DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF SCIENCE TECHNOLOGY AND AGRICULTURE
UNIVERSITY OF CALCUTTA

SURVIVABILITY PREDICTION OF CANCER PATIENTS USING MACHINE LEARNING ALGORITHMS ON MONGODB

A PROJECT REPORT SUBMITTED AS A PART OF FULFILLMENT OF MSC IN COMPUTER SCIENCE

SUBMITTED BY:
HANNAH BRIJIT
ROLL NO:- C91/CSC/201006
REGISTRATION NUMBER:- D01-1211-0010-20

RIPAN ADAK
ROLL NO:- C91/CSC/201017
REGISTRATION NUMBER:- 562-1111-0112-17

RUPAL MONDAL
ROLL NO:- C91/CSC/201018
REGISTRATION NUMBER:- 221-1212-0284-17

UNDER THE GUIDANCE OF
DR. SUNIRMAL KHATUA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF CALCUTTA

Acknowledgement

We are highly grateful to **Dr. Sunirmal Khatua**, our project supervisor, as he has given us the opportunity to work under his guidance on the most promising and emerging topic of today’s world. He has immensely motivated us as well as guided us to do our project. Along with him **Mr. Sudip Mondal**, one of the Ph.D. research scholars working under **Dr. Sunirmal Khatua**, too has helped us a lot in all aspects. We are obliged and thankful to them for their cooperation in every aspect of the project.

HANNAH BRIJIT
Roll No.: C91/CSC/201006
M.Sc. 4th Semester
University of Calcutta

RIPAN ADAK
Roll No.: C91/CSC/201017
M.Sc. 4th Semester
University of Calcutta

RUPAL MONDAL
Roll No.: C91/CSC/201018
M.Sc. 4th Semester
University of Calcutta

UNIVERSITY OF CALCUTTA

Certificate

This is to certify that the project entitled **“Survivability Prediction of Cancer Patients using Machine Learning Algorithms on MongoDB”** submitted for fulfillment of the requirements of 4th semester of M.Sc. Computer Science has been completed by
Ripan Adak (RollNo.C91/CSC/201017),
Rupal Mondal (Roll No. C91/CSC/201018) and
Hannah Brijit (Roll No. C91/CSC/201006) under the guidance of Dr. Sunirmal Khatua, Assistant Professor, Department of Computer Science & Engineering, University of Calcutta.

Signature of Supervisor

Signature of Chairman, BoS

Signature of Examiner

Preface

Survival Analysis is a study in the field of statistics for periodical analysis of an event among a particular population. To understand the proportional estimation of recovery and death for a particular disease, for example, estimating survival probability of a patient, the amount of risk factor for a treatment, several different regression modeling methods can be used. To measure the probability of survival for some future events along with the evaluation of death rate, a comparative study of several prediction algorithms has been made. We discuss in this article about the estimation of how good a model is for survival prediction and how to make it even better by some modification of its parameters, no. of features used for prediction or slightly altering the build of the model itself. The first section in our experiment describes the datasets that have been used for this study and their feature engineering. The second section focuses on analysis of the life cycle or predicting survival time and risk factors, using different survival regression and discussion of their respective performances. The third section highlights how to enhance the performance of these algorithms, by tuning their respective parameters, extracting the most important features from the data to improve running time and incorporating the above-mentioned study in a different data environment. We have performed the statistical regression using Kaplan-Meier (KM) for each predictor variable, Cox Proportional-Hazards (CoxPH) for multivariate analysis. To leverage the tree structure of the algorithms for survival prediction, Extreme Gradient Boosting (XGBoost) and Adaptive Regression (AdaBoost) have been implemented. Parameters of these algorithms are tuned and feature selection of the datasets are performed for better results. Our comparative study is tested upon two datasets, one is related to the event of heart failure and the other is of colon cancer patients. Both the datasets have shown consistent results for our study and XGBoost and its parameter tuned algorithm proved to be superior with respect to the others in terms of accuracy.

Table of Content

1. Problem Statement

Survival analysis of cancer patients on a distributed framework i.e., MongoDB using different machine learning ensemble methods, followed by a comparative study of their performances.

2. Introduction

Survival analysis, or more generally, time-to-event analysis, refers to a set of methods for analyzing the length of time until the occurrence of a well-defined end point of interest. The time starting from a defined point to the occurrence of a given event, is called survival time and the analysis of group data as survival analysis. The time until a person has an event of interest is represented by a time to event variable (e.g., heart attack, cancer remission, death). Because of the specific characteristics of time to event variables, statistical analysis of these variables necessitates approaches that differ from those discussed thus far for other types of outcomes. Survival analysis focuses on two important pieces of information:

1. Whether or not a participant suffers the event of interest during the study period. A dichotomous or indicator variable often coded as 1=event occurred or 0=event did not occur during the study observation period.
2. The follow up time for each individual being followed.

Time zero, or the time origin, is the time at which participants are considered at-risk for the outcome of interest. In many studies, time at risk is measured from the start of the study. Follow up time is measured from time zero until the event occurs, the study ends or the participant is lost; whichever comes first.

Medical researchers and data analysts first created and employed survival analysis to determine the life spans of a population. However, it has been utilized in a variety of other applications over the years, including anticipating churning customers/employees, estimating the lifetime of a machine, and so on.

The basic quantity employed to describe time-to-event phenomena is the survival function, the probability of an individual surviving beyond time x (experiencing the event after time x). It is defined as

$$S(x) = Pr\ (X > x)$$

In the field of engineering, S(x) is referred to as the reliability function. If X is a continuous random variable, then, S(x) is a continuous, strictly decreasing function. The survival curve is the graph of S(x) as a function of x. Survival analysis [1] [2] considers two variables, one is the survival variable, and another is the predictor variable(s). The predictor variables are called covariates. Here, the event of interest is known as a hazard. The difference in the actual result which is to happen and the predicted output is named as an error of the survival function. It is needed to make the error as small as possible, such that the predicted output becomes closer to the actual result. So, increasing the accuracy of the survival function becomes one of the major concerns.

For survival analysis, there are some dedicated models such as,
i) Kaplan Meier Method
ii) Cox Regression Method

Kaplan Meier is a non-parametric method, but it doesn’t handle multivariates and it is mainly used for classification problems. On the other hand, Cox Regression handles both classification and regression problems, also handles multivariates, but it is semi-parametric. To overcome the shortcomings of Kaplan-Meier and Cox regression, we use machine learning algorithms such as Adaptive Boosting Regression (Adaboost) and Extreme Gradient Boosting Regression (XGBoost).

Adaboost is a popular tree-based ensemble method. It takes many weak learners additively to form a strong learner. We apply them in survival prediction. We propose to use Adaboost with different weak learners such as decision tree, random forest, K-nearest neighbor (KNN) to see whether we can optimize the already existing algorithm, giving better accuracy. Although it is used for prediction in classification tasks, here we use adaboost as a regression problem.

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The main advantages of XGBoost are its very high speed compared to other algorithms, such as AdaBoost, and its regularization parameter that successfully reduces variance. XGBoost consists of a number of hyper-parameters that can be tuned, which is a primary advantage over gradient boosting machines. In this project, we tried to increase the accuracy of the model by tuning those parameters and compared all the results.

Data Storage is one of the most challenging tasks in a big data framework. Therefore, traditional data analysis is unfit to manage those systems. Moreover, different databases are required for horizontal and for vertical data integration because each of these approaches address different aspects of the integration problem. These problems can be easily handled using MongoDB, which is an open-source NoSQL database management program. We have implemented these algorithms on MongoDB as well to address the issue regarding data storage. All of these algorithms and methods will be covered in detail in the proposed methodologies section of this document.

3. Goal

The goal of this project is to study and execute different machine learning algorithms for survival analysis, optimize their performances by parameter tuning and implement them on a distributed framework, i.e., MongoDB.

4. Methodologies Applied

4.1. Kaplan Meier

Edward L. Kaplan and Paul Meier collaborated on a seminal article on dealing with incomplete observations in 1958 [3]. As a result, Kaplan-Meier curves and survival data estimations have become a common way of dealing with varying survival times, particularly when not all of the subjects remain in the research. The Kaplan-Meier estimate is one of the finest methods for calculating the percentage of subjects who live for a specific period of time post treatment. Clinical trials are controlled investigations that are done on human volunteers to compare efficacy and safety. The impact of an intervention is measured in clinical or community studies by calculating the number of individuals who survived or were saved after the intervention over time. The amount of time that elapses before an event occurs would be the variable of concern in situations when survival is the issue. In certain scenarios, such as cancer treatment, this period of time is quite lengthy; in such cases, the number of events such as death may be measured per unit of time. In some cases, the time it will take for a cancer to recur or for an infection to arise may be calculated. The survival time is defined as the time from a specified point to the occurrence of a specific event, and the survival analysis is defined as the study of group data.

These analyses are sometimes challenging when research subjects are uncooperative and refuse to stay in the study, or when some of the subjects may not undergo the event or die by the end of the research, or when we lose contact with them halfway through the study. We sometimes have subjects who join the research later, after a large amount of time has passed since the study began. For such subjects, we have a shorter observation period, and they may or may not experience the event within that time. However, we cannot eliminate such subjects since the study's sample size would otherwise be too small. Despite all of the challenges connected with subjects or circumstances, the Kaplan-Meier estimate is the simplest means of estimating survival over time.

Kaplan Meier (KM) method is one of the majorly used methods to analyse 'time-to-event' data. The KM method is a method of summarizing survival data, which uses all the cases in a series, not just those followed up until the selected cut-off. The technique is to divide the follow-up period into a number of small-time intervals, determining for each interval the number of cases followed up over that interval and the number of events of interest such as death, during each period. When the surviving proportion is multiplied by the surviving proportions for each of the preceding time periods, a probability of surviving to the end of that time period is obtained. This survival probability is then plotted against time. In statistical analysis, the method which is simply not based on the probability distribution is called nonparametric statistics [4]. Kaplan Meier estimator [5] is an example of nonparametric statistics. A function that shows results in the probability of survival of a patient or any object in a given time is denoted as survival function. Kaplan Meier estimator estimates survival function using "lifetime data". It provides the estimation of the survival curve, but any additional covariate can't be allowed. In this circumstance, the Cox Regression method comes into the platform. In cox regression, more than one covariates can take a part in the run. The ultimate purpose of the Cox proportional hazard method is to notice how different factors in our dataset impact the event of interest.

Let a random variable is $p \geq 0$. The main aim of the Kaplan Meier method is to approximate the survival function S underlying p. The survival function is interpreted as,

$$S(p) = \text{Prob}\ (p > t),$$

where t is denoted as the time and has the value equals to 0,1,2,… Let, p, p_2 , $p_3 \geq 0$ are variables that are independent, random, and identically distributed. Likewise, p_i is denoted as a random time when some event ‘i’ happened. The data which are required to estimate S is the pair-list,$((P_i, v_i))_{i=0,1,2,\ldots}$, where P_i is min(p_i,v_i) and i ϵ[n]. This information denotes whether the event i happened before the time v_i or not.

The Kaplan-Meier survival curve is defined as the probability of living for a given amount of time when time is divided into numerous short intervals. This study is based on three assumptions. To begin, we assume that patients who are censored have the same chances of survival as those who are tracked indefinitely. Second, we assume that the survival rates for subjects recruited early and late in the research are the same. Finally, we presume that the event occurs at the specified time. Individuals may be followed up on more regularly and at shorter time intervals to provide a more accurate estimate of survival. The "product limit estimate" is another name for the Kaplan-Meier estimate. It entails calculating the probabilities of an event occurring at a certain moment in time. To reach the final estimate, we multiply these sequential probabilities by any previously estimated probabilities.

Although Kaplan Meier is one of the most used survival analysis techniques, it has certain flaws that may be addressed by alternative approaches. Since the Cox Regression method is a multivariate regression approach, and KM Survival Analysis is univariate, meaning it can only operate with one predictor variable at a time, KM alone is not appropriate for our datasets with many predictor variables. In addition, Cox Regression is a semi-parametric method, while KM is a non-parametric technique. Finally, although Cox Regression may use both continuous and binary variables, KM Survival Analysis can only employ categorical predictors.

4.2. Cox Proportional Hazard Regression

The Cox proportional hazards (CPH) model, or simply, the Cox model (Cox 1972, 1975) is named after Sir David Cox, a statistician who developed it. The findings of Cox's original study have been extensively applied to issues in many scientific areas, and his work in this area will undoubtedly be acknowledged as one of the twentieth century's greatest statistical triumphs. The Cox model has a clear and straightforward logic.

The Cox proportional hazards model has been the most widely used procedure to analyse time-to-event curves, over many years of experience in many fields, especially medical research because of its applicability to a wide variety of types of clinical studies. It assesses the influence on survival of different attributes. It helps the analysis of the effect of defined variables on the rate of occurrence of a specific event at a specific moment. The CPH [6] is a semi-parametric model. The specific form of the baseline hazard rate, h0(t), is considered to be unknown and left unparameterized when the proportional hazards assumption applies in the Cox model. As a result, the Cox model is frequently referred to as a "semi-parametric" model: the (ordered) duration times are parameterized in terms of a set of covariates, but the duration times' precise distributional structure is not parameterized.

Cox PH regression model is represented by h(t), that is, the hazard function. So, it is possible to view the likelihood of death at time t as our hazard function.
Cox Hazard is valued as follows,

$$h(t) = h_0(t) \times \exp(k_1x_1 + k_2x_2 + \dots + k_px_p)$$

Here, "t" is used to represent the Survival time. The "h(t)" term, i.e., the hazard function depends on p number of covariates (x_1, x_2, ... x_p). The k_1,k_2,...,k_p are the regression coefficients, measured by the influence (the effect size) of covariates. This is the parametric part of the model. The h_0(t) term is known as the baseline hazard function. If the values of all x_i's are zero (the quantity exp(0) =1) then it is valued the same as the hazard. h(t) is a function of t, i.e., time, may fluctuate with time. This is the non-parametric part of the model.
The Cox PH regression is a multi-linear regression of the hazard's logarithm on x_i variables, where baseline hazard is the 'intercept' of the time dependent equation. The terms exp(b_i) are known as hazard ratios (HR). Here, $k_i > 0$, i.e., having a HR's value more than one means with the increasing value of i-th covariate, the event hazard becomes more, resulting in the decrease of survival length. We can summarize this as,
If the hazard remains same, HR = 1
If the hazard decreases, HR < 1
If the hazard increases, HR > 1

In the CPH model, no functional assumptions are made about the shape of the Hazard Function; instead, functional-form assumptions are made about the effects of the covariates alone.

It's a "robust" model in the sense that the Cox model's outputs will approximate the proper parametric model's results (even though the baseline hazard is not specified). We can also estimate the covariates and regression parameters in the exponential portion of the model even if the baseline hazard element of the model is unclear. The hazard function and the survival curves associated with it may then be calculated. When survival time information is known and censoring is present, it is preferable above the logistic model, since we can get more information.

The main assumption of the classical Cox hazards analysis is the proportionality of the hazards. In the event of a violation of the proportionality of hazard assumption, the use of a simple Cox regression model is incorrect [7]. Treating variables that strengthen as a hazard factor changes and/or disappear during follow-up as constant, significant risk factors of death, may result in a false inference. The scientific community has mostly adopted the CPH model because of its ability to swiftly calculate relevant outcomes and its simplicity of use. Despite being one of the most widely accepted survival prediction models, the CPH model has some downsides: for example, it's an inadequate model for high-dimensional situations, (which means when the number of data instances are lesser than the number of features). Also, it lacks the ability to accurately simulate unanticipated interaction effects and nonlinearities (both of which are ubiquitous in data). Although the Cox model can handle multiple covariates, it does not perform that well in general. The CPH model doesn't reflect adequate accuracy while dealing with real world scenarios.

In order to achieve a better model with better accuracy, we use some of the boosting mechanisms, such as Adaptive boosting and Extreme Gradient boosting.

4.3. Adaptive Boosting Regression

● **Ensemble Machine Learning Approach:** An ensemble [8] is a composite model, combines a series of low performing classifiers with the aim of creating an improved classifier. Here, individual classifier's vote and final prediction label are returned that performs majority voting. Ensembles offer more accuracy than individual or base classifier. Ensemble methods can parallelize by allocating each base learner to different-different machines. Finally, we can say Ensemble learning methods are meta-algorithms that combine several machine learning methods into a single predictive model to increase performance. Ensemble methods can decrease variance using a bagging approach, bias using a boosting approach, or improve predictions using a stacking approach.

Boosting [9] is a class of ensemble machine learning algorithms that involve combining the predictions from many weak learners. A weak learner is a model that is very simple, although has some skill. Boosting was a theoretical concept long before a practical algorithm could be developed, and the adaptive boosting (AdaBoost) algorithm [10] was the first successful approach for the idea. AdaBoost combines the predictions from short one-level decision trees, called decision stumps, although other algorithms can also be used. Decision stump algorithms are used as the AdaBoost algorithm seeks to use many weak models and correct their predictions by adding more weak models.

AdaBoost is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire. The algorithm was developed for classification and involves combining the predictions made by all decision trees [11] in the ensemble. A similar approach was also developed for regression problems where predictions are made by using the average of the decision trees. The contribution of each model to the ensemble prediction is weighted based on the performance of the model on the training dataset.

Decision Stumps [12] are the simplest model we could construct that would just guess the same label for every new example, no matter what it looked like. The accuracy of such a model would be best if we guess whichever answer, 1 or 0, is most common in the data. If, say, 60% of the examples are 1s, then we'll get 60% accuracy just by guessing 1 every time. Decision stumps improve upon this by splitting the examples into two subsets based on the value of one feature. Each stump chooses a feature, say X2, and a threshold, T, and then splits the examples into the two groups on either side of the threshold.

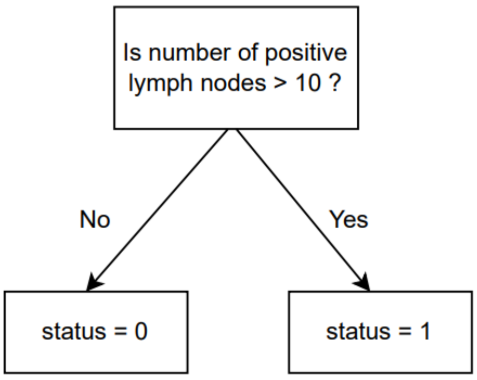


Fig 1: Decision stumps used in AdaBoost algorithm

To find the decision stump that best fits the examples, we can try every feature of the input along with every possible threshold and see which one gives the best accuracy. While it naively seems like there are an infinite number of choices for the threshold, two different thresholds are only meaningfully different if they put some examples on different sides of the split. To try every possibility, then, we can sort the examples by the feature in question and try one threshold falling between each adjacent pair of examples.

While Decision Tree Regressor [13] is used as the default base estimator by AdaBoost Regressor, we can also specify the base estimator from which the boosted ensemble is built. Other base estimators can be Random Forest Regressor, K-Nearest Neighbor (KNN) Regressor, Support Vector Machine (SVM) Regressor etc.

But AdaBoost is not the most accurate machine learning model always, because it performs better for a quality dataset. Noisy data and outliers have to be avoided before adopting an Adaboost algorithm. A newer machine learning model XGBoost performs better for most of the datasets, which was developed for rigorous datasets and executes much faster than AdaBoost.

4.4. Extreme Gradient Boosting Regression

Extreme Gradient Boosting (XGBoost) [14] is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm. Regression predictive modeling problems involve predicting a numerical value such as a dollar amount or a height. XGBoost can be used directly for regression predictive modeling. An efficient, scalable implementation of XGBoost has been published by Tianqi Chen and Carlos Guestrin.

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems. Models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, “gradient boosting” [15], as the loss gradient is minimized as the model is fit, much like a neural network.

Extreme Gradient Boosting, or XGBoost for short, is an efficient open-source implementation of the gradient boosting algorithm. As such, XGBoost is an algorithm, an open-source project, and a Python library. XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its objective function and base learners.

● Major steps for XGBoost are mentioned in brief:

i) Make an Initial Prediction and Calculate Residuals:

Generally, the initial prediction is the average of the values of dependent variable. The Residuals can be calculated by the formula:

Residuals = (Observed values - Predicted values)

ii) Build an XGBoost Tree:

Each tree starts with a single leaf and all the residuals go into that leaf.

Now we have to calculate the Similarity Score using the formula:

$Similarity\ Score = \frac{(Sum\ of\ Residuals)^2}{Number\ of\ Residuals + \lambda}$; where λ is the regularization parameter

Now split the leaf of the tree in different ways and calculate Similarity Score for the leaves.

Then calculate the Gain of splitting the Residuals into two groups. If the Gain is positive, then it’s a good idea to split, otherwise, it is not.

$Gain = Left_{Similarity} + Right_{Similarity} - Root_{Similarity}$

iii) Prune the Tree:

Pruning of the tree is done to avoid overfitting issues. If Gain — γ is positive then we keep the split, otherwise, we remove it. Default value for γ is 0. But it can range between [0, ∞].

iv) Calculate the Output values for Leaves:

The Output of the leaves can be calculated by the formula:

$$Output = \frac{Sum\ of\ Residuals}{Number\ of\ Residuals + \lambda}$$

v) Make New Predictions:

We can make predictions using this formula:

*New Prediction = Previous Prediction + (ε * Output from Final Tree)*

where, ε is Learning Rate

vi) Calculate Residuals Using the New Predictions:

The Residuals are calculated using the formula:

Residuals = (Observed values - New Predicted values)

vii) Repeat Steps (ii) to (vi):

We iterate through step (ii) to step (vi) until the Residuals are super small or we reached the maximum number of iterations we set for our algorithm.

