Bachelor Thesis

Computer Science and Engineering, 300 credits

AI applications on healthcare data

Computer Science and Engineering, 15 credits

Halmstad 2021-05-30

Oscar Andersson, Tim Andersson

HALMSTAD UNIVERSITY

Abstract

The purpose of this research is to get a better understanding of how different machine learning algorithms work with different amounts of data corruption. This is important since data corruption is an overbearing issue within data collection and thus, in extension, any work that relies on the collected data. The questions we were looking at were: What feature is the most important? How significant is the correlation of features? What algorithms should be used given the data available? And, How much noise (inaccurate or unhelpful captured data) is acceptable?

The study is structured to introduce AI in healthcare, data missingness, and the machine learning algorithms we used in the study. In the method section, we give a recommended workflow for handling data with machine learning in mind.

The results show us that when a dataset is filled with random values, the run-time of algorithms increases since many patterns are lost. Randomly removing values also caused less of a problem than first anticipated since we ran multiple trials, evening out any problems caused by the lost values. Lastly, imputation is a preferred way of handling missing data since it retained many dataset structures. One has to keep in mind if the imputation is done on categories or numerical values.

However, there is no easy "best-fit" for any dataset. It is hard to give a concrete answer when choosing a machine learning algorithm that fits any dataset. Nevertheless, since it is easy to simply plug-and-play with many algorithms, we would recommend any user try different ones before deciding which one fits a project the best.

oscand18@student.hh.se
timand18@student.hh.se

Acknowledgment

We chose this project to introduce ourselves to artificial intelligence and prepare for our masters, and we would like to thank Carmona for giving us the opportunity to work on this. Even though the project had to take a new direction and steer away from the original plan, you always stayed supportive and easy to work with.

We would also like to thank Alexander Galozy for doing an excellent job supervising this project. It would not have been possible without your guidance, both when writing the thesis and tackling machine learning problems. The weekly meetings kept us honest and helped us see the forest from the trees.

Lastly we would like to thank families and friends that have helped us keep sane during the pandemic and pushed us to completing this thesis.

oscand18@student.hh.se
timand18@student.hh.se

Contents

oscand18@student.hh.se
timand18@student.hh.se

oscand18@student.hh.se
timand18@student.hh.se

1 Introduction

Health problems have a huge impact on human living. During patient stay at medical clinics, data is collected of the patient and combined with data from the general public to reach a diagnosis and determine treatment. Therefore data has an important role in improving patient care and addressing health problems. Improved information and data collection is crucial to improving patient care.

The increased collection of data has made advances possible in many domains using machine learning. Such as image recognition [1], speech recognition [2], banking [3], traffic prediction [4] and self-driving cars [5].

The use of data has also improved care in many fields of healthcare like disease identification [6], robotic surgery [7] and personalized medicine [8]. Machine learning is an essential tool when extracting information from data, and data has a central role in healthcare, thus making research in machine learning critical for the future of healthcare. And with the ever increasing amount of data to feed machine learning models, the performance between AI and human experts has narrowed [9].

Be it surveys, clinical studies, observations, or interviews, the risk of data missingness is constantly plaguing data collection. Data missingness or missing values occurs when there is an empty field of data inside a dataset. There can be numerous causes of the missing data, but the fact remains that almost every data collection runs high risks of having missing data in them. The cause can vary from a question being sensitive, and the person taking a survey decides not to answer the question or the question not being relevant for the person and thus is skipped. Every type of missingness is a risk that may cause problems with classification depending on how it is managed [10]. There are various established ways of mitigating missing data by imputing values from the dataset itself or predicting what it could have been [11]. It is not ideal since actual data is always preferred; however, imputing is making the best of a bad situation. It can be enough to produce a satisfactory classification.

In this work we will cover the difficulties researchers will face when gathering there own or working with established data in terms of missingness. Failure to carefully consider the challenges may hinder machine learning models in terms of accuracy and invalidate them for healthcare.

1.1 Original Idea

The project was first meant to be a study of a machine learning classification model on Intermittent Claudication, suggested by Carmona [12], which is stage two of lower extremity arterial disease (LEAD). They were interested in Artificial Intelligence (AI) and saw this as an opportunity to get a prediction-style AI for a disease that needs more attention. The disease affects about 5% of the western population and is equally common between the sexes [13]. Every fourth patient diagnosed with the disease is the victim of a stroke, myocardial infarction, or dies from heart disease within five years of being diagnosed. Joakim Nordanstig, chief physician in vascular surgery at Sahlgrenska University Hospital in Sweden, calls it a 'Hidden and under-treated disease' during a lecture in September 2020 [14].

However, finding medical data is a very daunting task as most of it is personal information locked behind regulations such as General Data Protection Regulation (GDPR). With time as a factor, we were limited in what we realistically could access. As we were looking for datasets, we realized that we would need very specific features to be able to make classifications of the disease. Specifically, ankle-brachial index (ABI) proved to be one of the most challenging and vital features needed to build an AI to predict LEAD. ABI is used to identify the disease, and if the dataset does not have it as a feature, we can not tell our AI when it has made the correct prediction [15].

oscand18@student.hh.se
timand18@student.hh.se

1.2 Purpose & Problem description

Given the problem to be solved, data must be obtained to develop AI methods and algorithms. The quality and quantity of information gathered are essential since they will directly impact how well or poorly the model will work. We will focus on data collection by continually changing the amount of data we use and measuring how this impacts the model. We hope to be able to answer these questions:

I. **What feature is the most important?** - It's always important to know what the feature importance is for a dataset when designing a prediction AI. The reason for this is that more is not always better; features can serve as noise for the algorithms.

II. **How significant is the correlation of the features?** - Correlation can be used to get a general idea of the effect a feature has on another; high correlation means they most likely should be considered in the model. Although, too high of a correlation can deafen other features which, can be a problem.

III. **What algorithms should be used given the data available?** - Some algorithms work better with less data; figuring out which these algorithms are is essential when working on data corruption.

IV. **How much noise (faulty or unhelpful captured data) is acceptable?** - This last point is case-sensitive. In healthcare there is a need for a very high accuracy not to misdiagnose a patient, while in less critical fields, a lower accuracy is acceptable, and model complexity is more of an issue.

In answering these questions, we hope to give insight into how machine learning models will act and behave with different types of data corruption or data missingness. Furthermore, it gives researchers an idea of the baseline of data needed and metrics to look at when doing their research.

1.3 Disposition

In the background (Section 2) of this thesis, we introduce Carmona, which is the company that has worked with us on this project. Clarify what the dataset holds in terms of features and what a machine learning model is. Give a brief explanation of the different algorithms used in the report. We make use of nine different machine learning algorithms to make classifications. This is to achieve a wider grasp of how data affects the algorithms differently as well as gaining a more generalized view of how the algorithms preforms on a classification problem like the one we have. We use three linear algorithms and six non-linear algorithms. Data missingness will be explained as a concept and what can be done to prevent it.

In the method (Section 3), we explain our thought process and workflow when first setting up our prediction AI and then picking it apart using different data corruption techniques.

Our results (Section 4) show our findings when creating our prediction AI and how the different algorithms fared with data corruption and missingness using classification accuracy.

oscand18@student.hh.se
timand18@student.hh.se

2 Background

This chapter gives insight into our research partner Carmona and what their work is. What role AI serves in healthcare. What data missingness is, the different types of data missingness there is, and what measures can be taken to prevent it. We will also discuss machine learning models and algorithms, informing how a prediction AI is built and how the different algorithms used in the research work.

2.1 Carmona

Carmona is a market leader in IT services and solutions for specialist healthcare, focusing on several therapeutic areas. They are in a unique position where they have access to over 50 databases of patient data from the healthcare sector thanks to their collaboration with Halmstad University.

Carmona's main product is Compos. It is a system that collects information and processes it to make things easier for both patients and healthcare professionals. In addition to Compos, Carmona also had other services in development projects and registration studies.

The goal of Compos is to assist patients in need of medical advice or a healthcare professional in need of a large amount of data to make a decision. Relevant answers to questions are accessed through public data or data accessed with the data owner's permission.

2.2 AI in Healthcare

AI in medicine has evolved dramatically over the past five decades. While early AI focused on developing algorithms that could make decisions that previously could only be made by a human, we can now achieve more with advances in computational power paired with massive amounts of data generated in healthcare systems. AI is used in all current healthcare areas, everything from scheduling appointments to drug development or disease diagnostics [16][17][18]. The increase of AI assistance has reduced manual labor for primary care physicians and increases productivity, according to a study done in 2016 [19].

Predicting disease using machine learning AI models is not a new concept, and there are several examples of predicting LEAD [20][21][22][6].

The areas that do have uncertainty and have not been explored enough is the reliance on data [23][24][25].

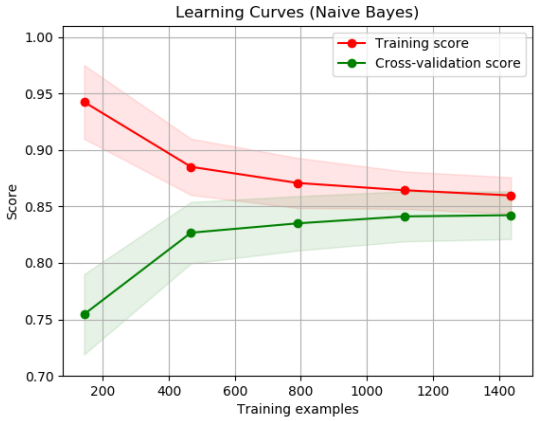
How much data is needed for a project is often unclear, and there are only guidelines. There are many factors at play and no golden rule, mostly guesswork and rule of thumbs to go after. One such rule of thumb is the 'one in ten rule,' [26] which is often used in regression problems [27], which states that one may want ten times as many data points as features to keep the risk of overfitting low [28]. Overfitting is the term for modeling an AI too closely on the training data and thus getting a worse result on the test data. All data have outliers, and focusing too heavily on classifying them will make models worse on new data as a result.

A way to better understand how much data a problem needs is to use a learning curve, a graph, displaying how a model improves with increased events.

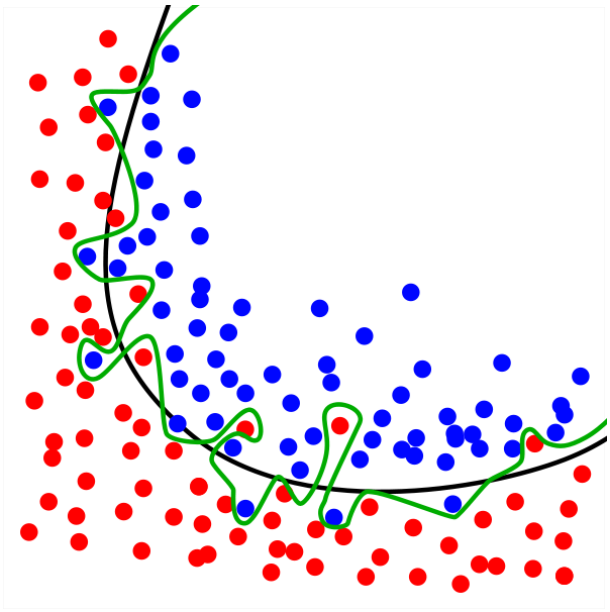
However, that in itself does not tell the whole story. Model selection, feature selection, and parameter tuning, to name a few, are methods that can be used to optimize models at different quantities of events and features. Also, non-linear algorithms need more data since they can learn complex non-linear relationships between input and output features by definition. Suppose a linear algorithm achieves good performance with hundreds of events per class. In that case, one may need thousands of events per class for a non-linear algorithm, like 'random forest' or an artificial neural network. Correlation between the features also plays a significant role in determining how much data is needed. Let say one has a classification problem where the goal is to classify if an object is human or not, and one of the features being looked at is the number of fingers the object has. Having ten fingers and being human have a high correlation meaning the AI would need fewer events to make the prediction.

oscand18@student.hh.se
timand18@student.hh.se

Machine learning is, in fact, a process of induction. The model can only capture what it has seen, and if the training data does not include a case, they will likely not be supported by the model.



(a) Learning curve showing training score and cross validation score [29].



(b) The green line represents an overfitted model and the black line represents a regularized model. The class of the data point is represented with the color red or blue. [28].

Figure 1: Parameter tuning

2.3 The dataset

During this project, we will use a version of the dataset named Heart Disease UCI with additional data points, which is free and accessible for anyone on the website Kaggle[1]. It contains 1025 rows of unique patients and 14 columns. The columns also called features, are: age, sex, cp, restbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal and target. Since the columns are not entirely self-explanatory following is table [1] with each column and brief clarification.

When picking this dataset, we wanted to be working with data from a disease with similarities to intermittent claudication, that being another type of vascular disease. Albeit, focused on the heart instead of the legs, thus leading us to 'heart.csv'.

[1] https://www.kaggle.com/zhaoyingzhu/heartcsv - Last accessed: 2021-05-07

oscand18@student.hh.se
timand18@student.hh.se

Column name	Explanation	Further explanation
age	The person's age in years	
sex		1 = male, 0 = female
cp	The chest pain experienced	Scale from 1 - 4
trestbps	The person's resting blood pressure	mm Hg
chol	The person's cholesterol	mg/dl
fbs	The person's fasting blood sugar	If more than 123mg/dl, 1 = true, 0 = false
restecg	Resting ECG measurement	Scale from 0 - 2
thalach	The person's maximum heart rate achieved	
exang	Exercise induced angina	1 = yes, 0 = no
oldpeak	ST induced by exercise	See footnote[2]
slope	The slope of the peak exercise ST segment	Scale from 1 - 3
ca	The number of major vessels	Scale from 0 - 3
thal	Thalassemia - A blood disorder	3 = normal, 6 = fixed defect, 7 = reversable defect
target	Heart disease	0 = no, 1 = yes

Table 1: heart.csv column clarification

2.4 Data missingness

Be it surveys, clinical studies, observations, or interviews, the risk of data missingness is constantly plaguing data collection. Data missingness or missing values occurs when there is an empty field of data inside a dataset. There can be numerous causes of the missing data, but the fact remains that almost every data collection runs high risks of having missing data in them. The cause can vary from a question being sensitive, and the person taking a survey decides not to answer the question or the question not being relevant for the person and thus is skipped. There are, at its core, four different types of data missingness:

- ***Structurally missing data*** - This is data that is missing for a logical reason. For example, in a survey about a patient's stay at a hospital. A question about the food given to the patient during the stay is only relevant if the patient was given any food. If not, then the field will be left empty [30].
- ***Missing completely at random (MCAR)*** - This occurs when data is missing randomly and without a systematic difference between both unobservable- and observable variables and parameters of interest. As an example, a patient typically has their vitals taken automatically by a machine every hour. If the machine, for whatever reason, breaks or misses to record vitals, we end up with data that is missing completely at random[31].

[2]More information about ST - https://litfl.com/st-segment-ecg-library/ - Last accessed: 2021-05-07

oscand18@student.hh.se
timand18@student.hh.se

- ***Missing at random (MAR)*** - When data is classified as missing at random, it is assumed that the missing values can be predicted using other data items. It has a somewhat misleading name. To make it clear, let us say a female patient in her mid-twenties has not had her blood pressure taken. Comparing the patient to other patients in a similar age range, a prediction can be made about where the blood pressure levels should be [32].[3]

- ***Missing not at random (nonignorable)*** - Lastly, when data is neither MCAR nor MAR, it is missing, not at random. This means that there is a relationship between a value's likelihood of being missing and its values. In a survey about drug use, a patient may leave areas blank out of fear of prosecution if they use an illegal drug. Fields are not blank out of randomness but left empty on purpose [32].

So why do we care if we are missing data? Because depending on the amount of missing data, it may cause several problems in any prediction or analysis problem. The lack of data decreases predictive strength and will lead to a bias in parameter estimation. It can minimize the sample's representativeness, and it may make the study's research more difficult. Any of these inaccuracies could jeopardize the trials' validity [11]. If the goal is to make predictions that are as accurate as possible, one would not want the algorithms to be affected by any bias.

There are several ways of handling missing data. Following are some common ways of doing so:

- ***Ignore the missing values*** - If the missing data is not MAR nor MNAR and under 10% for an observation, it can generally be ignored. The number of cases without missing data must also be sufficient for the chosen algorithm when incomplete cases are not considered.

- ***Drop the missing values*** - While it is possible to drop the missing value simply, it is not something that is recommended. Missing data can be indications of patterns that could prove helpful.

