the classes provided and if it is being used as a regressor then, it is used to predict the outcome.

A classification tree predicts the class of a new instance by taking the mode of all the class values at leaf nodes. A regression tree predicts the value by taking mean of all the values at leaf nodes [28]. There are many algorithms to build regression trees, some of them are AID, CART, M5 and GUIDE [27]. AID and CART construct piecewise constant regression trees. Among these algorithms, CART uses binary split on the node where as others use multiple split on the node.

(b) Ensemble Methods

The Random Forests [30], gradient-boosted trees [29] are called ensemble methods, for constructing multiple decision trees. Ensemble methods take multiple weak learners, such as decision trees, and construct a strong learner from them such as random forest. The Random Forests and gradient-boosted trees both can be used for classification and regression task.

In ensemble methods a classification task for a new instance will be done by taking the majority of votes from each tree. The class getting the highest votes will be chosen as the final target value for the new instance. Fig. 3 shows a simple illustration of random forest classifier.

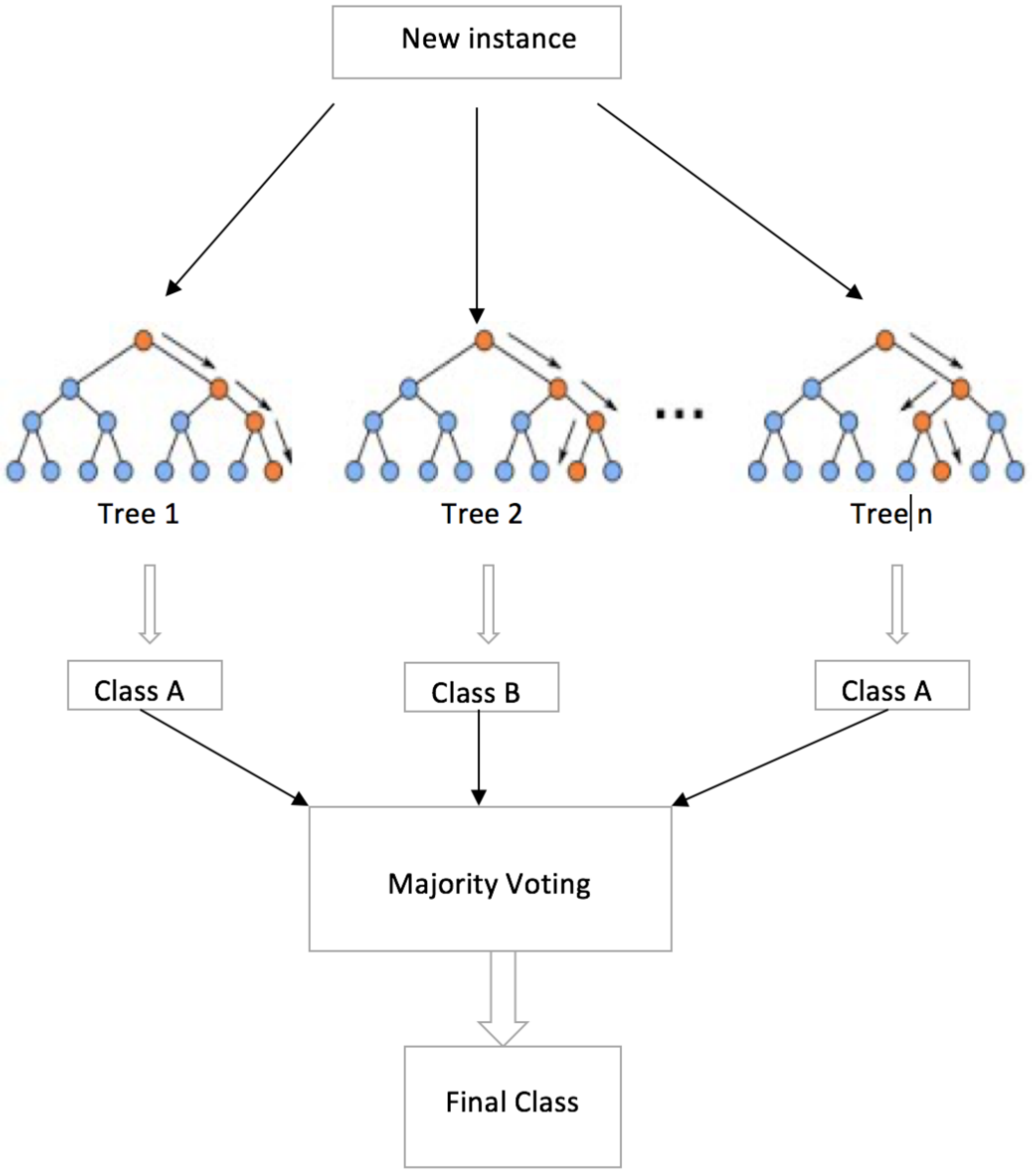


Fig. 3. An example of Random Forest classification

On the other hand, in a regression task, the new instance is passed through all the trees and the outcomes from all the individual trees are aggregated to produce an overall outcome. Fig. 4

shows a simple illustration of random forest regressor.

Like decision trees, we can calculate the importance of each feature in the ensemble method also. They are calculated by computing the importance of features of individual trees and then averaging them across the trees. Ensemble methods are more robust because they decrease the tendency of a single decision tree to overfit the training data.

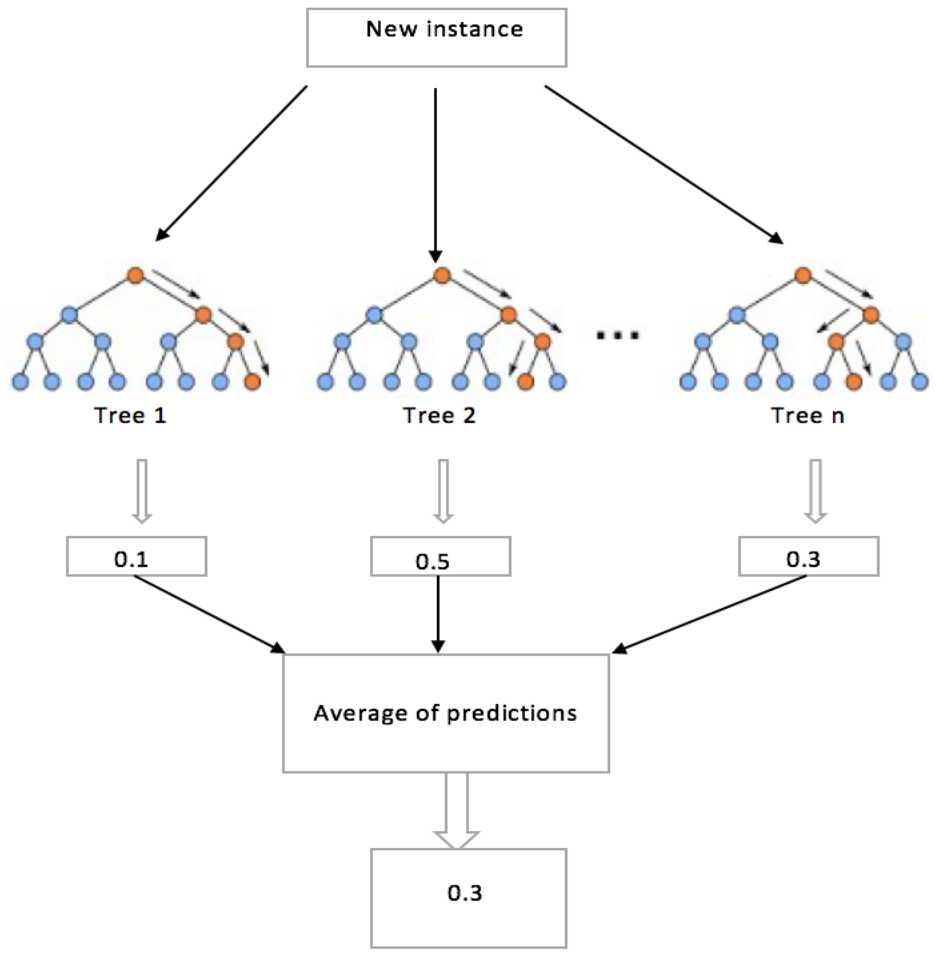


Fig. 4. An example of Random Forest regression

3. 2 Unsupervised Learning

The second type of machine learning technique is unsupervised learning. Unlike supervised learning, unsupervised learning algorithms don't learn from the input data. That is, the data in case of unsupervised learning don't have labels associated. An algorithm only takes input variables and finds patterns in the given data. It then tries to predict the right answer from those patterns. The most common examples of unsupervised learning are clustering algorithms and association rule mining algorithms. For example, in clustering the algorithm tries to find pattern in a given set of input variables. These patterns are called clusters. When the new input data comes for testing, an algorithm tries to put it into the right cluster formed during the training phase.

In the next chapter, we propose a medical price prediction system which will be used to predict medical prices. Additionally, we will provide the details of the dataset information and feature selection from the dataset.