4.5. Performance Metrics

Accuracy: We calculate accuracy in terms of confusion matrix [16] where accuracy is defined by, from all the classes (positive and negative) how many of them we have predicted correctly.

$$Accuracy = (\ all\ correct\ /\ all\ ) = \frac{TP + TN}{TP + TN + FP + FN}$$

There are 4 important terms to mention:

i) True Positive (TP) : We predicted Positive and it’s True.

ii) True Negative (TN) : We predicted Negative and it’s True.

iii) False Positive (FP) : We predicted Positive and it’s False. It is also known as Type-1 Error.

iv) False Negative (FN) : We predicted Negative and it’s False. It is also known as Type-2 Error.

Here, for our regression problem, whenever the predicted value is less than 0.5, then it is considered as a Boolean value False or 0, and for predicted value greater than or equal to 0.5, the outcome is considered as True or 1.

ROC Curve: An ROC curve (receiver operating characteristic curve) [17] is a graph showing the performance of a classification model at all classification thresholds. An ROC curve can be used to visualize the classifier performance at different threshold levels (from 0 to 1). This curve plots two parameters:

- True Positive Rate (Sensitivity)
- False Positive Rate (1 - Specificity)

True Positive Rate (TPR) is defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

4.6. Input datasets

Dataset 1- Chemotherapy for Stage B/C Colon Cancer:

These are data from one of the first successful trials of adjuvant chemotherapy for colon cancer. This dataset has survival and recurrence information on 929 people from a clinical trial on colon cancer chemotherapy. Levamisole is a low-toxicity compound previously used to treat worm infestations in animals; 5-FU is a moderately toxic (as these things go) chemotherapy agent. The study is originally described in Laurie (1989). The main report is found in Moertel (1990).[18] This data set is closest to that of the final report in Moertel (1991).

Data Dictionary of the dataset is mentioned below:

1. **id**: Unique identification number assigned to each patient in the study.

2. **study**: Shows whether a particular patient was present throughout the ***1*** for all patients.
3. **rx**: Treatment - ***1=Obs(ervation)***, ***2=Lev(amisole)***, ***3=Lev(amisole)+5-FU***.
4. **sex**: Gender of the patients in the study (***1=male***, ***0=female***).
5. **age**: Age of the patient in years (***minimum: 18, maximum: 53***).
6. **obstruct**: Obstruction in the colon by the tumor (***0=no obstruction, 1=obstructed***)
7. **perfor:** Gastrointestinal perforation, also known as ruptured bowel is a hole in the wall of part of the gastrointestinal tract (***0=no perforation***, ***1=perforation***).
8. **adhere**: Adherence of tumor to nearby organs (***0=no adherence***, ***1=adherence***).
9. **nodes**: Number of lymph nodes affected with detectable cancer (***minimum: 0, maximum: 33***).
10. **time**: Period of time until the event (death) or censoring (in number of days).
11. **status**: Patients' mortality status (***1=dead***, ***0=censored/alive during and after the event occurred***)
12. **differ**: Results after a biopsy is done on the tumor cell samples (***1=well differentiated***(looks more like normal cells under the microscope and the spread or growth is slower), ***2=moderately differentiated*** (not well-differentiated but also not poorly-differentiated, the cancer cells have started to look different from normal cells but not fully and the growth is bit faster than well-differentiated cells, ***3=poorly differentiated*** (the tumor cells don't look like normal cells at all and the growth is much faster compared to well and moderately differentiated cancer cells).
13. **extent**: Extent of local spread (***1=submucosa, 2=muscle, 3=serosa, 4=contiguous structures***).
14. **surg**: Span of time from registration of the patient to surgery (***0=short, 1=long***).
15. **node4**: More than 4 positive lymph nodes (***1=yes, 0=no***).

Dataset 2- Heart Failure Clinical Records Data Set:

This dataset contains the medical records of 299 patients who had heart failure, collected during their follow-up period, where each patient profile has 13 clinical features. The original dataset version was collected by Tanvir Ahmad, Assia Munir, Sajjad Haider Bhatti, Muhammad Aftab, and Muhammad Ali Raza (Government College University, Faisalabad, Pakistan). The current version of the dataset [19] was elaborated by Davide Chicco (Krembil Research Institute, Toronto, Canada).

Data Dictionary of the dataset is mentioned below:

1. **age:** Age of the patient in years, (***minimum: 40, maximum: 95***).

2. **anaemia**: A condition in which there is a deficiency of red blood cells or of hemoglobin in the blood, resulting in pallor and weariness, (***0: not anaemic, 1: Yes anaemic***).
3. **creatinine_phosphokinase**: Level of the CPK enzyme in a patient's blood in micrograms per liter (mcg/L), CPK enzyme is a protein that helps to elicit chemical changes in your body, (***lowest: 23, highest: 7861***).
4. **diabetes**: If the patient has diabetes or not, (***0: no, 1: yes***).
5. **ejection_fraction**: A measurement of the percentage of blood leaving the heart each time it squeezes, (***minimum: 14, maximum: 80***).
6. **high_blood_pressure**: If a patient has high blood pressure or not, (***0: no, 1: yes***).
7. **platelets**: Level of platelets in patient's blood in platelets/mL, platelets are small colorless disc-shaped cell fragments without a nucleus, found in large numbers in blood and involved in clotting, (***ranges between: 25100-850000***)
8. **serum_creatinine**: Level of creatinine in a patient's blood in mg/dL, creatinine is a chemical compound left over from energy-producing processes in your muscles. The creatinine level is a measure of how well kidneys are performing their job of filtering waste from the blood, (***minimum: 0.5, maximum: 9.4***)
9. **serum_sodium**: Level of sodium in patient's blood in mEq/L, (***highest: 113, lowest: 148***).
    a. Normal sodium levels in the blood are between 135 and 145 mEQ/L.
    b. The high level of sodium in the blood (>145 mEq/L) is known as Hypernatremia that is often caused by dehydration or kidney disease.
    c. Also, the low sodium concentration in the blood (<135 mEq/L) is called Hyponatremia that may be caused by diarrhea and vomiting and are most common in older adults.
10. **sex**: Gender of the patient, (***0: female, 1: male***)
11. **smoking**: If the patient smokes or not, (***0: no, 1: yes***)
12. **time**: Follow-up period in days, (***ranges from 4 to 285 days***). A period of time in which a person's health situation is monitored. This may also include keeping track of the health of people who participate in a clinical study or clinical trial for a period of time, both during the study and after the study ends.
13. **DEATH_EVENT**: If the patient is dead or not (due to heart attack), target feature, (***0:censored / alive during and after the event occured, 1: died***)

4.7. Feature Engineering

For Dataset 1:

1. In the ‘colon’ dataset, there are two records per person, one for recurrence and one for death.We have considered only the end results i.e. the records for the death.
2. Irrelevant features ‘id’ and ‘study’ have been dropped.
3. The null values have been replaced with the mean value of the corresponding column.
4. The categorical feature ‘rx’ has been label encoded in the following way:
   1=Obs, 2= Lev, 3=Lev+5-FU

For Dataset 2:

No modification had been done on this dataset.

4.8. Steps to apply Kaplan Meier method on both the datasets

For Dataset 1(Colon cancer dataset):

1. Do data preprocessing so that the dataset can be fitted perfectly.
2. The “duration” and “event” are initialized with the data “time” and “status”.
3. The data set is fitted to the KaplanMeierFitter by using three parameters “time”, “status” and “node4”.
4. The resultant graph is plotted showing the survival rate of patients depending upon their state of having 4 nodes (“node4”=1) with respect to time.
5. The data set is fitted to the KaplanMeierFitter by using three parameters “time”, “status” and “age”.
6. The resultant graph is plotted showing the of the survival rate of patients depending upon their age (grouped as, for age 0-20 = 1, for age 20-40 = 2, for age 40-60 = 3, for age 60-80 = 4,for age 80-100 = 5) with respect to time.
7. The data set is fitted to the KaplanMeierFitter by using three parameters “time”, “status” and “rx”.
8. The resultant graph is plotted showing the of the survival rate of patients depending upon “rx”, i.e. which medicine is used for treatment of the patient (observation = 1, amisole = 2, amisole+5-FU = 3) with respect to time
9. The data set is fitted to the KaplanMeierFitter by using three parameters “time”, “status” and “sex”.
10. The resultant graph is plotted showing the result of the survival rate of patients depending upon their gender with respect to time.
11. With respect to time duration, the rate of death is fitted to the Kaplan Meier, and the resultant graph is plotted.

For Dataset 2(Heart failure dataset):

1. Do data preprocessing so that the dataset can be fitted perfectly.
2. The "duration" and "event" are initialized with the data "time" and "DEATH_EVENT".
3. The data set is fitted to the KaplanMeierFitter by using three parameters "time", "DEATH_EVENT" and "anaemia".
4. The resultant graph is plotted showing the result of the survival of patients depending upon their state of having anaemia(yes=1, no=0) with respect to time(days passed).
5. The data set is fitted to the KaplanMeierFitter by using three parameters "time", "DEATH_EVENT" and "high_blood_pressure".
6. The resultant graph is plotted showing the survival rate of patients depending upon their state of having high blood pressure (yes=1, no=0) with respect to time.
7. The data set is fitted to the KaplanMeierFitter by using three parameters "time", "DEATH_EVENT" and "platelets".
8. The resultant graph is plotted showing the survival rate of patients depending upon their platelets count (grouped as, for platelets count < 15000 (low) = 1, for platelets count >= 15000 and <= 45000 (normal) = 2, for platelets count > 45000 (high) = 3) with respect to time.
9. The data set is fitted to the KaplanMeierFitter by using three parameters "time", "DEATH_EVENT" and "smoking".
10. The resultant graph is plotted showing the survival rate of patients depending upon whether they smoke (yes=1, 0=no) with respect to time.
11. With respect to time duration, the rate of death is fitted to the KaplanMeierfitter, and the resultant graph is plotted.

4.9. Steps to apply COX regression on both the datasets

For Dataset 1(Colon cancer dataset) :

1. Preprocess the data for good model fitting of the dataset.
2. Import required libraries.
3. Initialize the "predictor variables" with "df", "duration_col" with "time" and "event_col" with "status". Here, "df" is a pandas dataframe for the dataset.
4. Fit the dataset to the 'CoxPHFitter' using three parameters "df", "time" and "status".
5. Display the summary of the fitted model that contains the fit on the chosen dataset, the log hazard ratios, the coefficients, p-value, z-value, etc.
6. Plot a graph as a result to show the survival rate of randomly chosen patients from the fitttd dataset based on the predictor variables, i.e. "df" with respect to the time duration or time in study, i.e "time"
7. Plot the importance of the predictor variables and display them in terms of their log hazard ratios, having their magnitudes and standard errors.
8. Select only significant features which are evident from the graph in step 7.

9. Using the selected features from step 8 to initialize the “predictor variables” with “df2”, “duration_col” with “time” and “event_col” with “status”. Here, “df2” is a pandas dataframe for the attribute filtered dataset.
10. Repeat steps 4 to 7 using three parameters “df2”, “time” and “status”.

For Dataset 2(Heart failure dataset) :

1. Preprocess the data for good model fitting of the dataset.
2. Import required libraries.
3. Initialize the “predictor variables” with “data”, “duration_col” with “time” and “event_col” with “DEATH_EVENT”. Here, “data” is a pandas dataframe for the dataset.
4. Fit the dataset to the ‘CoxPHFitter’ using three parameters “data”, “time” and “DEATH_EVENT”.
5. Display the summary of the fitted model that contains the fit on the chosen dataset, the log hazard ratios, the coefficients, p-value, z-value, etc.
6. Plot a graph as a result to show the survival rate of randomly chosen patients from the fitttd dataset based on the predictor variables, i.e., “data” with respect to the time duration or time in study, i.e “time”
7. Plot the importance of the predictor variables and display them in terms of their log hazard ratios, having their magnitudes and standard errors.
8. Select only significant features which are evident from the graph in step 7.
9. Using the selected features from step 8 to initialize the “predictor variables” with “data1”, “duration_col” with “time” and “event_col” with “status”. Here, “data1” is a pandas dataframe for the attribute filtered dataset.
10. Repeat steps 4 to 7 using three parameters “data1”, “time” and “status”.

4.10. Steps to apply AdaBoost regression on Cancer dataset

1. Do data preprocessing so that the dataset can be fitted perfectly.
2. The dependent variables list X is initialized with ['rx', 'sex', 'age', 'obstruct', 'perfor', 'adhere', 'nodes', 'differ', 'extent', 'node4', 'surg', 'time'] fields of the dataset.
3. The independent variable Y is initialized with the ‘status’ field of the dataset.
4. Randomly split the dataset into Train_Dataset and Test_Dataset.
5. Randomly split the Train_Dataset into X_train and Y_train.
6. Randomly split the Test_Dataset into X_test and Y_test.
7. Create an AdaBoostRegressor object.
8. The Train_Dataset is fitted into the AdaBoostRegressor object passing the parameters X_train and Y_train.
9. After training the model, the prediction of the input dataset X_test is achieved and stored into Y_pred.

10. If Y_pred[i] < 0.5, for i=0, 1, 2, … we store Y_pred[i] = 0, otherwise we store Y_pred[i] = 1.
11. Now, values of TP, FP, TN and FN are calculated using our handwritten function and after that accuracy is evaluated using the formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

12. Lastly, we plot the ROC curve to visualize the accuracy to compare with other models.

We run the Adaptive Boosting algorithm on our dataset using 3 different Base Estimators. 1) Default Base Estimator (Decision Tree Regressor), 2) Random Forest Regressor and 3) K-Nearest Neighbors Regressor (KNN). We executed each of these algorithms at least 100 times and noted down the accuracies. And then average accuracy is calculated to get the better approximation of performance for each model.

$$Accuracy_{average} = \frac{\sum_{i=1}^{N} Accuracy_{\,i}}{N}$$

Then these accuracies are plotted and compared with other models to get our result.

4.11. Steps to apply XGBoost regression on Cancer dataset:

1. Do data preprocessing so that the dataset can be fitted perfectly.
2. The dependent variables list X is initialized with ['rx', 'sex', 'age', 'obstruct', 'perfor', 'adhere', 'nodes', 'differ', 'extent', 'node4', 'surg', 'time'] fields of the dataset.
3. The independent variable Y is initialized with the ‘status’ field of the dataset.
4. Randomly split the dataset into Train_Dataset and Test_Dataset.
5. Randomly split the Train_Dataset into X_train and Y_train.
6. Randomly split the Test_Dataset into X_test and Y_test.
7. Create an XGBRegressor object. All the parameters are kept as default values.
8. The Train_Dataset is fitted into the XGBRegressor object passing the parameters X_train and Y_train.
9. After training the model, the prediction of the input dataset X_test is achieved and stored into Y_pred.
10. If Y_pred[i] < 0.5, for i=0, 1, 2, … we store Y_pred[i] = 0, otherwise we store Y_pred[i] = 1.
11. Now, values of TP, FP, TN and FN are calculated using our handwritten function and after that accuracy is evaluated using the formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

12. Lastly, we plot the ROC curve to visualize the accuracy to compare with other models.

    This algorithm was executed at least 100 times and we noted down respective accuracy for every iteration. Then average accuracy was calculated to get a better approximation of accuracy.

$$Accuracy_{average} = \frac{\sum_{i=1}^{N} Accuracy_{\,i}}{N}$$

The average accuracy was then used to plot and compare with other models.

4.12. Parameter Optimization in XGBoost Algorithm (XGB-Mod)

- **Method - XGB-Mod**

1. Do data preprocessing so that the dataset can be fitted perfectly.
2. The dependent variables list X is initialized with ['rx', 'sex', 'age', 'obstruct', 'perfor', 'adhere', 'nodes', 'differ', 'extent', 'node4', 'surg', 'time'] fields of the dataset.
3. The independent variable Y is initialized with the 'status' field of the dataset.
4. Randomly split the dataset into Train_Dataset and Test_Dataset.
5. Randomly split the Train_Dataset into X_train and Y_train.
6. Randomly split the Test_Dataset into X_test and Y_test.
7. Create an XGBRegressor object. The parameters of XGBRegressor were tuned as follow:
   i) max_depth=3
   ii) learning_rate=0.02
   iii) n_estimators=200
   iv) min_child_weight=5
   v) gamma=0.4
   vi) colsample_bytree=0.6
   vii) subsample=0.8
   viii) reg_alpha=1.1
   Other parameters were kept as default.
8. The Train_Dataset is fitted into the XGBRegressor object passing the parameters X_train and Y_train.
9. After training the model, the prediction of the input dataset X_test is achieved and stored into Y_pred.
10. If Y_pred[i] < 0.5, for i=0, 1, 2, … we store Y_pred[i] = 0, otherwise we store Y_pred[i] = 1.
11. Now, values of TP, FP, TN and FN are calculated using our handwritten function and after that accuracy is evaluated using the formula:
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
12. Lastly, we plot the ROC curve to visualize the accuracy to compare with other models.

Here to mention, these hypertuned values of the parameters we obtained using GridSearchCV() library function reside in sklearn.model_selection library. Our Proposed XGB-Mod algorithm was executed at least 100 times and we noted down respective accuracy for every iteration. Then average accuracy was calculated to get a better approximation of accuracy.

$$Accuracy_{average} = \frac{\sum_{i=1}^{N} Accuracy_{\ i}}{N}$$

The average accuracy was then used to plot and compare with other models.

4.13. MongoDB

MongoDB [20] is most popular among the **NoSQL** (Not Only SQL) databases. For building data warehouses, it is a great tool especially because of its ability to fully utilize so-called “sharding-nothing cluster architecture.” It is an open-source database, which makes it ideal for building high performance data warehouses.

MongoDB is an open-source document database that provides **high performance**, **high availability**, and **automatic scaling**. A **record** in MongoDB is a **document**, which is a data structure **composed of field and value pairs**. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays and arrays of documents.

The advantages of using documents are:
• Documents (i.e., objects) correspond to native data types in many programming languages.
• Embedded documents and arrays reduce need for expensive joins.
• Dynamic schema supports fluent polymorphism.

A. Key Features of MongoDB

• High Performance – MongoDB provides high performance data persistence. In particular, it supports embedded data models, reduces I/O activity on database system, indexes support faster queries and can include keys from embedded documents and arrays.
• Rich Query Language - MongoDB supports a rich query language to support read and write operations (CRUD) as well as Data aggregation, Text Search.
• High Availability – MongoDb’s replication facility, called replica set, provides automatic failover and data redundancy. A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.
• Horizontal Scalability – MongoDB provides horizontal scalability as part of its core functionality. Sharding distributes data across a cluster of machines.

MongoDB works on the concept of collection and document.
• Database - Database is a physical container for collections. Each database gets its own set of files on the file system.
• Collection - Collection is a group of MongoDB documents. It is the equivalent of a RDBMS table. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
• Document - A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection’s documents may hold different types of data.

B. *Basic differences between MySql and MongoDB*

TABLE 1. MYSQL VS MONGODB TERMS

MySQL	**MongoDB**
Database	Database
Table	Collection
Index	Index
Row	BSON document
Column	BSON field
Join	Embedded documents and linking
Primary key	Primary key(Default key_id provided by mongodb itself)
Group by	Aggregation

As shown in Table 1, in MongoDB, some MySQL terms, such as table or row, get another name, namely ‘collection’, respectively BSON document. In other words, we can say that MongoDB contains collections, collections contain documents and a document contains multiple fields.

In the classical RDBMS model, the data is organized in the form of relations and is represented in a table consisting of rows and columns. Relational databases employ the usage of a parameter known as key. There are several types of keys available, albeit the primary key is one of the most important keys of the table; it is used to identify each row of the table uniquely. There are four main operations used to access the database they are known as CRUD namely, Create, Read, Update and Delete associated with the data. These operations use the Structured Query Language –SQL. ACID properties are one of the most significant and important attributes of a SQL database. This is the key difference between SQL and NoSQL database systems. The NewSQL approach on the other hand, conserves and supports the properties of relational models, at the same time incorporating the features of NoSQL model.

Unlike MySQL, where the database is presented graphically in the form of a table, in MongoDB, a database has the following graphic structure:

```
{
“_id”: “d4acaf3a76e4378b853eb15fde21672”,
“username”: “andra”,
“email”: andra@gmail.com,
}
{
```

“_id”: “d4rvgf3a76e4378b853eb15fde21672”,
“username”: “iona”,
“email”: iona@gmail.com,
}
The example above shows a database for users, each user having an id that is unique and automatically generated, a username and an email address.

The application will have 3 classes of users, namely the administrators, the moderators and the regular users. Each user has the right to create a private forum/subforum. Within a subforum, the moderators have the right to edit/delete the subforum and they can also moderate other users’ discussions, while regular users are only allowed to post discussions and leave comments. If a relational database has been used, the columns for forums and subforums should have appeared at all forum users, although normal users will never have the right to create, modify or delete them, unless of course, they are the administrators of that particular forum. Using MongoDB, these fields regarding the forum and subforum will appear only to users who have that right (moderators and administrators), thus significantly reducing storage space, which is much higher using MySQL.

As in relational databases, MongoDB also has one-to-many relationships, but in this case the concept of foreign key is not used; instead, the concept of annotations is used. Thus, in this case, regarding a forum, the connection between the forum and its subforums is as follows: in the forum document, the subforums are referenced using the annotation. MongoDB provided lower execution times than MySQL in all four basic operations (Insert, Select (query), Update, Delete), which is essential when an application should provide support to thousands of users simultaneously.

