CHALMERS
UNIVERSITY OF TECHNOLOGY

Finite Element Human Body Model-based Injury Prediction using Machine Learning

Development of simulation-based surrogate models for injury metrics

Master's thesis in Automotive Engineering

Yash Niranjan Poojary
Akhil Srinivas

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021:35

Finite Element Human Body Model-based Injury Prediction using Machine Learning

Development of simulation-based surrogate models for injury metrics

Yash Niranjan Poojary
Akhil Srinivas

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Injury Prevention
Chalmers University of Technology
Gothenburg, Sweden 2021

Finite Element Human Body Model-based Injury Prediction using Machine Learning
Development of simulation-based surrogate models for injury metrics
YASH NIRANJAN POOJARY
AKHIL SRINIVAS

Supervisor: Jobin John, Chalmers
Karl-Johan Larsson, Autoliv
Examiner: Johan Iraeus, Mechanics and Maritime Sciences, Chalmers

Master's Thesis 2021
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Injury Prevention
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Seated Human body model supported by a machine learning algorithm.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Finite Element Human Body Model-based Injury Prediction using Machine Learning
Development of simulation-based surrogate models for injury metrics
YASH NIRANJAN POOJARY
AKHIL SRINIVAS
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Models of the human body, both physical and virtual, have been vital tools for the design of safer vehicles. Finite element models of the human body are now becoming widely used for evaluating safety systems. With the introduction of future designs like autonomous vehicles that affect the nature of vehicle crashes, HBM will become a necessary tool to assess safety. Detailed human anatomy representation in HBM, however, makes it computationally expensive and consumes a lot of time to simulate crash scenarios. The aim of this thesis was to evaluate machine learning models as computationally inexpensive surrogates for injury metrics from human body models.

Machine learning was performed on outputs from two types of finite element models, crash test dummy (Hybrid III M50 fast model) and the human body (SAFER model). Different types of model outputs namely kinematics, kinetics, injury metrics, and injury risks used for injury evaluation were studied. Linear regression-based models (Ordinary, Lasso, Ridge) were used as the baseline models. Advanced tree-based methods (Random Forest, Gradient boosted, XGBoost, Histogram-based gradient boost) and Gaussian process regression were used to build additional machine learning models. The machine learning pipeline comprised of data preparation, model validation using the k-Fold cross-validation method, and model optimization using hyperparameter tuning methods such as random search and grid search. In these methods, the number of simulations required to achieve accurate models was evaluated.

This thesis provides representative model learning curves (model accuracy vs number of simulations) for the various HBM outputs. The accuracy of the models was seen to be highly dependent on data transformation if the HBM outputs were skewed. It was also seen that estimation of numerical noise needs to be considered as part of the ML pipeline for HBM data to avoid overfitting in the pursuit of accuracy. For the HBM outputs considered in this study, tree-based boosting methods provided accurate models in most cases.

Keywords:Finite Element (FE), Human Body Model (HBM), surrogate models, Injury risks, Injury criteria, Machine Learning, Hybrid III M50 fast model, SAFER HBM, Numerical Noise

Acknowledgements

This master thesis is carried out at the Department of Mechanics and Maritime Sciences along with Autoliv. We would like to thank our examiner Johan Iraeus and our supervisors Jobin John and Karl-Johan Larsson for the continuous support and guidance, thank you for all the time you spent on discussions and clearing our doubts throughout the project.

Furthermore, we are thankful to coderefinery and NVIDIA/ENCCS for giving us an opportunity to attend the seminar and hackathon on AI and software testing. We would like to thank DynaMore Nordic for an opportunity to attend the introductory course in LS-DYNA.

Yash Niranjan Poojary , Gothenburg, June 2021
Akhil Srinivas , Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

Approximately 1.35 million people die and 20-50 million suffer non-fatal injuries resulting in long-term disabilities in road crashes each year[1]. Though the number of fatalities is significant, the trend has been decreasing in the past decade[2]. Decrease in the number of occupant fatalities despite an increase in road vehicle users have been made possible due to stricter safety standards advancements in the field of vehicle safety.