- ***Dropping a feature*** - If possible, dropping data should be avoided whenever possible unless the number of missing data values in a feature is very high or if it is considered insignificant. If the missing data is more than 5%, then it could be left out. However, if the missing data is for a target feature, it is advised to delete dependent features. This is to avoid increasing artificial relationships between independent features[4].

- ***Case Deletion*** - If values for one or more features are missing, the whole row is dropped. An issue with this approach is that if the sample size is too small, too much data may be lost when deleting, causing bias in the dataset since data may not always be missing at random.

- ***Regression Methods*** - Theoretically, this method gives reasonable estimations of what values should be for a missing feature, but there are several disadvantages to letting an algorithm add the missing values. Since it adds values predicted using other values in the dataset, it tends to cause deflation in the standard errors. This is because the new values fit "too well."

- ***Imputation*** - Wherever possible, imputations should be considered rather than dropping data since it preserves more data. However, replacing missing data with estimated values taken from available data in the dataset may reduce variance and introduce a large amount of bias.

- ***K-Nearest Neighbour Imputation (KNN)*** - Using KNN techniques, the missing values are added relative to some distance to other values, and then the average is used as an imputation estimate.

[3] Because of the confusing terminology MCAR and MAR can easily be mixed up.

[4] https://www.kdnuggets.com/2020/06/missing-values-dataset.html - last access: 2021-05-07

oscand18@student.hh.se
timand18@student.hh.se

KNN can be used for both discrete and continuous attributes. However, the issue with KNN is that it works well with a low amount of features but becomes ineffective when there is a more considerable amount. This is because the more features there are in a dataset, the less influence each feature has on the resulting distance, which is also know as the curse of dimensionality [33].

- ***Imputation by Mean/Mode/Median*** - Missing numerical data can be imputed using the mean of the other values in the row. If there seem to be many outliers in the row, the median can be used instead. As for categorical values, the mode of the row can be chosen as the imputed value. The disadvantage of using this imputation is that the new values are just estimates and not related to other values in the dataset. Thus, there is a reduced correlation between the new values and the values used to impute them.
- ***Multiple Imputation*** - This is an iterative method where missing data is estimated using observed data. Involves three steps [34]:

  1. Imputation of missing values from an Imputation Model.
  2. Fitting of an Analysis Model to each of the imputed dataset separately.
  3. Pooling of the sets of estimates.

  The advantage of using Multiple Imputation is that one achieves an unbiased estimate of the missing data while also preserving the sample size. However, it requires the user to model a distribution of every variable with missing values regarding observed data. The results then depend on the model being done appropriately.

There is no single best or correct way of handling missing data since there are drawbacks to each method. However, some methods are more popular than others. According to an article from BioMed Central that looked into randomized controlled trials (RCT). The most common methods for handling missing data were as follows: case analysis (45%), simple imputation (27%), model-based methods (19%), and multiple imputation (8%). The conclusions made in the mentioned article are that missing data continues to be a problem in RCTs. There is a large gap between the research related to data that is missing and the use of methods in applied setting in top medical journals [35].

Preferable ways of handling missing data are to plan out whatever data collection method as well as possible in advance to avoid the issue altogether, hopefully. However, as stated in the first paragraph of this section, data missingness is almost always plaguing data collection. Hyun Kang suggests a seven-step plan for minimizing the amount of data missingness in clinical research in 2013 [11]. He proposed the following:

1. Limit the study and who is participating in it.
2. Produce detailed documentation of the study before beginning the clinical research.
3. Instruct all participating personnel in all aspects of the study.
4. Perform a minor in-scale pilot study before the main trial.
5. Decide on what levels of missing data is acceptable.
6. Pursue and engage the participants who may be at risk of being lost during follow-up.
7. If a patient decides to withdraw from the study, record the reason for it, to then be interpreted in the results[5].

[5] A deeper explanation of each proposed step can be seen here: [11]

oscand18@student.hh.se
timand18@student.hh.se

Note that the above is a recommendation for mitigating missing data in a study where it is possible to prepare and collect specific data points. Often, researchers may want to try and predict something using retrospective data, which means data that has already been collected without a specific study in mind. When working with retrospective data, missingness can be more prevalent, and thus one may need to find ways of solving the missingness problem more actively.

2.5 Model

A model in supervised learning usually refers to the mathematical structure used to make the prediction y_i from the input x_i. In linear models, the prediction is given as:

$$y_i = \theta_0 + \sum_{j=1}^{m} \theta_j x_{ij} \tag{1}$$

Where x_{ij} for j = 1, ... , m, is the value of the j-th explanatory variable for data point i and θ_0, ..., θ_m are the coefficients indicating the relative effect of a particular explanatory variable on the outcome [36].

To put it simply, a linear combination of weighted input features. The parameters or weights are undetermined, and the task of training the model involves finding parameters θ that best fit the input x_i and output y_i.

We train the model by defining the objective function to measure how well the model fits the training data. The objective function consists of two parts, training loss $L(\theta)$ and regularization $\Omega(\theta)$:

$$obj(\theta) = L(\theta) + \Omega(\theta) \tag{2}$$

Regularization is a term added to the objective function to control fluctuation and prevent overfitting.

The training loss is a metric that indicates how well our model predicts the training data. A common choice and something we use is the mean squared error which is given as:

$$\sum_i (y_i - \hat{y}_i)^2 \tag{3}$$

Where y_i is the observed value and $\hat{y}_i$ is the predicted value.

Therefore, a model in machine learning is the output of a machine learning algorithm run on data. A model represents what was learned by a machine learning algorithm.

oscand18@student.hh.se
timand18@student.hh.se

2.6 Algorithms

In this section, we will briefly present the machine learning algorithms we will use to do classifications. Out of the nine different algorithms, three of them are linear algorithms, and six are non-linear. The linear algorithms are linear regression, linear SVC, and elastic net. The non-linear algorithms are k-nearest neighbor, Naive Bayes, Decision tree, Random forest, Gradient boosting trees, and multilayer perceptron.

2.6.1 K-Nearest Neighbor

The k-nearest neighbor algorithm is a non-parametric classification method, and there are no assumptions about the distribution of the underlying data. Meaning, the structure of the model is determined from the dataset.

The k-nearest neighbor algorithm is very versatile because it can be used for both classification and regression predictions. It relies on calculating the approximated distance between the categorized data point and other data points near it. 'K' is the number of neighbors used to define the category for the undetermined data point.

To determine the distance to a data points nearest neighbor, the euclidean distance formula is used, in our case since we make use of the scikit-learn [37] library for our algorithms:

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2} \tag{4}$$

Where p and q are two points on the real line, and the distance between them is $|p-q|$. Given multiple dimensions simply increase n in the formula giving us multiple p_i and q_i where i increases with n.

Using the k-nearest neighbor algorithm, normalization of the data used may be required if features differ significantly in values or scales. The advantage of using k-nearest neighbor is that it has a simple implementation and, as mentioned before, makes no prior assumptions of the data. However, the prediction time can be relatively high as it needs to measure the distance between every data point for its predictions.

2.6.2 Naive Bayes

Naive Bayes classifiers are a collection of algorithms based on the Bayes' theorem[6], which describes the probability of an event occurring given the probability of another event that has already occurred.

$$P(A \mid B)=\frac{P(B \mid A)P(A)}{P(B)} \tag{5}$$

Which is the same as:

$$posterior=\frac{prior\cdot likelihood}{evidence} \tag{6}$$

With regards to our dataset:

$$P(y \mid X)=\frac{P(X \mid y)P(y)}{P(X)} \tag{7}$$

Where y is a class variable (positive or negative class) and X is an independent feature vector.

The algorithm is naive because it makes the naive assumption that every pair of features is independent of each other and makes an equal contribution to the outcome. Meaning there is no correlation between

[6] https://machinelearningmastery.com/bayes-theorem-for-machine-learning/ - Last accessed: 2021-05-07

oscand18@student.hh.se
timand18@student.hh.se

the features. Despite Naive Bayes having a simple design, it has been proven to work well in complex real-world situations. [38].

We use Gaussian Naive Bayes, which is used when working with continuous data, and data values associated with each feature are assumed to be distributed according to a Gaussian distribution, also called Normal distribution [39]. The likelihood of the features is therefore given by:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{8}$$

Where x_i is feature number i, σ_y^2 is the variance and μ_y is the mean of the target value.

2.6.3 Decision Tree

Decision tree is an algorithm designed as a tree containing a root, nodes, and leaves (the last node in the tree). By measuring impurity, it is determined which feature is the root and if a node should continue to branch the tree or not. There are different ways of measuring impurity. We use Gini, which for binary values are:

$$G(k) = \sum_{i=1}^{J} P(i)(1 - P(i)) \tag{9}$$

Where $P(i)$ is the probability of a certain classification i per the training dataset.

By design, the tree will recursively split at each node until each leaf is left with a single outcome reaching the globally optimal solution, resulting in overfitting. To avoid overfitting, several different pruning approaches give a higher accuracy on the test data but at the expense of the training accuracy and complexity of the model. Out of this problem, Random forest was born.

2.6.4 Random Forest

Random forest is a supervised learning algorithm. Meaning it will learn the relation between training examples and their associated target variables, then apply that learned relationship to classify entirely new inputs without targets. The core of random forest is that it builds multiple decision trees and then combines the predictions of the trees to achieve a more accurate prediction.

While increasing the trees, random forest adds more randomness to the model. When splitting a node, it looks for the best feature among a random subset of features rather than the most appropriate one. As a consequence, there is a lot of variety, which leads to a better model.

As a result, in Random forest, the algorithm for splitting a node only considers a random subset of the features.

Random forest is an easy-to-use algorithm because the default hyperparameters[7] it uses often produces a good prediction result and are few and easy to understand. A big problem in machine learning is overfitting which random forest prevent mainly with the random forest classifier. If there are enough trees in the forest, the classifier will not overfit the model. It, however, comes with limitations which are that it is slow and therefore unsuitable for real-time predictions.

[7] https://deepai.org/machine-learning-glossary-and-terms/hyperparameter

oscand18@student.hh.se
timand18@student.hh.se

2.6.5 Gradient Boosting Trees

Gradient boosting as a concept was born out of the idea of converting weak learners into strong learners. A weak learner is an algorithm with a performance slightly higher than random chance (50% in classification problems). The idea behind boosting is to break down the problem, sequentially adding more weak learners to handle complex patterns, creating a strong learner.

The loss function used could be, for example, the mean squared error explained earlier. Although the gradient boosting framework has the advantage of not requiring creating a new boosting algorithm for each loss function that may be used; instead, it is a general enough framework that any differentiable loss function may be used.

The weak learners used are decision trees whose output can be added together to allow sequential models to correct and improve the difference between observed and predicted data. The trees are constructed greedily, choosing the best split points, but are constrained in ways such as a maximum number of layers, nodes, splits, or leaf nodes. These constraints ensure that the learner remains weak but can still be greedy.

As weak learners are added to the model, one at a time, gradient descent [8] is used to minimize the loss. Instead of parameters, gradient boosting models use decision trees. After calculating the loss function, we add trees to the model that reduces it.

2.6.6 Logistic Regression

Logistic regression has many similarities with linear regression, the main difference being what they are used for. Linear regression is used for regression problems, while logistic regression is used for classification problems. Logistic regression is named after its core function; the logistic function also called the sigmoid function. It is an S-shaped curve that can map any real-valued number to a value between 0 and 1.

$$\frac{1}{(1 + e^{-value})} \tag{10}$$

The curve from the sigmoid function indicates the likelihood of a target value and is fitted using maximum likelihood. Using the concept of a threshold value, the target value can be assigned. If the threshold value is 0.4, any possibility below is considered 0, and everything equal or above equals 1.

Models can be simple, fitting only a few features or more complicated. In more complicated models, the usefulness of features is calculated using Wald's Test [40].

2.6.7 Linear SVC

Linear SVC performs well with a large number of samples and is most commonly used for solving classification problem. Its object is to fit data provided by a dataset and then return a "best fit" hyperplane that is used to categorizes the data. After having a hyperplane splitting the categories, features can be fed to the classifier to see where the data point would be classified. Since the hyperplane is drawn between two classes, there will always be a closest sample from each class to the hyperplane, this is the support vector. The distance of the closest samples to the hyperplane describes how well the categories are separated, and is the margin. The goal of linear SVC is to maximize the width of this margin.

[8] https://machinelearningmastery.com/gradient-descent-for-machine-learning/ - Last access: 2021-05-07

oscand18@student.hh.se
timand18@student.hh.se

2.6.8 Elastic Net

Elastic net regression is a linear regression model, meaning it assumes a linear relationship between the target variable and the input variables. The elastic net algorithm is a combination of ridge regression (equation 11) [41] and least absolute shrinkage and selection operation (LASSO) regression (equation 12) [42]. Elastic net (equation 13) came to fruition due to the critique on the LASSO regression, which relied too heavily on data for variable selection. To mitigate this, it was combined with ridge regression to achieve the best of both worlds.

$$L2_{penalty} = \sum_{i=0}^{p} \beta_j^2 \tag{11}$$

$$L1_{penalty} = \sum_{i=0}^{p} |\beta_j| \tag{12}$$

$$ElasticNet_{penalty} = (\alpha \cdot L1_{penalty}) + ((1-\alpha) \cdot L2_{penalty}) \tag{13}$$

Ridge regression is known as L2 penalty, which means it is the sum of squared coefficient values while LASSO as an L1 penalty is summing absolute coefficient values. For the elastic net equation, we can then add a hyperparameter α to assign how much weight is given to either the L1 or L2 penalty. The weight, α, can be a value between 0 and 1 [43].

2.6.9 Multilayer Perceptron

Multilayer perceptron or MLP is a deep learning artificial neural network composed of more than one perceptron. It comprises an input layer that receives a signal, an output layer that makes a judgment or prediction based on the input, and an arbitrary number of hidden layers that serve as the MLP's computational engine.

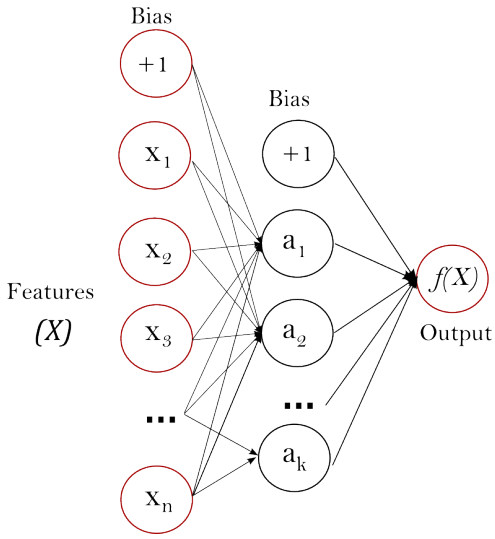


Figure 2: One hidden layer MLP [44].

In Figure 2 the layers mentioned above can be seen from left to right (input, hidden, output). x_n represents the input features that enter the hidden layers which transforms the values with a weighted linear summation

$$w_1x_1 + w_2x_2 + \cdots + w_mx_m \tag{14}$$

which is followed by a non-linear activation function $g(\cdot) : R \rightarrow R$ - like the hyperbolic tan function [45] which is analogues of trigonometric functions but are defined using a hyperbola instead of a circle.

As a part of the deep learning family of algorithms, MLP can be used on and learn from non-linear models. However, the algorithm is sensitive to feature scaling and may require tuning of hyperparameters such as the number of hidden neurons, layers, and iterations.

oscand18@student.hh.se
timand18@student.hh.se

3 Method

In this section, we give a detailed explanation of the steps taken to reach our results. We followed a well-known structure for working with machine learning as a base to produce our baseline prediction. We then moved on to data corruption in the form of feature removal, row removal, noise corruption, and value removal. After the corruption steps were complete, we moved on to mitigating missingness in the value removal section by imputing the missing values differently. Lastly, we run our various algorithms on the different files affected by different types of corruption to see what results we can obtain.