C. *Why MongoDB ?*

Any relational database has a typical schema design that shows the number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship. Advantages of MongoDB over RDBMS can be described as:

- MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Unlike RDBMS, there are no integrity constraints in MongoDB.
- MongoDB does not need any defined data schema; every document here could have different data.
- Structure of a single object is clear.
- No complex joins.
- Deep query ability. MongoDB supports dynamic queries on documents using a document-based query language that’s nearly as powerful as SQL.
- Ease of scale out. MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

D. *MongoDB in Survival Analysis*

1. Data Storage is one of the most challenging tasks in a big data framework. Therefore, traditional data analysis is unfit to manage those systems. This component may be HDFS, NoSQL such as MongoDB and SQL databases, or a combination of all of them. Therefore, it is more scalable and ensures high storage capabilities.
2. Different databases are required for horizontal and for vertical data integration [21] because each of these approaches address different aspects of the integration problem. Horizontal data integration deals with unstructured and heterogeneous data. Thus, we use a document-based database (such as MongoDB), which can handle different data types and formats.
3. All the data from the experimental datasets are integrated horizontally with NoSQL (MongoDB) technology and represented as a semi-structure. This results in a semi-structure per data type. All the raw and metadata are stored in MongoDB in JSON format.

E. *Operations on dataset stored in MongoDB*

1. *Fitting dataset into MongoDB:*

   First of all, we have to create an instance of MongoClient[22] which connects the frontend with the MongoDB database. Then we create a new database. After that each row of the dataset is read and transformed into the JSON-like document. Then the documents are stored into the MongoDB one by one.

   Examples:

   i) Inserting one document:

   ```
   db.colonCancer.insert_one(document)
   ```

   ii) Inserting more than one documents:

   ```
   db.colonCancer.insert_many(documents)
   ```

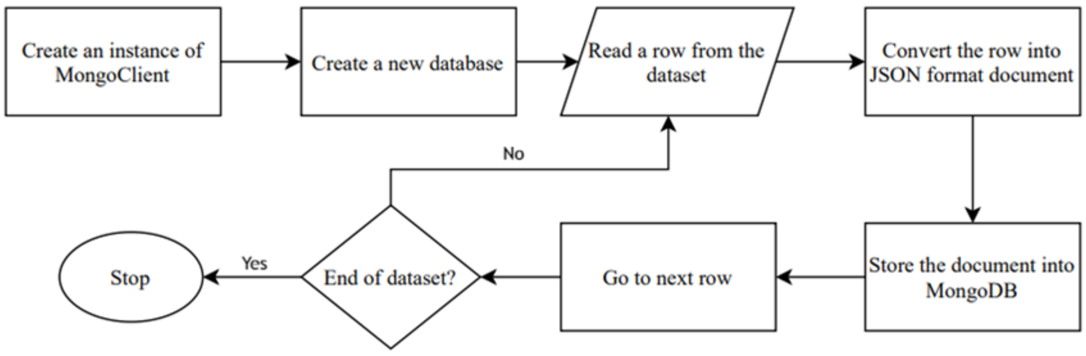

Create an instance of MongoClient
Create a new database
Read a row from the dataset
Convert the row into JSON format document
Store the document into MongoDB
Go to next row
End of dataset?
No
Yes
Stop

Fig. 2: Flowchart to store the dataset into MongoDB

2. *Fetching documents from MongoDB:*

In the beginning, we have to create an instance of MongoClient which connects the frontend with the MongoDB database. Then we define a database object which points to the desired database in MongoDB. Now to get a particular collection from the database, we define a collection object which points to the collection. After that each document of the collection is fetched. Beside that more than documents can also be fetched depending upon the necessity.

Examples:

i) Reading one document:

```
oneRecord = myCollection.find_one()
```

ii) Reading all documents:

```
allRecords = myCollection.find()
```

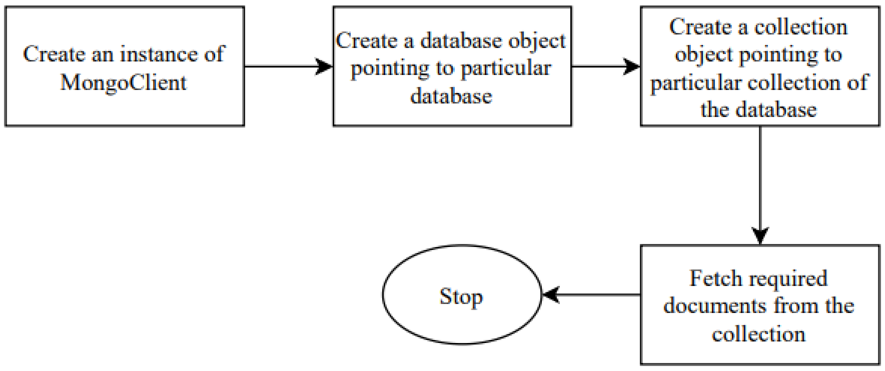

Create an instance of MongoClient
Create a database object pointing to particular database
Create a collection object pointing to particular collection of the database
Fetch required documents from the collection
Stop

Fig. 3: Flowchart to fetch documents MongoDB

3. *Query processing in MongoDB:*

To process a query, we have to create an instance of MongoClient which connects the frontend with the MongoDB database. Then we define a database object which points to the desired database in MongoDB. Now to get a particular collection from the database, we define a collection object which points to the collection. After that fetching of the required documents is done along with the conditions to get the only desired documents.

Examples:

i) Getting all documents having age = 80

```
filter = myCollection.find({"age": 80})
```

ii) Getting status from documents having nodes > 20 and age >= 45

```
filter2 = myCollection.find({$and: [{"nodes": {"$gt": 20.0}},{"age": {"$gte": 45}}]}, {"status": 1, "_id": 0})
```

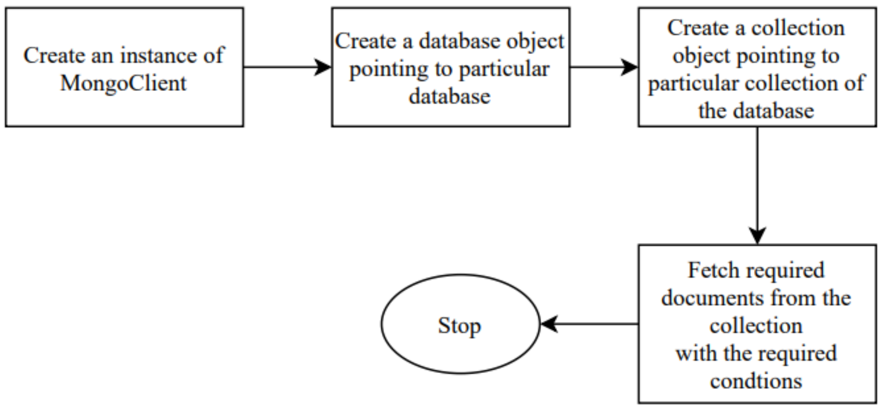

Create an instance of MongoClient
Create a database object pointing to particular database
Create a collection object pointing to particular collection of the database
Fetch required documents from the collection with the required condtions
Stop

Fig. 4: Flowchart to query processing in MongoDB

5. Results

5.1. Kaplan Meier Regression

The graphical representation of the KM curve defines an event's probability in a finite amount of time.

For Dataset 1 :

1) **Survival Probability Distribution with respect to Age**

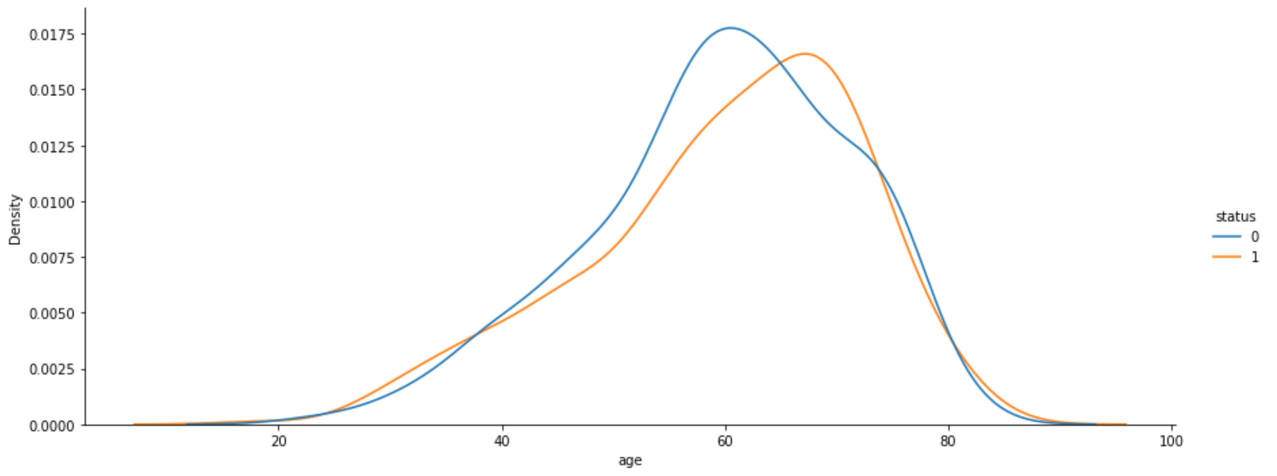


Fig. 5: Survival Probability Distribution w.r.t. Age of Patients

2) **Survival Probability Distribution with respect to Number of Positive Lymph Nodes**

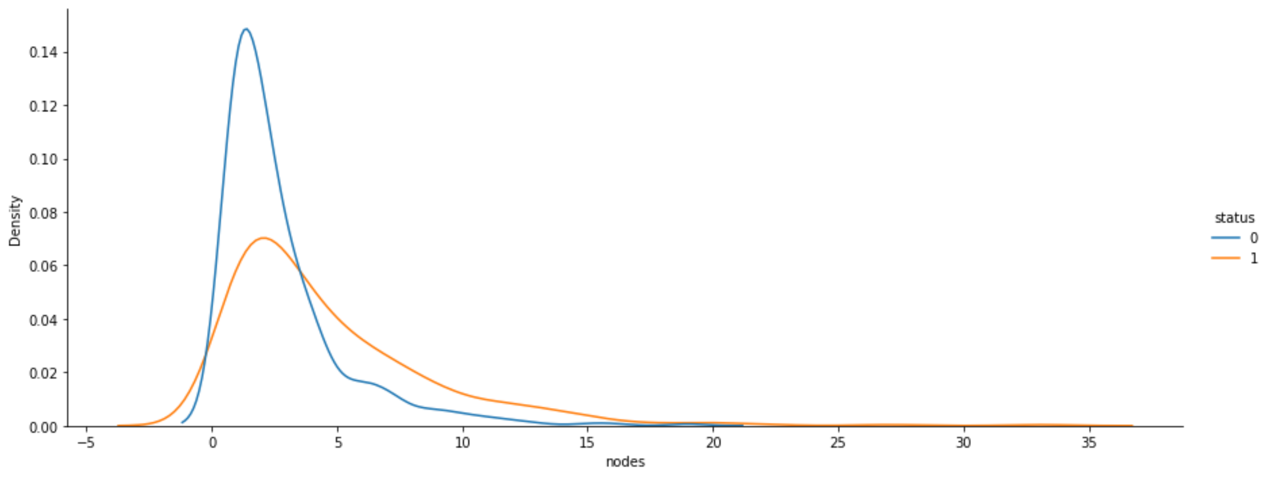


Fig. 6: Survival Probability Distribution w.r.t. Positive Lymph Nodes Detected

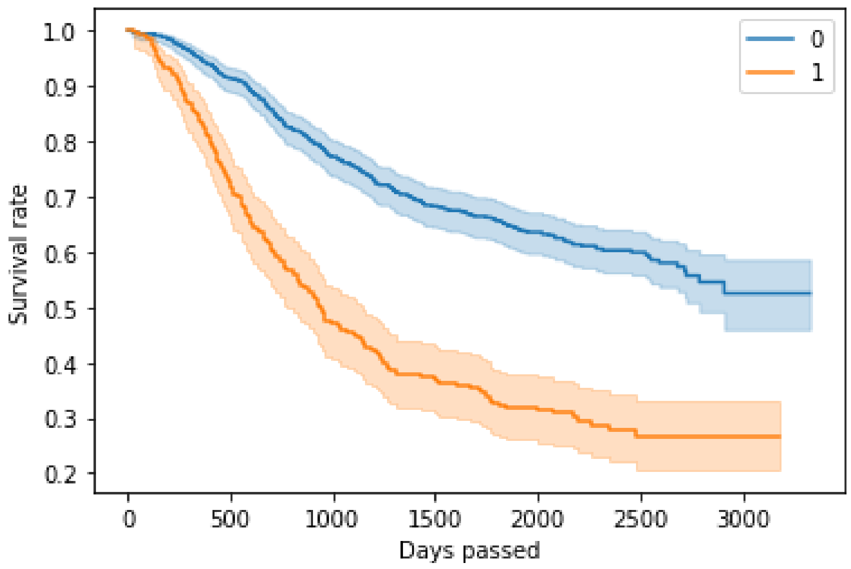


Fig. 7: Kaplan Meier Survival curve plotted for the feature node4, i.e. more than 4 positive lymphatic nodes detected or not (yes = 1, no = 0).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

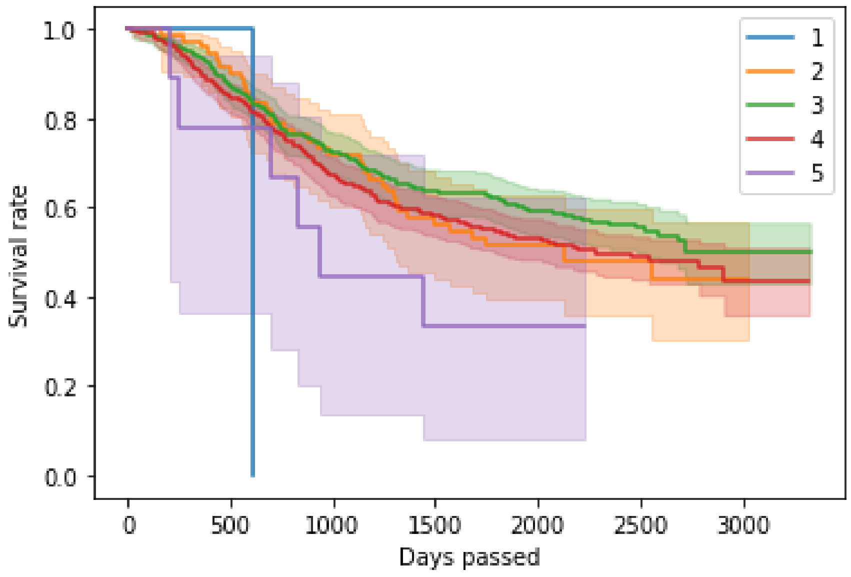


Fig. 8: Kaplan Meier Survival curve plotted for feature age, grouped as,
for age 0-20 = 1, for age 20-40 = 2, for age 40-60 = 3, for age 60-80 = 4,
for age 80-100 = 5
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

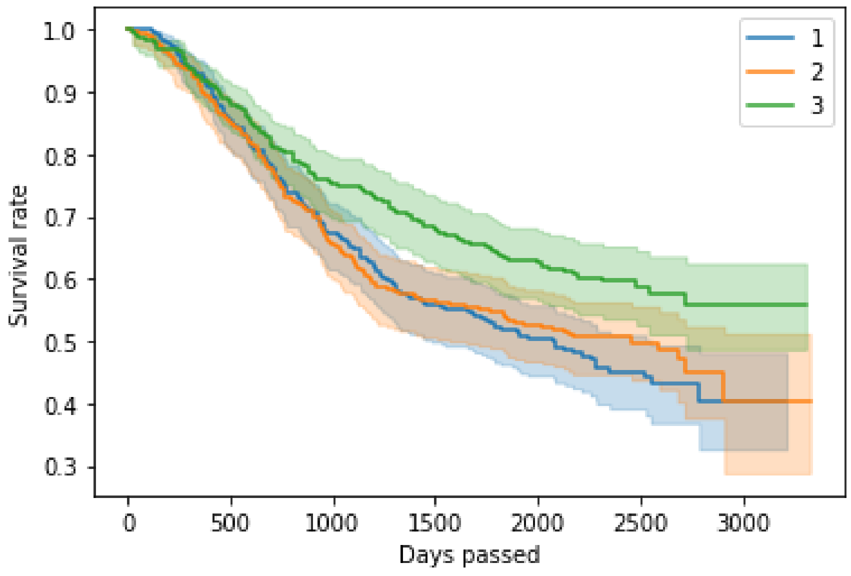


Fig. 9: Kaplan Meier Survival curve plotted for the feature rx, i.e. which medicine is used for treatment of the patient (ervation = 1, amisole = 2, amisole+5-FU = 3).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

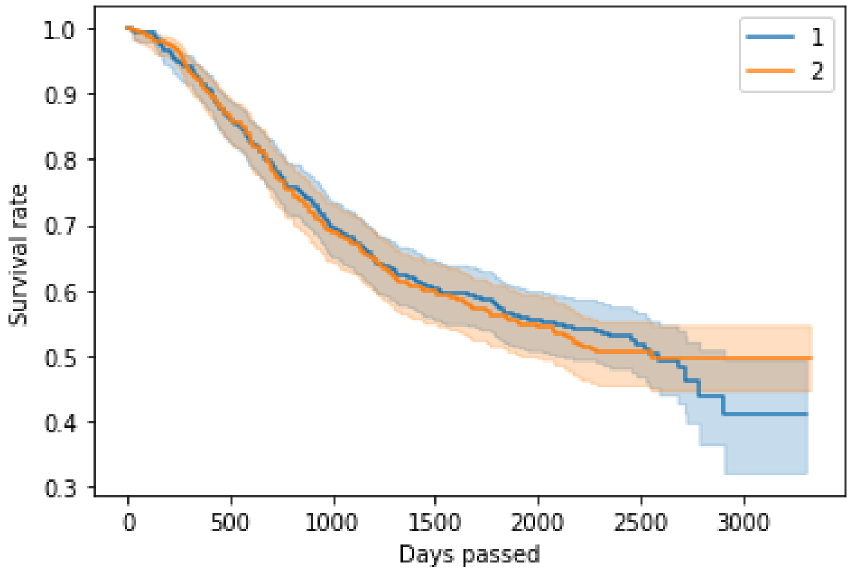


Fig. 10: Kaplan Meier Survival curve plotted for the feature gender of the patient (male = 1, female = 2).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

For Dataset 2 :

3) <u>Survival Probability Distribution with respect to Amount of Creatinine Phosphokinase</u>

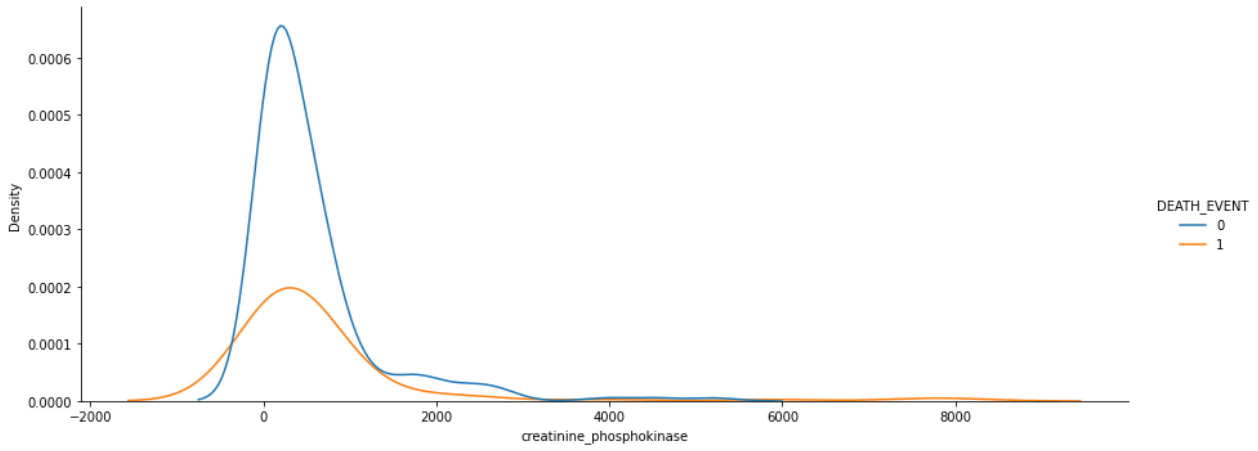


Fig. 11: Survival Probability Distribution w.r.t. Amount of Creatinine Phosphokinase