CHAPTER 4

Proposed Medical Price Prediction System

As seen in the earlier chapters, a lot of work has been done in the area of medical price prediction and researchers have used various techniques to predict various medical prices. A new proposed medical price prediction system will predict the costs of treating DRGs at a hospital located anywhere in the U.S. using Medicare payment datasets. The problem of a price prediction is a regression task and for this use of machine learning models to perform regression will be suitable solution. The system will use random forest regression algorithm as a machine learning algorithm to predict the prices.

4. 1 Dataset Information

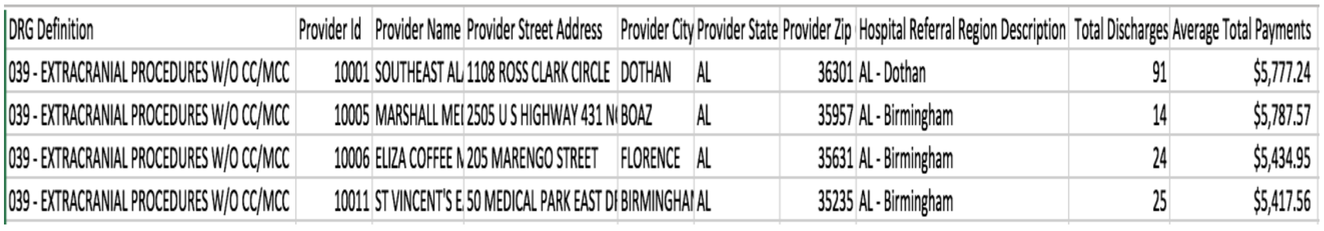
The input dataset for our proposed system will be a dataset which will be a combination of two separate datasets. In our final input dataset, we are going to incorporate some columns from the inpatient Medicare payment data [31] and one column from the Zillow Data [32]. We will examine these columns in detail in the next section.

The first dataset which is a medicare payment data, includes hospital-level charges for over 3000 U.S. hospitals for the top 100 most frequently billed Diagnostic-Related Groups (DRGs). The payment for the top 100 DRGs constitutes 60% of the total inpatient-related Medicare payments and it represents 7 million discharges [31].

Table I lists four randomly selected rows from the payment dataset for exposition [31].

TABLE I

Sample rows from the payment dataset



DRG Definition	Provider Id	Provider Name	Provider Street Address	Provider City	Provider State	Provider Zip	Hospital Referral Region Description	Total Discharges	Average Total Payments
039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10001	SOUTHEAST AL	1108 ROSS CLARK CIRCLE	DOTHAN	AL	36301	AL - Dothan	91	$5,777.24
039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10005	MARSHALL ME	2505 U S HIGHWAY 431 N	BOAZ	AL	35957	AL - Birmingham	14	$5,787.57
039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10006	ELIZA COFFEE N	205 MARENGO STREET	FLORENCE	AL	35631	AL - Birmingham	24	$5,434.95
039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10011	ST VINCENT'S E	50 MEDICAL PARK EAST D	BIRMINGHAI	AL	35235	AL - Birmingham	25	$5,417.56

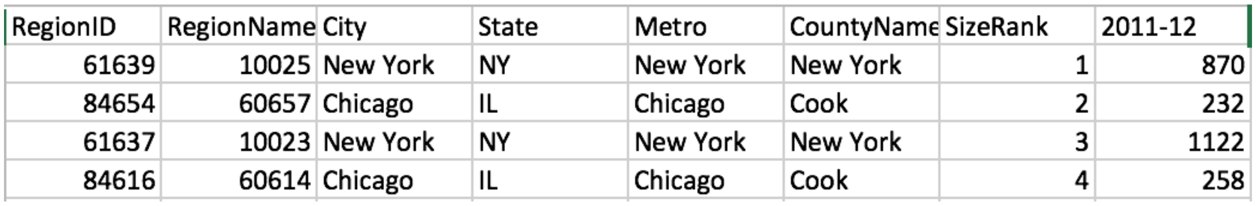
Each row consists of the DRG definition column, the provider ID, the provider name, address, ZIP code, state where the hospital is located, the number of discharges from the hospital for the fiscal year in the DRG category, and the average payment across all discharges. The DRGs are a patient-classification system [33] used by Medicare as the basis for hospital payments. The patients within a DRG are deemed clinically similar and are expected use similar quantity of hospital resources and hence, are expected to be billed similarly. Even so, a wide variation in the prices for a given DRG among different providers is observed.

The second dataset is a median house price per square feet data. It includes median house prices per square feet by zip code for all states in US from the year 1996 to 2017. In this dataset we are only concerned about the median house prices for the year 2011 in every zip code because our medicare payment data is for the year 2011.

Table II lists four randomly selected rows from the house price dataset for exposition [32]. Each row in this dataset consists of RegionID which is a zip code for that region. Other columns in the dataset are Region Name, City, State, Metro area and County Names for the particular region. The last column in the table is the most important column for us which is median house prices for the particular region for the year 2011.

TABLE II

Sample rows from the house price dataset



RegionID	RegionName	City	State	Metro	CountyName	SizeRank	2011-12
61639	10025	New York	NY	New York	New York	1	870
84654	60657	Chicago	IL	Chicago	Cook	2	232
61637	10023	New York	NY	New York	New York	3	1122
84616	60614	Chicago	IL	Chicago	Cook	4	258

We will examine which columns to select as features for our price prediction problem and why from above two datasets in detail in the next section.

4. 2 Feature Selection

As seen in the previous section, each row in the payment dataset has ten columns. Each of these columns represent a feature in the field of machine learning. A feature is an important property on which a prediction variable under consideration is dependent upon. Every problem under observation consist of a set of independent features which help to build an accurate machine learning model.

4.2.1 Types of Features

There are total four types of features. They are following [34]:

1. Binary: This type of feature has only two values. For example, feature “Eat Pizza” will have only two values “Yes” or “No”.
2. Nominal/Categorical: This type of feature has multiple values. For example, in our dataset “DRG Definition” is a categorical feature. It has 100 unique values.

3. Ordinal: The values of this type of features are in order and that order cannot be changed. For example, consider feature “T-shirt Size” it will have the values as “Small”, “Medium” and “Large” etc. which are in order.
4. Continuous: This type of feature is usually numerical. They can have as many values as defined in continuous range of numbers. For example, consider feature “House Price”, it can have any value in a given range of data.

4.2.2 Selection of Features from the Dataset

It is a very crucial task to select only important features from the given set of features which are more relevant and build a robust model. That is why, from our medicare dataset we are going to select only those features which will independently help us to predict medical prices. Columns such as provider address, ZIP code, state, city and hospital region referral description, all represent the location of the provider. Hence, instead of considering all of them we will only take one of them as a feature in our feature set. We will choose ‘hospital region referral description’ as it is not as specific like provider address or city and not broad like state. Other independent features we will choose from medical payment data are DRG Definition and Total Discharges.

Along with these features in the Medicare payment dataset, a new feature which is real estate prices. These prices are the prices of real estates present in the locality of hospitals for the same year as the medical data. Our vision is, this feature can also be dominant feature in predicting prices. The hypothesis is that a hospital’s cost of operation factors into the price charged by the hospital. Among the various costs a hospital has to bear, a significant one relates to real estate cost – the cost of owning a building or renting it. The real estate cost is a surrogate for other costs too in that if a certain region has high real estate costs, then it likely has higher costs in the other categories as well such as salary paid to doctors, staff etc [1]. Therefore, real estate cost will also be included in the feature set, in particular the median per-square foot price of homes – MedianPerSqFtPrice – within the hospital's ZIP code [32] compiled by the real estate portal www.zillow.com. Fig. 5 plots the real estate prices against the payment amounts for the DRG that has the highest number of prices for it, viz., "194 - SIMPLE PNEUMONIA & PLEURISY W CC" [1]. The linear trend line generated using the least-squares fit is shown. It is shown clearly in the figure that as the per-sq-ft prices go up in a ZIP code, the medical prices also go up in the area.

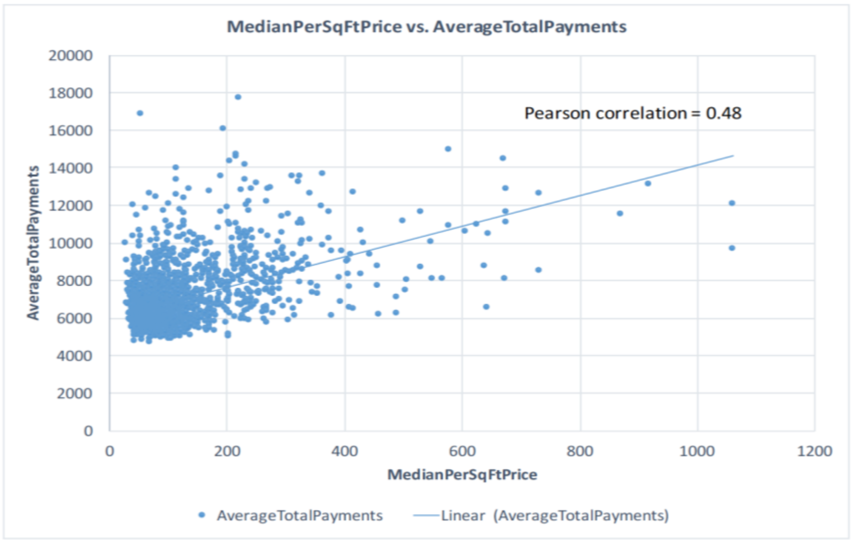


Fig. 5. The relation between real estate and medical prices for the DRG: 194 [1]

4.3 The Total Feature Set

The system uses following features to train and test the machine learning model.