Current safety systems are evaluated and developed using anthropomorphic test devices (ATD) also known as crash test dummies. These ATD's are instrumented to record the data during a crash. ATD's have a simplified representation of the human anatomy which limits their similarity to humans, in order to be robust and do not break during repeated crash tests. To overcome this, in recent years Finite element Human body models (HBM) are increasingly used as they capture more realistic human responses during crashes. HBM are essential tools to evaluate safety as they give a detailed understanding of an injury mechanism during a crash [3].

Despite HBM being very crucial in assessing safety in crashes they are computationally expensive and take a long time to reconstruct crash scenarios. This can be overcome with the use of Machine Learning (ML). The use of ML has been a fast growing field of FEM [4]. The use of ML to predict injury risk and kinematics of crashes can save time and computational effort. Hence ML is a potential alternative that can be used to reduce the time to evaluate crash scenarios. But the ML model needs a sufficiently large set of input-output data in order to learn the underlying structure of the simulation case it will be used to predict. Hence a number of FE simulations must be run, in order to provide the ML method with training data. So exploring how well ML methods predict when they only have a set number of examples, which ML method learns the fastest can be evaluated.

1.1 Background

Prediction of injury severity and risk is an important measure to evaluate crash safety. Many statistical models like logistic regression, multinomial logit, and mixed logit [5] models have been used to calculate the risk and occurrence of traffic crashes from the accident data. But these models are dependent on many assumptions of the distribution of data. A study by Khaled et. al [6] has explored on implementation of ML models like Neural Networks(NN) and support vector machines (SVM) to

predict the crash injury severity from a crash database to learn. The risk for crashes was modeled for inputs Driver attributes, vehicle attributes, and road condition attributes.

Real world human responses are not completely captured by the current crash evaluation techniques. Human and environmental Human variability, such as age, height, and weight, can influence the outcome of an accident. Similarly, the injury outcome depends on the specific accident scenario (impact speed, impacting object, impact direction, vehicle structure design) [7]. Studies conducted to evaluate head intrusion to door impacts and thoracic injuries as a function of restraint parameters using Neural Networks (NN) in Monte Carlo simulations [7]. The resulting response surface was validated using real world field data. This study illustrated the methodology to generate response surfaces incorporating various input parameters for injury predictions which could be used for design considerations.

Joodaki et al.[8] compared the performances of several ML methods on predicting human body model responses in restraint design simulations. The study was done to predict the effects of the restraint system inputs on the behavioral response of the occupant during a crash. The study indicated that hyper-parameter tuning is essential to build surrogates to predict HBM responses. Ensemble methods performed well compared to lasso, support vector regression, Random Forest, Neural Network, and ordinary least squares to predict HBM response in restraint design simulations.

As made clear from the above, previous investigations into using ML to predict HBM responses have only studied by modeling to predict the behavioral response and variability in crashes. There is a lot of variability to evaluate safety during a crash. The use of HBM are expensive to explore all the variability. Hence, we explore the use of ML models to predict injury criteria, injury risks, kinetics, and kinematics during crashes.

1.2 Objectives

The aim of this thesis is to evaluate the use of machine learning methods to build fast surrogates for human body simulation responses in crash scenarios. This involves:

1. Evaluating the number of simulations required to build accurate machine learning models for a given set of FE parameters
2. Implementing machine learning pipelines comprising data preparation, model validation and optimization for various types of responses (kinematics, kinetics, injury metrics, and injury risks) from a Human Body Model.

1.3 Limitations

The development of the ML model in the first objective is done using hybrid III HBM. The generation of data is done with approximated parameter ranges. Ranges defined for the 90th percentile of the real world crashes are used.

Data generated by SAFER HBM from a previous parametric study was used. The data consisted of 1000 simulations and will not be extended if it is not enough for the development and evaluation of ML models.