4) <u>Survival Probability Distribution with respect to Number of Platelets</u>

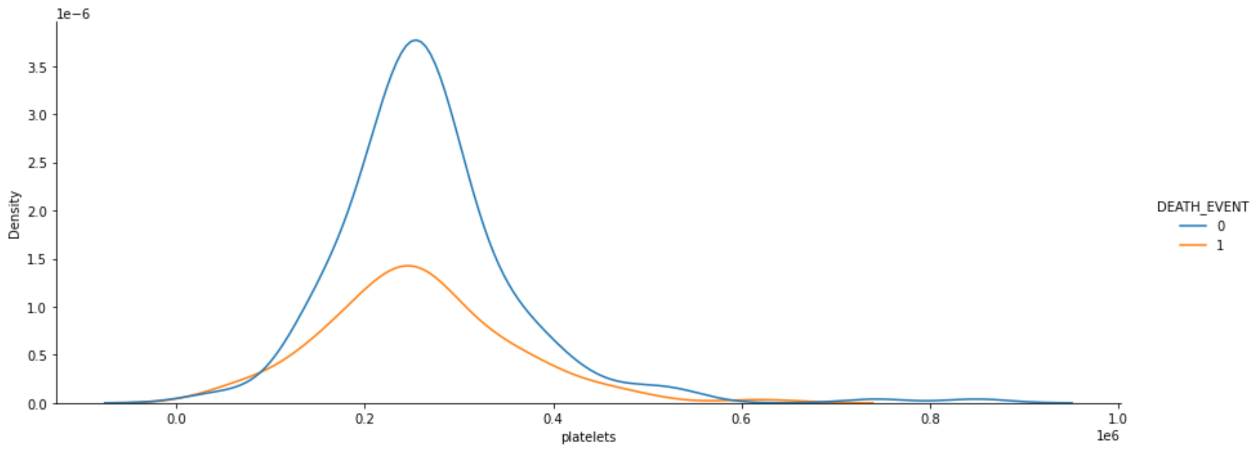


Fig. 12: Survival Probability Distribution w.r.t. Number of Platelets

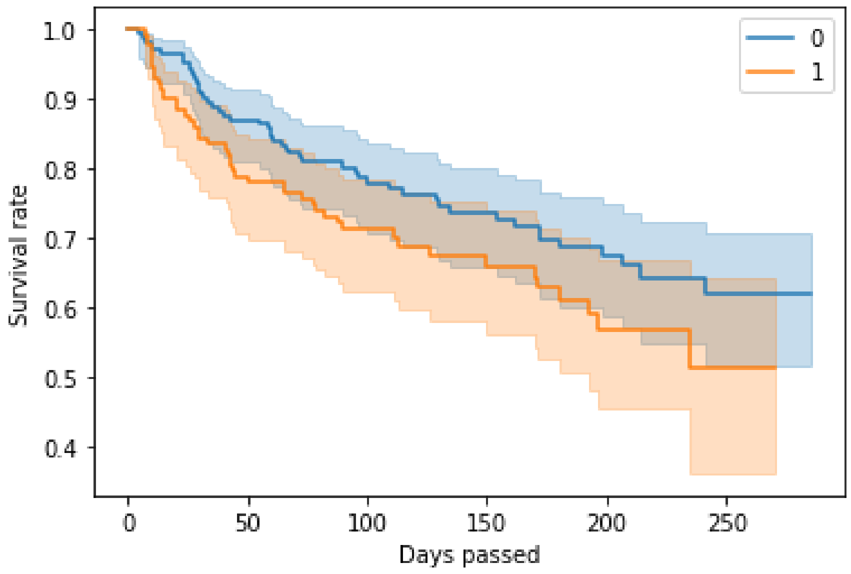


Fig. 13: Kaplan Meier Survival curve plotted for the feature anaemia, i.e. anaemia is present for the patient or not (yes = 1, no = 0).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

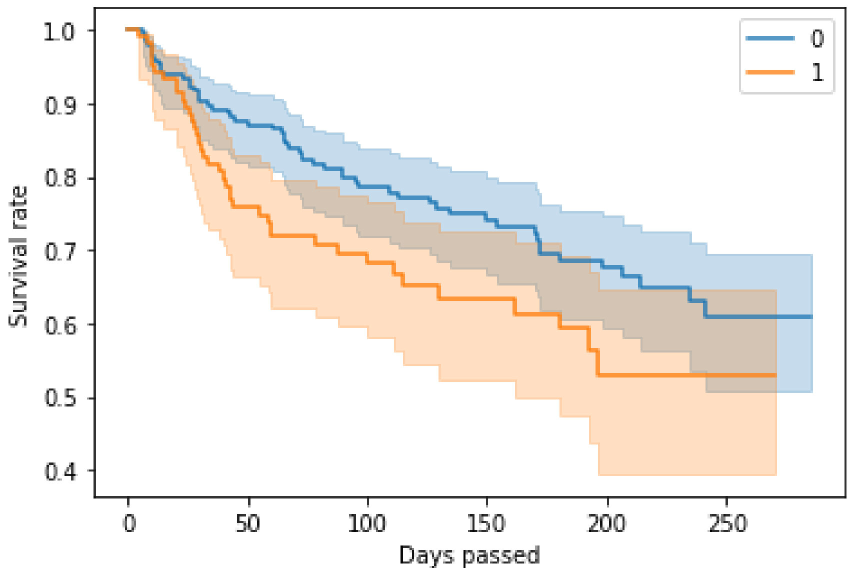


Fig. 14: Kaplan Meier Survival curve plotted for the feature high_blood_pressure, i.e. high blood pressure is present for the patient or not (yes = 1, no = 0).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

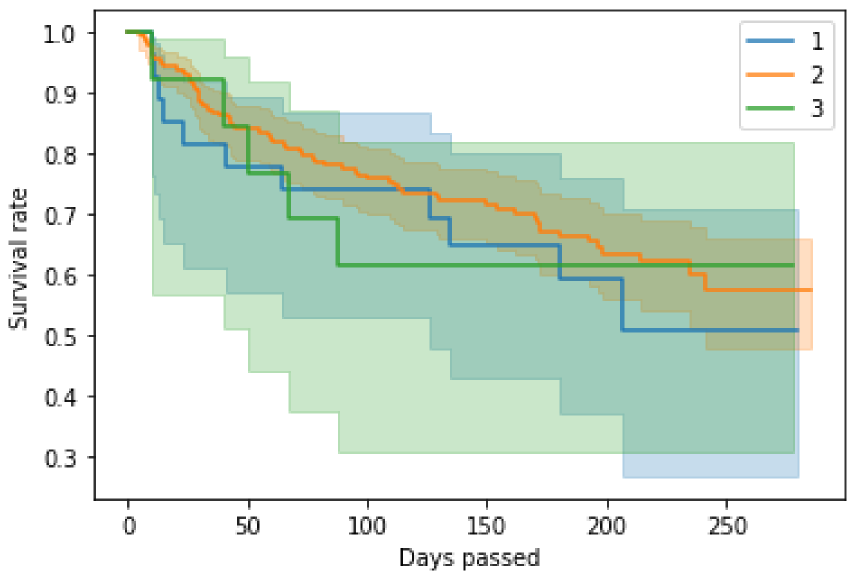


Fig. 15 : Kaplan Meier Survival curve plotted for feature platelets, grouped as, for platelets count < 15000 (low) = 1, for platelets count >= 15000 and <= 45000 (normal) = 2, for platelets count > 45000 (high) = 3
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

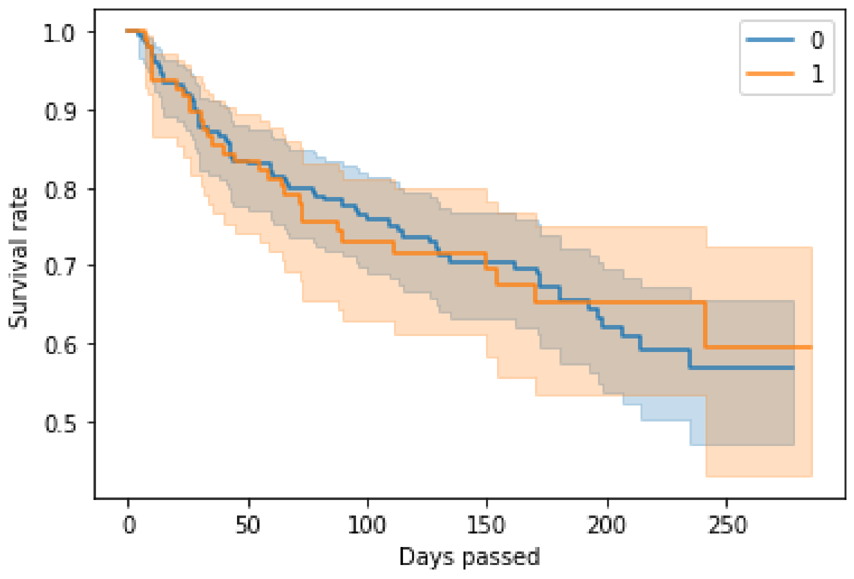


Fig. 16: Kaplan Meier Survival curve plotted for the feature smoking, i.e. the patient smokes or not (yes = 1, no = 0).
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

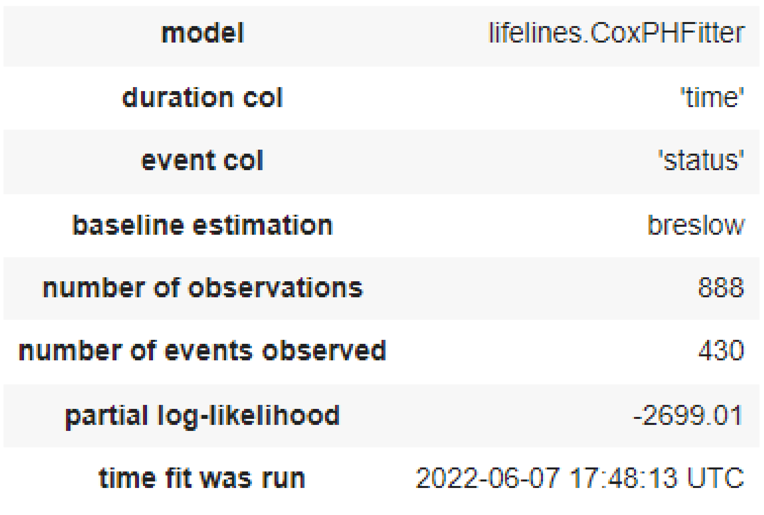
5.2. Cox Proportional Hazard Regression

CoxPHFitter class in the Lifelines package of python implements fitting Cox's proportional hazard model. Here, the baseline hazard, h0(t) is modeled non-parametrically, using Breslow's method. In this case, the entire model is the traditional semi-parametric Cox model. Ties are handled using Efron's method.

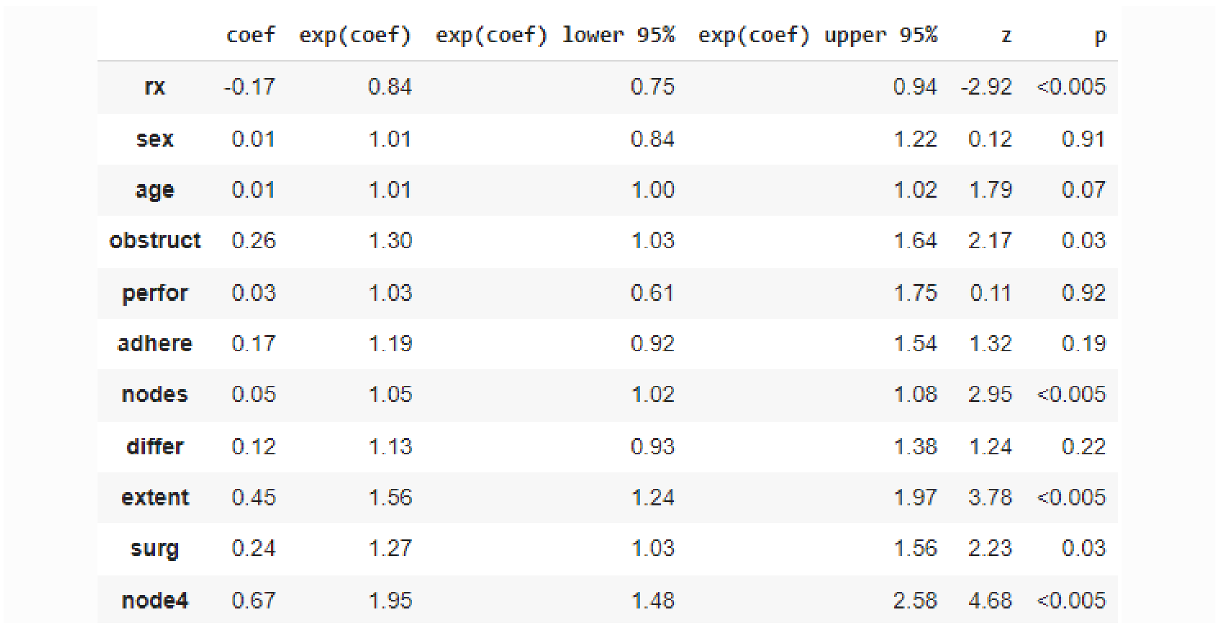
The datasets are pre-processed and fitted using Coxph fitter model and the summary of the fitted result has been shown along with the importance of the different features used in the datasets. We have also shown some patients' (for dataset 1, 1st to 5th patient and for dataset 2, 100th to 104th patient) survival curve plotted using Cox regression method.

For Dataset 1:

In the below Cox PH model, p-value and the Hazard Ratio (HR) of the feature node4, are <0.005 and 0.67 respectively, which implies that for other covariates constant value, patients having detected more than 4 positive lymphatic nodes increase the hazard by a factor of 0.67 or 33%, i.e., survival chances are significantly lower for the colon cancer patients having more than 4 cancerous lymphatic nodes already. Again, the p-value and the Hazard Ratio (HR) for the feature extent are <0.005 and 0.45, which indicates there is significant difference between the colon cancer patients having extent of local spread already, as risk of death increases by 55%. On the other hand, we can see, p-value and the Hazard Ratio(HR) of the feature sex, are 0.91 and 0.01 respectively. This implies that the gender of the patient does not impact significantly on the survival of the colon cancer patients. Same rules follow for the other features also.



model	lifelines.CoxPHFitter
duration col	'time'
event col	'status'
baseline estimation	breslow
number of observations	888
number of events observed	430
partial log-likelihood	-2699.01
time fit was run	2022-06-07 17:48:13 UTC



	coef	exp(coef)	exp(coef) lower 95%	exp(coef) upper 95%	z	p
rx	-0.17	0.84	0.75	0.94	-2.92	<0.005
sex	0.01	1.01	0.84	1.22	0.12	0.91
age	0.01	1.01	1.00	1.02	1.79	0.07
obstruct	0.26	1.30	1.03	1.64	2.17	0.03
perfor	0.03	1.03	0.61	1.75	0.11	0.92
adhere	0.17	1.19	0.92	1.54	1.32	0.19
nodes	0.05	1.05	1.02	1.08	2.95	<0.005
differ	0.12	1.13	0.93	1.38	1.24	0.22
extent	0.45	1.56	1.24	1.97	3.78	<0.005
surg	0.24	1.27	1.03	1.56	2.23	0.03
node4	0.67	1.95	1.48	2.58	4.68	<0.005

Fig. 17: Cox Proportional Hazard model summary

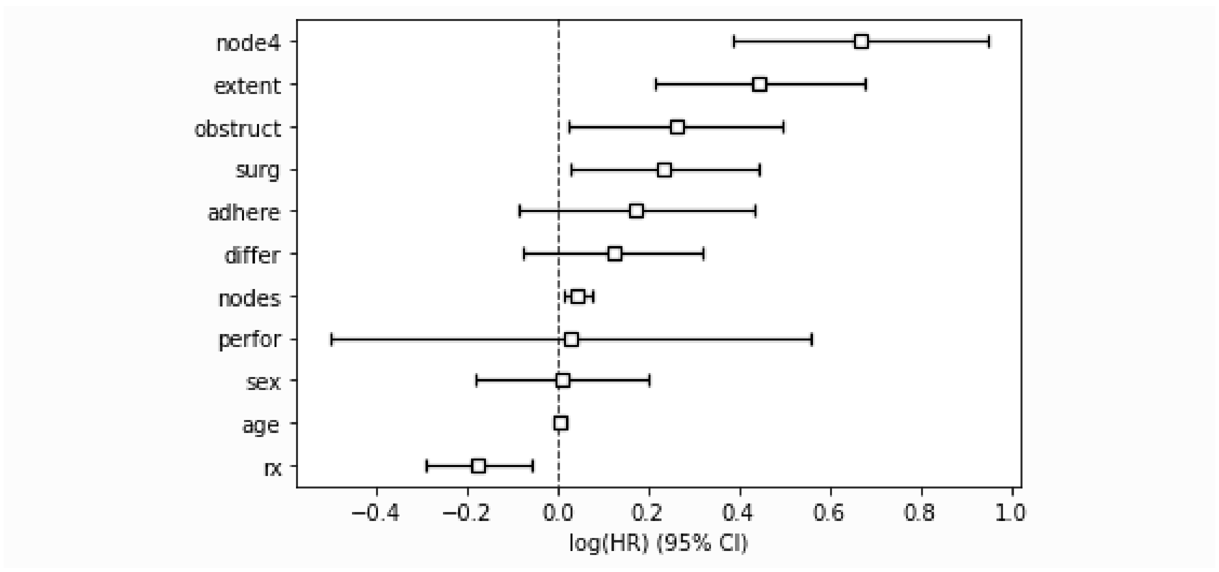


Fig. 18: Feature importance graphical representation

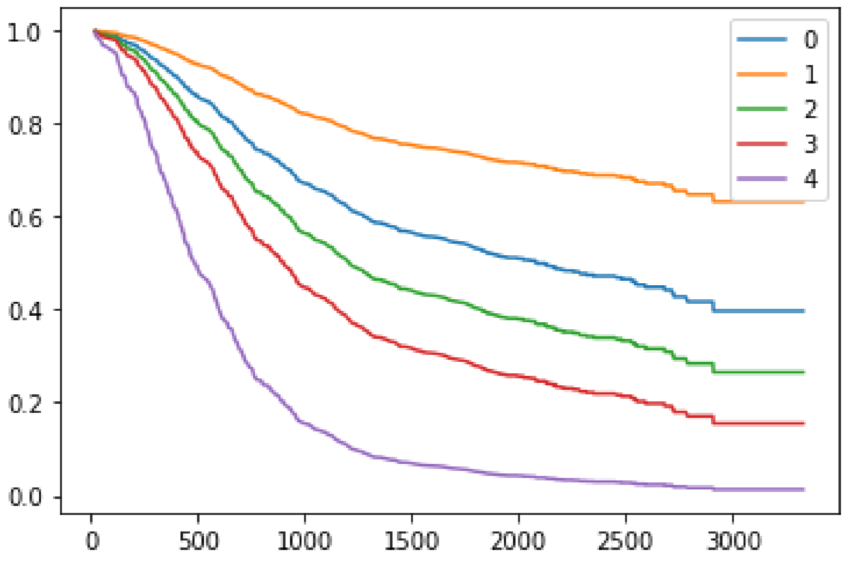


Fig. 19: Survival curve plotted using Cox Regression method for patients indexed 0-4.
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

- **After reducing some features that do not have significant impact:**

  Instead of taking all 11 features we are taking only 6 significant features

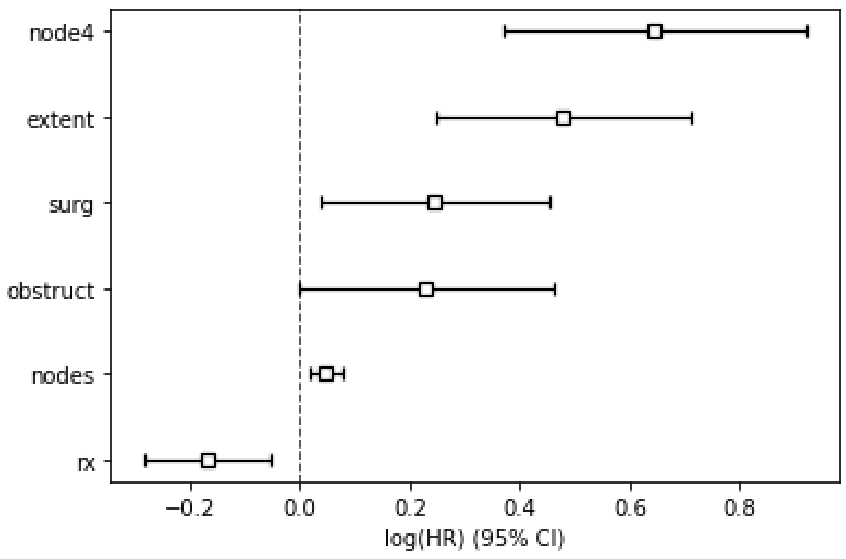


Fig. 20: Feature Importance graph

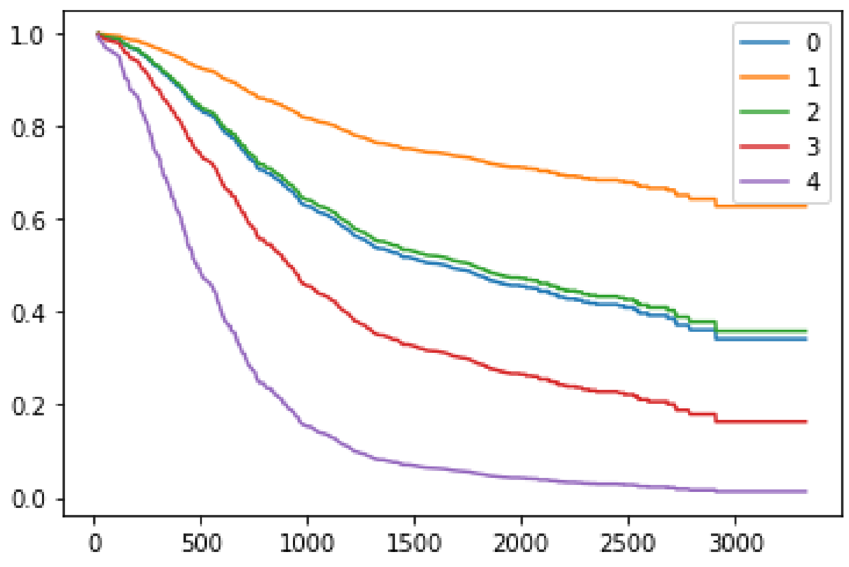
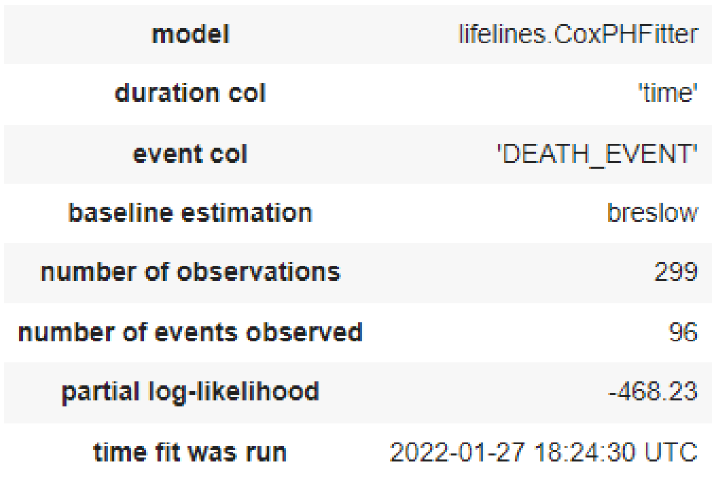