3.1 Baseline

While working on our baseline model we followed the steps below:

- Visualize the data
  - The first thing we do is explore our dataset to gain a better understanding of what we are working with. Often it is easier to understand the data and draw conclusions when looking at graphics compared to numerical values. We make plots and graphs to see if we can find patterns between the features in our dataset. We are looking at patients, ages, genders, amount of patients with a heart disease versus no heart disease, correlations between having heart disease, and other features (resting blood pressure, fasting blood sugar).
- Define prediction task
  - When the visualization is complete, and we have a better understanding of what correlations there are in the dataset, we can decide what we would like to predict and what we can predict with the data. For example, a patient's age, how much a specific house is worth, or what a particular stock will look like at the end of the day. In our case, we are looking to classify if a patient has heart disease or not depending on specific parameters taken during patient checkups.
- Gather features or create features
  - With an idea of what we aim to predict, we need to go through the data again and find features and parameters that relate to our defined prediction task. For example, if it is possible to predict a patient's age from their resting blood pressure, cholesterol, and fasting blood sugar. In the dataset we are working on, we did not need to go out of our way to find features since it is all condensed into one .csv file. In other cases, we may have to look through multiple data files and create a new file with everything we are interested in.
- Preprocessing
  - After deciding on a prediction task and its features, we move on to working with the data. For the algorithms to work best, the data may need to be cleaned up. Null values should be normalized by filling them with the mean, median, or mode of the dataset, or the data point can be completely removed if the dataset has a large amount of data and would not be affected by the loss of data. If we are working with categories, they need to be turned into dummy values ([north, west, south, east]→[0,1,2,3]) for the algorithm to be able to work with them. This part of the process can be a very time-consuming step, depending on what data one has access to. Again, we can avoid parts of this step with the dataset we are working with since it is already in a processed state. For example, it is already cleared of null values.

oscand18@student.hh.se
timand18@student.hh.se

- Generate training/validation/test
  - When the data is clean, we use the python library(sklearn [37]) 'train_test_split' to split the data into an 80/20 training and testing set. Meaning that 80% of our data will be used to train the algorithms while the other 20% will be used to test our predictions. When the data is split is complete, data may need to be scaled if values are of different types(binary, categorical, numerical) and have a large spread (for example, 120, 542.6, 2). Doing this prevents the larger numbers from having more impact on the predictions. The scaler we make use of is the 'MinMaxScaler' from the sklearn library, where the transformation is given by:

$$\mathbf{X}std = \frac{X - min(X)}{max(X) - min(X)} \tag{15}$$

    Where X refers to a two-dimensional array of data points we wish to normalize, and min and max equal the feature range (between 0 and 1).

- Train models and validate
  - Using Python libraries, we can now use our training sets with different algorithms to find our case's best fit. The step before and after this is somewhat automated, thanks to the sklearn library (for example, train_test_split, MinMaxScaler, VarianceThreshold). We run our training data through different machine learning algorithms: Random Forest, K-Nearest Neighbor, Naive Bayes, Multilayer Perception (MLP), Gradient Boosting Trees, and Decision Tree, Elastic Net, Linear SVC, Logistic Regression. This gives us a spread of different types of algorithms, both linear and non-linear.

- Check final generalization performance
  - Lastly, we use our test set on the optimized model and look at our results(Seen in 'Results so far'). As stated in "Generate training/validation/test" we are currently splitting our data into two camps: one for training and one for testing. However, when moving on to distorting our data we aim to do another split of 70/15/15 where the last 15% of the data is kept separate from the training and testing sets to then later be used to validate our final tuned models with an unbiased estimator of the skill of the algorithms.

- Loop if needed
  - When the model is tested and the results have been presented, we may either be happy with the results or we loop back to the start of the process and do it again while looking at new features. We repeat the steps until the results are satisfactory and then move on to visualizing the results so that they can be presented in a manner that is easier to understand. The last 15% of our data mentioned above will be used at this stage when we are happy with our results to do testing.

This will be the general workflow regardless of what dataset one is working with. Some points may be less relevant in some cases, for example, when the dataset is already pre-processed. However, the structure will still remain the same.

oscand18@student.hh.se
timand18@student.hh.se

3.2 Data Corruption

At this stage, we know what the dataset looks like and have made classification models for the different algorithms in use so we can move on to data corruption. Our methods for performing data corruption were split into five parts. We add patients to the dataset to mimic noise; we find what features have the most significant effect on classifications and removed them. We randomly pick out values from the more important features and then remove cases containing missing values and impute missing values using four imputation methods mentioned in Section 2.4

3.2.1 Noise Corruption

The initial type of corruption we wanted to implement was noise corruption, as we wanted to test how different levels of noise would affect our algorithms. This lead us to create dummy patients that could potentially mimic excising patients from the dataset. They were given the same parameters as any other patients; however, the ranges of values added to the columns were randomly created. We attempted to keep the values within a "realistic" range, meaning that the patients in our dataset were between the ages of 29 - 77 years old. So when a random patient was created, they were giving an age within that range. The same was done for each column.

We then made a loop that would append a new patient to the original 'heart.csv' file. Nine different files were created to add an increasing amount of noise, starting with 10% moving up to 70% added noise. We were interested to see how the dummy patients would affect the algorithms when they were nearing an absolute majority, so we also created a file where there were as many dummy patients as real patients and lastly, one where dummy patients were 10 to 1.

3.2.2 Feature Removal

As for feature removal, we wanted to remove a feature completely and attempt to see its effect on the overall predictions. Since we are working with a dataset with 14 features where one of them is our target variable, we were left with 13 features we could potentially remove. However, not all of the features affect our target variable enough to be worthy of removing (we saw this during the data visualization step where we produced a heatmap to see how the different features correlated to our target variable).

To find out what features could potentially affect the predictions, we ran ten types of feature selections on the dataset. The ten types of feature selection being: logistic regression, decision tree, gradient boosting trees, random forest, permutation KNN, low variance, variance threshold, univariate feature selection, recursive feature selection, sequential feature selection using KNN (forward and backward feature selection).

All of the different types of feature selections were run on the base .csv file and on the previously created noise files to see if the noise had any effect on the importance of feature selection.

3.2.3 Value Removal

After we decided what features had a more significant effect on the target variable, we wanted to see how the removal of values within those columns would affect the predictions. In juxtaposition to how we added noise, for the removal of values, we removed 10% at a time creating seven new .csv files with less and less data in them. This was initially done for the four columns we deemed most important from the feature selection process, meaning we had 28 .csv files with varying amounts of data removed from columns. The data removed were randomly selected each time to attempt not to cause any bias in the removal.

Finally, we chose three of the features (which were all categorical as opposed to numerical) and removed values simultaneously from all three to see if there were a more significant effect on predictions when data were missing from multiple columns at the same time.

oscand18@student.hh.se
timand18@student.hh.se

3.2.4 Case Removal

As mentioned in the data missingness chapter, one way of handling data missingness is to remove a row with missing values. It is unadvised to do so if the dataset is on the smaller side, meaning a reasonably large percent is removed. We wanted to see how much the removal affected the predictions.

Using a random seed, we selected rows and started removing them by a certain percentage at a time, starting at 10% up to 90%. This step was also done in two different ways: a completely random approach and a pseudo-random approach.

The completely random approach means that we take our base dataset, delete a percentage of data and run a test. For the next test with an increased percentage removed, we once again take the base dataset and remove a percentage of the data. The problem with this strategy is that the randomness is increased since 10% removed data could potentially have a bigger impact than 50% removed data if essential data is deleted in the 10% sample but not in the 50% one.

The pseudo-random approach means that we remove an increasing amount of data from the same dataset as a strategy to reduce variance; this meant that we removed 102 data points each iteration. The strategies have less and less of an effect on the result with increased trials. We ran the algorithms 20 times.

3.2.5 Missing value imputation

The last step aims to combat the created missingness made from the value removal step. We make use of four different ways to impute missing values. The four different techniques are simple imputation, an iterative imputation using a random forest regressor, KNN imputation, and linear regression. The techniques were the same for the most part, but some adjustments had to be made for one of the columns since it contained numerical values instead of categorical.

Firstly, we use a 'simple imputer' where we specify what values were to be imputed (null values in our case) and what strategy to use. We used three different strategies for the categorical values: 'mean,' 'median,' and 'most_frequent.' For the numerical column, we did not use the 'most_frequent' strategy.

Then we move on to a 'KNN imputer,' which is a bit more involved when predicting categories. For our numerical feature, we just specify the number of neighbors we are calculating the distance to and then fit the dataset to the imputation algorithm. For the categorical columns, we had to find the category mappings and encode integers before calling the KNN-imputer class on the dataset.

The last imputer we used was a 'linear regression imputer.' Here we predicted values to add by using a linear regression model. However, since we cannot make classifications on empty values, we had to add a deterministic column that we filled with values. The random values in the deterministic column were based on other values in the same column. By using the imputed column as our y and the remaining as X we can fit the regression model and predict the missing values.

3.3 Evaluation

We analyze our results using classification accuracy and confusion matrices. Confusion matrices are used to get a more accurate idea of a models performance. It displays the result in a 2x2 matrix with the cells representing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). A false negative would be to diagnose a patient as healthy when it is, in fact, sick.

Accuracy as a metric is calculated by taking the models correct classifications and dividing them with the total. Which is the metric we will use during testing of data corruption as we will plot the accuracy of the algorithms at different stages of missingness to see the performance changes. These performance changes will be interesting to look at when testing our hypotheses of feature importance and imputator techniques.

oscand18@student.hh.se
timand18@student.hh.se

4 Results

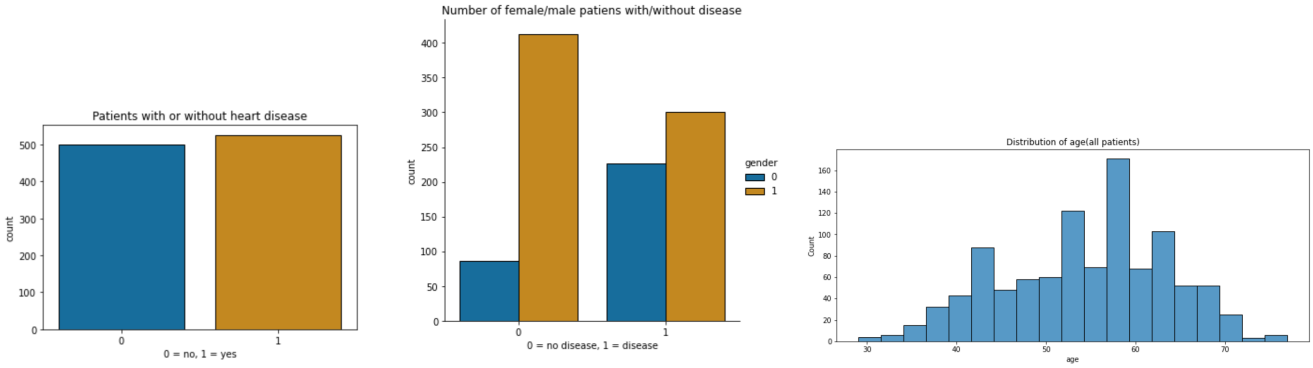
In this chapter, we present the results of our research. It is a crucial part that gives answers to the research questions and problem statements. It is split into two sections, one where we show the results for our classification AI, using algorithms to classify sick and healthy patients. The other section shows our results with different types of data corruption such as case removal, feature removal, noise, and ways to combat missingness using imputation techniques.

4.1 Baseline

4.1.1 Data Preparation

We start by inspecting the dataset. How many data points are there? How many features? How many null values? We have 1025 data points, 14 features, including our target feature, and no null values.

Next, we figure out what demographics we are working with, such as gender split, sick and healthy, age distribution, resting blood pressure groups, and how these demographics correlate with our target value; heart disease.



(a) Heart disease (b) Male/female (c) Age

Figure 3: Demographics

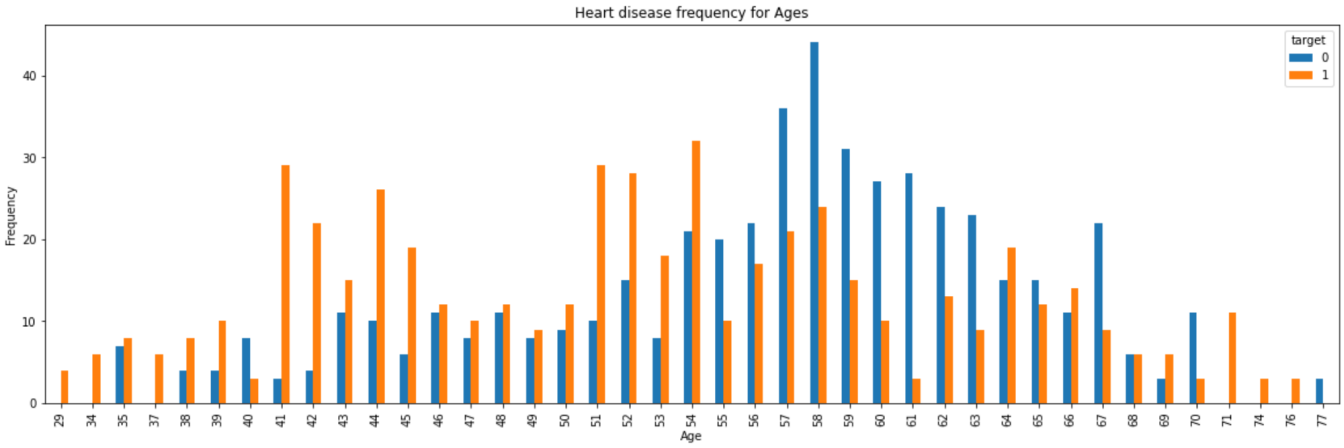
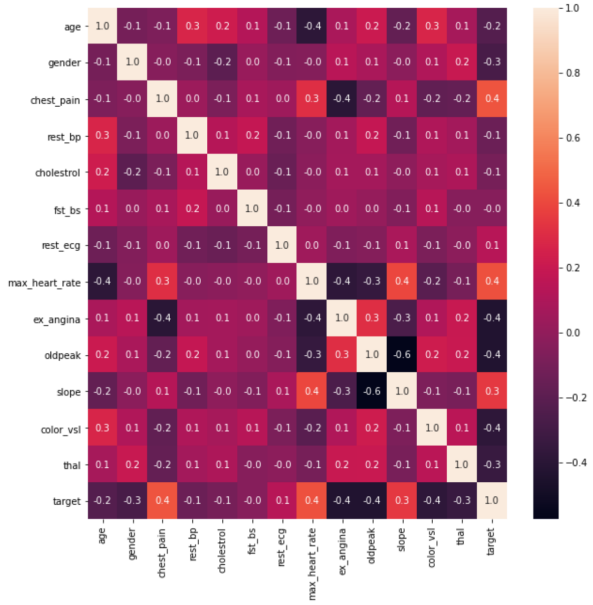


Figure 4: Heart disease frequency for ages

oscand18@student.hh.se
timand18@student.hh.se

The dataset has almost a fifty-fifty split between sick and healthy patients, seen in figure 3, which is pretty uncommon, and the case can be made that it is somewhat biased. The age distribution is almost of normal distribution leaning towards older age. We also see a significant discrepancy in the genders; a large majority of females are sick; are they more prone to sickness? Although we did find that there is a correlation between gender and sickness, it is pretty low. We also see spikes of disease in patients 40-45 years old and 51-54 in figure 4. The reason for a high representation of sickness in younger groups might be because they die as they become older. Moreover, the reason for the slight increase at later ages is that the patient gets old and weak, therefore being more prone to sickness.

As we can see in figure 5 the correlations are low across the board, fst_bs (The person's fasting blood sugar), cholesterol, rest_ecg (Resting electrocardiographic measurement) are the lowest. However, it does not tell the whole story, it might not correlate highly with our target variable, but it might help our model in combination with other features.



(a) Heatmap

fst_bs	0.041164
cholestrol	0.099966
rest_ecg	0.134468
rest_bp	0.138772
age	0.229324
gender	0.279501
thal	0.337838
slope	0.345512
color_vsl	0.382085
max_heart_rate	0.422895
chest_pain	0.434854
ex_angina	0.438029
oldpeak	0.438441
target	1.000000

Name: target, dtype: float64

(b) Numerical values, Correlation coefficient used is Spearman

Figure 5: Correlation between features

4.1.2 Feature Selection

We use as many different types of feature selection methods to get an idea of how they differ. One method is feature importance which is a built-in function in sklearn.

As seen in figure 10 tree-based algorithms place high importance on chest pain, thal (the person's maximum heart rate achieved), oldpeak (ST depression induced by exercise relative to rest) and color vsl (the number of major vessels). The linear model, logistic regression puts a significant weight on gender, chest pain, ex angina (exercise-induced angina) and thalach.

Next using low variance feature selection we found that the features with the lowest variance and therefore the most constant and non-interesting are cholesterol, resting blood sugar, fasting blood sugar and rest ecg.

oscand18@student.hh.se
timand18@student.hh.se

Univariate Selection can also be used in combination with the SelectKBest class to score features depending on their relationship with the target variable. It retains the K highest scoring features, using a scoring function, our scoring function is chi2 and our k is 10. We have also done the test with increased noise to see the differences in feature selection, presented in figure 11.