Evaluation of the machine learning models during this thesis was done using existing python libraries like sklearn [21]. For this study we only studied the ML-model prediction of the following FE-simulation outputs:

1. **Hybrid III M50 fast model**: HIC15, maximum head acceleration,maximum neck force and maximum chest deflection.
2. **SAFER HBM v9**: HIC15, maximum head acceleration, maximum pelvis acceleration, Rib strains, risk of 2+ rib fractures and maximum chest deflection

2
Theory

Theoretical background of the methods used in this thesis is given in this chapter.

2.1 Machine Learning

Machine learning (ML) is a technique that involves algorithms to learn automatically from data and improve on preceding experiences without being distinctly programmed. Such algorithms work using existing data to build and continuously process models to perform predictions. Conventionally, ML forms the core of data mining to exhibit a hidden pattern in large data sets [15]. ML is classified into supervised learning and unsupervised learning. In supervised learning, the algorithms are trained with specifically labeled data set and analyses training data to generates anticipated predictions on new data[22]. Supervised learning is further classified into Regression and classification.

Regression modelling is the task to find the approximate mapping function to map input variables($\mathbf{X}$) to continuous output variable ($\mathbf{y}$). This method predicts a continuous variable which is a real value such as an integer or real number. This study is mainly focused to evaluate regression modeling of supervised learning methods.

Every ML project is distinct as the type of data used and the aim of the project is different. However, for every project the steps to train ML algorithms generally the same. This process is referred to as the applied machine learning process. The four steps involved in the applied machine learning process are represented in Figure 2.1.

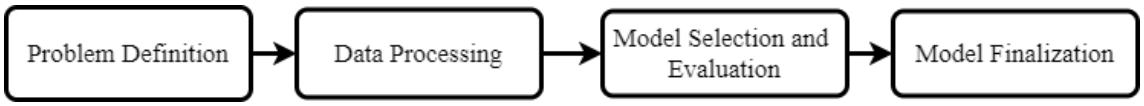


Figure 2.1: Machine Learning Process Flow

2.2 Problem definition

This step involves studying the project objectives to select the framing of the prediction task. Framing includes regression, classification, and unsupervised learning methods. The problem definition includes Data collection from the respective

sources, selecting input and output variables for modeling, encapsulation of collected data using statistical functions to describe mean, standard deviation, min and max values, percentiles values to describe the data structure. These are the steps considered before data processing.

2.3 Data processing

Data processing is to analyze the raw data for any abnormalities and outliers. If any of these are present in the raw data, transformations are used to re-structure the data. Transformations is a data preparation technique that is very specific to the raw data and ML models.

Depending on the ML model used, the type of raw data to train and the relationship between input and output variables will affect the prediction. The raw data obtained from the simulations would presumably be skewed, possibly reducing the ML performance. ML models perform better when data is evenly distributed in the training range.

1. Skewness is a statistical measure to describe the symmetry of the distribution in data about the mean [37]. Positive skewness represents more number of data points on the right and negative skewness when more data is concentrated to the left, as shown in Figure 2.2.

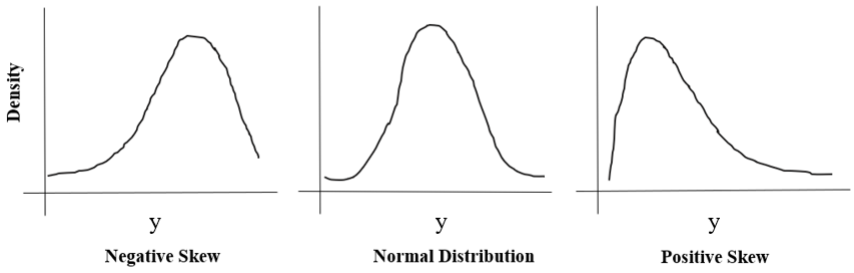


Figure 2.2: Types of Data Skewness

2. Skewness in the data is visualized using Kernel Density Estimate (KDE) plots[38]. Y-axis of KDE plot represents a probability density function and X-axis is an actual variable value. Histogram can also be integrated into KDE plots, where the height of the histogram represents the probability density of the value occurrence.