1) DRG Definition

2) Hospital Region Referral Description

3) Total Discharges

4) MedianPerSqFtPrice

In the feature set, the DRG Definition and the Hospital Region Referral Description are categorical features where the Total Discharges and the MedianPerSqFtPrice are continuous features. While training the model, the Average Total Payment column becomes the “label” or the value being predicted by the model at the time of testing.

4.4 Basic Architecture of the Proposed System

The system architecture is shown in Fig. 6 Medicare payment and real estate datasets are used to train various machine learning models. The efficacy of the different trained models will be decided using a test dataset. The system will be useful to patients and policymakers for predicting prices of medical procedures.

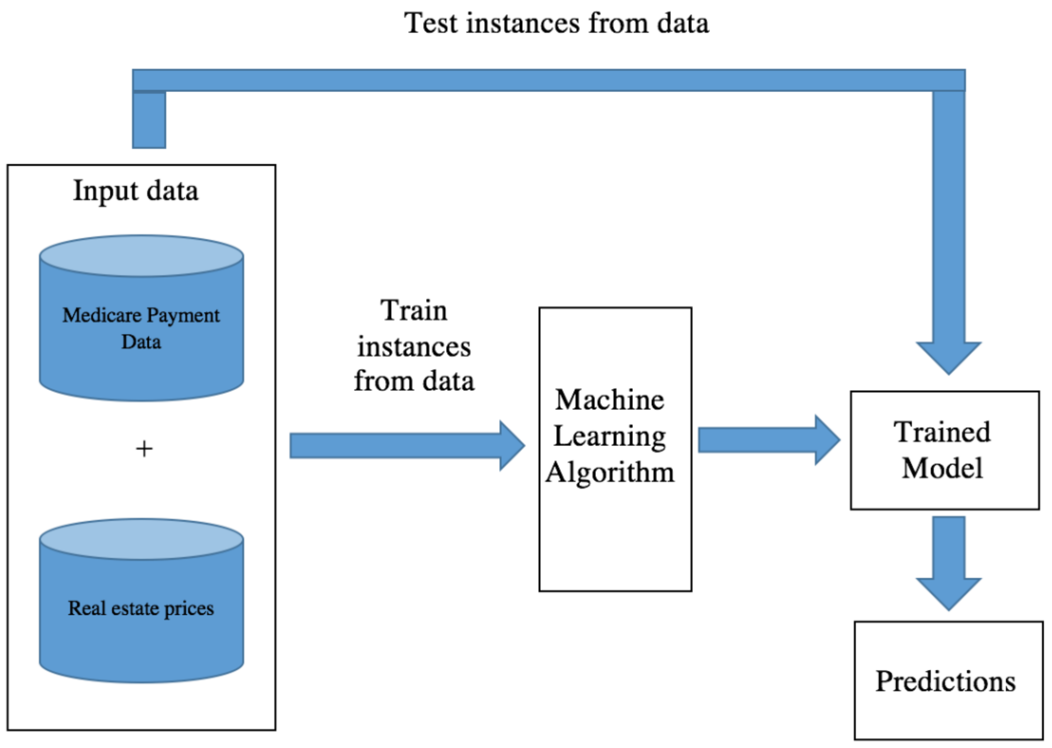


Fig. 6. System Architecture

In the next chapter, we will provide the implementation details of the data pre-processing task.

The data pre-processing task includes following steps:

1. Data Integration
2. Data Cleaning
3. Data Transformation

We will describe each of these steps in detail.

CHAPTER 5

Data Pre-Processing

One of the most important parts of machine learning problems is the data to be used to solve the problem at hand. In a typical machine learning project, data preparation takes around sixty to seventy percent of the total time of the project [35]. The correct data for the problem at hand is very important in getting good results. Data preparation in general is a combination of feature selection and pre-processing those features. Hence, once the feature selection is done from the large amount of data, the next important part is to pre-process those features. Because the data in its raw form is not useful. The purpose of pre-processing here is, to make features suitable for the machine learning model which we are going to apply. Better results can be achieved from the model if the features are in proper manner. Also, the format of data are different for different models. For example, there are machine learning models which do not accept null values in the data. So, before using the raw data for those models, we have to manage null values properly to get the best results.

5. 1 Data Pre-Processing Steps

There are many steps involved in data pre-processing. Some of them are data cleaning, data integration, data transformation and data reduction. In data pre-processing the steps to use from the mentioned steps is dependent upon the nature of raw data, the problem statement and the machine learning algorithm to be used to solve the problem. Sometimes all of the steps of data pre-processing are required in order to get the results or sometimes only some of them are required.

Hence, from the above mentioned steps of data pre-processing we mainly need three steps for our price prediction problem.

5.1.1 Step 1: Data Integration

Data integration involves identifying different data sources to be required for processing and consolidating their data into one. This step is very important for any system which requires huge data processing to be done to solve the problem at hand. There are different tools in the industry to integrate data from varied sources and combine it. Such tools can be used where data is huge and there are many data sources. But for our machine learning problem these tools are not required because our data is very less. So we have used a simple python program to integrate our data sources.

As mentioned in the previous chapter, we will be using two datasets for our price prediction problem. One of them is medical payment data and the other is median house price data. Both of these datasets have “Zip Code” column in common. Therefore, we applied inner join on this column. Inner join selects only those records from both datasets which are matching the specified condition. In our case, the matching condition is zip code values. After performing this step, we get the combined dataset which will be further used for data cleaning and transformation.

5.1.2 Step 2: Data Cleaning

Raw data can have many impurities. These impurities can affect the final outcome especially in machine learning problems. Therefore, once the data is integrated, it needs to be cleaned. Data cleaning involves detecting the impurities such as inaccurate entries, irrelevant and inconsistent

records and removing these impurities. There are many techniques by which we can perform data cleaning. Some of the techniques in data cleaning are using automated tools, manual intervention and writing scripts to programmatically clean the data according to our needs.
As discussed in the previous section our data is not big enough to use automated tools. Therefore, we have used manual intervention and python script to clean the data based on our requirement. Firstly, we checked by manual intervention whether our records do not have any erroneous or wrong entries. After that, we removed null records using the script as Random Forest Algorithm does not accept such values. In addition to this, our script removed unexpected spaces between the values. Once the task of data cleaning is done, the cleaned data can be transformed into required format for the problem.

5.1.3 Step 3: Data Transformation

When data integration and data cleaning are done the next step is to transform the integrated and clean data into suitable format required by our system. Typically, data transformation consists in converting source data into the format required by the destination data. In our case the source data will be the data we received after data cleaning step and the destination data will be the data to be fed to the machine learning model. Data transformation can be done using automated tools and writing scripts. For our price prediction problem, to perform data transformation we have written python script which transforms the data into required format.
Data for our problem is the set of features used to predict the prices. Though theoretically every machine learning algorithm takes all types of features e.g. categorical, numerical etc. and tries to build the model with the help of them, technically these features have to be converted into

numerical format in order to understand by the computer. Hence, we have to transform all categorical features fed to our machine learning algorithm into numerical ones.

There will be two types of transformations done on the input data. First is, the conversion of categorical features into numerical ones and second is, removing unnecessary characters in feature values.

We have done the data transformation step by step. The first step is data discovery in which we identified which columns in the data to transform. For our problem, the columns to be transformed are “DRG Definition”, “Hospital Referral Region Description” and “Average Total Payments”. The next two steps after data discovery are data mapping and converting data into required format with the help of a program.

From the above three mentioned columns “DRG Definition” and “Hospital Referral Region Description” are categorical features. There are two ways to transform categorical features into numerical features. First is, using one-hot encoding and other is writing a program which will convert feature formats into required ones.

One-hot encoding generates one boolean column for each unique category present for the particular categorical feature. Only one of these columns could take the true value that is 1 for each record. For our dataset one-hot encoding is not suitable because, categorical features in our dataset has too many unique categories. Adding Boolean column for each of those categories will make data grow horizontally unnecessarily. It will make the model very complex. For example, “Hospital Referral Region Description” column has 101 unique values hence we have to add 101 boolean columns for each category value which is very complex to process even further.

Therefore, we decided to write a program to convert all categorical features into numerical ones according to our needs which will be simple.

Transforming categorical feature “DRG Definition” into numerical feature:

The actual value for “DRG Definition” column was in the following format: “039 - EXTRACRANIAL PROCEDURES W/O CC/MCC”. We identified a pattern in this format which is a number followed by the DRG description. It made sense to just extract first three characters from the original string. These three characters are actually numbers corresponding to each unique DRG. Hence, the old value is transformed into new value, which is “039”. Finally, we converted this three characters string to number giving us the end result as number 39.