Recursive feature selection is another method that can be used with the algorithm of your choice showing optimal number of features, we are using random forest in this graph 6 and the metric is accuracy.

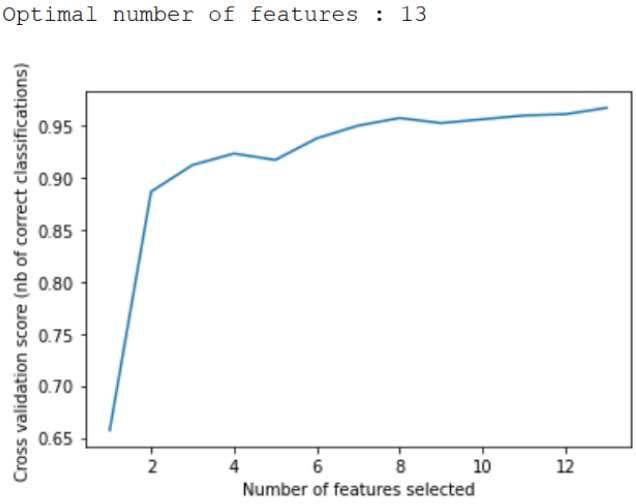


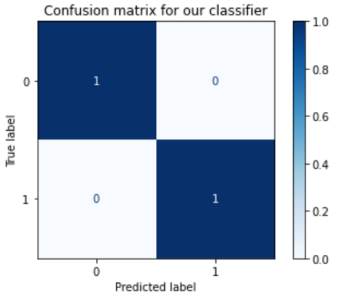
Figure 6: Feature Selection - Recursive

We can see that this dataset is very healthy, and Random Forest can achieve very high accuracy with only 3-4 features; this also leads us to believe that there are definite tells the model can look at when determining the patient's state.

oscand18@student.hh.se
timand18@student.hh.se

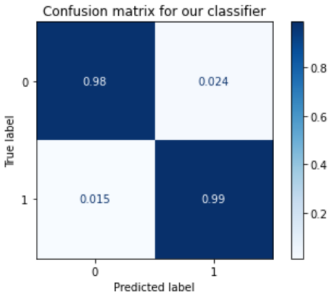
4.1.3 Prediction

After the tuning of hyperparameters and feature selection, we achieved these classification scores for the different algorithms. Important to note, this is not a ranking of algorithms since that would be unfair; not all algorithms were equally tuned.



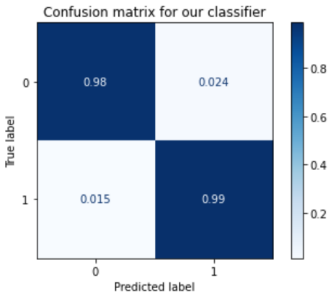
MSE train data: 0.000, MSE test data: 0.000
R2 train data: 1.000, R2 test data: 1.000
Accuracy: 1.0

(a) Decision Tree



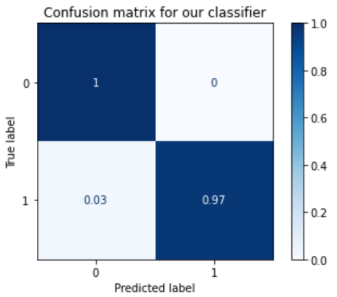
MSE train data: 0.180, MSE test data: 0.140
R2 train data: 0.281, R2 test data: 0.439
Accuracy: 0.8599221789883269

(b) Elastic Net



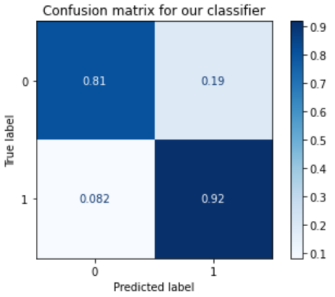
MSE train data: 0.016, MSE test data: 0.019
R2 train data: 0.937, R2 test data: 0.922
Accuracy: 0.980544747081712

(c) Gradient Boostring Trees



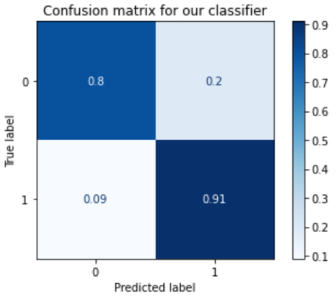
MSE train data: 0.003, MSE test data: 0.016
R2 train data: 0.990, R2 test data: 0.938
Accuracy: 0.9844357976653697

(d) KNN



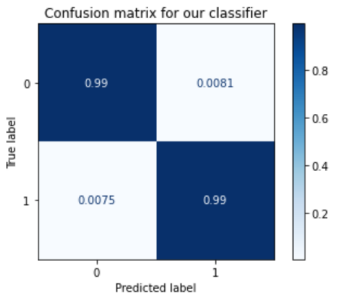
MSE train data: 0.177, MSE test data: 0.132
R2 train data: 0.291, R2 test data: 0.470
Accuracy: 0.867704280155642

(e) SVC



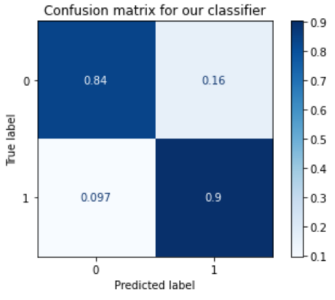
MSE train data: 0.181, MSE test data: 0.140
R2 train data: 0.276, R2 test data: 0.439
Accuracy: 0.8599221789883269

(f) Logistic Regression



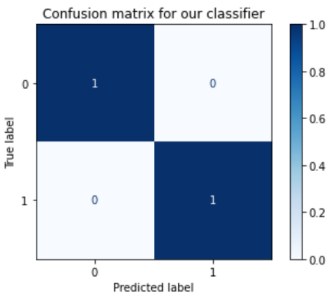
MSE train data: 0.007, MSE test data: 0.008
R2 train data: 0.974, R2 test data: 0.969
Accuracy: 0.9922178988326849

(g) MLP



MSE train data: 0.164, MSE test data: 0.128
R2 train data: 0.343, R2 test data: 0.485
Accuracy: 0.8715953307392996

(h) Naive Bayes



MSE train data: 0.000, MSE test data: 0.000
R2 train data: 1.000, R2 test data: 1.000
Accuracy: 1.0

(i) Random forest

Figure 7: Predictions

The trees do exceptionally well, and so do MLP and KNN. We can also see that this dataset is too complex for a linear model like elastic net, SVC, and the naive correlations assumed by Naive Bayes.

oscand18@student.hh.se
timand18@student.hh.se

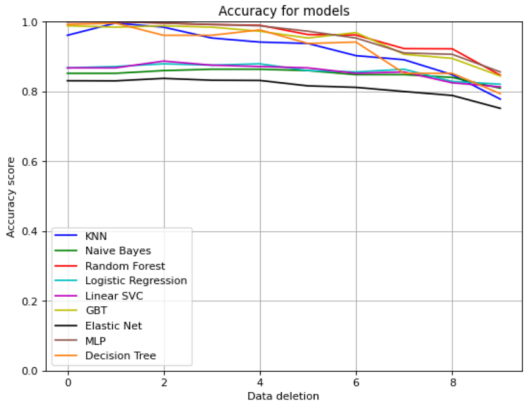
4.2 Data Corruption

4.2.1 Imputator

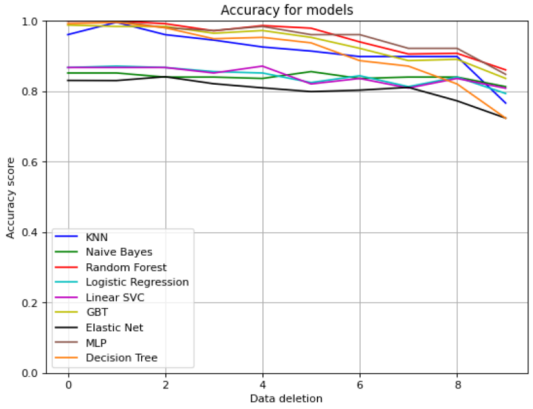
Our approach for imputators is to delete data in three of our most important features, old peak, chest pain and thal, in increments of 10% to 70%. The plots (12)(13)(14)(15)(16)(17) clearly show that imputators have a positive effect on the accuracy of the algorithm with the exception being logistic regression which actually see an accuracy decrease when imputing values.

4.2.2 Case Removal

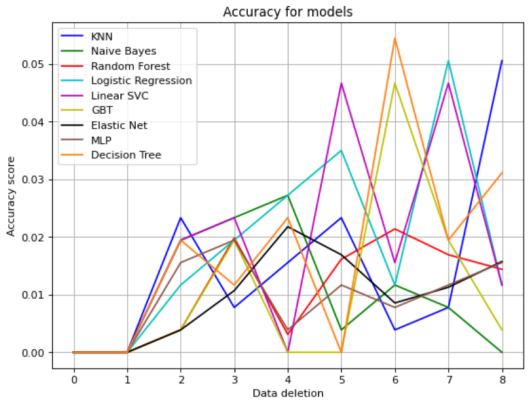
We can once again see that this dataset is healthy. Our best classification algorithms, MLP, and the tree structures do not see a noticeable drop until around half the dataset has been removed. The least affected algorithms overall are the linear ones and Naive Bayes, with the exception of Elastic net, which saw a significant drop in accuracy.



(a) Accuracy for random



(b) Acurracy for pseudo-random



(c) Difference between random and pseudo-random in terms of accuracy

Figure 8: Case removal

oscand18@student.hh.se
timand18@student.hh.se

4.2.3 Feature Removal

We started our test with feature removal by removing the three highest-scoring features, oldpeak, thal and chest_pain and saw little no effect on our algorithms, as can be seen in fig (18).

We then gradually started to remove more and more features starting with max_heart_rate and color_vsl seen here (19). We start to see a dip in accuracy from algorithms that have done well in the past, hinting towards a reliance on this combination of features to classify.

Next, we removed the age feature, seen in figure (20). We now see an increase in performance in some algorithms showing that less is sometimes better when it comes to feature selection, which we are familiar with.

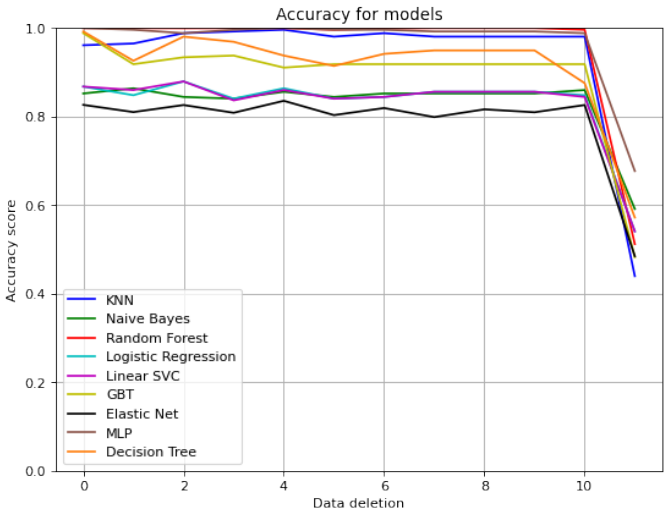
After that, we removed cholesterol and plotted the result seen here (21). We can see a significant decrease in accuracy from the tree structures and MLP. The linear algorithms and Naive Bayes have seen a minimal difference throughout these tests proving that they have not utilized these features entirely.

Lastly, we removed rest_bp where we can see a substantial drop in accuracy across all the algorithms rendering them unsuitable for healthcare use (22). We are now left with the features ex_angina, slope, gender, rest_ecg, and fst_bs, which have been some of our lowest scoring features.

To get the complete picture, we also ran tests using only our four highest-scoring features, oldpeak, thal, chest_pain, and max_heart_rate, to see how well the algorithms would do. We can in these plots (23) see that even with only these features, the tree structures, Random Forest and Decision Tree is able to achieve an accuracy of 98%.

4.2.4 Noise

Noise has little effect on the algorithms; it is not until we add a substantial amount of noise (1000%), meaning there are ten dummy patients to every actual patient, that we see a decrease in accuracy as seen in figure 24a.



(a) Accuracy for algorithms with added noise

Figure 9: Noise

oscand18@student.hh.se
timand18@student.hh.se

5 Analysis

We started the project with the idea of using actual patient data and got an insight into how hard actual patient data is to come by. Even with personal information such as name and social security number being removed, data is heavily regulated due to integrity requirements. Although making our work and the job of establishing datasets harder it serves an important role in protecting patients involved in research from harm and preserving their rights.

Ultimately we decided to use a source of data publicly accessible meant for machine learning. Because it was made with machine learning in mind, it is very robust with little to no signs of missingness, making it a viable dataset to break down and perform tests on.

Following are the different corruption methods tested in this work and our analysis of the implementation of the tests, the strengths and weaknesses.

- **Noise** - Having done tests for as much noise as having as many fake patients as real, we can conclude that when noise is done in a random fashion it has little effect on the accuracy of the model. But what noise does it that it increases the complexity of the dataset resulting in high run-time.

  To improve the noise section, we would have liked to add more realistic patients, where we took an existing patient from the dataset and tweaked one or two rows to mimic data that is mislabeled data. The reason adding noise to the dataset had little effect might have been because the noise was added randomly instead of a more targeted approach. When algorithms make classification, they fit the line in a way to separate classes as well as possible without taking the outliers into account where it makes sense, not to overfit. When adding random patients, the data points will often fall as outliers having little to no effect on the decisions made by the algorithm, being rightfully detected as noise. With fake patients falling closer and closer to the fitted line, we could see that accuracy decreases as the algorithms start to overfit the model taking the fake data into account.

- **Feature Removal** - The approach of testing several different feature selection methods gives a better understanding of what features are the most important. The work shows a trend throughout the different methods and what ultimately matters is the algorithm being used. Using a pairplot of random forest seen in figure (24) looking at the features oldpeak, chest_pain, and cholesterol we can find "islands" in the data where it is very consistent whether a data point is classified as a 0 or 1. We suspect that this is partly the reason why the algorithms do so well with this dataset and why it values these features highly, as it can use these islands to make the correct classification.

  While feature removal is something that can be done in many different ways, testing numerous combinations, this work focuses on the high impact features.

- **Value Removal** - Doing two different types of removal was an implementation decision to combat accuracy variance as the removal was done randomly. Different results can be obtained at the removal thresholds by chance, removing essential data at one and reinstating that data at a later stage, and removing something else. The approaches had less impact as we scaled the tests up by introducing more trials and taking an average, but it is still something to take in mind when working with data. The approach and implementation play a significant role.

  Potential improvements to the work would have been to remove values in specific areas, such as all patients within the age group of 50-55. With no data of patients in that age group classification proves harder as the algorithms are presented with new patients part of it. Machine learning algorithms and models can only classify what has been observed.

oscand18@student.hh.se
timand18@student.hh.se

- **Imputation** - Our reason for choosing the four imputation methods we did stemmed mainly from accessibility and ease of use. Three of the imputation methods are a part of the scikit-learn [37] library (simple imputater, multiple imputater, and KNN imputation), making them easier to implement. Linear regression imputation was also used since we wanted to see if we could predict the missing values. We mention in section 2.4 that some imputations methods are more common than others, so we wanted to pick two from both sides of the spectrum (simple imputation and multiple imputation).

  Looking at the performance of the algorithms in figures (12)(13)(14)(15)(16)(17) we can see that all of the imputations are performing very well and stay in line with the original file. However, the logistic regression algorithm always performed worse with an imputed file (albeit not by much). We have not performed enough tests to draw a concrete conclusion why this is. It is hard to see much of a difference in accuracy in general, and we should have perhaps tried to remove even more data to see a more significant difference between the methods.

5.1 Answer to research questions

I. **What feature is the most important?** - The results reflect that feature importance depends on the algorithm being used. Tree structures favored chest pain, thal, oldpeak and color vsl, while the linear models favor gender, chest pain, ex angina, and thal. Feature importance changes slightly as noise is introduced to the dataset, but more tests need to be done to determine whether this is due to noise not being equally distributed across the features.

   Feature selection showed that removing unimportant features saw an increase in accuracy, which is a well-known fact in machine learning [46]. By removing unimportant features, also called dimensionality reduction, we prevent overfitting, make a simpler model that is easier to interpret, and lower computational time.