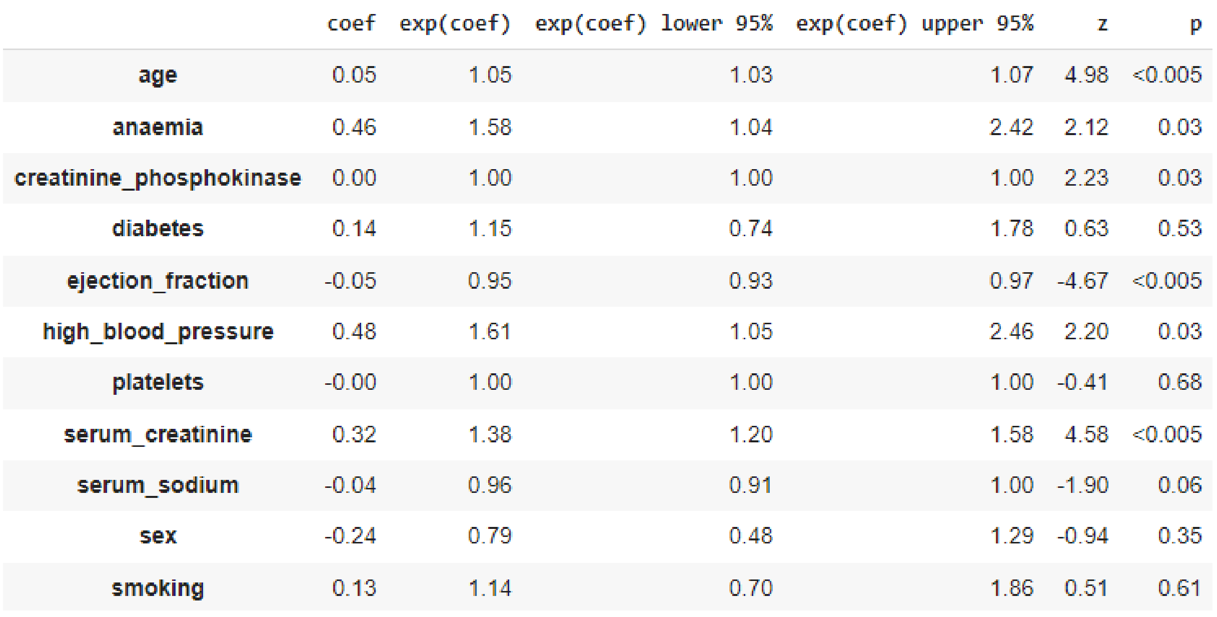
Fig. 21: Survival curve plotted using Cox regression method with reduced features for patients indexed 0-4.
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

For Dataset 2:

In the below Cox PH model, p-value and the Hazard Ratio(HR) of the feature age, i.e. age of the heart failure patients, are <0.005 and 0.05 respectively, which implies that for other covariates constant value, lower age decreases the hazard by a factor of up to 0.05 or 95%, i.e. survival chances are higher for old young patients. Again, the p-value for ejection_fraction is <0.005, and Hazard Ratio(HR) is -0.05, meaning that patients with higher ejection_fraction had 95% lower chance of death compared to patients with lower ejection_fraction. But for the feature smoking, the p-value and Hazard Ratio (HR) are 0.64 and 0.12 respectively which indicates there is no significant difference of the hazard by the fact that the patient smokes or not.



model	lifelines.CoxPHFitter
duration col	'time'
event col	'DEATH_EVENT'
baseline estimation	breslow
number of observations	299
number of events observed	96
partial log-likelihood	-468.23
time fit was run	2022-01-27 18:24:30 UTC



	coef	exp(coef)	exp(coef) lower 95%	exp(coef) upper 95%	z	p
age	0.05	1.05	1.03	1.07	4.98	<0.005
anaemia	0.46	1.58	1.04	2.42	2.12	0.03
creatinine_phosphokinase	0.00	1.00	1.00	1.00	2.23	0.03
diabetes	0.14	1.15	0.74	1.78	0.63	0.53
ejection_fraction	-0.05	0.95	0.93	0.97	-4.67	<0.005
high_blood_pressure	0.48	1.61	1.05	2.46	2.20	0.03
platelets	-0.00	1.00	1.00	1.00	-0.41	0.68
serum_creatinine	0.32	1.38	1.20	1.58	4.58	<0.005
serum_sodium	-0.04	0.96	0.91	1.00	-1.90	0.06
sex	-0.24	0.79	0.48	1.29	-0.94	0.35
smoking	0.13	1.14	0.70	1.86	0.51	0.61

Fig. 22: Cox Proportional Hazard model summary

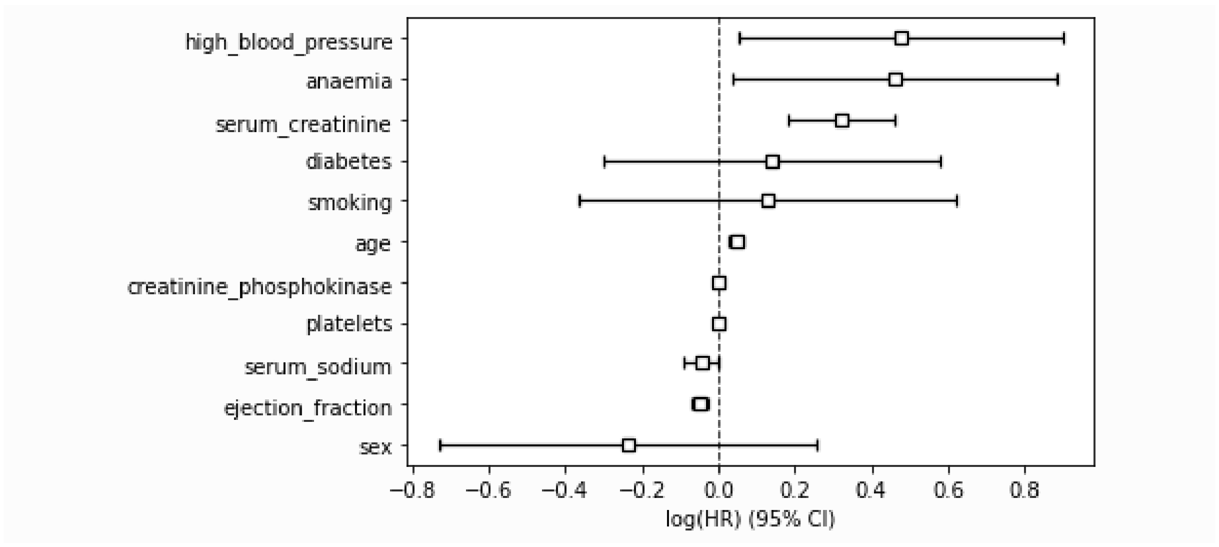


Fig. 23: Feature importance graphical representation

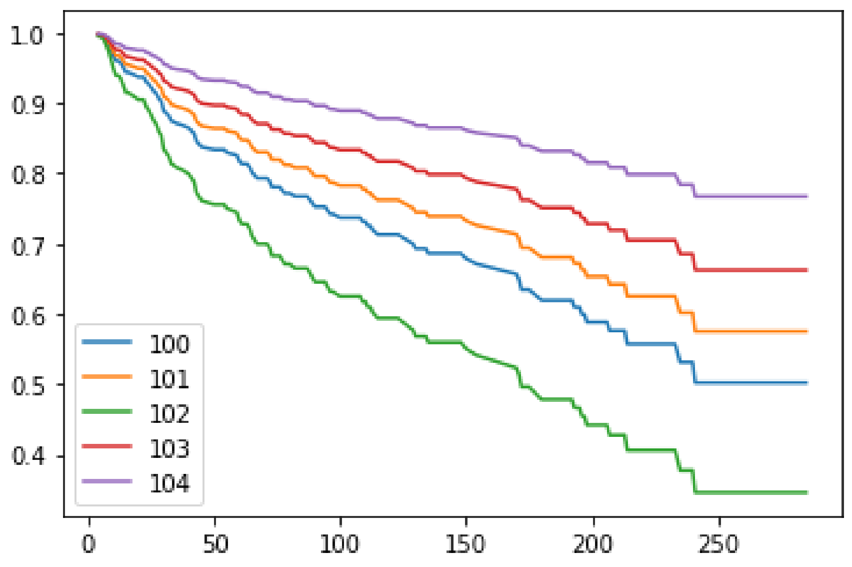


Fig. 24: Survival curve plotted using Cox Regression method for patients indexed 100-104. Survival Duration is on X axis (in days) Survival Probability is on the Y axis.

- **After reducing some features that do not have significant impact:**

  Instead of taking all 11 features we are taking only 6 significant features

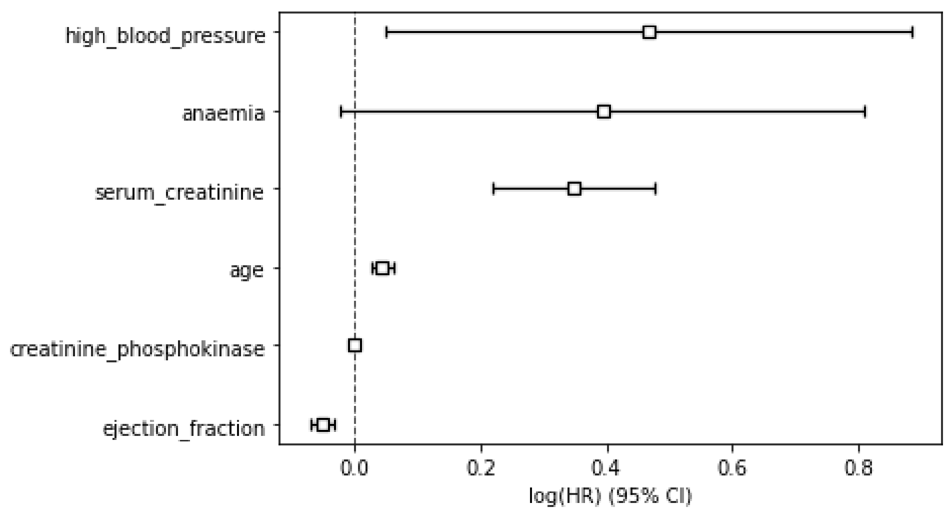


Fig. 25: Feature Importance graph

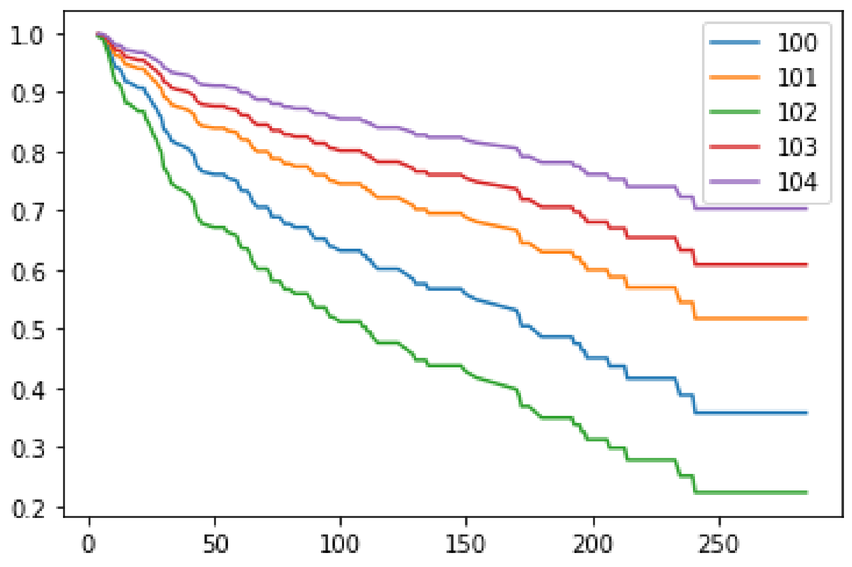


Fig. 26: Survival curve plotted using Cox regression method with reduced features for patients indexed 100-104.
Survival Duration is on X axis (in days)
Survival Probability is on the Y axis.

5.3. AdaBoost Regression

In order to achieve a better range in accuracy for prediction of survivability of several colon cancer patients, we have implemented the adaptive boosting regression technique using the AdaBoostRegressor model from the ensemble module in the Scikit-learn package of python. The Colon cancer dataset is fitted using the sklearn AdaBoostRegressor model and then average accuracy is calculated.

The model is fitted using 3 different base_estimators: 1-DecisionTreeRegressor, which is the default base estimator for AdaBoost, initialized with max_depth=3.
2-RandomForestRegressor(rfr) initialized with n_estimators = 100, and
3-KNeighborsRegressor(knn) initialized with n_neighbors=20, metric='euclidean'.
Prediction for accuracy of the model is being made with the total number of true positives, true negatives, false positives and false negatives in the data using the actual and predicted values of the dependent variable. This part of the code has been executed 100 times for each of the base estimators mentioned above and the respective accuracies are stored in three different text files. The average accuracy is then calculated for those base estimators of AdaBoostRegressor and plotted in the form of histograms with models versus the average accuracy. Visualization tools like pyplot have been used for this.

We have taken different sets of such average accuracies to finally conclude that AdaBoostRegressor with KNeighborsRegressor(knn) as its base_estimator performs with the best accuracy among the three. Although, sometimes the default base_estimator, i.e., DecisionTreeRegressor has been observed to perform well but never better than knn as base_estimator.

For the Colon cancer dataset :

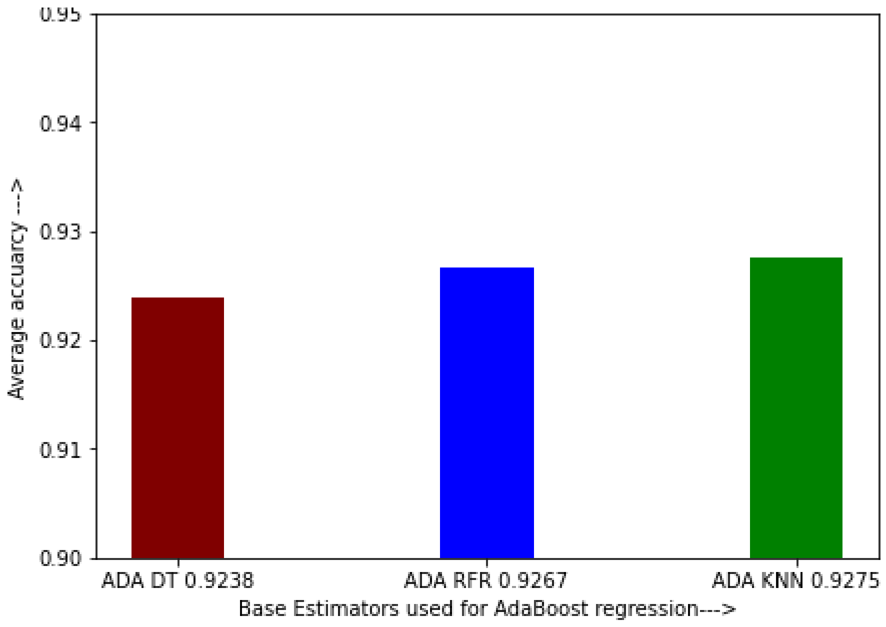


Fig. 27: Performance comparison of AdaBoost Regression with different types of Base Estimators; where, ADA DT implies DecisionTreeRegressor, ADA RFR implies RandomForestRegressor, and ADA KNN implies KNeighborsRegressor. Base Estimators used are plotted over the X-axis and Average accuracy on the Y-axis (Run 1)

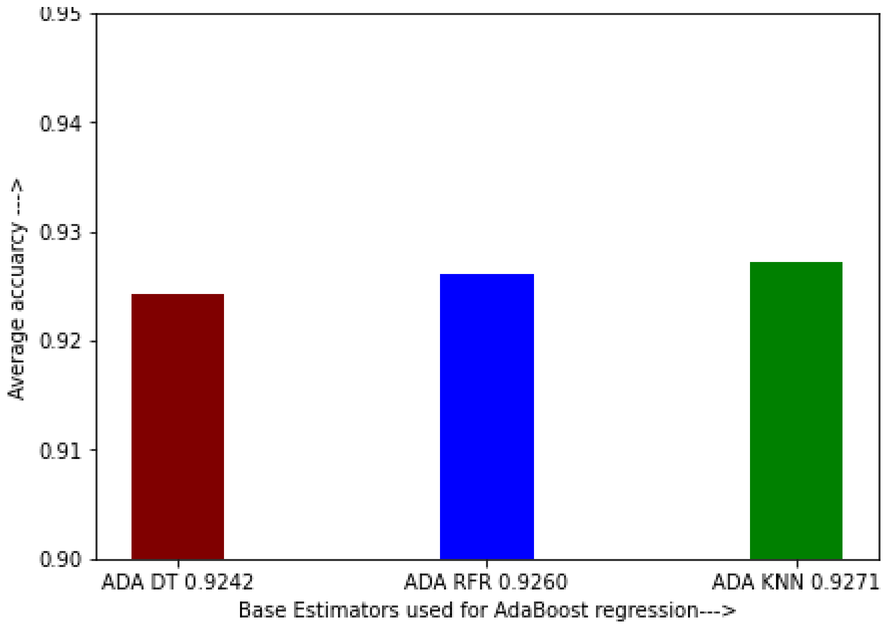


Fig. 28: Performance comparison of AdaBoost Regression with different types of Base Estimators; where, ADA DT implies DecisionTreeRegressor, ADA RFR implies RandomForestRegressor, and ADA KNN implies KNeighborsRegressor. Base Estimators used are plotted over the X-axis and Average accuracy on the Y-axis (Run 2)

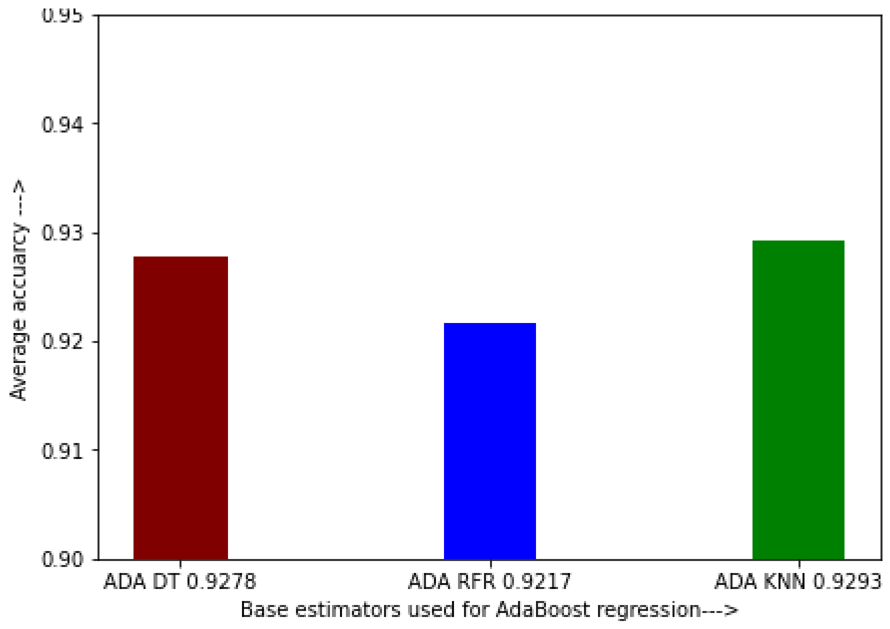


Fig. 29: Performance comparison of AdaBoost Regression with different types of Base Estimators; where, ADA DT implies DecisionTreeRegressor, ADA RFR implies RandomForestRegressor, and ADA KNN implies KNeighborsRegressor. Base Estimators used are plotted over the X-axis and Average accuracy on the Y-axis (Run 3)

5.4. XGBoost Regression in comparison to AdaBoost

Further, in order to achieve a better range in accuracy for prediction of survivability of several colon cancer patients, we have implemented the extreme gradient boosting regression technique using the XGBoostRegressor model from the ensemble module in the Scikit-learn package of python. The Colon cancer dataset is fitted using the sklearn XGBoostRegressor model and then average accuracy is calculated. We then compare the average accuracy of XGBoost regression with average accuracy of AdaBoost regression.