The skewness of the raw data cannot be corrected completely using transformations but, it can be reduced for better performance of the ML model. The commonly used transformations are log, Quantile, and Logit transformers [21] and these are shown in Table 2.1. Quantile transformer converts data to a normal or uniform distribution, it has an inbuilt inverse function.

Table 2.1: Transformation methods

Transformation technique	Formula	Inverse
Log transform	$log(y)$	$exp(y)$
logit transform	$\log(\frac{y}{1-y})$	$\frac{\exp y}{\exp y+1}$
Quantile transform	sklearn.preprocessing.QuantileTransformer .fit_transform(y)	.inverse_transform(y)

2.4 Model selection and Evaluation

Every ML model is different and it's performance depends on the type of data and its distribution as mentioned in the previous section. Assessments on the model to check its performance on test data is referred as model evaluation. Test data consists of data not seen by the model before. Based on the Evaluation results The model is selected.

To prepare a training set we split the data into train and test data. The train data will be used to train the ML model and the test data set will be used to evaluate the model performance. The split of the data from the original is shown in figure 2.3. The data is split evenly between the test and train preventing any out of range data to be in test set. This will be verified using the KDE plot.

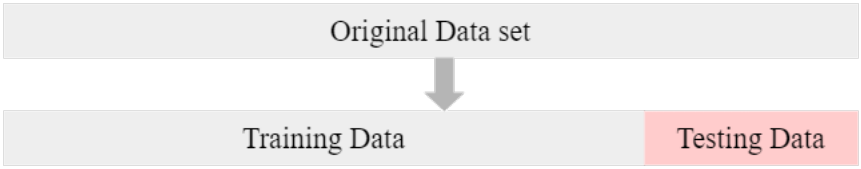


Figure 2.3: Split the original data to test and train

2.4.1 k-Fold Cross-validation

k-Fold cross validation (CV) works by randomly splitting the training data set into k groups as described in the Figure 2.4 illustrating a 4 fold CV. In the first step, one of the groups will be used as a test set referred to as validation set, the red group as shown in the Figure 2.4. The rest of the group data will be used to train the ML model. The model is fit and the validation data is used to evaluate the prediction error. The prediction error is retained and the model is discarded. This process iterates for the defined k times. During this process every group becomes a training and the testing set. In the end of the process k different predictions for k different test groups is obtained.

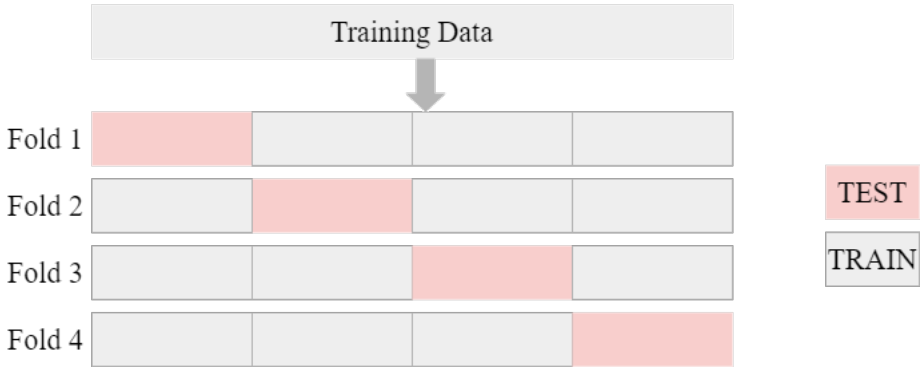


Figure 2.4: K-Fold Cross Validation interpretation with 4 folds. Test set is represented in red and the train group by grey.

k-Fold CV makes the model more accurate in predicting. It is also a more efficient way of using the training data as it prevents data loss as test data will also be used for training in the process.