II. **How significant is the correlation of the features?** - Correlation proved less valuable than we initially thought. A correlation between features should work as a guideline when presented with a dataset to show possible predictions with the data. If there is a feature with a low correlation to the rest, it might not be a suitable target variable. Features correlated to the target variable can be viewed as hints to make a good guess; with more correlated features, there is a higher chance of making a good guess, classifying the data correctly.

III. **What algorithms should be used given the data available?** - Random Forest, Decision Tree, MLP, Gradient Boosting Trees and to an extent KNN have consistently done well in our tests. They are able to identify patterns in the data that the linear models can not capture. This also means that removal of data hits these algorithms the hardest as they lose their edge. Our research showed that removal of data points narrowed the gap between the algorithms. Algorithms that previously could not make use of the data points were less effected. Random Forest still proved to be the highest performing algorithm after removal of 90% of the data but only by a small margin.

IV. **How much noise (faulty or unhelpful captured data) is acceptable?** - This question proved to be vague and hard to answer as noise comes in two forms, attribute and target noise [47]. Attribute noise can also differ significantly depending on which feature is affected and how. Randomly added attribute and target noise showed little effect on the classification accuracy in our tests, and a lot of noise had to be introduced to the dataset before we saw an unacceptable drop in accuracy. Noise, however, introduced a lot of performance issues in terms of speed as the dataset became more and more complex. Although, when working with human lives, the speed of the predictions should not be a factor.

oscand18@student.hh.se
timand18@student.hh.se

6 Conclusion

AI could be used in more areas of the healthcare sector as it could assist in the classification of various diseases. However, there are limitations of data access caused by confidentiality, meaning it is harder to get the helpful data needed to create proper algorithms for classification. There is also an issue with how data is collected that can make machine learning harder to use in classifications of rare diseases. Unless the data is specifically collected for the project, there is an issue of having the data and enough of it, since missingness is often not considered when performing data collection over more extended periods of time.

Since we decided to try and handle missing data in various ways, we got a notion of how simple it could be to impute missing values. However, the more basic imputations did not produce ideal results for the files where a majority of cases were missing values. Taking the mean value of a feature and assigning that value to every case where a value is missing should be avoided unless it is only done for a smaller fraction of a dataset. This issue is not very prevalent in the graphs showing our results since the machine learning algorithms were able to mitigate the odd values by using the remaining features to perform classifications. From the different methods we used, our preferred methods were KNN and linear regression imputations since they diversified the missing values giving somewhat of a more "realistic" result.

As for the nine different algorithms we used during this project, we can not rule out any algorithm as worse than the other since we did not tune them during testing and wanted to see how they performed at a base level. However, with how easy it is to plug-and-play the algorithms using ready-made libraries, we would heavily encourage users to test different algorithms before deciding to stick with one for any specific dataset to see if there is a "best fit." There are guidelines for when specific algorithms could, in general, be preferable to use, but as seen from our results, they perform on a relatively even level, no matter what was done to the dataset.

oscand18@student.hh.se
timand18@student.hh.se

7 References

[1] Meiyin Wu and Li Chen. Image recognition based on deep learning. In *2015 Chinese Automation Congress (CAC)*, pages 542–546, 2015.

[2] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

[3] Martin Leo, Suneel Sharma, and K. Maddulety. Machine learning in banking risk management: A literature review. *Risks*, 7(1), 2019.

[4] Jarosław Rzeszótko and Sinh Hoa Nguyen. Machine learning for traffic prediction. *Fundamenta Informaticae*, 119:407–420, 2012. 3-4.

[5] Mike Daily, Swarup Medasani, Reinhold Behringer, and Mohan Trivedi. Self-driving cars. *Computer*, 50(12):18–23, 2017.

[6] Elsie Gyang Ross, Nigam H. Shah, Ronald L. Dalman, Kevin T. Nead, John P. Cooke, and Nicholas J. Leeper. The use of machine learning for the identification of peripheral artery disease and future mortality risk. *Journal of Vascular Surgery*, 64(5):1515–1522.e3, 2016.

[7] Daniil Pakhomov, Vittal Premachandran, Max Allan, Mahdi Azizian, and Nassir Navab. Deep residual learning for instrument segmentation in robotic surgery. In Heung-Il Suk, Mingxia Liu, Pingkun Yan, and Chunfeng Lian, editors, *Machine Learning in Medical Imaging*, pages 566–573, Cham, 2019. Springer International Publishing.

[8] Jeremy C. Weiss, Sriraam Natarajan, Peggy L. Peissig, Catherine A. McCarty, and David Page. Machine learning for personalized medicine: Predicting primary myocardial infarction from electronic health records. *AI Magazine*, 33(4):33, Dec. 2012.

[9] Eric J. Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25(1):44–56, Jan 2019.

[10] Benjamin M Marlin. Missing data problems in machine learning. 2008.

[11] H. Kang. The prevention and handling of the missing data. *Korean J Anesthesiol*, 64(5):402–406, May 2013.

[12] Carmona: Improving everyday life. https://www2.carmona.se/. Accessed: 2021-01-29.

[13] F G R Fowkes, E Housley, E H H Cawood, C C A Machintyre, C V Ruckley, and R J Prescott. Edinburgh Artery Study: Prevalence of Asymptomatic and Symptomatic Peripheral Arterial Disease in the General Population. *International Journal of Epidemiology*, 20(2):384–392, 06 1991.

[14] J Nordanstig. Benartärsjukdom - dold och underbehandlad folksjukdom. https://www.youtube.com/watch?v=s78ML7wnAig, September 2020.

[15] M. Rac-Albu, L. Iliuta, S. M. Guberna, and C. Sinescu. The role of ankle-brachial index for predicting peripheral arterial disease. *Maedica (Bucur)*, 9(3):295–302, Sep 2014.

[16] CognitiveScale. Amplify: Ai powered patient scheduling. https://www.cognitivescale.com/wp-content/uploads/2019/04/Amplify_AI_Powered_Patient_Scheduling_AI_Anatomy.pdf.

oscand18@student.hh.se
timand18@student.hh.se

[17] Eric Smalley. Ai-powered drug discovery captures pharma interest. *Nature Biotechnology*, 7:604–605, 2017.

[18] P. Szolovits, R. S. Patil, and W. B. Schwartz. Artificial intelligence in medical diagnosis. *Ann Intern Med*, 108(1):80–87, Jan 1988.

[19] C. Sinsky, L. Colligan, L. Li, M. Prgomet, S. Reynolds, L. Goeders, J. Westbrook, M. Tutty, and G. Blike. Allocation of Physician Time in Ambulatory Practice: A Time and Motion Study in 4 Specialties. *Ann Intern Med*, 165(11):753–760, Dec 2016.

[20] Emile R. Mohler III. Peripheral Arterial Disease: Identification and Implications. *Archives of Internal Medicine*, 163(19):2306–2314, 10 2003.

[21] Andrew J. Steele, Spiros C. Denaxas, Anoop D. Shah, Harry Hemingway, and Nicholas M. Luscombe. Machine learning models in electronic health records can outperform conventional survival models for predicting patient mortality in coronary artery disease. *PLOS ONE*, 13(8):1–20, 08 2018.

[22] Manish Motwani, Damini Dey, Daniel S. Berman, Guido Germano, Stephan Achenbach, Mouaz H. Al-Mallah, Daniele Andreini, Matthew J. Budoff, Filippo Cademartiri, Tracy Q. Callister, Hyuk-Jae Chang, Kavitha Chinnaiyan, Benjamin J.W. Chow, Ricardo C. Cury, Augustin Delago, Millie Gomez, Heidi Gransar, Martin Hadamitzky, Joerg Hausleiter, Niree Hindoyan, Gudrun Feuchtner, Philipp A. Kaufmann, Yong-Jin Kim, Jonathon Leipsic, Fay Y. Lin, Erica Maffei, Hugo Marques, Gianluca Pontone, Gilbert Raff, Ronen Rubinshtein, Leslee J. Shaw, Julia Stehli, Todd C. Villines, Allison Dunning, James K. Min, and Piotr J. Slomka. Machine learning for prediction of all-cause mortality in patients with suspected coronary artery disease: a 5-year multicentre prospective registry analysis. *European Heart Journal*, 38(7):500–507, 06 2016.

[23] Changho Shin, Seungeun Rho, Hyoseop Lee, and Wonjong Rhee. Data requirements for applying machine learning to energy disaggregation. *Energies*, 12(9), 2019.

[24] A. K. Jain and B. Chandrasekaran. 39 dimensionality and sample size considerations in pattern recognition practice. In *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 835–855. Elsevier, 1982.

[25] S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.

[26] Frank E. Harrell. *Case Study in Binary Logistic Regression, Model Selection and Approximation: Predicting Cause of Death*, pages 275–289. Springer International Publishing, Cham, 2015.

[27] Wikipedia contributors. Regression analysis — Wikipedia, the free encyclopedia, 2021. [Online; accessed 7-March-2021].

[28] Wikipedia contributors. Overfitting — Wikipedia, the free encyclopedia, 2021.

[29] Wikipedia contributors. Learning curve (machine learning) — Wikipedia, the free encyclopedia, 2021. [Online; accessed 9-March-2021].

[30] Stan Lipovetsky and Ewa Nowakowska. Modeling with structurally missing data by ols and shapley value regressions. *International J. of Operations and Quantitative Management*, 19:169–178, 09 2013.

oscand18@student.hh.se
timand18@student.hh.se

[31] J. A. Sterne, I. R. White, J. B. Carlin, M. Spratt, P. Royston, M. G. Kenward, A. M. Wood, and J. R. Carpenter. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ*, 338:b2393, Jun 2009.

[32] K. Bhaskaran and L. Smeeth. What is the difference between missing completely at random and missing at random? *Int J Epidemiol*, 43(4):1336–1339, Aug 2014.

[33] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

[34] S. Chevret, S. Seaman, and M. Resche-Rigon. Multiple imputation: a mature approach to dealing with missing data. *Intensive Care Med*, 41(2):348–350, Feb 2015.

[35] Melanie L. Bell, Mallorie Fiero, Nicholas J. Horton, and Chiu-Hsieh Hsu. Handling missing data in rcts; a review of the top medical journals. *BMC Medical Research Methodology*, 14(1), 2014.

[36] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Harry Zhang. The optimality of naive bayes. volume 2, 01 2004.

[39] Richard G. Brereton. The normal distribution. *Journal of Chemometrics*, 28(11):789–792, 2014.

[40] Christian Gouriéroux, Alberto Holly, and Alain Monfort. Likelihood ratio test, wald test, and kuhn-tucker test in linear models with inequality constraints on the regression parameters. *Econometrica*, 50(1):63–80, 1982.

[41] Donald E. Hilt; Donald W. Seegrist. Ridge: a computer program for calculating ridge regression estimates. *Research Note NE-236. Upper Darby, PA: U.S. Department of Agriculture, Forest Service, Northeastern Forest Experiment Station*, page 7p, 1977.

[42] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[43] Trevor Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 01 2009.

[44] scikit-learn developers (BSD License). 1.17. neural network models (supervised), 2021. [Online; accessed 6-Maj-2021].

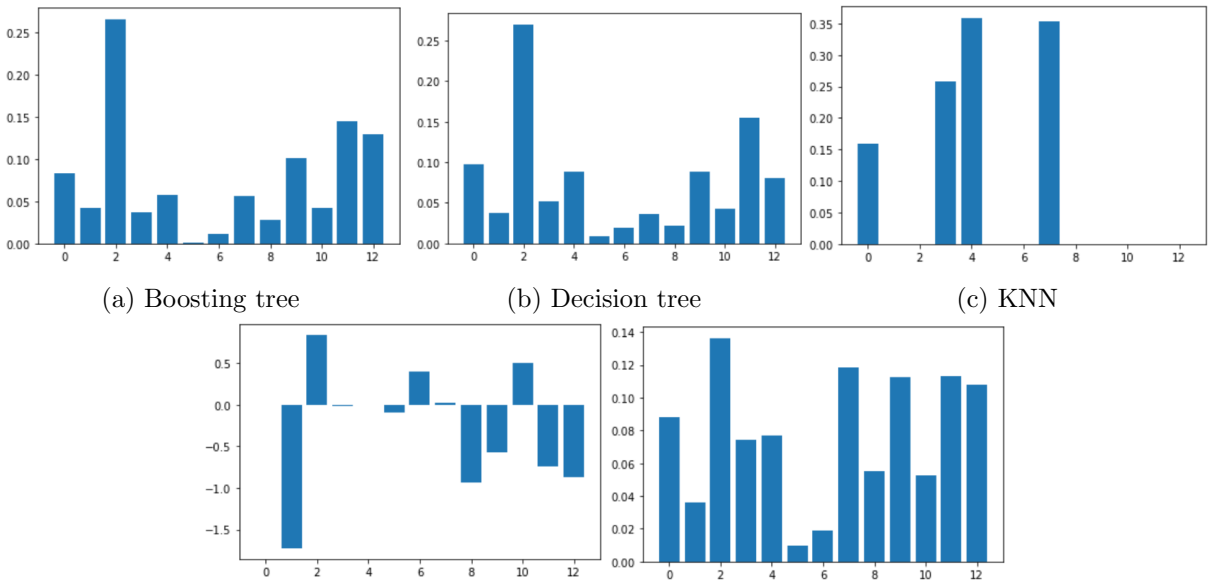
[45] Wolfram Research, Inc. Introduction to the hyperbolic tangent function. Champaign, IL, 2020.

[46] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10:66–71, 2009.

[47] Shivani Gupta and Atul Gupta. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, 161:466–474, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

oscand18@student.hh.se
timand18@student.hh.se

8 Appendices



(a) Boosting tree (b) Decision tree (c) KNN

(d) Logistic regression (e) Random forest

Figure 10: Feature Selection - Feature Importance

oscand18@student.hh.se
timand18@student.hh.se

	Specs	Score
7	thalach	650.008493
9	oldpeak	253.653461
2	cp	217.823922
11	ca	210.625919
8	exang	130.470927
4	chol	110.723364
0	age	81.425368
3	trestbps	45.974069
10	slope	33.673948
1	sex	24.373650

(a) No noise

	Specs	Score
7	thalach	547.036912
9	oldpeak	220.323909
11	ca	188.139163
2	cp	175.766670
8	exang	128.342084
4	chol	89.374990
0	age	72.681775
3	trestbps	42.282532
10	slope	30.035063
1	sex	21.747162

(b) 10% added noise

	Specs	Score
7	thalach	628.499108
9	oldpeak	198.844751
2	cp	175.281234
11	ca	172.233814
4	chol	121.865770
8	exang	96.795524
0	age	65.195807
3	trestbps	62.217457
10	slope	33.368312
1	sex	19.340331

(c) 20% added noise

	Specs	Score
7	thalach	416.828659
9	oldpeak	180.739939
11	ca	147.076865
2	cp	138.821750
4	chol	124.466992
8	exang	89.800578
0	age	77.192850
3	trestbps	35.521454
10	slope	23.493628
1	sex	17.241862

(d) 30% added noise

	Specs	Score
7	thalach	509.891874
9	oldpeak	168.866854
11	ca	142.422188
2	cp	131.234253
4	chol	96.478431
8	exang	81.225904
0	age	44.645035
3	trestbps	22.975916
10	slope	22.459101
1	sex	20.762091

(e) 40% added noise

	Specs	Score
7	thalach	456.762927
4	chol	317.076646
9	oldpeak	147.446897
11	ca	131.681716
2	cp	113.743590
0	age	79.927895
8	exang	67.922129
3	trestbps	34.772921
10	slope	33.689221
1	sex	17.184802

(f) 50% added noise

	Specs	Score
7	thalach	470.036235
9	oldpeak	167.461946
11	ca	154.011391
2	cp	94.518772
8	exang	70.503839
4	chol	50.613835
0	age	43.855870
10	slope	28.157650
1	sex	20.576689
3	trestbps	19.971758

(g) 60% added noise

	Specs	Score
7	thalach	315.727040
9	oldpeak	139.745696
11	ca	129.685679
2	cp	116.520456
8	exang	66.457390
4	chol	63.215001
0	age	57.252143
3	trestbps	57.161364
1	sex	22.382740
10	slope	18.141298