Transforming categorical feature “Hospital Referral Region Description” into numerical feature:

The actual value for “Hospital Referral Region Description” column was in the following format: “AK – Anchorage”. The pattern we identified in this format is State Code followed by region within the particular city i.e. zip code region. This column has total 101 unique values. We assigned 101 random numbers to these string values. Hence, for example all the rows having “AK- Anchorage” value will have the numerical value 1.

Transforming the label “Average Total Payments” in numerical format:

The actual value for the label “Average Total Payments” following: $5787.57

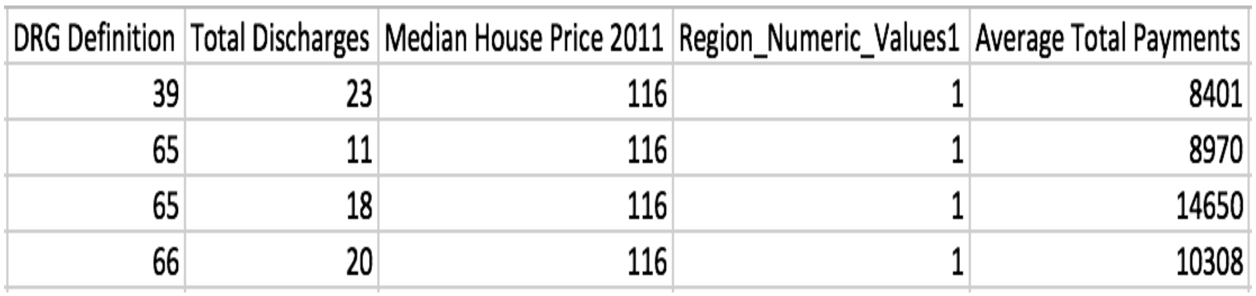
The pattern here is “$” followed by a number which is float. At the time of applying the algorithm on such value we do not need “$” sign hence, we removed dollar sign from the values of this column. Also, for the simplicity we converted float values to integer.

5. 2 Totally Pre-Processed Data

After performing all of the previously mentioned steps, we get our final data which can be fed to our machine learning algorithm to predict the prices. Table III shows the sample four rows from the final pre-processed data.

TABLE III

Sample rows from the final dataset



DRG Definition	Total Discharges	Median House Price 2011	Region_Numeric_Values1	Average Total Payments
39	23	116	1	8401
65	11	116	1	8970
65	18	116	1	14650
66	20	116	1	10308

In the next chapter, we provide the implementation details of the regression tree algorithm.

CHAPTER 6

Regression Tree and Random Forest Regression Implementation

6.1 Regression Trees

We provided a description of decision trees in Chapter 3. Decision trees are one of the techniques used in the predictive modeling. Predictive modeling is the technique in which the data at hand is used to predict the value for new data. As described in Chapter 3, a new instance is either classified or predicted in the predictive modeling. Based on classification or prediction, a decision tree is also divided into two types. The first type is classification tree and the other is regression tree. A classification tree is used when the label of the data is categorical, that is, the label has discrete values. When the new record is passed to the classification tree, it will classify the record into one of the classes the label represents. On the other hand, a regression tree is used when the label is a continuous value, for example, medical price. Hence, when the new record is passed to the regression tree, it will predict the value of the label for that record based on the data at hand. The label values in this case can be any value.

Any decision tree has three types of nodes. The nodes in a decision tree are a root node, intermediate nodes and leaf nodes. A root node will always have all the data before splitting. Then, as we go down in the hierarchy, the data are further split using intermediate decision nodes. The leaf nodes will have the prediction value. The root node and the intermediate nodes will represent the features and their values. The leaf nodes will have the label values or predictions. Once the tree is created, for the new record it can be traversed through the branches and nodes of a tree based on the feature values the record has until the leaf node is reached. Fig. 7 shows the tree structure.

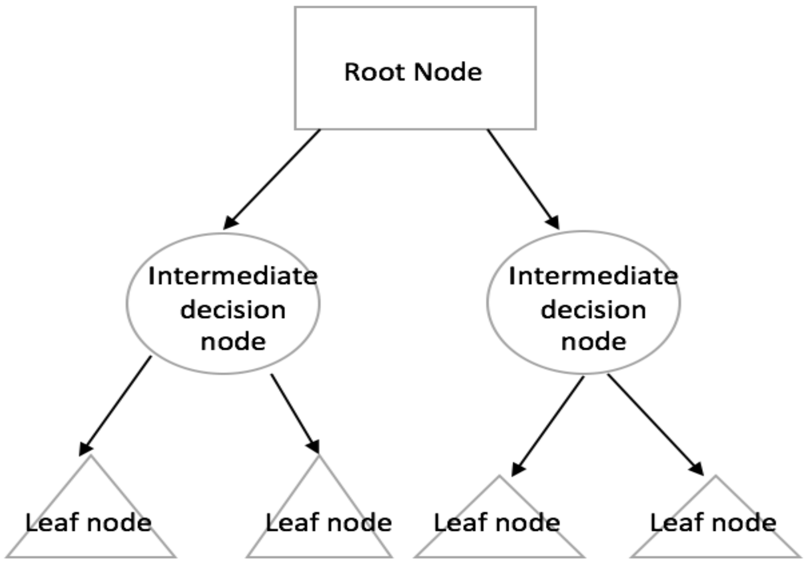


Fig. 7. A basic tree structure

6.1.1 Working of a machine learning model

Once the data is pre-processed, we need to divide it into train and test records. As discussed in Chapter 3, supervised learning methods need to perform this step in order to build the machine learning model to train the model to do the prediction for new records that are test records. For our project the machine learning models are regression trees and random forest.

As discussed in Chapter 4, our price prediction system will use the following steps in order to predict the prices.

1. Pre-process the data: We discussed pre-processing of input data in Chapter 5.

2. Dividing the pre-processed data into train and test records.

3. Build the model using train data.

4. Test the built model using test records by predicting the values of label.

There are many techniques which are used to divide the input data into training and testing records. Some of them divide the input data into training and testing records randomly or by fixed number. In random division, the records in input data will be randomly selected and divided into training and testing sets based on the user specified ratio of training is to testing. For example, if the input data has 100 records and user specified train is to test ratio is 80:20, then the training set will have 80 randomly selected records from 100 and testing set will have 20 randomly selected records from 100. On the other hand, in case of fixed split user can specify the range of record indices which will go in training and testing sets respectively. Though these methods are easy to implement and very simple, they have problems associated with them. A machine learning model can overfit because of these methods. Overfitting means that model is trained too much on the training data and it is not generalized. That is the model is well trained for the training records but when the new records are passed to it, the model does not predict the results accurately. To avoid this problem of overfitting the solution is to use a method cross-validation split. To split the input data into training and testing sets, we have used cross-validation split in our project.

6.1.2 Cross-validation Split

Cross-validation splitting tries to minimize the overfitting and generalizes the model. We used k-fold cross-validation which comes under the type Non-exhaustive cross-validation [36]. In k-fold cross-validation, entire input data is divided into k number of subsets. Among these k subsets, k-1 subsets are used as a training set and kth set is used as a testing set. Selection of k-1 sets is random. This process of selection is repeated k times. In each iteration exactly one fold is used as a testing set. When all the iterations are done all the results got from each iteration are averaged to get the final result. There are many other ways to use cross-validations which are slightly different from k-fold cross validation [37]. It depends upon the input dataset which type of cross-validation split to use. Fig. 8 shows the working of k-fold cross-validation split.

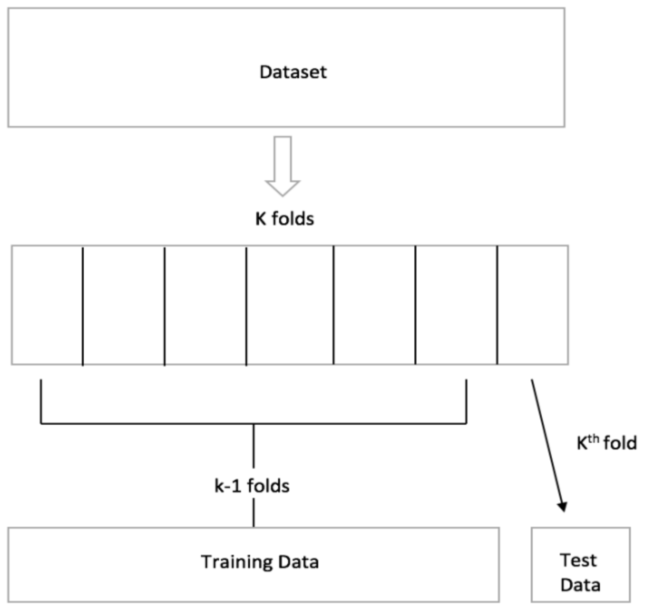


Fig.8. k-fold Cross-validation split

6.1.3 CART Algorithm

Once the input data is divided into training and testing sets, the next step is to build the machine learning model using the training set. As discussed in Chapter 2, price prediction is a regression task so, we have used regression tree to predict medical prices for our data. The algorithm we have implemented in order to build the trees is CART [37]. CART uses recursive partitioning technique. Recursive partitioning partitions the input data recursively [37]. It starts partitioning the input data starting from the root node and keeps partitioning the data until the leaf node is reached that is, there are no more records for partitioning are left. The recursive partitioning in case of CART is binary. Binary partitioning divides the data into at-most two parts or regions. This is called dividing the input space. There are three main components in recursive partitioning. They are as follows:

1. Selection of a feature from the set of features to split the data.
2. Selection of a particular value of the selected feature in step 1.
3. Termination criteria, that is, when should the splitting stop and records are reached to the leaf node.