At first, XGBoostRegressor model is fitted with all the parameter as default values and AdaBoostRegressor model is fitted with DecisionTreeRegressor which is the default base_estimator for AdaBoost, initialized with max_depth=3. Next, XGBoostRegressor model is fitted with all the parameter as default values and AdaBoostRegressor model is fitted with RandomForestRegressor(rfr) as base_estimator initialized with n_estimators = 100 and AdaBoostRegressor model is fitted with KNeighborsRegressor(knn) as base_estimator initialized with n_neighbors=20, metric='euclidean'.

Prediction for accuracy of the model is being made with the total number of true positives, true negatives, false positives and false negatives in the data using the actual and predicted values of the dependent variable. This part of the code has been executed 100 times for each of the models mentioned above and the respective accuracies are stored in three different text files. The average accuracy is then calculated for those models of XGBoostRegressor and AdaBoostRegressor and plotted in the form of histograms with models versus the average accuracy. Visualization tools like pyplot have been used for this.

We have taken different sets of such average accuracies to finally conclude that XGBoost performs with the best accuracy in comparison to AdaBoost with the default base estimator, RandomForestRegressor as base estimator and KNeighborsRegressor as base estimator.

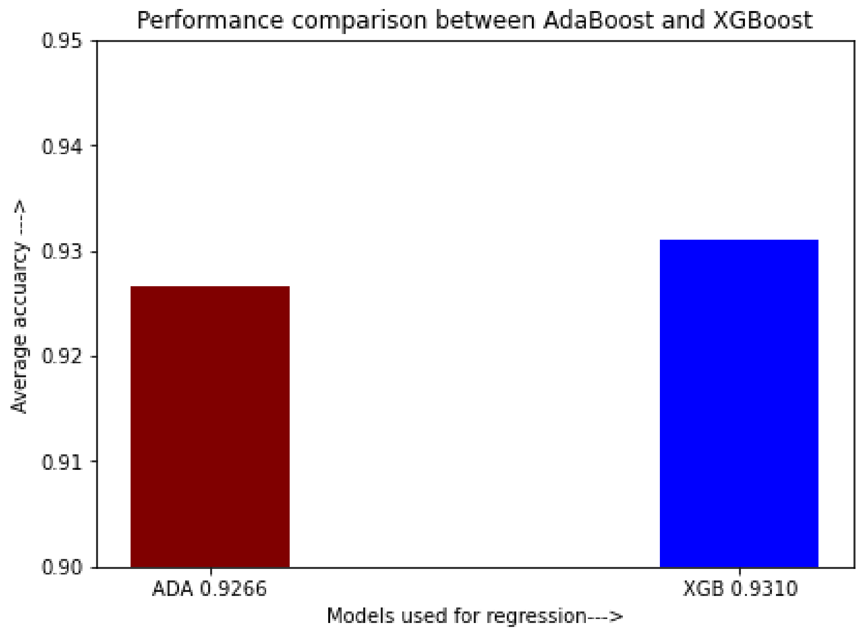


Fig. 30: Performance comparison of AdaBoost Regression and XGBoost Regression Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 1)

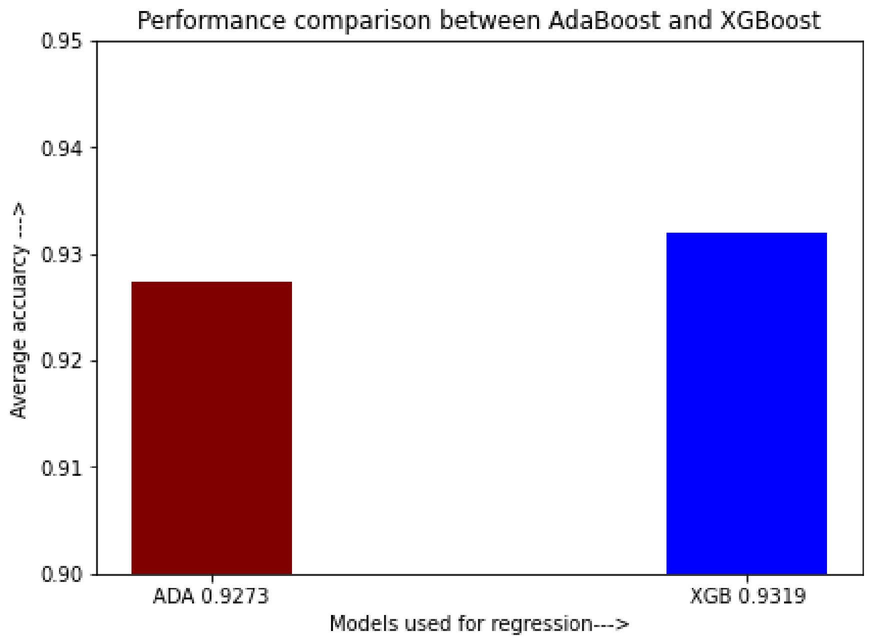


Fig. 31: Performance comparison of AdaBoost Regression and XGBoost Regression Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 2)

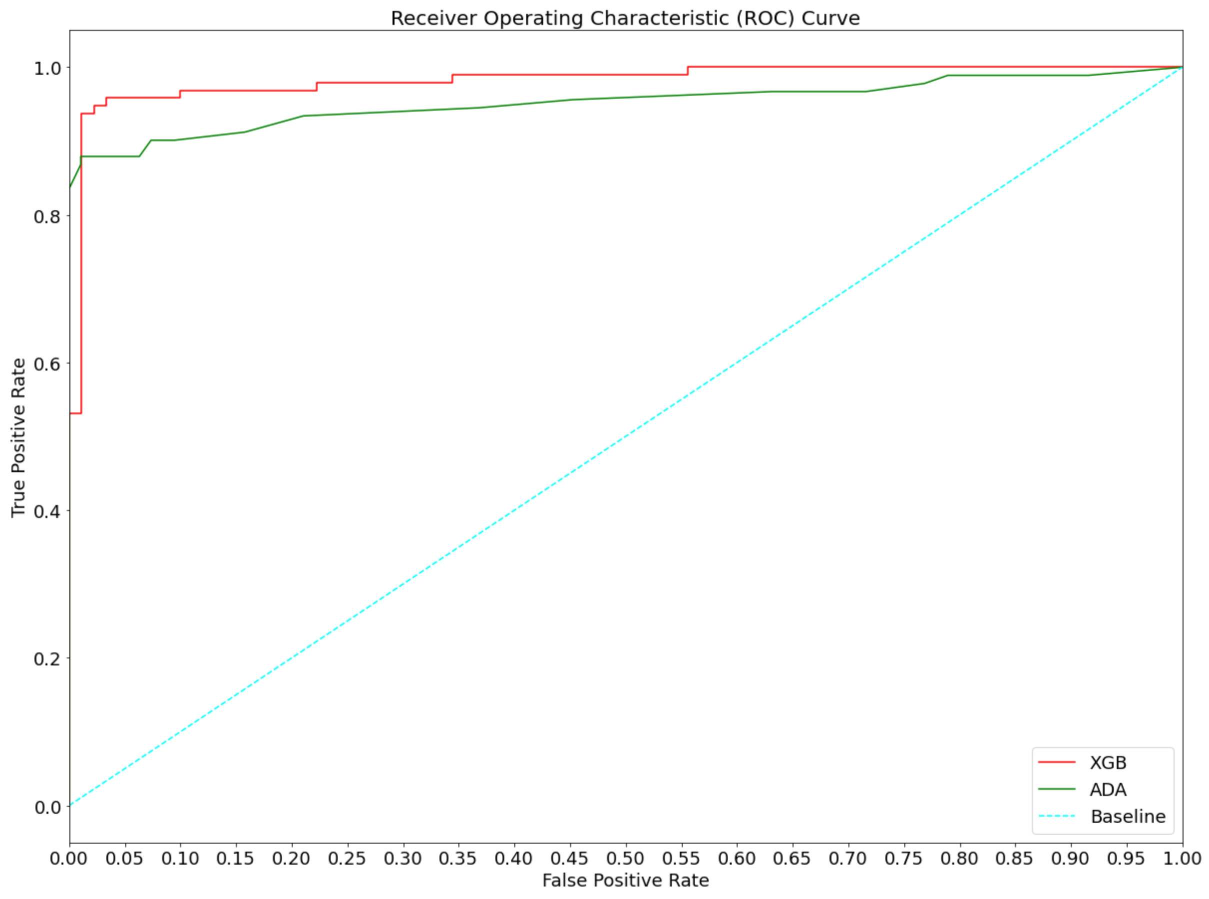


Fig. 32: ROC curve of XGBoost Regression and AdaBoost Regression

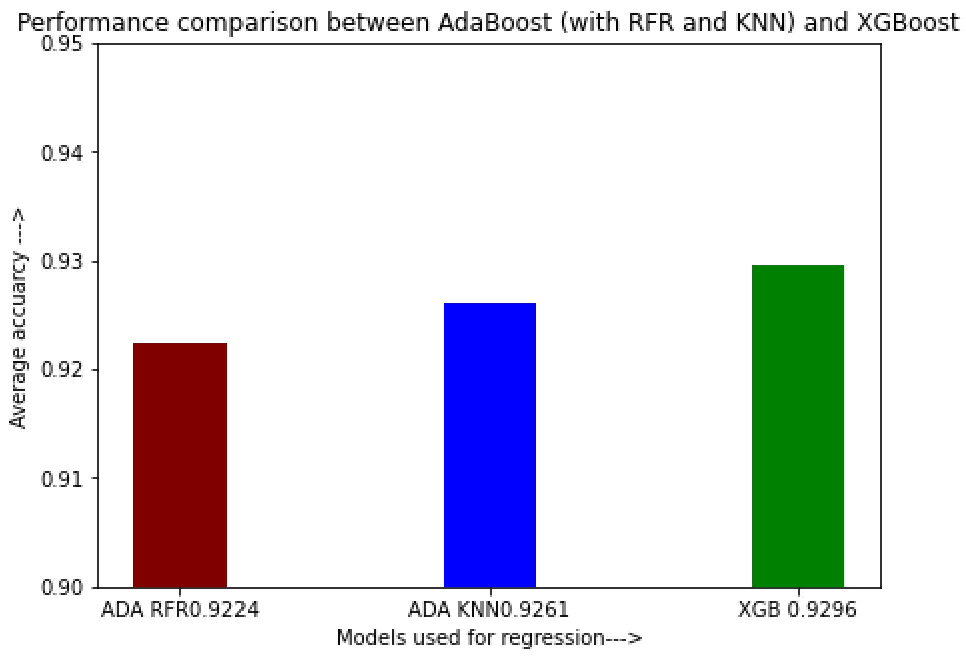


Fig. 33: Performance comparison of AdaBoost Regression with different types of Base Estimators; where, ADA RFR implies RandomForestRegressor, and ADA KNN implies KNeighborsRegressor and XGBoost Regression.
Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 1)

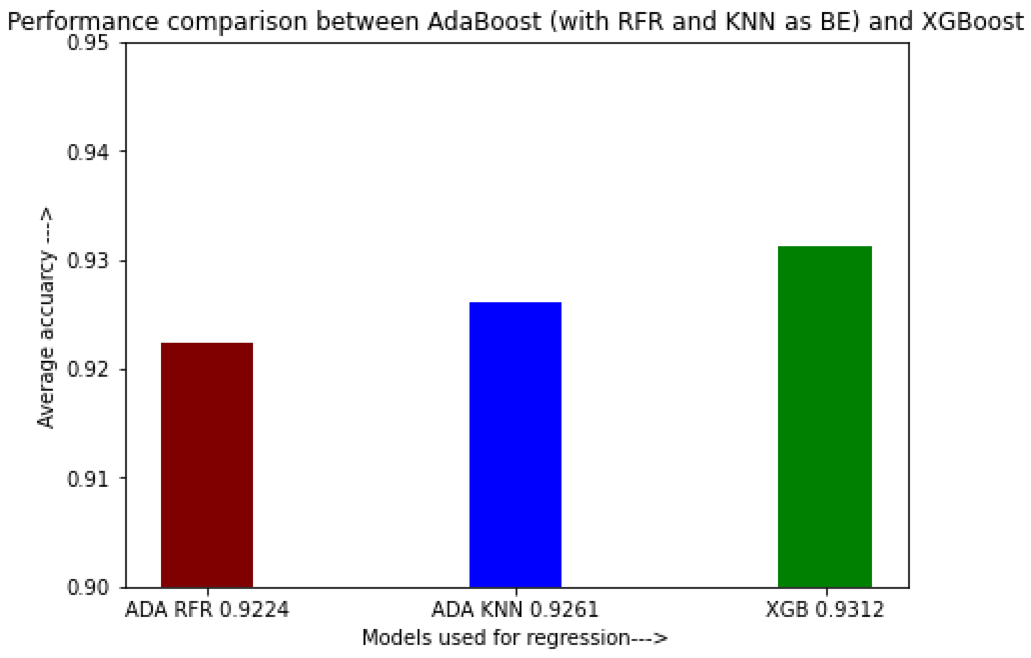


Fig. 34: Performance comparison of AdaBoost Regression with different types of Base Estimators; where, ADA RFR implies RandomForestRegressor, and ADA KNN implies KNeighborsRegressor and XGBoost Regression.
Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 2)

5.5. XGBoost Regression (Parameter Tuned) in Comparison to XGBoost (Default Parameter)

Further, in order to achieve a better range in accuracy for prediction of survivability of several colon cancer patients, we have implemented the extreme gradient boosting regression technique using the XGBoostRegressor model with parameter tuning. The Colon cancer dataset is fitted using the sklearn XGBoostRegressor model and then average accuracy is calculated. We then compare the average accuracy of XGBoost regression (parameter tuned) with average accuracy of XGBoost regression (default parameter).

XGBoostRegressor (parameter tuned) model is fitted with the parameter values as: max_depth=3, learning_rate=0.02, n_estimators=200, min_child_weight=5, gamma=0.4, colsample_bytree=0.6, subsample=0.8, reg_alpha=1.1.

The optimal values of the parameter are found using the GridSearchCV object residing in the model_selection module of sklearn library. A set of possible optimal values are stored in a list namely param_test. Then an GridSearchCV object is created passing the parameters as: estimator = XGBRegressor(max_depth=3, n_estimators=200, learning_rate=0.05), param_grid = param_test, scoring='roc_auc', n_jobs=4, cv=5. The GridSearchCV object is then fitted with X_train and Y_train data. The best parameter value and best score are found from best_params_ and best_score_ attributes of the GridSearchCV object.

XGBoostRegressor (default parameter) model is fitted with all the parameters as their default values.

Prediction for accuracy of the model is being made with the total number of true positives, true negatives, false positives and false negatives in the data using the actual and predicted values of the dependent variable. This part of the code has been executed 100 times for each of the models mentioned above and the respective accuracies are stored in three different text files. The average accuracy is then calculated for those models of XGBoostRegressor (parameter tuned) and XGBoostRegressor (default parameter) and plotted in the form of histograms with models versus the average accuracy. Visualization tools like pyplot have been used for this.

We have taken different sets of such average accuracies to finally conclude that XGBoost (parameter tuned) performs with the best accuracy in comparison to XGBoost (default parameter). This is the highest accuracy we have got in our project span.

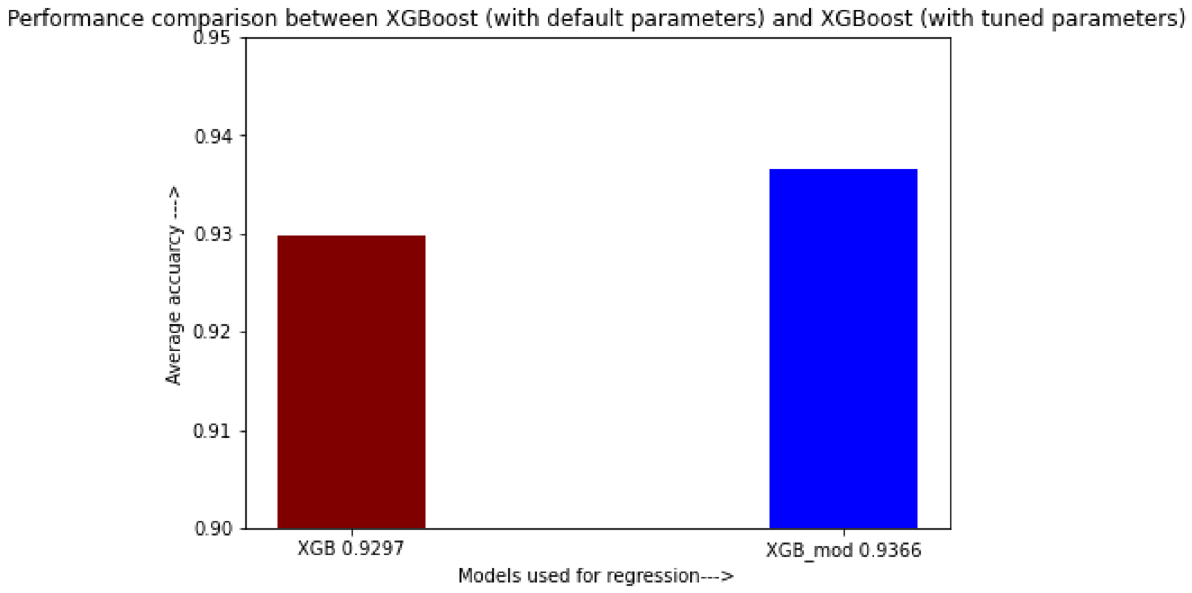


Fig. 35: Performance comparison of XGBoost Regression (default parameter) and XGBoost Regression (parameter tuned)
Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 1)

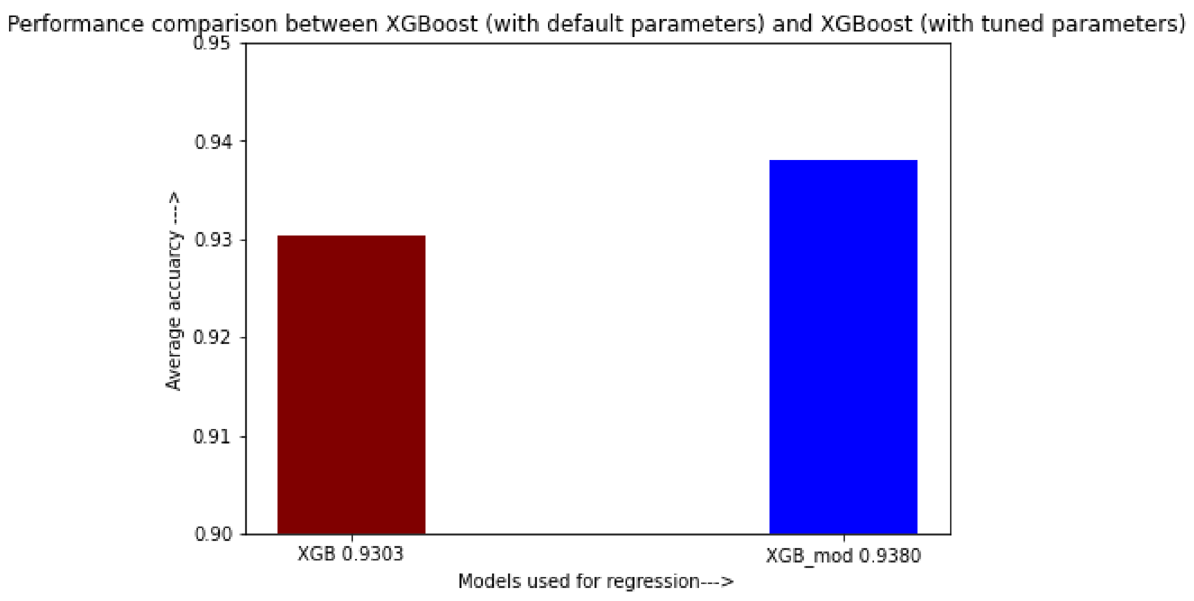


Fig. 36: Performance comparison of XGBoost Regression (default parameter) and XGBoost Regression (parameter tuned)
Models are plotted over the X-axis and Average accuracy on the Y-axis (Run 2)