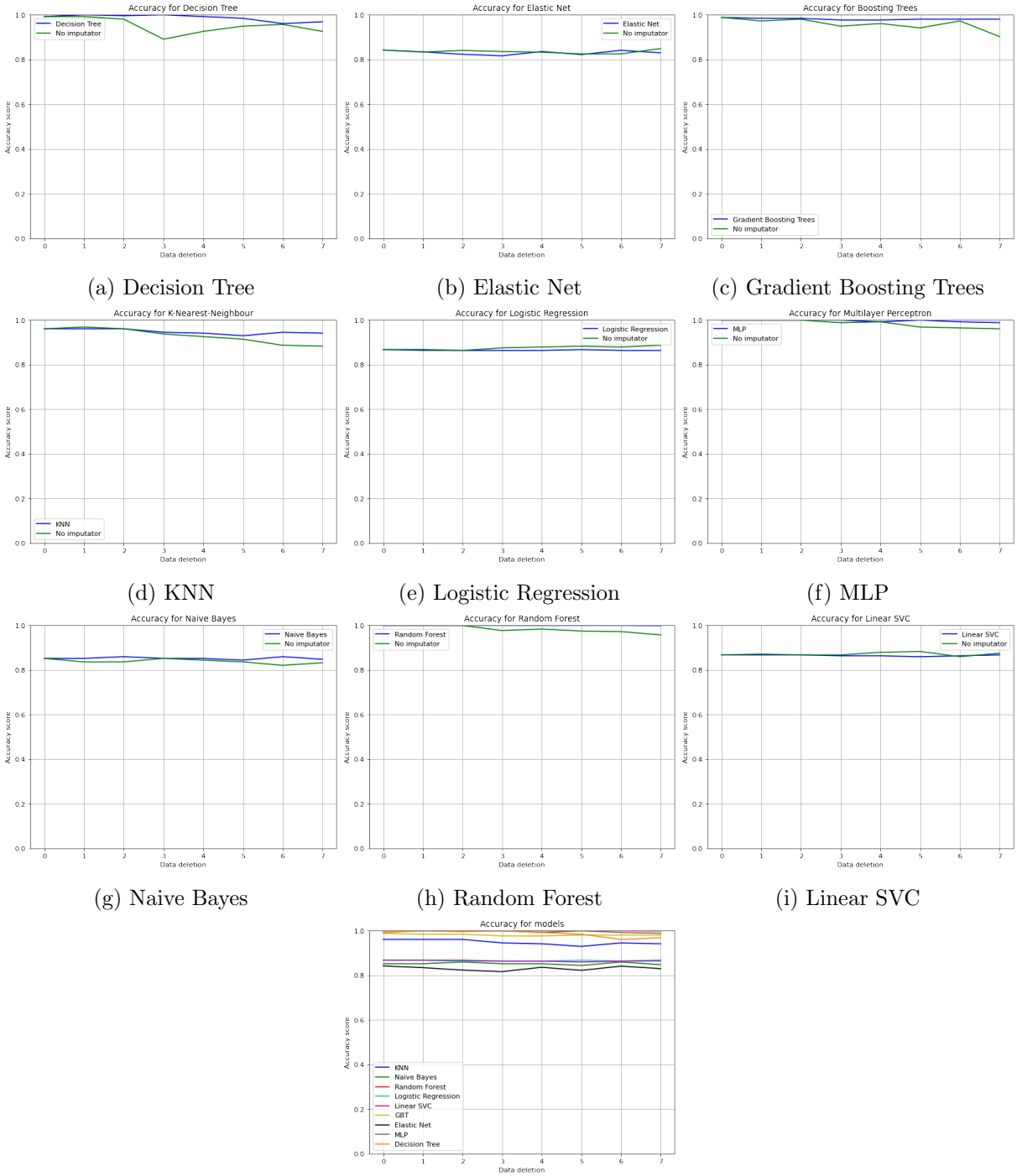
(h) 70% added noise

	Specs	Score
7	thalach	280.679135
9	oldpeak	136.781717
11	ca	83.252665
2	cp	62.739713
8	exang	49.100101
4	chol	48.840853
3	trestbps	43.086315
0	age	39.119602
12	thal	20.271456
10	slope	18.006134

(i) 100% added noise

Figure 11: Feature Selection - Univariate

oscand18@student.hh.se
timand18@student.hh.se



(a) Decision Tree (b) Elastic Net (c) Gradient Boosting Trees

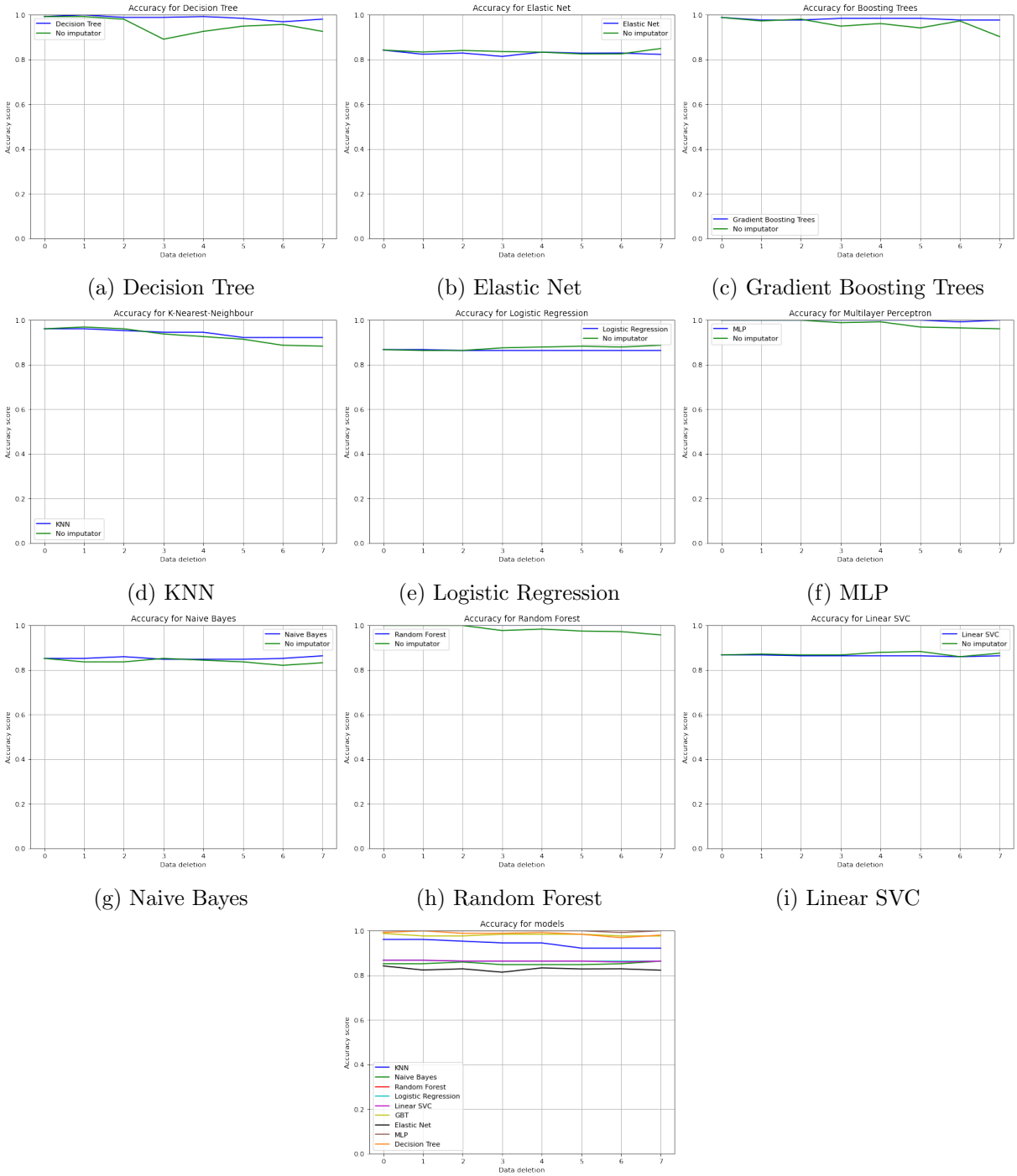
(d) KNN (e) Logistic Regression (f) MLP

(g) Naive Bayes (h) Random Forest (i) Linear SVC

(j) All algorithms

Figure 12: Iterative mean

oscand18@student.hh.se
timand18@student.hh.se



(a) Decision Tree (b) Elastic Net (c) Gradient Boosting Trees

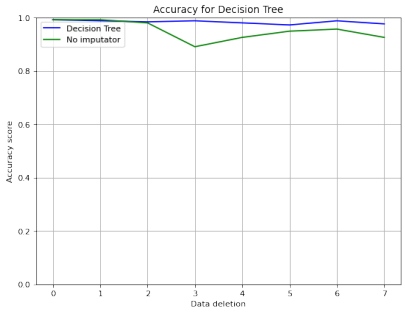
(d) KNN (e) Logistic Regression (f) MLP

(g) Naive Bayes (h) Random Forest (i) Linear SVC

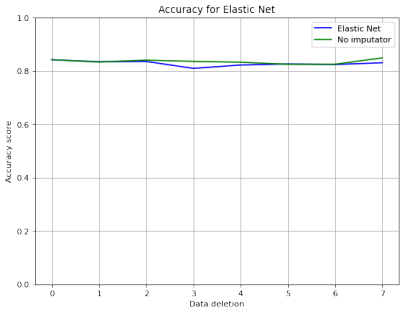
(j) All algorithms

Figure 13: Iterative median

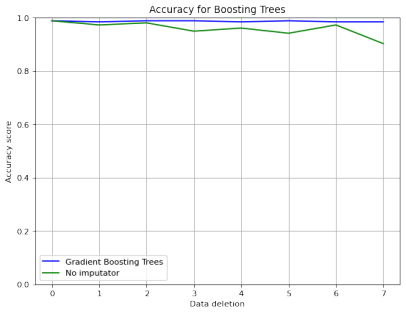
oscand18@student.hh.se
timand18@student.hh.se



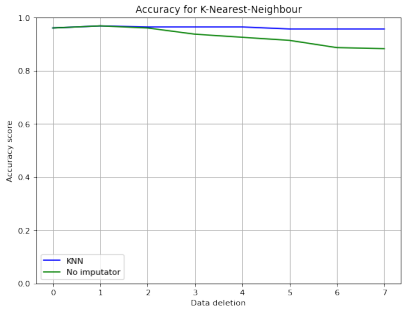
(a) Decision Tree



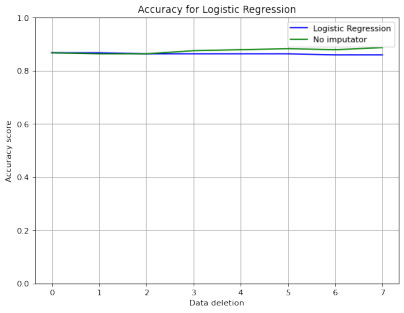
(b) Elastic Net



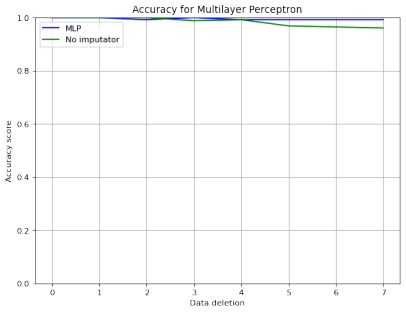
(c) Gradient Boosting Trees



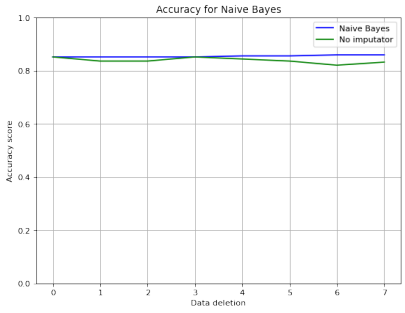
(d) KNN



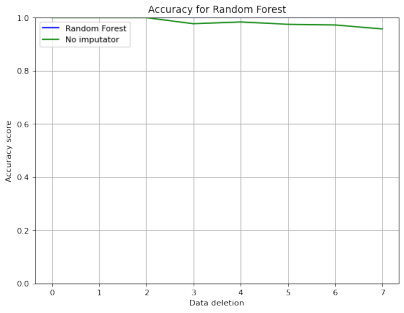
(e) Logistic Regression



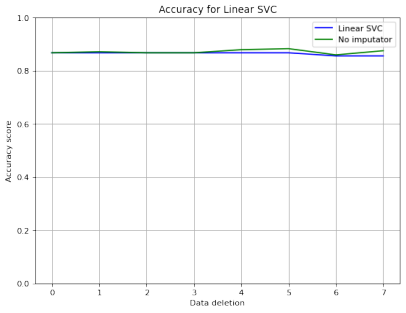
(f) MLP



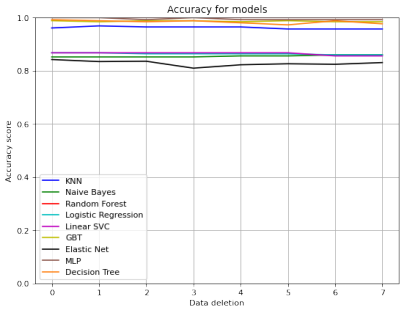
(g) Naive Bayes



(h) Random Forest



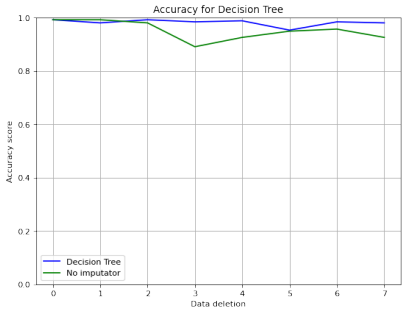
(i) Linear SVC



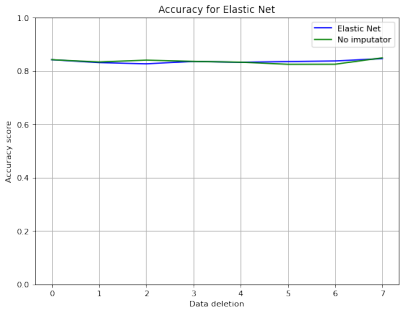
(j) All algorithms

Figure 14: KNN

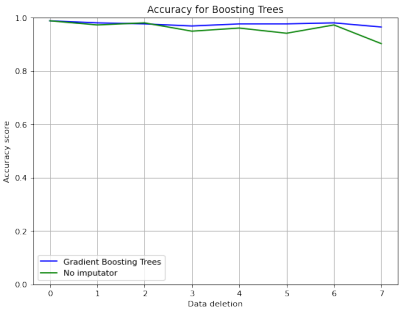
oscand18@student.hh.se
timand18@student.hh.se