Selection of a feature from the set of features to split the input data

When a particular feature and its value is selected for splitting, a model will try to minimize the cost function defined and will try to maximize the information gain from this split. This strategy of dividing the input space is called greedy strategy as each time a node is selected which will give the best split. The features and its value to split will be selected based on the loss function. For regression trees, mainly two cost functions are used across all the training records which fall within each divided input space. These functions are the sum squared error and the weighted variance.

Sum squared error:

$$SSE = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

where n is the total number of the observations falling under the selected input-space. X_i is the ith observation and X bar is the mean of the values of all observations.

So, the split with minimum sum squared error is selected.

Variance Reduction:

Variance reduction is the other criteria used to select the split. We have employed weighted variance reduction in our implementation of regression trees as it gave better results than the sum of squared error for our input data.

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

where n is the total number of the observations falling under the selected input-space. X is the value of observation and X bar is the mean of the values of all observations.

So, the split with minimum variance is selected. To calculate weighted variance, first the variance of each node on which we want to split the data is calculated. Then after that, we calculate variance for each split as weighted average of each node variance.

Once the variance is calculated, we calculate the information gain for the split. The idea is to maximize the information gain. A split is selected such that it gives maximum information gain which means for that split we are getting homogenous records. Fig. 9 shows homogenous splits. In the Fig.9, a dataset has two types of classes, triangles and squares. A node to split on will be selected such that two splits we will be getting, will have homogenous records that is records of the same class. As in the Fig.9, we are getting homogenous splits such split is desirable and gives maximum information gain.

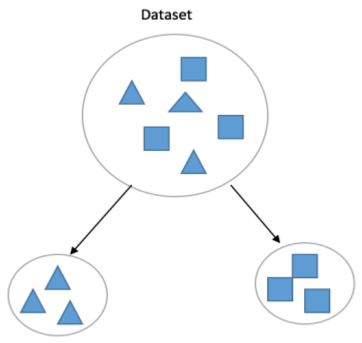


Fig. 9. An example of Homogenous split

Selection of a particular value of the selected feature in step 1

To calculate variance and eventually information gain, when we select a particular feature to split on at that time we also select the particular value of that feature which will be used as a condition to split. For example, in our dataset there is a feature “DRG definition”. If this feature is selected at the time of tree building for splitting, then we select its particular value as a condition to divide the records into regions. DRG definition has set of values. Hence among those values, for example, “DRG definition <= 56” will be a condition. This condition will be used to divide the records into two separate regions. Region 1 will have all the records matching the condition and region 2 will have all the records which are not matching the condition. The notion here is, to determine how well the split condition performs. For that, we compare the degree of impurity of a parent node before splitting and degree of impurity of child nodes after splitting, this difference should be larger to consider the split as the best split [36]. The larger the difference, the more the is split homogenous.

There are many techniques to choose a value of a particular feature so that it will provide the best information gain. These techniques are different for different types of features as mentioned in Chapter 4. We converted all of the features required for our price prediction task into continuous features. For example, if there is a continuous feature called “F” and it holds a value “V” then split conditions can be one of the following conditions mentioned:

1. $F < V$
2. $F<=V$
3. $F> V$
4. $F>=V$

With these conditions to split continuous attributes, there are mainly four techniques [34]. First technique is to go through all the values the feature holds in order to find the best split which will provide the maximum information gain. For example, if there is a feature F which has total n values, then the earlier mentioned technique will go over each value. While checking the condition for each value, it calculates the variance and information gain for this split. Finally, it chooses the value with minimum variance. This technique is computationally expensive as it requires O(N) operations to compute the variance and information gain at each candidate split position. There are a total of N features, so the total time complexity required by this task will be $O(N^2)$.

The modification in this technique is the next technique. This technique sorts the values of features before starting the comparison. Then it selects only those values as split points which are the midpoints between two adjacent values. This approach reduces time complexity of the task to O(N log N). We do not need to evaluate all values of the feature but just the midpoints of two adjacent values.

The third technique is binning the values of a feature. If the feature represents wide range of values, then we can put certain range of values in a bin. For example, if feature “F” has values ranging from 1 to 100, then we can make 10 bins, each of these will hold 10 values. For example, bin 1 will have values 1 to 10, bin 2 will have 11 to 20 and so on. This will reduce the comparison time if the feature has too many values.

The fourth technique is to calculate the mean of all values a feature holds and always compare the other values with the mean. This technique is not robust as it may miss the best split value which can give the maximum information gain ultimately making the model strong.

The technique we have used in our project to find the best split which will give us the minimum variance each time, firstly sorts all the values a particular feature holds. It then, checks the comparison condition for only those values which are unique for this feature and finds the split point which gives the minimum variance each time. The comparison condition we used is "<=". For example, if we want to check split for a feature "DRG Definition" for a value 56 then the condition will be "DRG Definition<=56". This condition will divide the records into two parts. Records matching the condition and records which are not satisfying the condition.

Termination criteria

For building trees, CART uses recursive partitioning algorithm because of which trees can grow large. The algorithm uses the greedy approach to find the best split which leads to grow larger trees. This strategy can induce errors in the machine learning model. The most important error which can occur is overfitting because as the tree becomes larger the model learns too much from the data and fails to give correct prediction when the new record is passed to it [34]. To avoid this error, the technique called pruning is used. Pruning cut the tree when it grows very large. There are two types of pruning techniques, the first one is pre-pruning and the second one is post-pruning. Both of the techniques reduce the tree size. In our tree building process, we have used pre-pruning as the strategy to stop the growth of the tree because post-pruning is computationally time consuming.

Pre-pruning: Pre-pruning is also called as early stopping [34] because in this approach tree is pruned at the time of building. The stopping criteria we used are following:

1. Number of records reached to the leaf node: We have provided a parameter which can hold the value of user defined number of records reached to the leaf node. The value of this parameter is always checked when a particular record reaches the leaf node. If the number of records reached to the leaf node are greater than this parameter, then the tree should be stopped from growing.
2. Maximum depth of a tree: We have provided a parameter which can hold the value of user defined depth of a tree. If the depth of the tree is greater than equal to the maximum depth of a tree given by user, then the tree should be stopped growing.
3. Information gain threshold: We have provided a variable parameter which holds the minimum threshold value for the information gain. If the information gain from the current split on a feature is below the threshold, then this split should not be considered for building the tree. This will help to consider only those splits which are relevant ultimately reducing the size of the tree.
4. Partitioning of records: When the partitioning is performed on the particular value of a feature which gives either one or both, the number of true records and the number of false records as zero, then such a split should not be considered.

Post-pruning: As mentioned in the previous section we have not implemented post-pruning technique in our project as it is computationally complex and we have less number of features at the time of growing of a tree. Due to a low number of features for our project pre-pruning is sufficient to reduce the size of the tree. Post-pruning can be done by replacing an entire sub-tree with a new leaf node which will have the prediction from the most frequently used branch of the subtree [34].

6.1.4 Error calculation

To calculate the accuracy of the model the criterion we used is called mean absolute error [38]. Mean absolute error is the average of the absolute difference between actual value of label and predicted value of label by the model in a set of test records. The formula for mean absolute error is given as following:

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

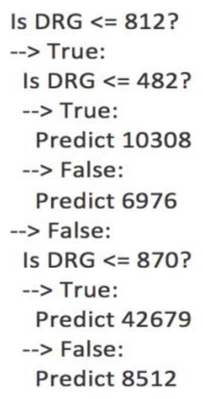
where n is the total number of records in the test set, j is the record in a test set, y_j is the actual value of a label and $\hat{y}_j$ is a prediction.

We used this metric as it can be used for all machine learning models we have implemented. Hence, it will be easy to compare the results for each model.

6.1.5 Regression Tree Figures

The following are some example figures of regression trees built from our regression tree implementation:

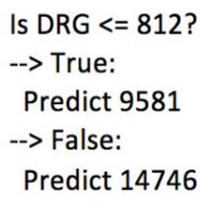
1. Example 1:



```
Is DRG <= 812?
--> True:
 Is DRG <= 482?
 --> True:
  Predict 10308
 --> False:
  Predict 6976
--> False:
 Is DRG <= 870?
 --> True:
  Predict 42679
 --> False:
  Predict 8512
```

Fig.10. Example 1: Regression Tree built from implemented algorithm

2. Example 2:



```
Is DRG <= 812?
--> True:
 Predict 9581
--> False:
 Predict 14746
```

Fig.11. Example 2: Regression Tree built from implemented algorithm

In the next section, we provide the implementation details of the random forest regression algorithm.