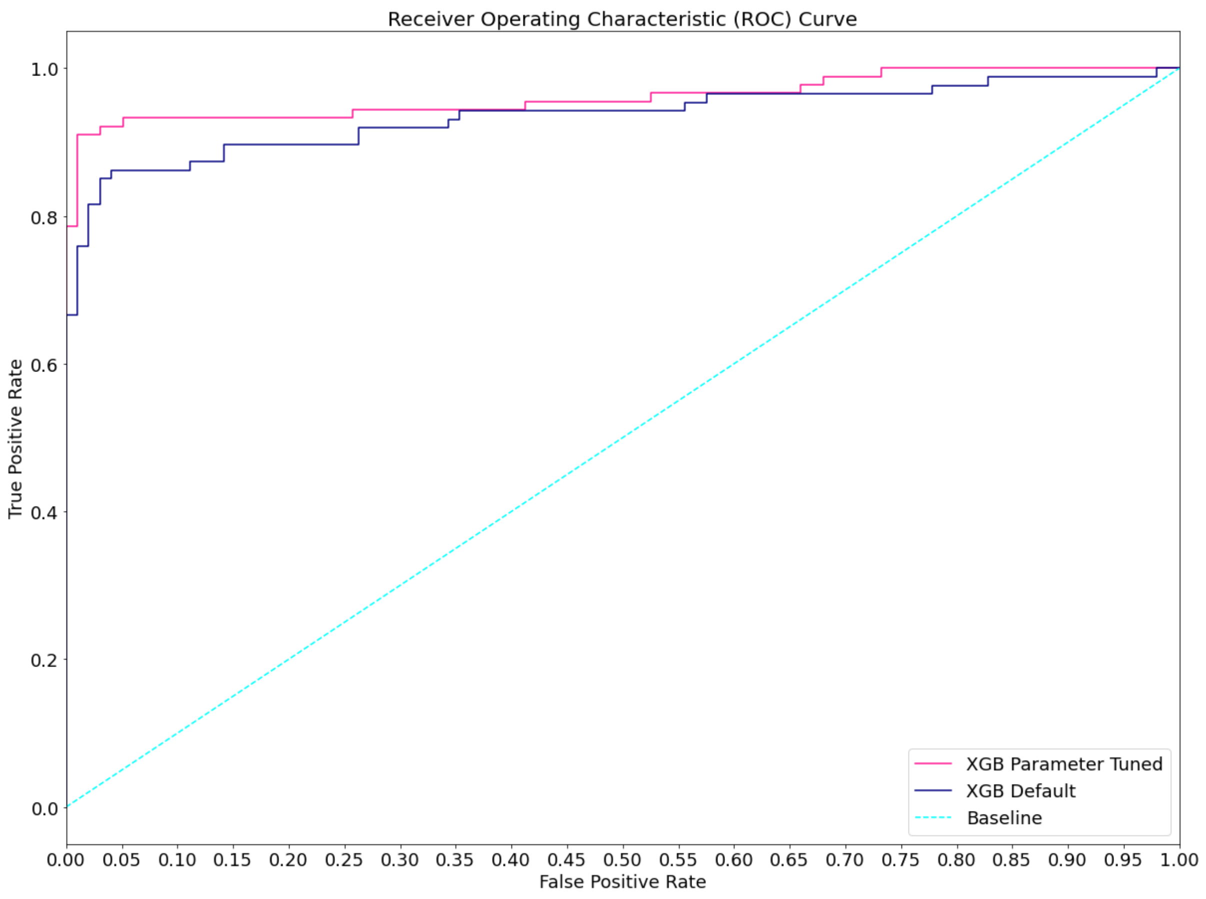


Fig. 37: ROC curve of XGBoost Regression (default parameter) and XGBoost Regression (parameter tuned)

6. Conclusion

To analyze time-to-event data, we have applied Kaplan- Meier method and Cox Regression, which are dedicated models for survival analysis , on both Colon Cancer and Heart failure datasets. To predict the survival probability of Colon Cancer patients, we have applied ensemble learning methods such as Adaptive Boosting Regression(AdaBoost) and Extreme Gradient Boosting Regression(XGBoost) as these models overcome the drawbacks of Kaplan-Meier and Cox Regression methods. The AdaBoost has been run using three different base estimators; DecisionTreeRegressor, which is the default base estimator, KNeighbors, and RandomForestRegressor. We observed that AdaBoostRegressor with KNeighborsRegressor(knn) as its base_estimator performs with the best accuracy (92.93%) among the three. Further, in order to achieve a better range in accuracy for prediction of survivability of several colon cancer patients, we have implemented the XGBoost. We observed that XGBoost (93.10%) always performed better (0.44%) than AdaBoost (92.66%), when fitted with three different base estimators; DecisionTreeRegressor, which is the default base estimator, KNeighbors, and RandomForestRegressor. In order to improve the accuracy of the XGBoost model, we tuned the parameters by finding out optimal values of each parameter. Our parameter tuned XGBoost model (93.80%) performed better (0.77%) than XGBoost with default parameters (93.03%), giving best accuracy among all the models. To overcome the limitations of performing survival analysis on a centralized system, we have implemented our parameter tuned XGBoost algorithm on a distributed framework, MongoDB, and observed that the algorithm performs equally good on MongoDB. In the future, we would like to work on increasing the accuracy of predicting the survival probability of colon cancer patients using deep learning methods.

References

[1] Dudley, W.N., Wickham, R. and Coombs, N., 2016. An introduction to survival statistics: Kaplan-Meier analysis. *Journal of the advanced practitioner in oncology*, *7*(1), p.91.

[2] George, B., Seals, S. and Aban, I., 2014. Survival analysis and regression models. *Journal of nuclear cardiology*, *21*(4), pp.686-694.

[3] Bland, J.M. and Altman, D.G., 1998. Survival probabilities (the Kaplan-Meier method). *Bmj*, *317*(7172), pp.1572-1580.

[4] Lira, R.P.C., Antunes-Foschini, R. and Rocha, E.M., 2020. Survival analysis (Kaplan-Meier curves): a method to predict the future. *Arquivos Brasileiros de Oftalmologia*, *83*, pp.V-VII.

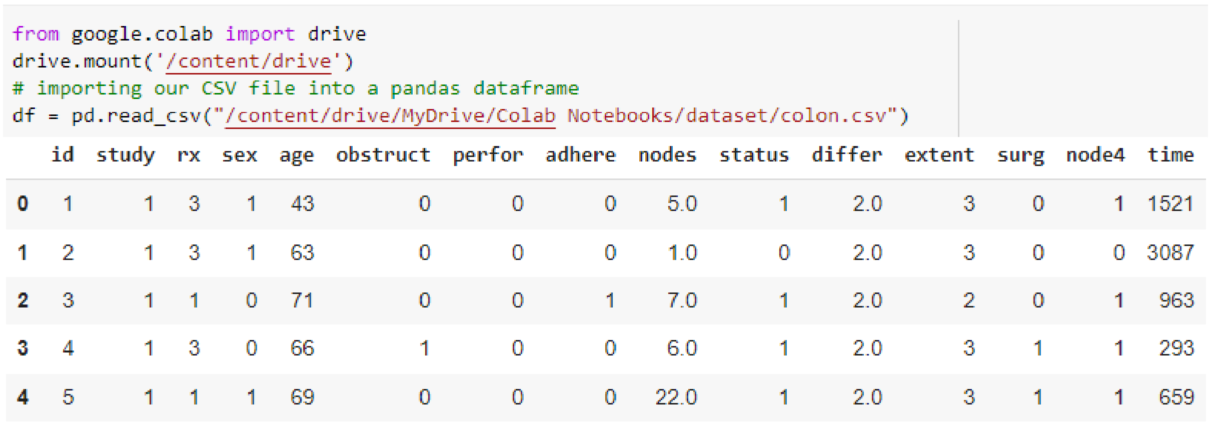
[5] Efron, B., 1988. Logistic regression, survival analysis, and the Kaplan-Meier curve. *Journal of the American Statistical Association*, *83*(402), pp.414-425.

[6] Adelian, R., Jamali, J., Zare, N., Ayatollahi, S.M.T., Pooladfar, G.R. and Roustaei, N.A.R.G.E.S., 2015. Comparison of Cox’s regression model and parametric models in evaluating the prognostic factors for survival after liver transplantation in Shiraz during 2000–2012. *International journal of organ transplantation medicine*, *6*(3), p.119.

[7] Babińska, M., Chudek, J., Chełmecka, E., Janik, M., Klimek, K. and Owczarek, A., 2015. Limitations of cox proportional hazards analysis in mortality prediction of patients with acute coronary syndrome. *Studies in Logic, Grammar and Rhetoric*, 43(1), pp.33-48.
[8] Dietterich, T.G., 2000, June. Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Springer, Berlin, Heidelberg.
[9] Schapire, R.E., 2003. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pp.149-171.
[10] Schapire, R.E., 2013. Explaining adaboost. In *Empirical inference* (pp. 37-52). Springer, Berlin, Heidelberg.
[11] Song, Y.Y. and Ying, L.U., 2015. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2), p.130.
[12] Oliver, J.J. and Hand, D., 1994, April. Averaging over decision stumps. In *European conference on machine learning* (pp. 231-241). Springer, Berlin, Heidelberg.
[13] Drucker, H., 1997, July. Improving regressors using boosting techniques. In *ICML* (Vol. 97, pp. 107-115).
[14] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H. and Chen, K., 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), pp.1-4.
[15] Bentéjac, C., Csörgő, A. and Martínez-Muñoz, G., 2021. A comparative analysis of gradient boosting algorithms. Artificial Intelligence Review, 54(3), pp.1937-1967.
[16] Šimundić, A.M., 2009. Measures of diagnostic accuracy: basic definitions. *ejifcc*, *19*(4), p.203.
[17] Hoo, Z.H., Candlish, J. and Teare, D., 2017. What is an ROC curve?. *Emergency Medicine Journal*, *34*(6), pp.357-359.
[18] Moertel, C.G., Fleming, T.R., Macdonald, J.S., Haller, D.G., Laurie, J.A., Tangen, C.M., Ungerleider, J.S., Emerson, W.A., Tormey, D.C., Glick, J.H. and Veeder, M.H., 1995. Fluorouracil plus levamisole as effective adjuvant therapy after resection of stage III colon carcinoma: a final report. *Annals of internal medicine*, 122(5), pp.321-326.
[19] Chicco, D. and Jurman, G., 2020. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC medical informatics and decision making*, 20(1), pp.1-16.
[20] Chauhan, A., 2019. A Review on Various Aspects of MongoDb Databases. *International Journal of Engineering Research & Technology* (IJERT), 8(5).
[21] Benhlima, L., 2018. Big data management for healthcare systems: architecture, requirements, and implementation. Advances in bioinformatics, 2018.
[22] What is MongoDB? — MongoDB Manual - https://docs.mongodb.com/manual/

Supplementary Materials

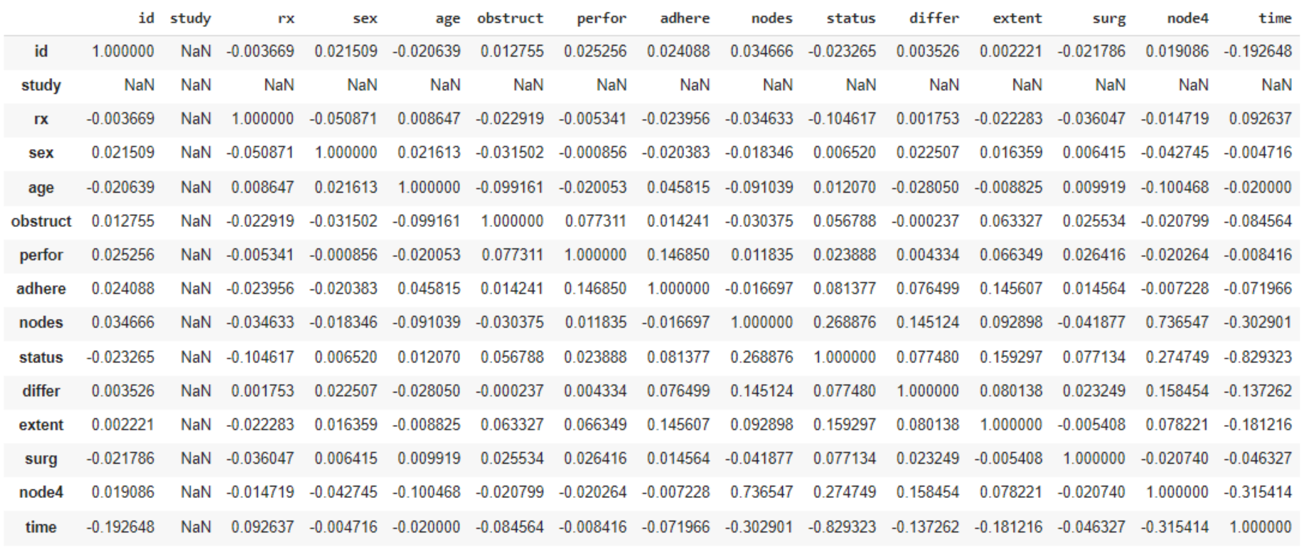
Datasets:



```
from google.colab import drive
drive.mount('/content/drive')
# importing our CSV file into a pandas dataframe
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/dataset/colon.csv")
```

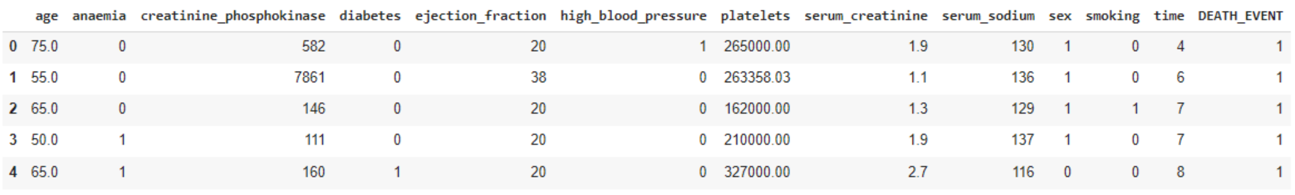
	id	study	rx	sex	age	obstruct	perfor	adhere	nodes	status	differ	extent	surg	node4	time
0	1	1	3	1	43	0	0	0	5.0	1	2.0	3	0	1	1521
1	2	1	3	1	63	0	0	0	1.0	0	2.0	3	0	0	3087
2	3	1	1	0	71	0	0	1	7.0	1	2.0	2	0	1	963
3	4	1	3	0	66	1	0	0	6.0	1	2.0	3	1	1	293
4	5	1	1	1	69	0	0	0	22.0	1	2.0	3	1	1	659

Fig. 38: First 5 samples of dataset 1



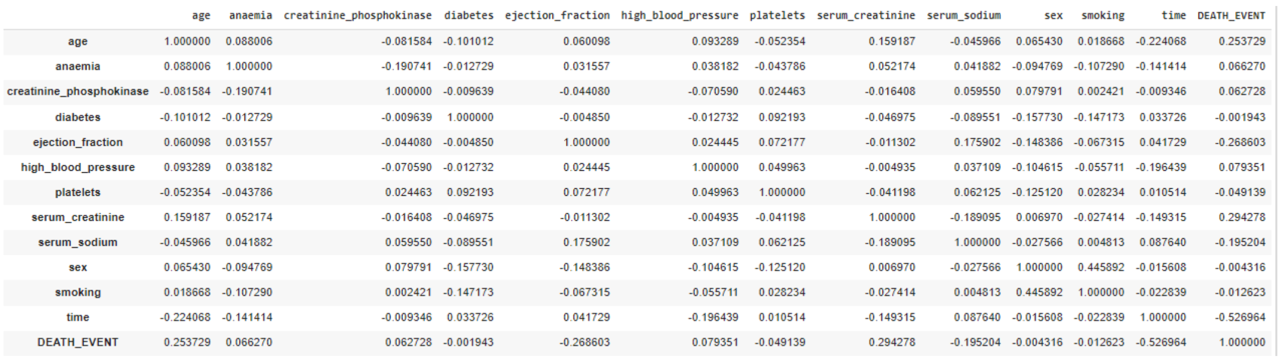
	id	study	rx	sex	age	obstruct	perfor	adhere	nodes	status	differ	extent	surg	node4	time
id	1.000000	NaN	-0.003669	0.021509	-0.020639	0.012755	0.025256	0.024088	0.034666	-0.023265	0.003526	0.002221	-0.021786	0.019086	-0.192648
study	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rx	-0.003669	NaN	1.000000	-0.050871	0.008647	-0.022919	-0.005341	-0.023956	-0.034633	-0.104617	0.001753	-0.022283	-0.036047	-0.014719	0.092637
sex	0.021509	NaN	-0.050871	1.000000	0.021613	-0.031502	-0.000856	-0.020383	-0.018346	0.006520	0.022507	0.016359	0.006415	-0.042745	-0.004716
age	-0.020639	NaN	0.008647	0.021613	1.000000	-0.099161	-0.020053	0.045815	-0.091039	0.012070	-0.028050	-0.008825	0.009919	-0.100468	-0.020000
obstruct	0.012755	NaN	-0.022919	-0.031502	-0.099161	1.000000	0.077311	0.014241	-0.030375	0.056788	-0.000237	0.063327	0.025534	-0.020799	-0.084564
perfor	0.025256	NaN	-0.005341	-0.000856	-0.020053	0.077311	1.000000	0.146850	0.011835	0.023888	0.004334	0.066349	0.026416	-0.020264	-0.008416
adhere	0.024088	NaN	-0.023956	-0.020383	0.045815	0.014241	0.146850	1.000000	-0.016697	0.081377	0.076499	0.145607	0.014564	-0.007228	-0.071966
nodes	0.034666	NaN	-0.034633	-0.018346	-0.091039	-0.030375	0.011835	-0.016697	1.000000	0.268876	0.145124	0.092898	-0.041877	0.736547	-0.302901
status	-0.023265	NaN	-0.104617	0.006520	0.012070	0.056788	0.023888	0.081377	0.268876	1.000000	0.077480	0.159297	0.077134	0.274749	-0.829323
differ	0.003526	NaN	0.001753	0.022507	-0.028050	-0.000237	0.004334	0.076499	0.145124	0.077480	1.000000	0.080138	0.023249	0.158454	-0.137262
extent	0.002221	NaN	-0.022283	0.016359	-0.008825	0.063327	0.066349	0.145607	0.092898	0.159297	0.080138	1.000000	-0.005408	0.078221	-0.181216
surg	-0.021786	NaN	-0.036047	0.006415	0.009919	0.025534	0.026416	0.014564	-0.041877	0.077134	0.023249	-0.005408	1.000000	-0.020740	-0.046327
node4	0.019086	NaN	-0.014719	-0.042745	-0.100468	-0.020799	-0.020264	-0.007228	0.736547	0.274749	0.158454	0.078221	-0.020740	1.000000	-0.315414
time	-0.192648	NaN	0.092637	-0.004716	-0.020000	-0.084564	-0.008416	-0.071966	-0.302901	-0.829323	-0.137262	-0.181216	-0.046327	-0.315414	1.000000

Fig. 39: Pearson Correlation between the features of dataset 1



	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1

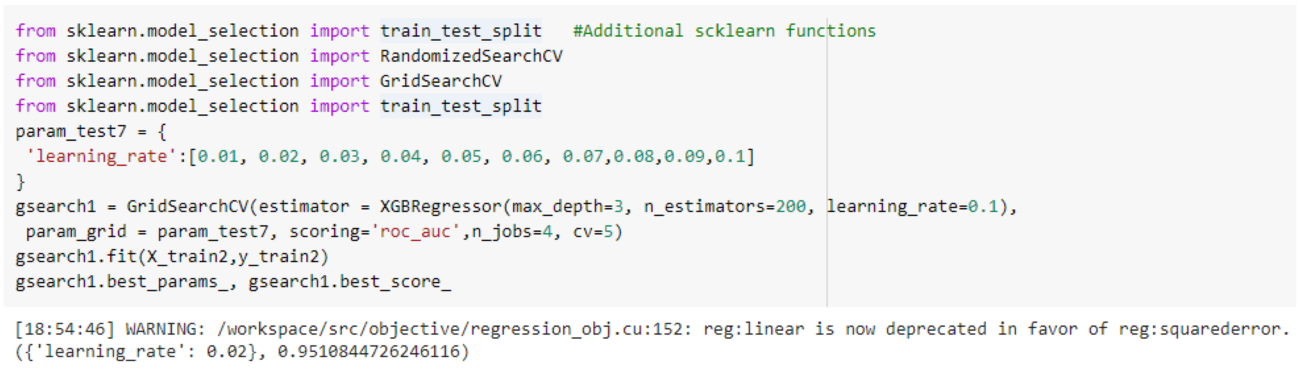
Fig. 40: First 5 samples of dataset 2



	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
age	1.000000	0.088006	-0.081584	-0.101012	0.060098	0.093289	-0.052354	0.159187	-0.045966	0.065430	0.018668	-0.224068	0.253729
anaemia	0.088006	1.000000	-0.190741	-0.012729	0.031557	0.038182	-0.043786	0.052174	0.041882	-0.094769	-0.107290	-0.141414	0.066270
creatinine_phosphokinase	-0.081584	-0.190741	1.000000	-0.009639	-0.044080	-0.070590	0.024463	-0.016408	0.059550	0.079791	0.002421	-0.009346	0.062728
diabetes	-0.101012	-0.012729	-0.009639	1.000000	-0.004850	-0.012732	0.092193	-0.046975	-0.089551	-0.157730	-0.147173	0.033726	-0.001943
ejection_fraction	0.060098	0.031557	-0.044080	-0.004850	1.000000	0.024445	0.072177	-0.011302	0.175902	-0.148386	-0.067315	0.041729	-0.268603
high_blood_pressure	0.093289	0.038182	-0.070590	-0.012732	0.024445	1.000000	0.049963	-0.004935	0.037109	-0.104615	-0.055711	-0.196439	0.079351
platelets	-0.052354	-0.043786	0.024463	0.092193	0.072177	0.049963	1.000000	-0.041198	0.062125	-0.125120	0.028234	0.010514	-0.049139
serum_creatinine	0.159187	0.052174	-0.016408	-0.046975	-0.011302	-0.004935	-0.041198	1.000000	-0.189095	0.006970	-0.027414	-0.149315	0.294278
serum_sodium	-0.045966	0.041882	0.059550	-0.089551	0.175902	0.037109	0.062125	-0.189095	1.000000	-0.027566	0.004813	0.087640	-0.195204
sex	0.065430	-0.094769	0.079791	-0.157730	-0.148386	-0.104615	-0.125120	0.006970	-0.027566	1.000000	0.445892	-0.015608	-0.004316
smoking	0.018668	-0.107290	0.002421	-0.147173	-0.067315	-0.055711	0.028234	-0.027414	0.004813	0.445892	1.000000	-0.022839	-0.012623
time	-0.224068	-0.141414	-0.009346	0.033726	0.041729	-0.196439	0.010514	-0.149315	0.087640	-0.015608	-0.022839	1.000000	-0.526964
DEATH_EVENT	0.253729	0.066270	0.062728	-0.001943	-0.268603	0.079351	-0.049139	0.294278	-0.195204	-0.004316	-0.012623	-0.526964	1.000000