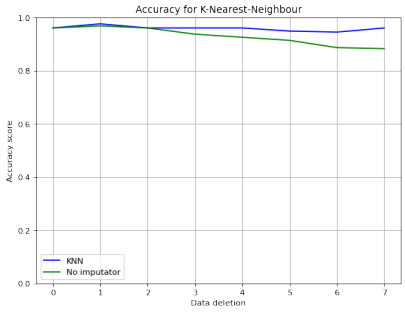
(a) Decision Tree



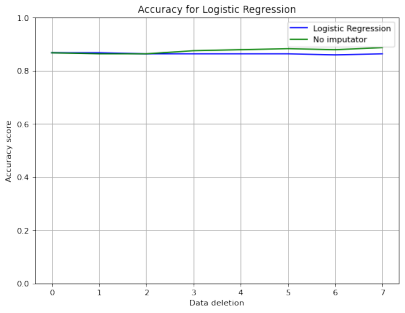
(b) Elastic Net



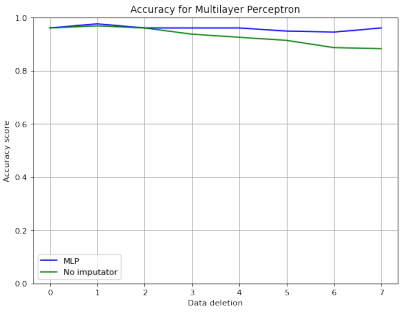
(c) Gradient Boosting Trees



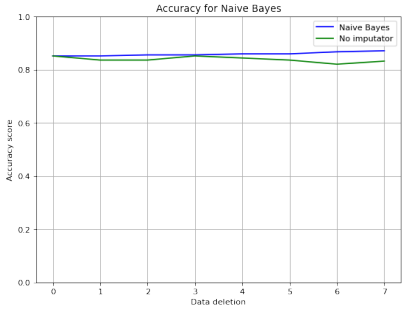
(d) KNN



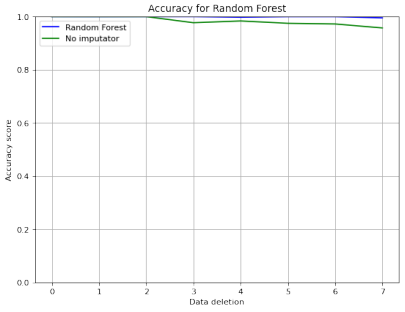
(e) Logistic Regression



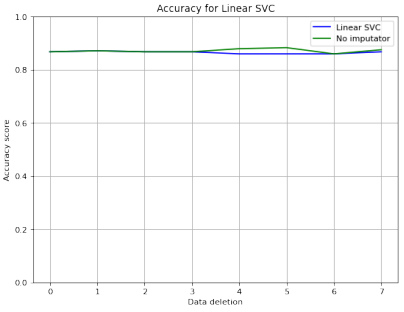
(f) MLP



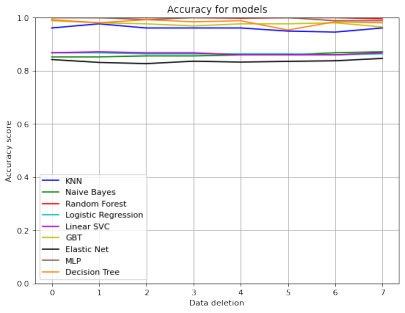
(g) Naive Bayes



(h) Random Forest



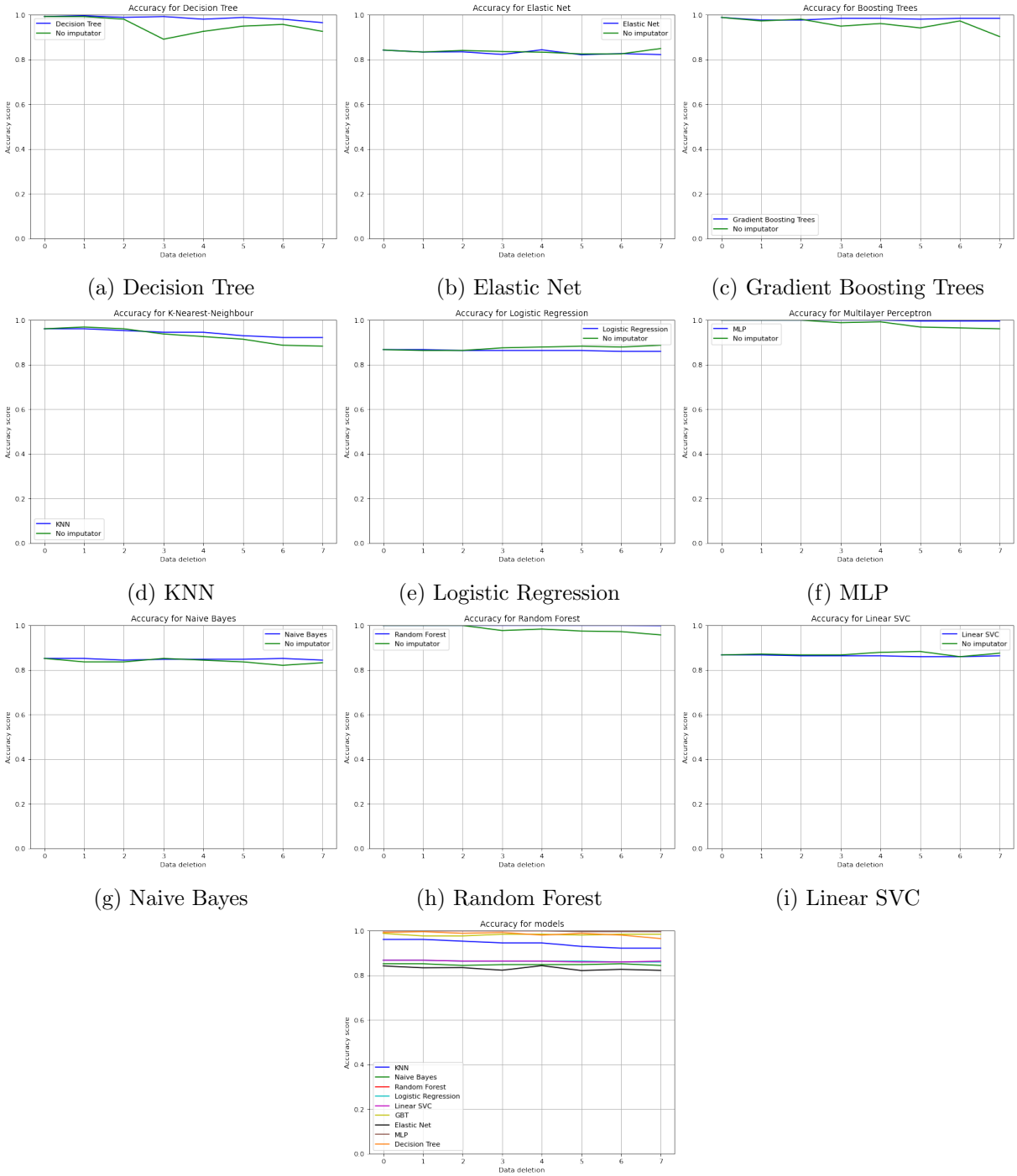
(i) Linear SVC



(j) All algorithms

Figure 15: Regression

oscand18@student.hh.se
timand18@student.hh.se



(a) Decision Tree (b) Elastic Net (c) Gradient Boosting Trees

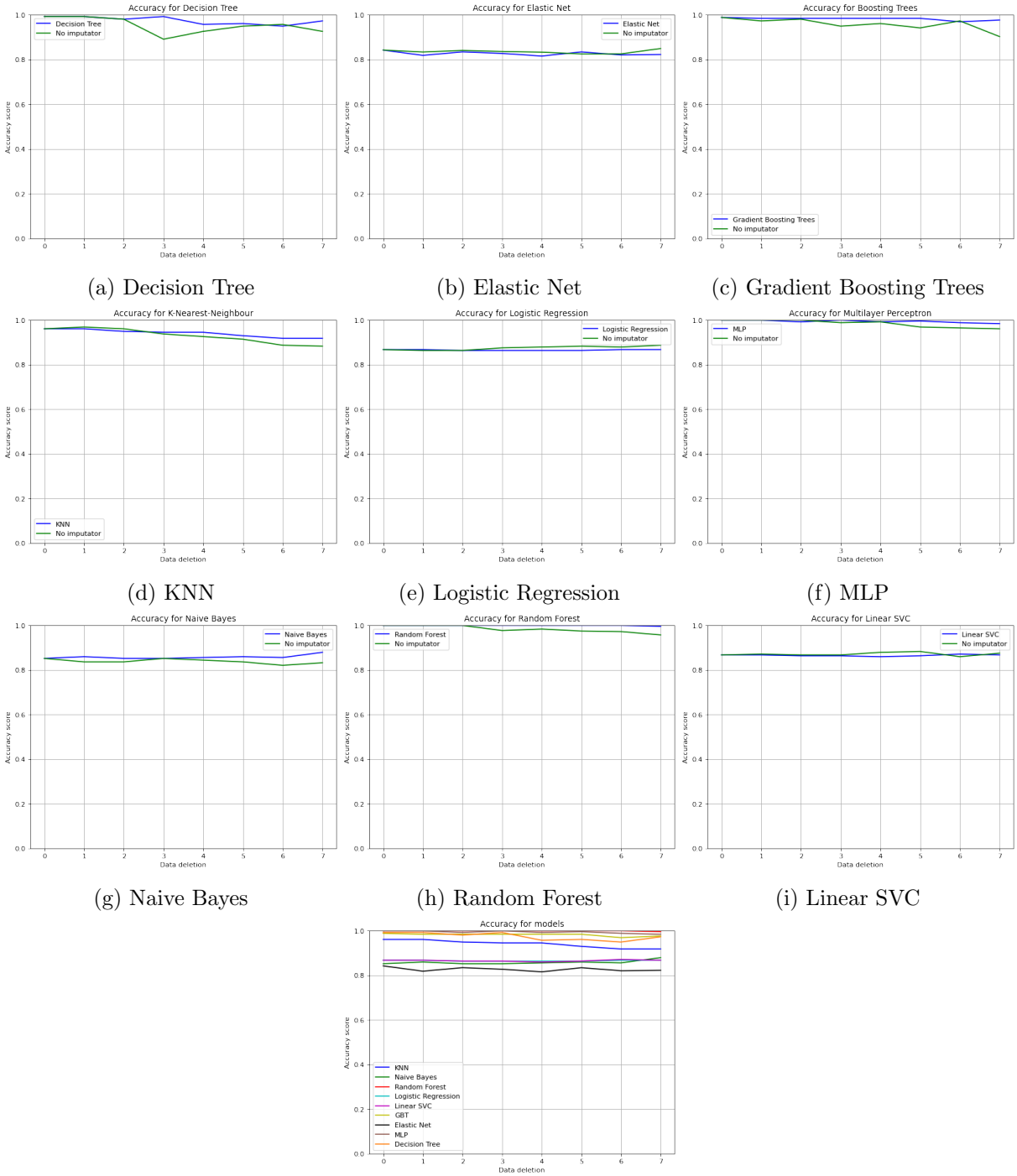
(d) KNN (e) Logistic Regression (f) MLP

(g) Naive Bayes (h) Random Forest (i) Linear SVC

(j) All algorithms

Figure 16: Simple mean

oscand18@student.hh.se
timand18@student.hh.se



(a) Decision Tree (b) Elastic Net (c) Gradient Boosting Trees

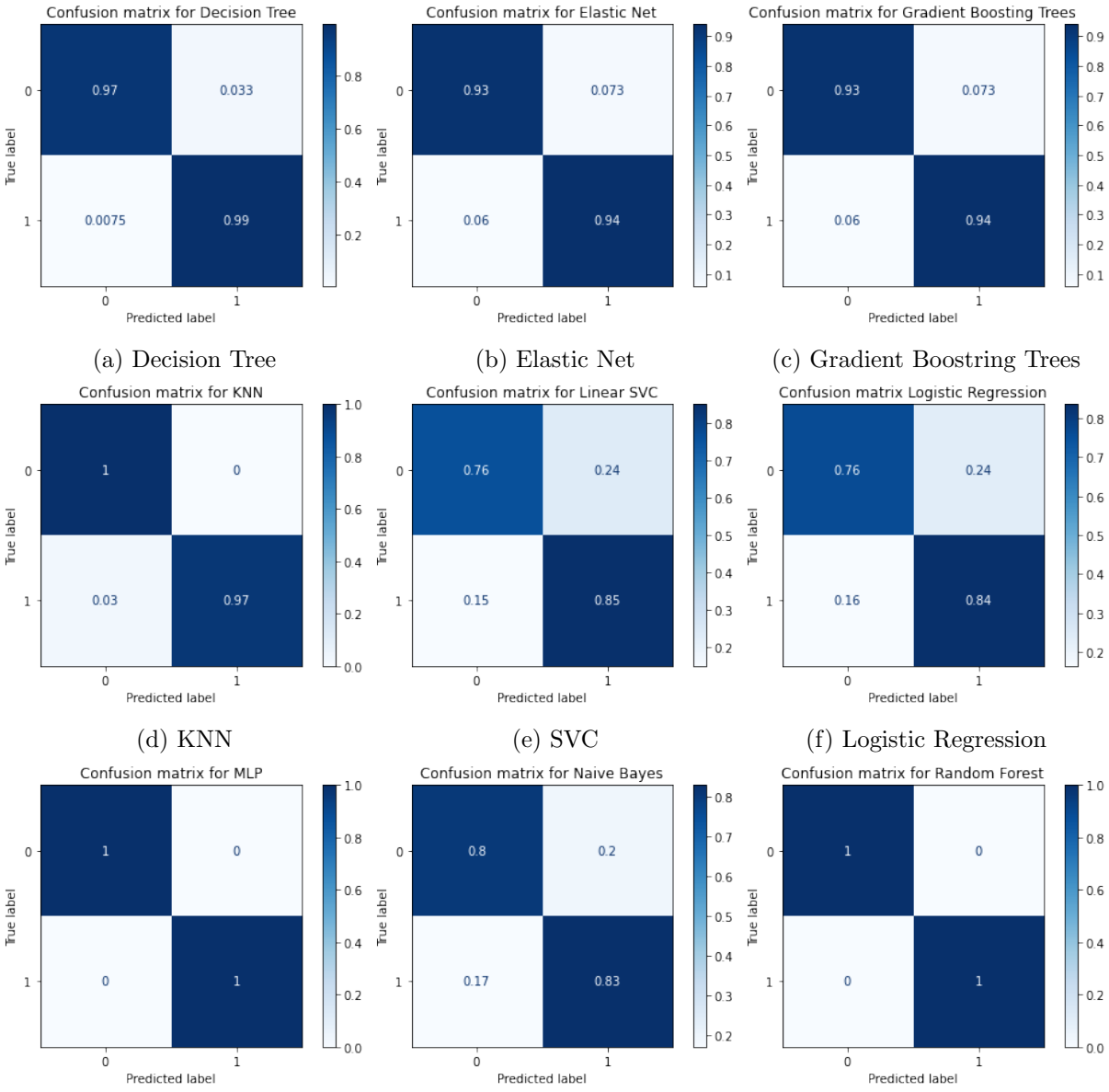
(d) KNN (e) Logistic Regression (f) MLP

(g) Naive Bayes (h) Random Forest (i) Linear SVC

(j) All algorithms

Figure 17: Simple median

oscand18@student.hh.se
timand18@student.hh.se



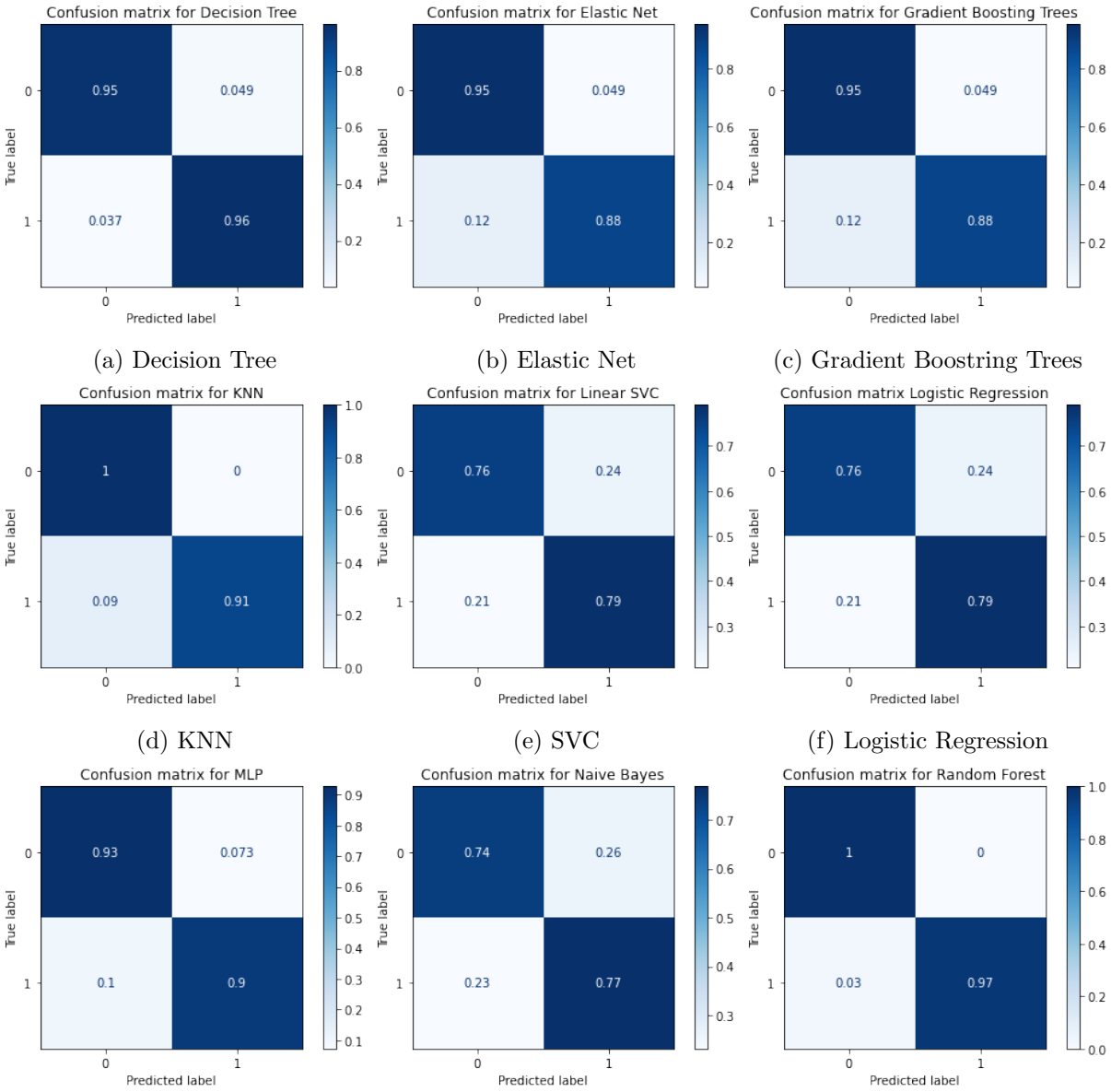
(a) Decision Tree (b) Elastic Net (c) Gradient Boostring Trees