6.2 Random Forest Regression Implementation

We introduced ensemble learners and random forests in the Chapter 3. In this section, we will give the implementation details of a Random Forest Regression algorithm which we implemented, run and tested in our project. Like CART, Random Forest algorithm can also be used for both regression as well as classification tasks. It is a supervised learning algorithm where we train the algorithm on some data and test its performance on new data. The problem with decision trees is they can be easily overfit. Random forest tries to avoid this problem of overfitting. As the name suggests Random Forest consist of number of trees. Decision Trees act as a weak learner for it. The strategy used while building the random forest is creating a strong learner from multiple weak learners which will minimize the errors produced by weak learners. Random Forest tries to minimize overfitting by introducing randomness in the creation of trees in the forest. The randomness in introduced in selection of features and selection of subsamples used while building each tree.

6.2.1 Working of Random Forest Regression

The algorithm works in a following way [30]:

1. Suppose there are n number of trees in the forest. Then for each tree a random subsample of train data with replacement is selected.
2. Suppose there are total m features in the feature set. Then, for each tree in the forest, the algorithm selects "k" features out of "m". It depends upon the value of m, what should be the value of k. For example, you can select k=1 or you can select all m values to build each tree, in that case k==m. If you are selecting the criteria as k<<m then the possible values of m are, 1/2√m, √m, 2 √m.
3. Once the features are selected for each tree, then each tree is built using the regression tree algorithm we specified in previous sections.

4. Steps 1, 2 and 3 are repeated until all n number of trees are built. Once all trees are built, then test records are passed to each tree for prediction. Each tree predicts the output and the final prediction is the average of all predictions from each tree in the forest.

The method Random Forest uses to build trees and make prediction is called bagging or bootstrap aggregation. Fig.12 explains random forest regression [39].

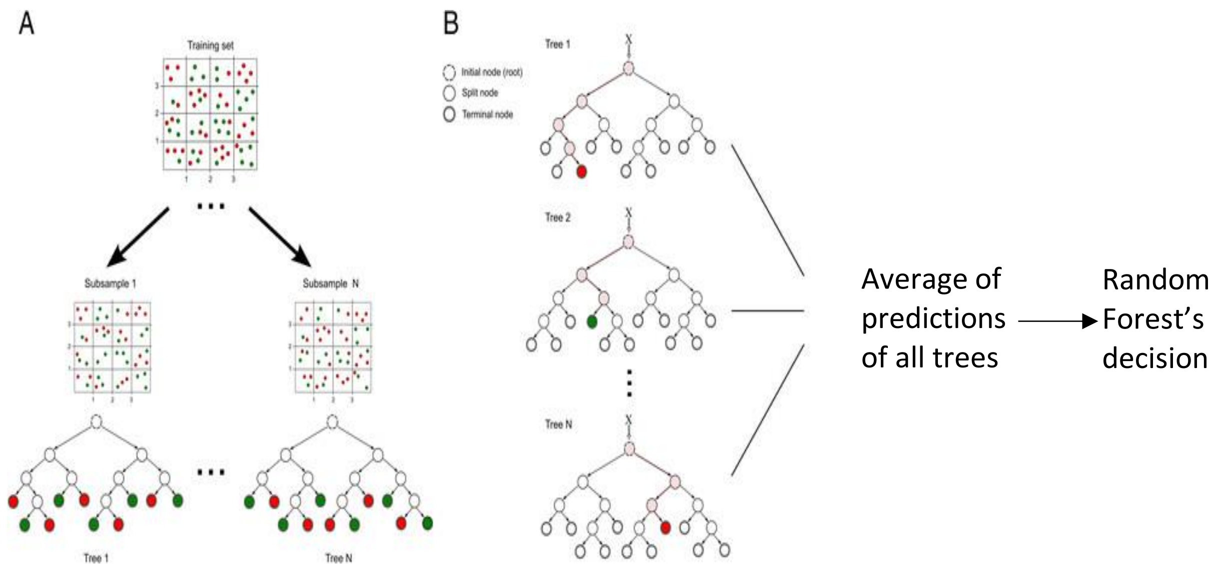


Fig. 12. Subsampling and Random Forest

In Fig. 10, step A shows a subsample of a train data given to each tree in the forest. Picking a subsample with replacement means when a record "r" is selected for subsample 1 then it can be again picked for subsample 2. This method does not remove a picked record from the original train data. The approach of subsampling without replacement can also be used to create subsamples. Step B in Fig 10. shows how each tree will be different when built.

6.2.2 Error Calculation

To calculate the accuracy of a model we used the same criteria as the regression trees which is mean absolute error.

In the next chapter we give the description of two algorithms Gradient Boosted Trees and Linear Regression.

CHAPTER 7

Gradient Boosted Trees and Linear Regression

In the previous chapter, we gave the implementation details of algorithms which we have implemented from scratch. The second part of our project is to make use of two other algorithms and get the price prediction results for the same dataset which we have used earlier. The algorithms we are going to use are Gradient Boosted Decision Trees [29] and Linear Regression [38]. To implement these algorithms on our dataset we are going to use available libraries, GradientBoostingRegressor and LinearRegression from "scikit learn" toolkit [6]. Scikit learn toolkit is a free machine learning library for python programming language. It has a wide range of machine learning as well as feature pre-processing in-built libraries. A user has to import these libraries in the program in order to use them.

8.1 Gradient Boosted Decision Trees

Gradient Boosted Decision Trees are considered as an ensemble learning method which use decision trees as weak learners for prediction. They can be used for classification as well as regression task. To build the trees, they use a mechanism called boosting.

Boosting consists of iteratively training weak learners with respect to a distribution of the training data and adding these weak learners to a final strong learner. When they are added, they are weighted in way that is usually related to the weak learners' accuracy. Each time a weak learner is added, the data are reweighted that is, the examples that are not predicted correctly gain weight and examples that are predicted correctly lose weight. It depends on the particular boosting algorithm to be used which strategy to use to add weights. Thus, future weak learners focus more on the examples that previous weak learners did not predict well.

Boosting is considered a form of numerical optimization problem. A numerical optimization problem tries to minimize a certain function f(X). In the case of gradient boosting, the goal is to minimize the loss of the entire model by adding weak learners using a gradient descent like procedure.

8.1.1 Gradient Boosting Working

As mentioned in the previous section, a gradient boosting tries to minimize the defined loss function. Hence there are three important things to be considered in case of gradient boosting. Those are loss of a function, weak learner and additive model strategy to be used to minimize the loss function.

Loss Function:

Depending upon the problem to be solved a loss function can be any function which is differentiable. For example, for the regression task a squared error is a common loss function which can be used on the other hand for the classification task a logarithmic loss is a common loss function which can be used.

Weak Learner - Decision Trees:

Predictions are made using decision trees as weak learners in gradient boosted decision trees. Specifically, regression trees are used which output real values for splits and whose output can be added together. They allow subsequent models' outputs to be added and "correct" the residuals in the predictions.

Trees are constructed in a greedy manner like random forest, choosing the best splits to minimize the loss or to maximize the information gain.

Additive Model Strategy:

Trees are added in a sequential manner. The entire model is constructed in a stage wise fashion. At each stage, new tree is added and which does not impact the already existing trees. At each stage while adding the new tree the model tries to minimize the total loss. This strategy is called as additive model strategy in which each tree is added in additive manner at each stage minimizing the loss every time. After calculating error or loss at each stage, the weights are updated to minimize that loss. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model. Fig.13 shows gradient boosting procedure [41].

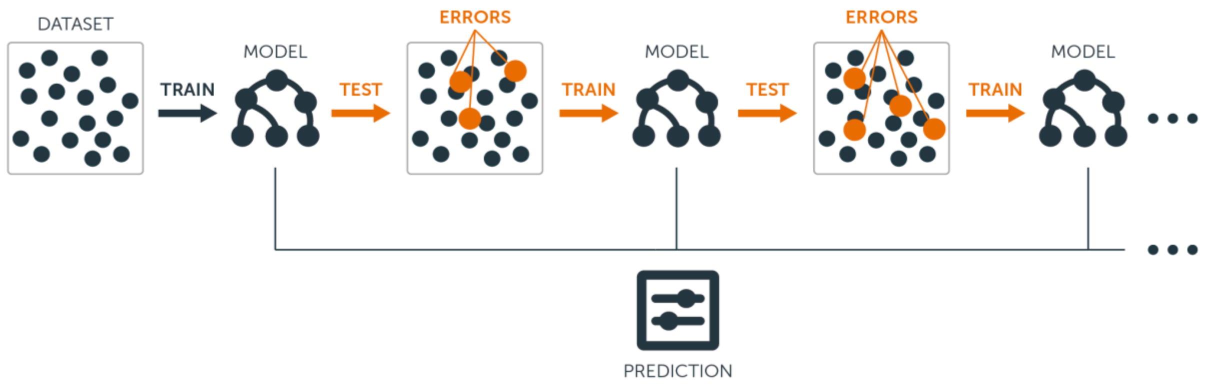


Fig. 13. An example of Gradient Boosted Decision Trees

The final output for a new record in Fig. 13 is done by taking weighted average of predictions from all the trees.