2.4.2 Metric Functions

An assessment criteria to evaluate the performance of the ML model is referred as a metric. For regression we generally use error metrices like mean square error, root mean square error and mean absolute error. In figure 2.5 the predicted value is plotted against the true value. The 45 deg line represents $X = Y$ i.e.,if a point is on the line the prediction is accurate. The red line represents the residuals or the variation of the prediction.

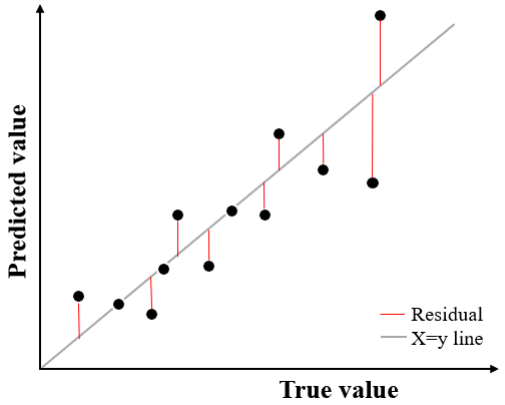


Figure 2.5: Error description due to deviation of the prediction from the actual point

1. **Mean Squared Error(MSE)**
   Mean squared error calculates the difference between the true and the predicted value, represented in the 2.5 in red. Squared the values and gives the mean of this for all the predictions.
$$MSE = \frac{1}{N_{samples}} \sum (y_{true} - y_{prediction})^2 \tag{2.1}$$

2. **Root Mean Squared Error(RMSE)**
   The RMSE is the root of the MSE. This value is more easy for interpretation of the results as the value scale is same as the true values.
$$RMSE = \sqrt{\frac{1}{N_{samples}} \sum (y_{true} - y_{prediction})^2} \tag{2.2}$$

3. **Mean Absolute Error(MAE)**
   MAE is the mean of the absolute value of the residuals. This metric is used for loss functions for various fit in ML algorithms.
$$MAE = \frac{1}{N_{samples}} \sum |(y_{true} - y_{prediction})| \tag{2.3}$$

2.4.3 Bias and variance

The goal of a ML method is to generalise the knowledge obtained from training in order to predict values from new inputs with maximum possible accuracy. Every generated data will have a certain amount of noise. The exact value of any real world system cannot be measured, similarly complex computer simulations for example crash simulations runs over many time-steps with finite accuracy. Every time step generates small errors due to many reasons for example rounding off. So the ML model needs to study the underlying structure of the data and learn.

Every ML method has a unique framework to generalise the relationship of the inputs and outputs. Sometimes this framework limits the ability of the model to learn during the training process. This limits causes bias in the model. High biased model prevents effective learning during the training, this is also referred as underfitting.

When the built model learns too much during training i.e., the model learns the noise in the data along with useful information, it causes high variance in the predictions. In this case the model does not generalise the relation between the input and the output and it's called overfitting.

A high biased model is limited by not being to train effectively and causing underfitting. A high variance model will not generalise as it learns too much during training causing overfitting [36]. The model should be selected trying to avoid both these scenarios. To find this visualisation of the error is done using learning curve and validation curve to do this.

2.4.4 Validation curve

A validation curve is the plot of the error against the features of the ML model. The features are the hyperparameters influencing the prediction accuracy of the model. The hyperparameters affects the complexity of the model. Both the training and the validation errors are plotted as represented in figure 2.6. Using the nature of the curves of both training and validation a optimum model is selected.