(d) KNN (e) SVC (f) Logistic Regression

(g) MLP (h) Naive Bayes (i) Random forest

Figure 18: Feature Removal of oldpeak, thal and chest pain

oscand18@student.hh.se
timand18@student.hh.se



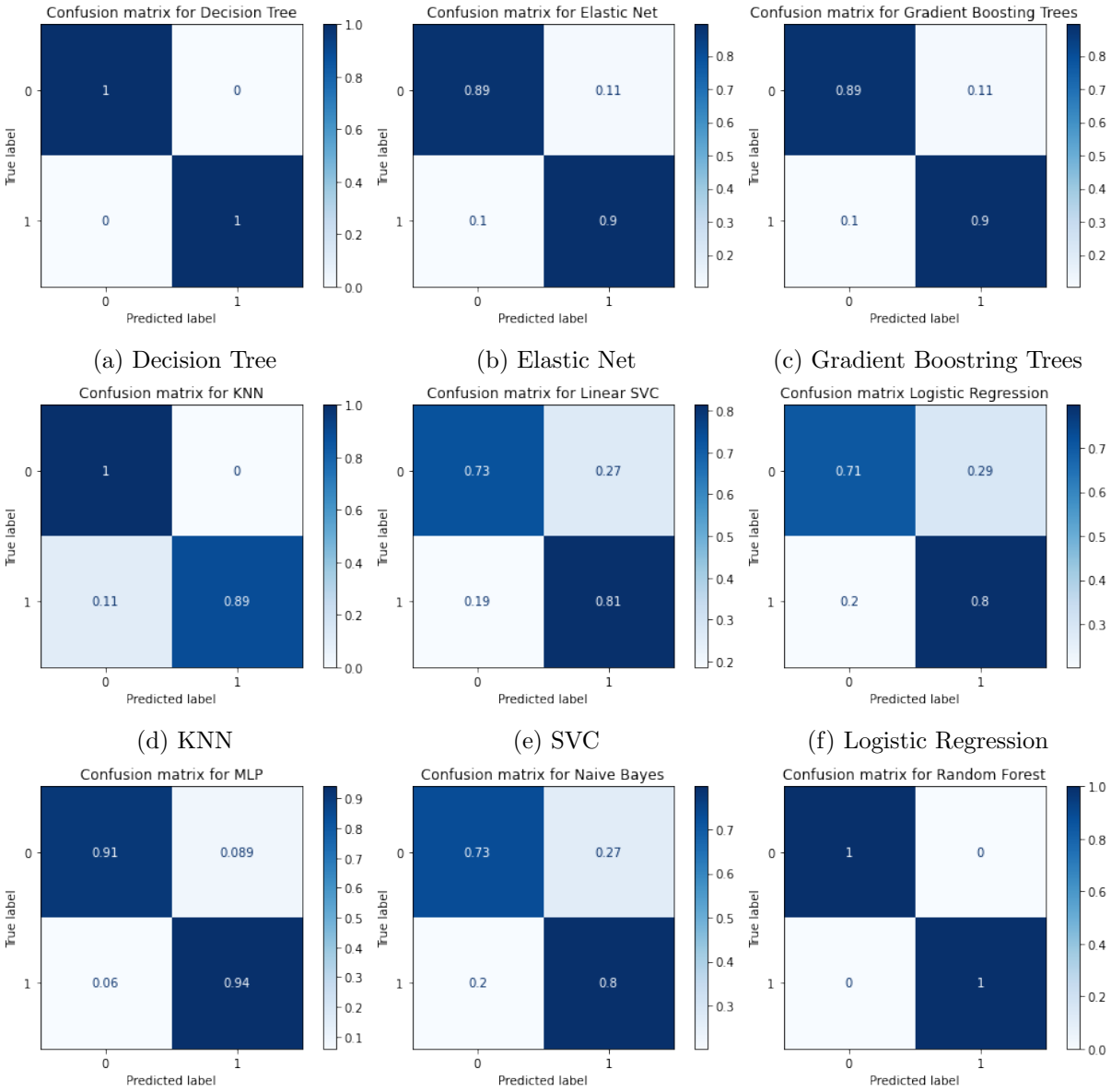
(a) Decision Tree (b) Elastic Net (c) Gradient Boostring Trees

(d) KNN (e) SVC (f) Logistic Regression

(g) MLP (h) Naive Bayes (i) Random forest

Figure 19: Feature Removal of oldpeak, thal, chest pain, max heart rate and color vsl

oscand18@student.hh.se
timand18@student.hh.se



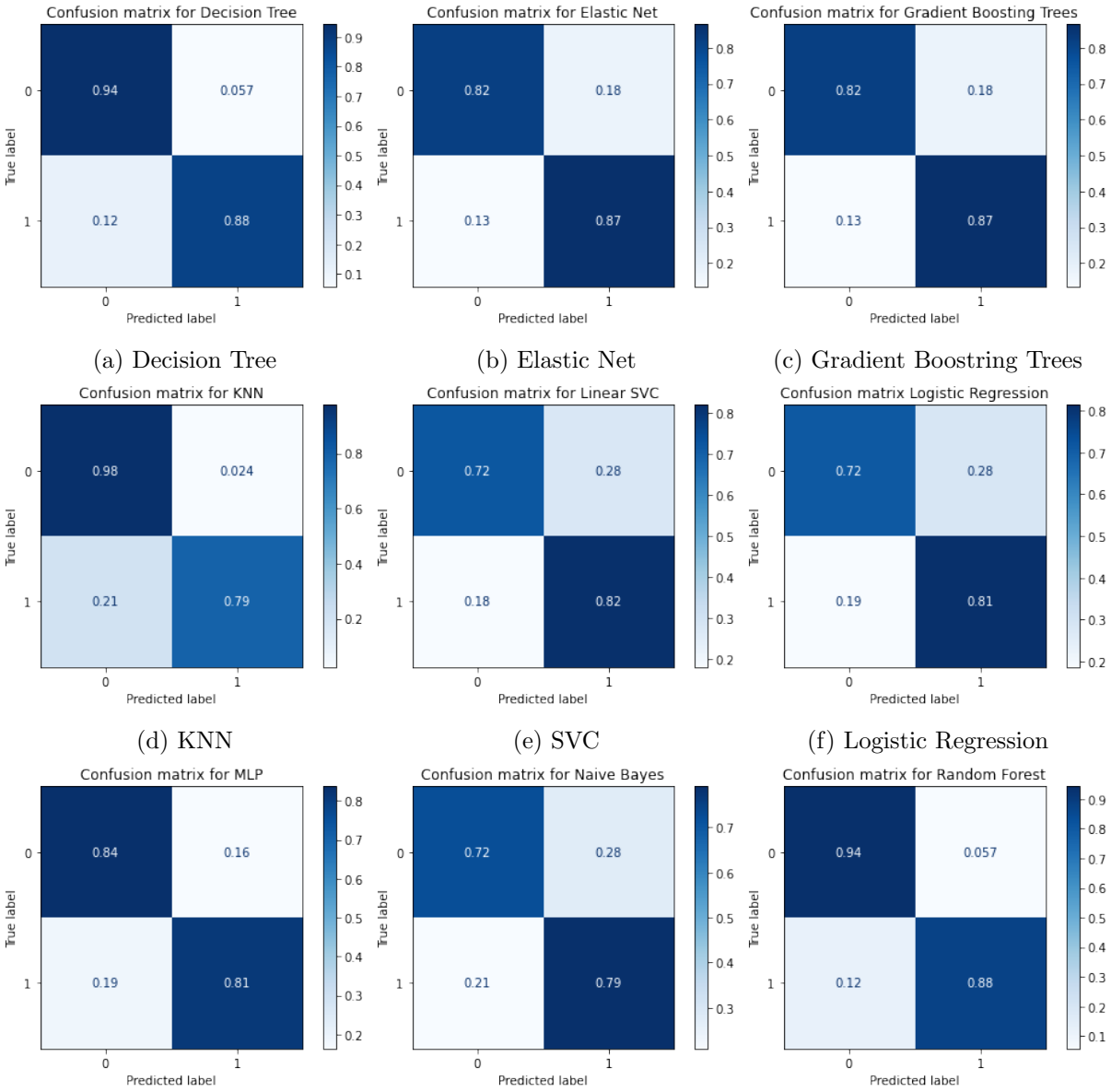
(a) Decision Tree (b) Elastic Net (c) Gradient Boostring Trees

(d) KNN (e) SVC (f) Logistic Regression

(g) MLP (h) Naive Bayes (i) Random forest

Figure 20: Feature Removal of oldpeak, thal, chest pain, max heart rate, color vsl and age

oscand18@student.hh.se
timand18@student.hh.se



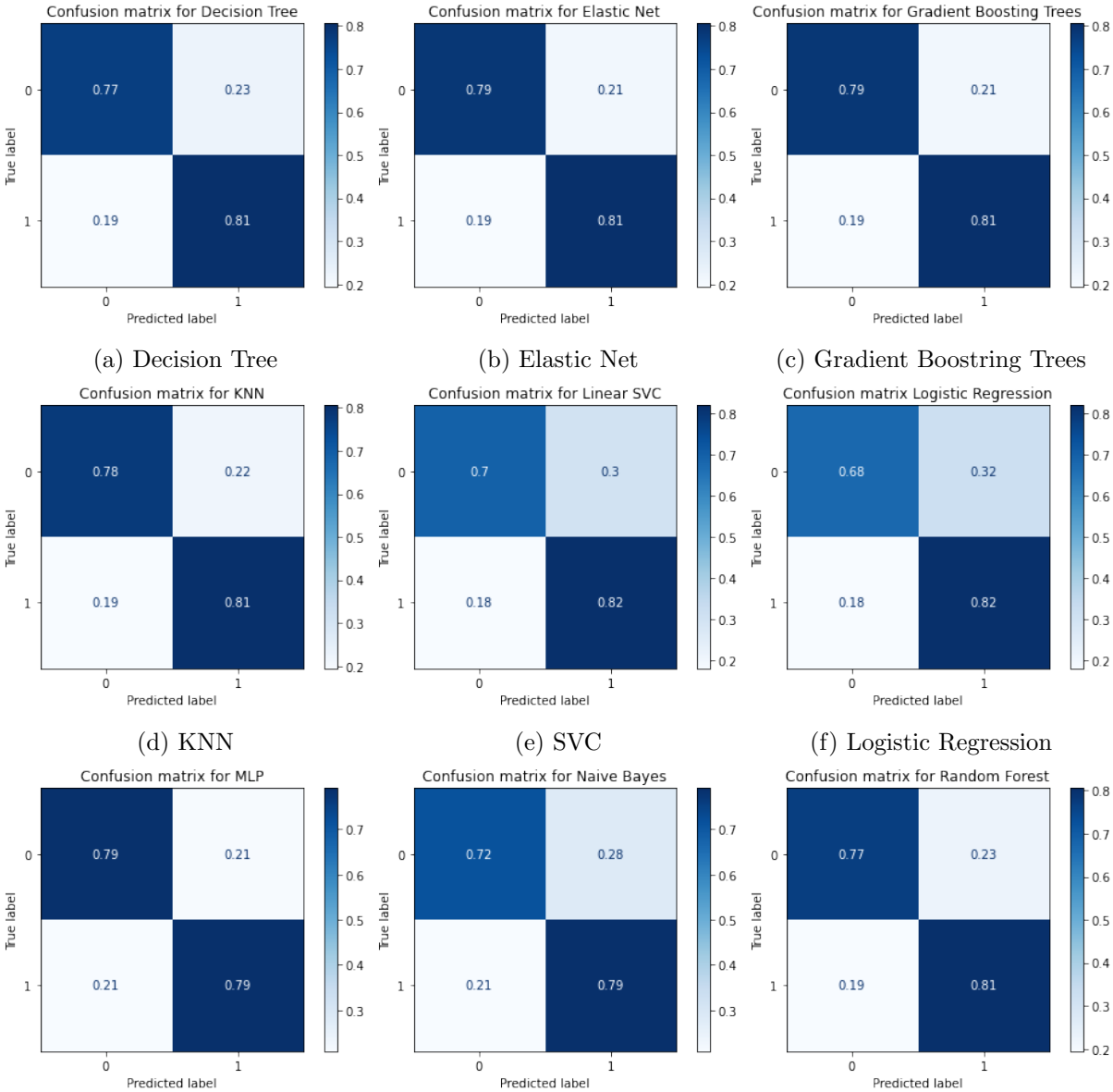
(a) Decision Tree (b) Elastic Net (c) Gradient Boostring Trees

(d) KNN (e) SVC (f) Logistic Regression

(g) MLP (h) Naive Bayes (i) Random forest

Figure 21: Feature Removal of oldpeak, thal, chest pain, max heart rate, color vsl, age and cholesterol

oscand18@student.hh.se
timand18@student.hh.se



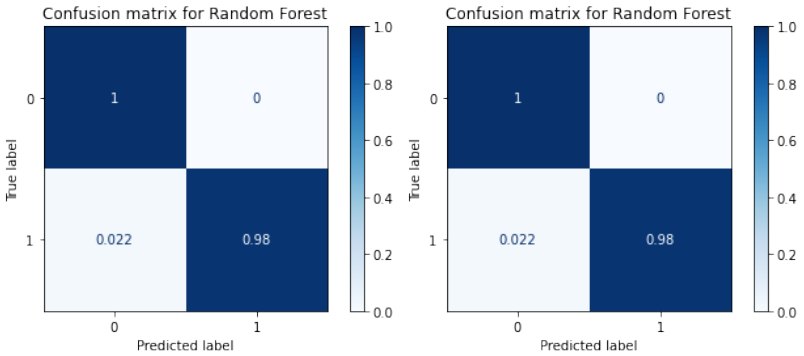
(a) Decision Tree (b) Elastic Net (c) Gradient Boostring Trees

(d) KNN (e) SVC (f) Logistic Regression

(g) MLP (h) Naive Bayes (i) Random forest

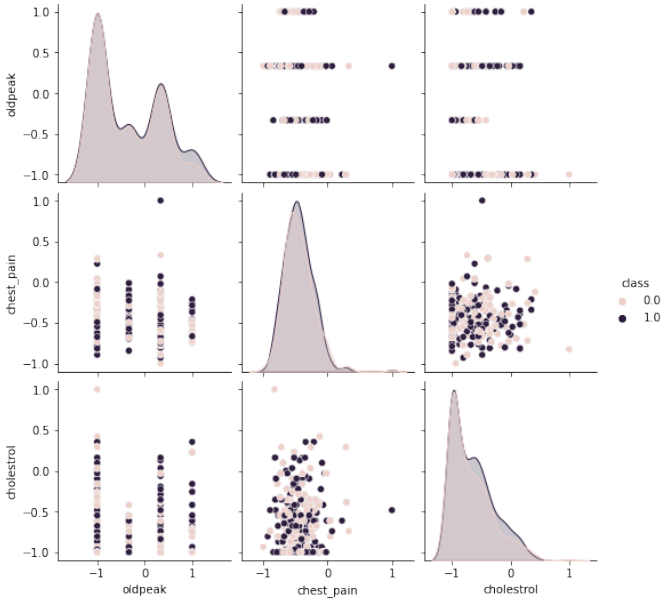
Figure 22: Feature Removal of oldpeak, thal, chest pain, max heart rate, color vsl, age, cholesterol and resting blood pressure

oscand18@student.hh.se
timand18@student.hh.se



(a) Random Forest (b) Decision Tree

Figure 23: Feature Removal of every feature but oldpeak, thal, chest pain and max heart rate



(a) Random Forest

Figure 24: Pairplot - Random Forest

oscand18@student.hh.se
timand18@student.hh.se

Oscar Andersson, student at
Halmstad University

Tim Andersson, student at Halmstad
University

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se