8.1.2 Difference between Random Forest and Gradient Boosted Trees

Table IV lists the differences between Random Forest and Gradient Boosted Trees [27].

TABLE IV

Difference between Random Forest and Gradient Boosted Trees

Random Forest	**Gradient Boosted Trees**
Reduces error mainly by reducing variance.	Reduces error mainly by reducing bias.
Trees are built in parallel manner.	Trees are built in sequential manner.
The tree built in each stage is trained independently from all other previously trained trees.	The tree built in each stage is trained to improve the performance of already trained trees.
Each time the tree is built by taking random subsample from the training data with replacement. This method is called bagging.	Each time the tree is built by taking random subsample from the training data but the subsequent samples depend on weights given to records in the previous sample which did not predict correctly. This method is called boosting.
The final prediction is the average of all the predictions from all the trees.	The final prediction is the weighted average of predictions from all the trees.
Can be built and tuned with keeping only few hyper-parameters in consideration.	There are too many hyper-parameters to be considered to build and tune the tree.

8.2 Linear Regression

In statistics, a linear regression is a method to find a relationship between a scalar dependent variable y and one or more explanatory variables or independent variables called X. When there is only one independent variable it is called a simple linear regression. On the other hand, when there are more than one independent variables, it is called as a multiple linear regression.

In machine learning, a linear regression model tries to model the relationship between two variables by fitting a linear equation to observed data. Observed data is the training data.

For example, a modeler might want to relate the area of a rectangles to their heights using linear regression.

A linear regression line has an equation of the form,

$$Y = c + mX$$

where X is the independent variable and Y is the dependent variable. The slope of the line is m, and c is the intercept that is the value of y when x = 0.

In machine learning, we pass the training data to a model and train it so as when the new data come a model predicts the outcome for it. For example, (x1, y1), (x2, y2), (x3, y3) till (xm, ym) can be training data. Then the model uses these set of points to find the coefficient m and the constant c such that they fit in the equation mentioned above.

Once it is done, when the new point (xn) is given as a test record the model predicts the value of yn for this value for x.

The most common method for fitting a regression line is the method of least-squares. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. If a point lies on the fitted line exactly, then its vertical deviation is 0. Fig.14 shows simple linear regression [40].

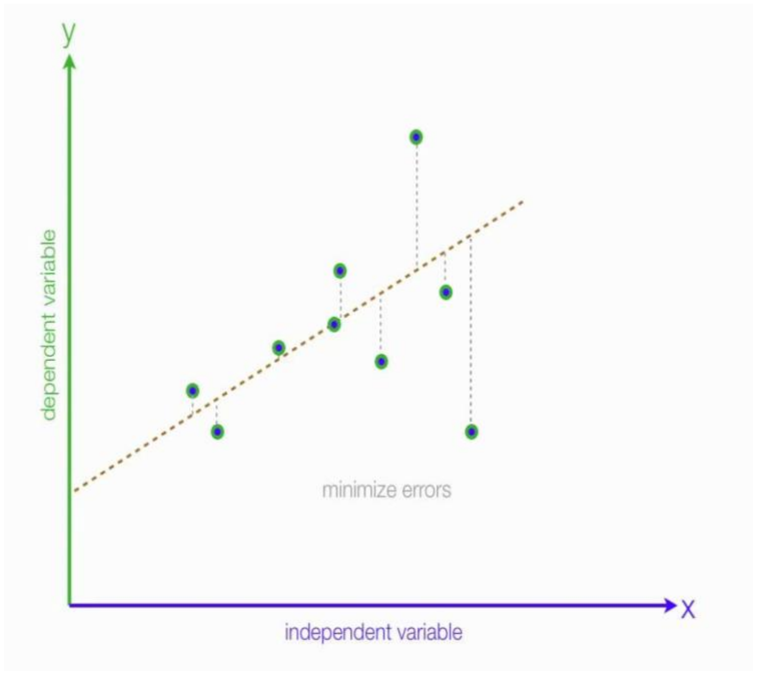


Fig. 14. Simple Linear Regression

CHAPTER 8

In this chapter, we report and then discuss the results of prediction from the algorithms mentioned in Chapter 6 and 7.

Results

1. Parameters:

Maximum depth of a tree = 1

Number of folds in cross-validation= 2

Minimum samples reached at leaf= 1

Number of trees in Random Forest (used only for Random Forest) = 2

Loss function used for Regression Tree and Random Forest= variance reduction

Loss function used for Gradient Boosted Regression Trees = least absolute deviation

TABLE V

Mean Absolute Error of each model for Configuration 1

Regression Tree	Random Forest Regression	Gradient Boosted Regression Trees	Linear Regression
Mean Absolute Error = 0.612	Mean Absolute Error = 0.617	Mean Absolute Error = 0.385	Mean Absolute Error = 0.610

2. Parameters:

Maximum depth of a tree = 3

Number of folds in cross-validation= 5

Minimum samples reached at leaf= 1

Number of trees in Random Forest (used only for Random Forest) = 5

Loss function used for Regression Tree and Random Forest= variance reduction

Loss function used for Gradient Boosted Regression Trees = least absolute deviation

TABLE VI

Mean Absolute Error of each model for Configuration 2

Regression Tree	Random Forest Regression	Gradient Boosted Regression Trees	Linear Regression
Mean Absolute Error = 0.546	Mean Absolute Error = 0.510	Mean Absolute Error = 0.243	Mean Absolute Error = 0.610

3. Parameters:

Maximum depth of a tree = 5

Number of folds in cross-validation= 5

Minimum samples reached at leaf= 2

Number of trees in Random Forest (used only for Random Forest) = 5

Loss function used for Regression Tree and Random Forest= variance reduction

Loss function used for Gradient Boosted Regression Trees = least absolute deviation

TABLE VII

Mean Absolute Error of each model for Configuration 3

Regression Tree	Random Forest Regression	Gradient Boosted Regression Trees	Linear Regression
Mean Absolute Error = 0.460	Mean Absolute Error = 0.450	Mean Absolute Error = 0.175	Mean Absolute Error = 0.610

4. Parameters:

Maximum depth of a tree = 5

Number of folds in cross-validation= 3

Minimum samples reached at leaf= 5

Number of trees in Random Forest (used only for Random Forest) = 5

Loss function used for Regression Tree and Random Forest= variance reduction

Loss function used for Gradient Boosted Regression Trees = least absolute deviation

TABLE VIII

Mean Absolute Error of each model for Configuration 4

Regression Tree	Random Forest Regression	Gradient Boosted Regression Trees	Linear Regression
Mean Absolute Error = 0.452	Mean Absolute Error = 0.418	Mean Absolute Error = 0.181	Mean Absolute Error = 0.613

5. Parameters:

Maximum depth of a tree = 7

Number of folds in cross-validation= 7

Minimum samples reached at leaf= 5

Number of trees in Random Forest (used only for Random Forest) = 7

Loss function used for Regression Tree and Random Forest= variance reduction

Loss function used for Gradient Boosted Regression Trees = least absolute deviation

TABLE IX

Mean Absolute Error of each model for Configuration 5

Regression Tree	Random Forest Regression	Gradient Boosted Regression Trees	Linear Regression
Mean Absolute Error = 0.329	Mean Absolute Error = 0.320	Mean Absolute Error = 0.124	Mean Absolute Error = 0.615

In this work, we test each model on a few test configuration parameters. In the future, we can test the results on all possible combinations of the test configuration parameters. The results may vary when tested on all possible combinations.

In the mentioned configurations, we are gradually increasing the depth of a tree, number of folds in cross-validation, number of trees in the forest and number of samples reached at leaf. From the results, it is clear that for tree based machine learning models such as regression trees, random forest regression and gradient boosted regression trees the performance improved with increasing the previously mentioned parameters. For linear regression, we performed testing with only one parameter which is number of folds in cross-validation.

Table IX shows that Gradient Boosted Regression Trees performed the best giving ~88% accuracy. Regression Trees and Random Forest Regression, which we implemented from scratch gave ~70% accuracy. Linear regression gave ~39% accuracy. These results can be improved by testing on all the possible combinations of test parameters and also adding more parameters while testing.

In the next chapter, we conclude the work and suggest future directions.

CHAPTER 9

Conclusion

In this project, we have successfully implemented Regression Trees and Random Forest Regression algorithms from scratch to predict the medical prices from the input dataset. We also compared the results of Regression Trees, Random Forest Regression, Gradient Boosted Regression Trees and Linear Regression for the same dataset. From the results, we cannot conclude which model performed the best because the model performance can vary depending upon the configuration tried while testing. Hence, the model performing best for some configurations can give unsatisfactory results for some other configurations. Overall for the test configuration parameters, the order of performance of each model from the best to worst is Gradient Boosted Regression Trees, Random Forest Regression, Regression Trees and Linear Regression. The average medical payments predicted by Gradient Boosted Regression Trees, Random Forest Regression and Regression Trees are close to the actual values of payments.