Fig. 41: Pearson Correlation between the features of dataset 2

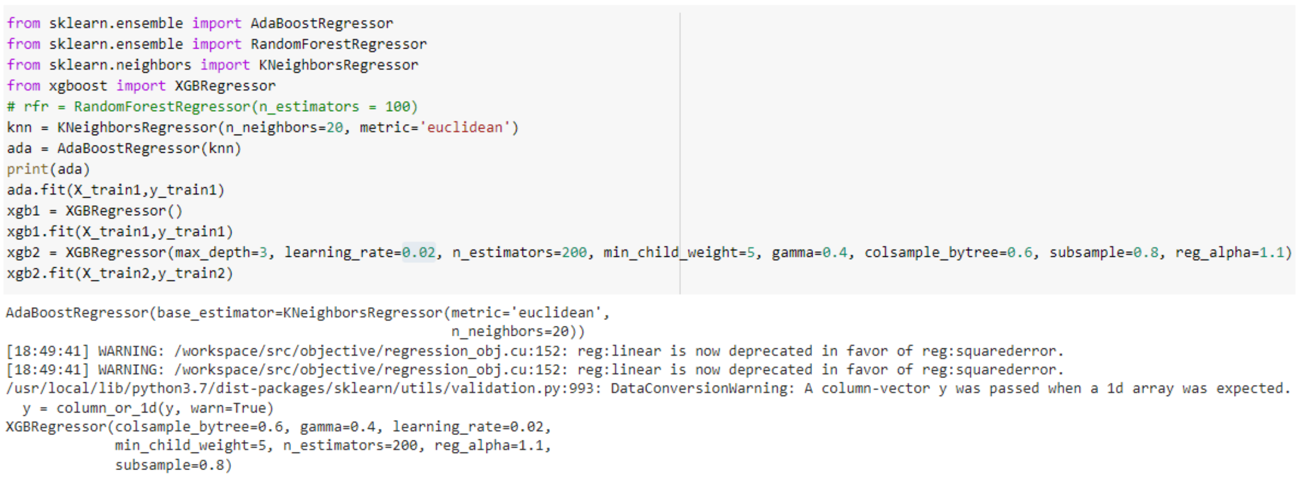
XGBRegressor and Parameter Selection:



```
from sklearn.model_selection import train_test_split   #Additional scklearn functions
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
param_test7 = {
 'learning_rate':[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07,0.08,0.09,0.1]
}
gsearch1 = GridSearchCV(estimator = XGBRegressor(max_depth=3, n_estimators=200, learning_rate=0.1),
 param_grid = param_test7, scoring='roc_auc',n_jobs=4, cv=5)
gsearch1.fit(X_train2,y_train2)
gsearch1.best_params_, gsearch1.best_score_
```

```
[18:54:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
({'learning_rate': 0.02}, 0.9510844726246116)
```

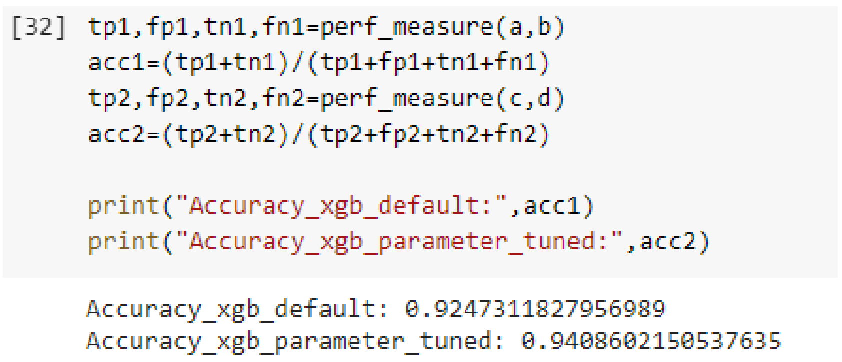
Fig. 42: Choosing best parameter value for 'learning_rate' in XGBRegressor



```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
# rfr = RandomForestRegressor(n_estimators = 100)
knn = KNeighborsRegressor(n_neighbors=20, metric='euclidean')
ada = AdaBoostRegressor(knn)
print(ada)
ada.fit(X_train1,y_train1)
xgb1 = XGBRegressor()
xgb1.fit(X_train1,y_train1)
xgb2 = XGBRegressor(max_depth=3, learning_rate=0.02, n_estimators=200, min_child_weight=5, gamma=0.4, colsample_bytree=0.6, subsample=0.8, reg_alpha=1.1)
xgb2.fit(X_train2,y_train2)
```

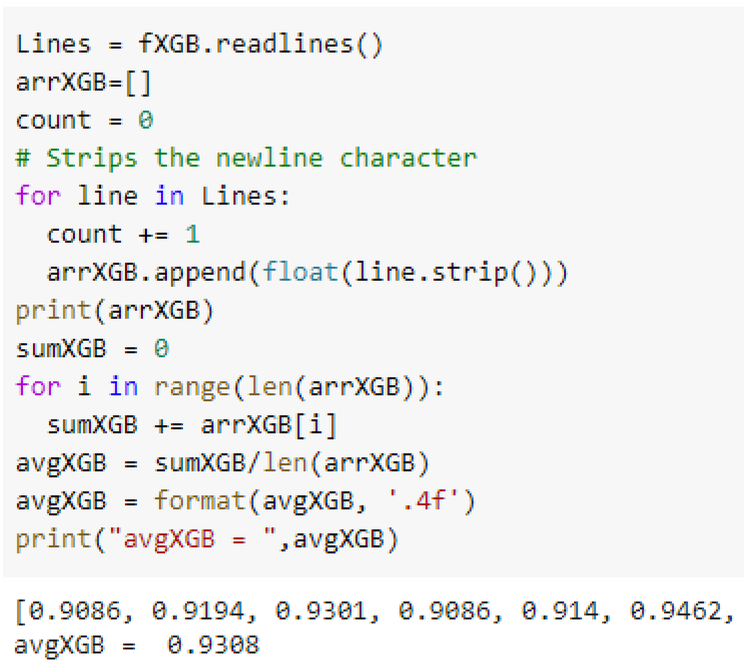
```
AdaBoostRegressor(base_estimator=KNeighborsRegressor(metric='euclidean',
                                                     n_neighbors=20))
[18:49:41] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[18:49:41] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
  y = column_or_1d(y, warn=True)
XGBRegressor(colsample_bytree=0.6, gamma=0.4, learning_rate=0.02,
             min_child_weight=5, n_estimators=200, reg_alpha=1.1,
             subsample=0.8)
```

Fig. 43: Fitting XGBRegressor with the chosen value of 'learning_rate' using the method shown in Fig. 42



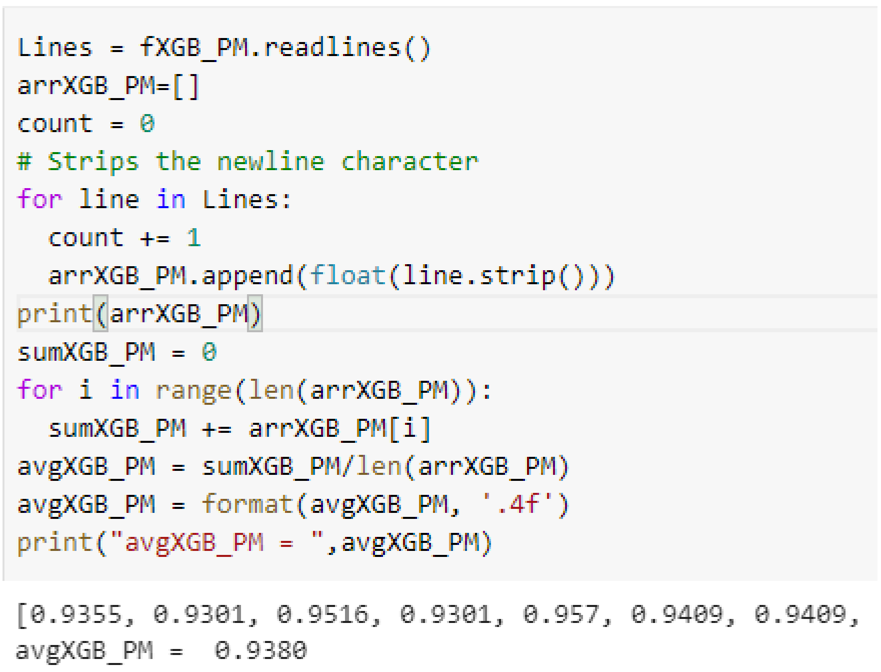
```
[32] tp1,fp1,tn1,fn1=perf_measure(a,b)
     acc1=(tp1+tn1)/(tp1+fp1+tn1+fn1)
     tp2,fp2,tn2,fn2=perf_measure(c,d)
     acc2=(tp2+tn2)/(tp2+fp2+tn2+fn2)

     print("Accuracy_xgb_default:",acc1)
     print("Accuracy_xgb_parameter_tuned:",acc2)

     Accuracy_xgb_default: 0.9247311827956989
     Accuracy_xgb_parameter_tuned: 0.9408602150537635
```

Fig. 44: Comparing the accuracies between XGBRegressor (default parameter) and XGBRegressor (parameter tuned)



```
Lines = fXGB.readlines()
arrXGB=[]
count = 0
# Strips the newline character
for line in Lines:
  count += 1
  arrXGB.append(float(line.strip()))
print(arrXGB)
sumXGB = 0
for i in range(len(arrXGB)):
  sumXGB += arrXGB[i]
avgXGB = sumXGB/len(arrXGB)
avgXGB = format(avgXGB, '.4f')
print("avgXGB = ",avgXGB)

[0.9086, 0.9194, 0.9301, 0.9086, 0.914, 0.9462,
avgXGB =  0.9308
```

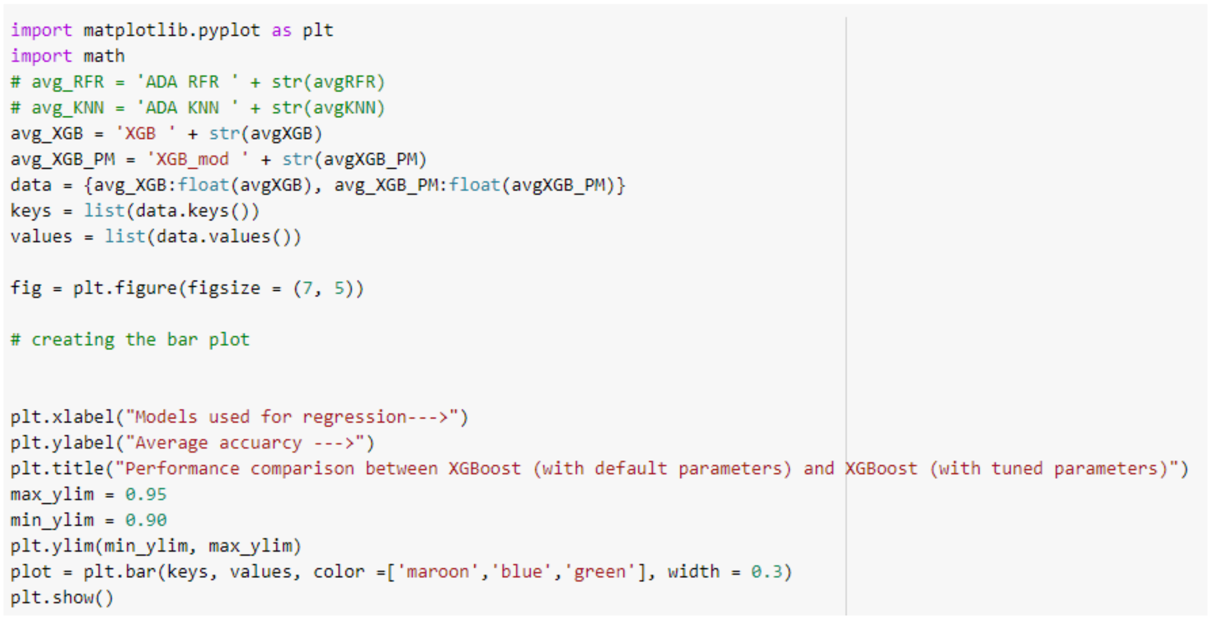
Fig. 45: Calculating average accuracy over 100 such accuracies computed for XGBRegressor



```
Lines = fXGB_PM.readlines()
arrXGB_PM=[]
count = 0
# Strips the newline character
for line in Lines:
  count += 1
  arrXGB_PM.append(float(line.strip()))
print(arrXGB_PM)
sumXGB_PM = 0
for i in range(len(arrXGB_PM)):
  sumXGB_PM += arrXGB_PM[i]
avgXGB_PM = sumXGB_PM/len(arrXGB_PM)
avgXGB_PM = format(avgXGB_PM, '.4f')
print("avgXGB_PM = ",avgXGB_PM)
```

```
[0.9355, 0.9301, 0.9516, 0.9301, 0.957, 0.9409, 0.9409,
avgXGB_PM =  0.9380
```

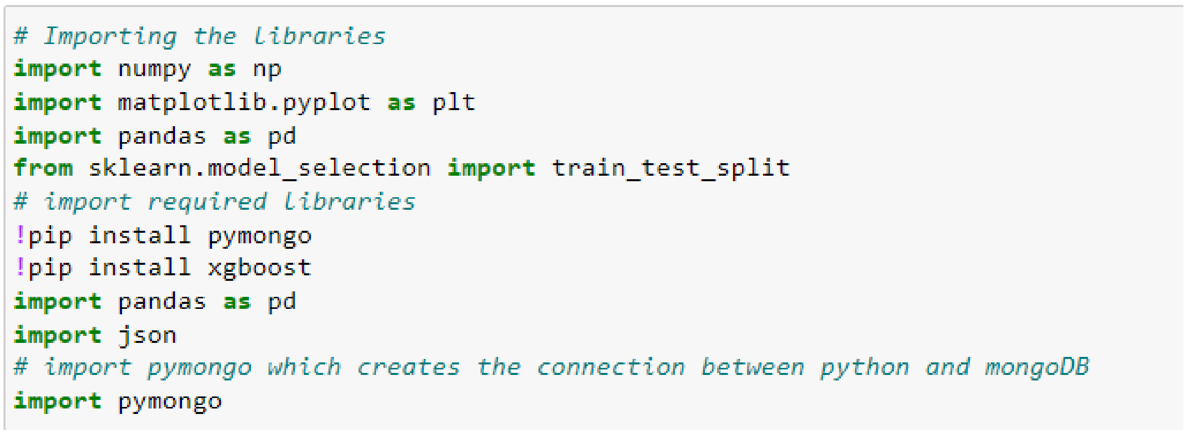
Fig. 46: Calculating average accuracy over 100 such accuracies computed for XGBRegressor (with parameter tuning)



```
import matplotlib.pyplot as plt
import math
# avg_RFR = 'ADA RFR ' + str(avgRFR)
# avg_KNN = 'ADA KNN ' + str(avgKNN)
avg_XGB = 'XGB ' + str(avgXGB)
avg_XGB_PM = 'XGB_mod ' + str(avgXGB_PM)
data = {avg_XGB:float(avgXGB), avg_XGB_PM:float(avgXGB_PM)}
keys = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (7, 5))

# creating the bar plot


plt.xlabel("Models used for regression--->")
plt.ylabel("Average accuarcy --->")
plt.title("Performance comparison between XGBoost (with default parameters) and XGBoost (with tuned parameters)")
max_ylim = 0.95
min_ylim = 0.90
plt.ylim(min_ylim, max_ylim)
plot = plt.bar(keys, values, color =['maroon','blue','green'], width = 0.3)
plt.show()
```

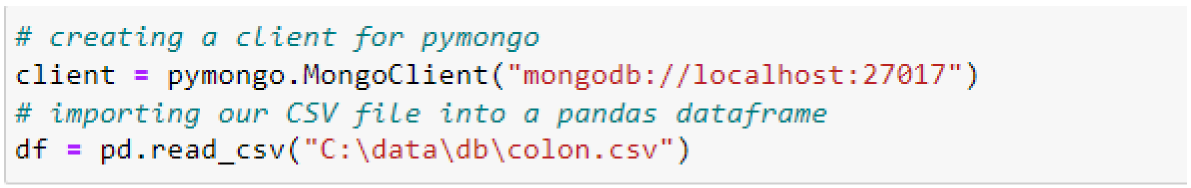
Fig. 47: Plotting average accuracies for XGBRegressor (default parameter) and XGBRegressor (parameter tuned)

MongoDB:



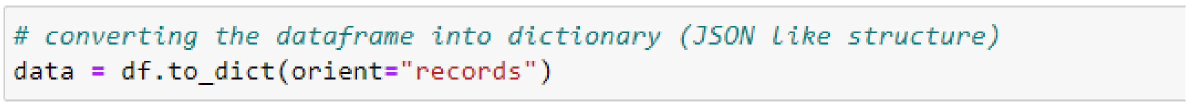
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
# import required libraries
!pip install pymongo
!pip install xgboost
import pandas as pd
import json
# import pymongo which creates the connection between python and mongoDB
import pymongo
```

Fig. 45: Importing PyMongo library



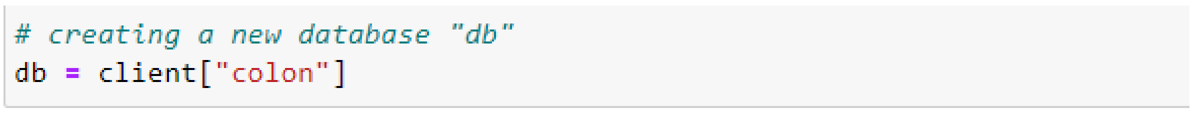
```
# creating a client for pymongo
client = pymongo.MongoClient("mongodb://localhost:27017")
# importing our CSV file into a pandas dataframe
df = pd.read_csv("C:\data\db\colon.csv")
```

Fig. 46: Creating a client for PyMongo, which connects python with MongoDB



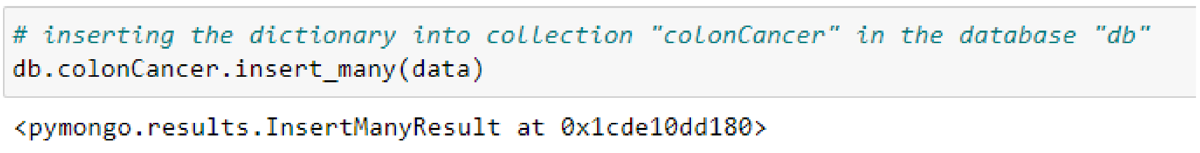
```
# converting the dataframe into dictionary (JSON like structure)
data = df.to_dict(orient="records")
```

Fig. 47: Converting the dataframe into dictionary



```
# creating a new database "db"
db = client["colon"]
```

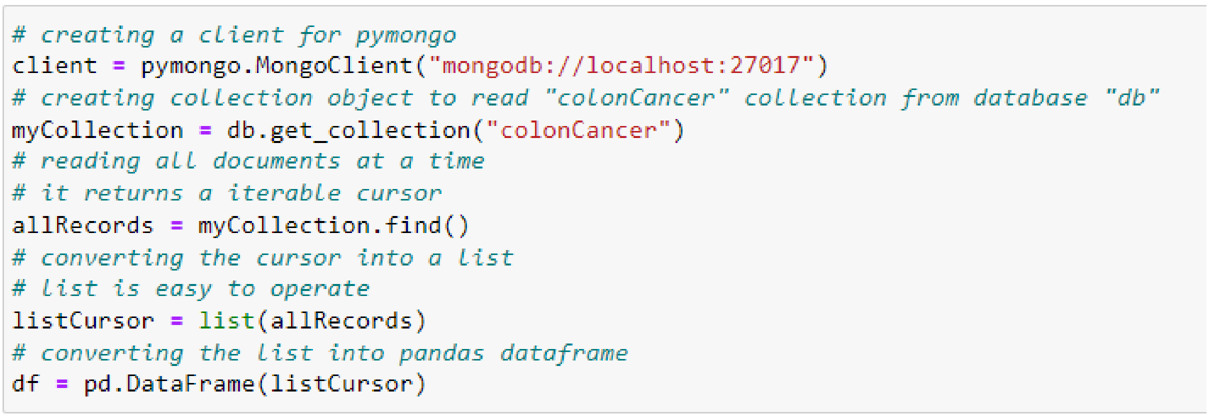
Fig. 48: Creating a new database in MongoDB



```
# inserting the dictionary into collection "colonCancer" in the database "db"
db.colonCancer.insert_many(data)
```

```
<pymongo.results.InsertManyResult at 0x1cde10dd180>
```

Fig. 49: Inserting the dictionary into collection



```
# creating a client for pymongo
client = pymongo.MongoClient("mongodb://localhost:27017")
# creating collection object to read "colonCancer" collection from database "db"
myCollection = db.get_collection("colonCancer")
# reading all documents at a time
# it returns a iterable cursor
allRecords = myCollection.find()
# converting the cursor into a list
# list is easy to operate
listCursor = list(allRecords)
# converting the list into pandas dataframe
df = pd.DataFrame(listCursor)
```

Fig. 50: Reading dataset from MongoDB