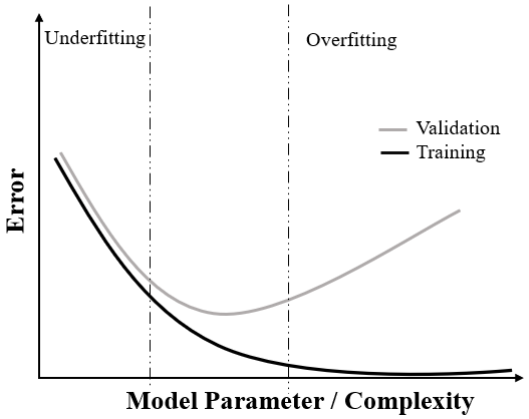


Figure 2.6: Validation curve representing overfitting and underfitting

In the figure 2.6 a part of the plot when the both training and validation error are high, the model has high bias causing underfitting as the model is not able to learn during training. The region of the plot where the errors diverge, the training error close to zero as the model learns too much during training. The validation error is high as the model was unable to generalise to the new data, this resembles overfitting.

2.4.5 Learning curves

Learning curve is the plot of the error against the training data sizes. Similar to the validation curve both the validation and the training error is plotted to find the suitable model. When the training and the validation errors converge early as seen in figure 2.7 the model is highly biased. For this model with increase in the sample size the error does not improve significantly as the model is underfitted. This can be improved by tuning the ML method by adding more input parameters. In figure 2.8 there is a distance between the curves represented by a red line, this is due to high variance. This case of overfitting can be improved by increasing the sample size for training.

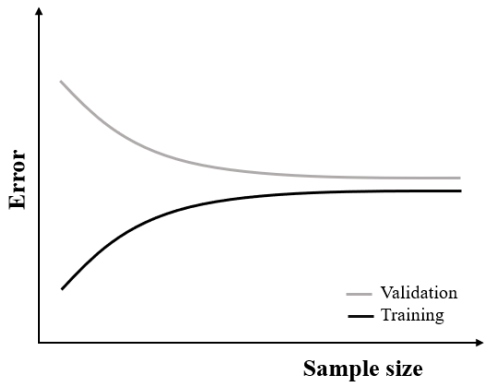


Figure 2.7: Learning curve representing Bias

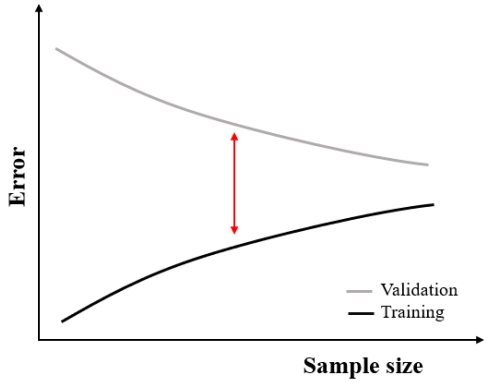


Figure 2.8: Learning curve representing variance

2.4.6 Hyper-parameter tuning

ML methods have unique features which defines the working of the method, these are referred to as hyperparamters. Searching for the most suitable and ideal hyperparameters for the model to perform at its best for a data is called as Hyperparameter tuning. This is an important aspect influencing the model performance. Many methods are used to tune the features like Random search and Grid search. Discrete values of the features need to given these methods so that it can find the best combination among the provided values. From the Validation curve an effective range of features to avoid either overfitting or underfitting is selected.

1. **Random search**
   Randomsearch is used to find a combination of hyperparametrs using limited number of iterations. This method might overlook some combinations due to limited iterations specified. But this gives a better picture on the range of the hyperparameters.

2. **Grid search**
   Gridsearch is the final step followed for fine tuning the ML model, it fits the ML model to all the possible combination of from the set of discrete values specifid in the range which is attained for Randomsearch. This gives the final combination of features to a model.

2.5 Machine Learning Algorithms:

The implementation of Machine learning algorithms is dependent on the type of objective defined for the project. Our objective involves finding the most suitable ML model for different training samples with independent variables (**X**, input FE parameters) with dependent variables (**y**, injury risk, and kinetics) using the post processed data from FE simulations. Regression models are used and it involves the following workflow.