Future Work

As possible future work for this project, we can add new features to the dataset we are already using. These newly added features can be totally new or can be derived from the dataset itself. For example, we can calculate distinct DRG counts feature from the DRG Definition column. It will give the count of each unique DRG in the entire dataset. We think it will be useful because, the dataset has the top 100 most frequently billed Diagnostic-Related Groups (DRGs) and a wide variation in the prices for a given DRG among different providers is observed. Hence, it will be important to know if the particular DRG has more impact on the medical price. The addition of more features can improve the accuracy of prediction as we can add randomness in the selection of features while building the individual trees for the Random Forest algorithm.

Another addition in the future work can be, making the system even more scalable. Right now we are using few thousands of records to train and test the algorithm. In future, we can try to scale the algorithm for a larger dataset having at least a million records and see the results for it. To make the system scalable we can make use of distributed frameworks like Spark and Hadoop. These frameworks can handle big data efficiently.

References:

[1] A. Tike and S. Tavarageri. (2017). A Medical Price Prediction System using Hierarchical Decision Trees. In: IEEE Big Data Conference 2017. IEEE.

[2] "National Health Expenditures 2015 Highlights," CMS.gov, 2015. [Online]. Available:https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/downloads/highlights.pdf

[3] J. Cubanski, C. Swoop and T. Neuman,"How Much Is Enough? Out-of-Pocket Spending Among Medicare Beneficiaries: A Chartbook," The Henry J. Kaiser Family Foundation, Menlo Park, CA, 2014. [Online]. Available: http://files.kff.org/attachment/ how-much-is-enough-out-of-pocket-spending-among-medicare-beneficiaries-a-chartbook-

[4] J. Cubanski and T. Neuman,"The Facts on Medicare Spending and Financing," The Henry J. Kaiser Family Foundation, Menlo Park, CA, 2017. [Online]. Available: http://files.kff.org/attachment/Issue-Brief-The-Facts-on-Medicare-Spending-and-Financing. [Accessed Nov. 2, 2017].

[5] The Henry J. Kaiser Family Foundation. (2017). Total Number of Medicare Beneficiaries. [Online]. Available: https://www.kff.org/ medicare/state-indicator/total-medicare-beneficiaries/. [Accessed Nov. 2, 2017].

[6] Scikit-learn.org. (2018). About us — scikit-learn 0.19.1 documentation. [online] Available at: http://scikit-learn.org/stable/about.html#people [Accessed 23 Apr. 2018].

[7] R. Hafezi, J. Shahrabi, and E. Hadavandi, "A bat-neural network multi- agent system (bnnmas) for stock price prediction: Case study of dax stock price," Applied Soft Computing, vol. 29, pp. 196–210, 2015.

[8] H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky, "On the importance of text analysis for stock price prediction." in LREC, 2014, pp. 1170–1175.

[9] P.-C. Chang and C.-H. Liu, "A tsk type fuzzy rule based system for stock price prediction," Expert Systems with applications, vol. 34, no. 1,pp. 135–144, 2008.

[10] K. Kohara, T. Ishikawa, Y. Fukuhara, and Y. Nakamura, "Stock price prediction using prior knowledge and neural networks," Intelligent systems in accounting, finance and management, vol. 6, no. 1, pp. 11–22, 1997.

[11] K.-j. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," Expert systems with Applications, vol. 19, no. 2, pp. 125–132, 2000.

[12] S. C. Bourassa, E. Cantoni, and M. Hoesli, “Spatial dependence, housing submarkets, and house price prediction,” The Journal of Real Estate Finance and Economics, vol. 35, no. 2, pp. 143–160, 2007.

[13] Park and J. K. Bae, “Using machine learning algorithms for housing price prediction: The case of fairfax county, virginia housing data,”Expert Systems with Applications, vol. 42, no. 6, pp. 2928–2934, 2015.

[14] A.-H. Mohsenian-Rad and A. Leon-Garcia, “Optimal residential load control with price prediction in real-time electricity pricing environments,” IEEE transactions on Smart Grid, vol. 1, no. 2, pp. 120–133, 2010.

[15] J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina, “Day-ahead electricity price forecasting using the wavelet transform and arima models,” IEEE transactions on power systems, vol. 20, no. 2, pp. 1035–1042, 2005.

[16] P. Mandal, T. Senjyu, N. Urasaki, T. Funabashi, and A. K. Srivastava, “A novel approach to forecast electricity price for pjm using neural network and similar days method,” IEEE Transactions on Power Systems, vol. 22, no. 4, pp. 2058–2065, 2007.

[17] J. L. Moran, P. J. Solomon, A. R. Peisach, and J. Martin, “New models for old questions: generalized linear models for cost prediction,” Journal of evaluation in clinical practice, vol. 13, no. 3, pp. 381–389, 2007.

[18] S. Sushmita, S. Newman, J. Marquardt, P. Ram, V. Prasad, M. D. Cock, and A. Teredesai, “Population cost prediction on public healthcare datasets,” in Proceedings of the 5th International Conference on Digital Health 2015. ACM, 2015, pp. 87–94.

[19] Lahiri and N. Agarwal, “Predicting healthcare expenditure increase for an individual from medicare data,” in Proceedings of the ACM SIGKDD Workshop on Health Informatics, 2014.

[20] Gregori, M. Petrinco, S. Bo, A. Desideri, F. Merletti, and E. Pagano, “Regression models for analyzing costs and their determinants in health care: an introductory review,” International Journal for Quality in Health Care, vol. 23, no. 3, pp. 331–341, 2011.

[21] Bertsimas, M. V. Bjarnad´ottir, M. A. Kane, J. C. Kryder, R. Pandey, S. Vempala, and G. Wang, “Algorithmic prediction of health-care costs,” Operations Research, vol. 56, no. 6, pp. 1382–1392, 2008.

[22] P. C. Austin, V. Goel, and C. van Walraven, “An introduction to multilevel regression models,” Canadian Journal of Public Health, vol. 92, no. 2, p. 150, 2001.

[23] Y.-L. Chen, H.-W. Hu, and K. Tang, “Constructing a decision tree from data with hierarchical class labels,” Expert systems with applications, vol. 36, no. 3, pp. 4838–4847, 2009.

[24] Differencebetween.com. (2017). Difference Between Classification and Regression. [Online]. Available: http://www.differencebetween.com/difference-between-classification-and-vs-regression/ [Accessed 16 Nov. 2017]

[25] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” IEEE transactions on systems, man, and cybernetics,vol. 21, no. 3, pp. 660–674, 1991.

[26] J. Mingers, “An empirical comparison of selection measures for decision-tree induction,” Machine learning, vol. 3, no. 4, pp. 319–342,1989.

[27] Cda.psych.uiuc.edu. (2017). [Online]. Available: http://cda.psych.uiuc.edu/multivariate_fall_2012/systat_cart_manual.pdf [Accessed 8 Nov. 2017].

[28] Liaw, M. Wiener et al., “Classification and regression by randomforest,” R news, vol. 2, no. 3, pp. 18–22, 2002.

[29] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” Annals of statistics, pp. 1189–1232, 2001.

[30] L. Breiman. Random forests. Machine Learning, 45:5–32, 2001.

[31] “Inpatient Prospective Payment System (IPPS) Provider Summary for the Top 100 Diagnosis-Related Groups (DRG) - FY2011”. [online] Available at: https://data.cms.gov/Medicare-Inpatient/ Inpatient-Prospective-Payment-System-IPPS-Provider/97k6-zzx3, 2017.

[32] “Zillow Data”. [online] Available at: https://www.zillow.com/research/data/

[33] “Design and development of the Diagnosis Related Group (DRG),” https://www.cms.gov/ICD10Manual/version34-fullcode-cms/fullcode cms/Design and development of the Diagnosis Related Group (DRGs) PBL-038.pdf.

[34] TSK IV. (1992). Frankfurt am Main: Fachbereich Geowiss. der Johann-Wolfgang-Goethe-Univ

[35] Stat.wisc.edu. (2018). [online] Available at: http://www.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf [Accessed 23 Apr. 2018].

[36] En.wikipedia.org. (2018). Cross-validation (statistics). [online] Available at: https://en.wikipedia.org/wiki/Cross-validation_(statistics) [Accessed 23 Apr. 2018].

[37] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.

[38] En.wikipedia.org. (2018). Mean absolute error. [online] Available at: https://en.wikipedia.org/wiki/Mean_absolute_error [Accessed 23 Apr. 2018].

[39] Aggiwal, R. (2018). Introduction to Random forest | Dimensionless: Learn Data Science & Analytics. [online] DIMENSIONLESS TECHNOLOGIES PVT.LTD. Available at: https://dimensionless.in/introduction-to-random-forest/ [Accessed 23 Apr. 2018].

[40] Statistics Solutions. (2018). What is Linear Regression? - Statistics Solutions. [online] Available at: https://www.statisticssolutions.com/what-is-linear-regression/ [Accessed 23 Apr. 2018].

[41] The Official Blog of BigML.com. (2018). Introduction to Boosted Trees. [online] Available at: https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/ [Accessed 23 Apr. 2018].