The training data set is used by the ML algorithm to learn and develop a hypothesis. The hypothesis is developed relative to the ML algorithm to create a mapping function $f(\mathbf{X}, \mathbf{y})$.

The mapping function, $f(\mathbf{X}, \mathbf{y})$ generalizes based on the most suitable ML model to balance bias and variance by reducing the residual errors. The errors are computed through cost function and gradient descent algorithms to minimize the errors.

Basic Principles in Machine learning:

1. **Cost-Function**
   In machine learning, cost-functions are implemented to measure the goodness of a model in realising its ability to evaluate the correlation between $\mathbf{X}$ and $\mathbf{y}$. It is calculated as the average difference between the predicted values (results from the hypothesis) and actual output values $\mathbf{y}$.

   $$J = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \tag{2.4}$$

   where, J is cost-function, y_i is the actual value and $\hat{y}_i$ is the predicted value from the ML model.

   The above function is similar to MSE metric and $\frac{1}{2m}$ is included in the equation to make mathematics simple and convenient to gradient descent computations.

2. **Cost-function minimization - Gradient Descent**
   The Machine learning models learn by minimizing a cost- function using gradient descent function[23]. Therefore, Gradient descent is an optimizing algorithm that repeatedly attempts to get a local minimum of function[23].

   Mathematically, gradient descent algorithm performs by taking the derivative of cost function, where derivative is the slope of the tangent at that point. The slope of cost-function is stepped down in the conduct with steepest descent. As the ML models iterate, the function finds a convergence point with slope zero inferring minimal residual error for those parameters of the hypothesis.

   The iterative step size is determined by hyper-parameter called learning rate α and is tuned manually[23]. When α is preferred far from the optimal value α^* the optimization algorithm diverges and if $\alpha < \alpha^*$ converges slowly[23].

Goal:
Minimize Cost-function until reaches convergence point/local minimum(zero slope). Basically there are distinguished techniques to overcome overfitting for linear models and non-linear models respectively. The below techniques/methods assist to generalise well to the data and learn well to prevent from overfitting or underfitting.

2.5.1 Linear Model

Linear model in regression modelling finds the line that best fits the data points and gives a linear relationship between independent variables (**X**) and dependent variable (**y**). When linear regression model comprises of a large number of **X** as in Equation 2.5 will be affected by overfitting and multicollinearity, which exits when independent variables are correlated with each other. Thus, the above conditions reduce prediction accuracy of linear regression model on test data set.

Regularisation is a configuration of regression [26] which are imposed on linear models in order to minimize cost function to prevent overfitting. This is achieved by penalizing models with large parameter values [25]. The common techniques are L1 and L2 regularisations[21], which are complimented to algorithm which minimizes cost-function i.e. minimize($(h_\theta(X)-y_i)^2 + \lambda\,||\theta|\,|$) for L1 and minimize($(h_\theta(X)-y_i)^2 + \lambda\,||\theta|\,|^2$) for L2, where θ = parameters vector, and

$$h_\theta(X) = \theta_0 + \theta_1 X_1 + ... + \theta_n X_n \tag{2.5}$$

In this technique, the parameter "λ" is introduced which weights the penalty in opposition to complex models with increasing variance in the data errors. This results in giving increasing penalty as complexity of model increases [24].
The models developed based on regularisation technique are,

1. **Ridge Regression**
   In Ridge regression, the cost-function is penalized/modified by adding a penalty[21] parameter λ to square of the co-efficients(θ).Therefore, this method is computed until minimizing the cost function. The cost function used for ridge regression is,
   $$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(X^i) - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right) \tag{2.6}$$
   where, j = number of co-efficient including intercept(θ_0),y^i is actual value and λ is the regularisation parameter[28] which puts constraint on co-efficients by regularizing so that when co-efficients take large values, the function shrinks the co-efficients and aids to reduce model complexity. This gives small amount of bias due to penalty and reduces variance.
   The value of λ is chosen between 0 to positive infinity, so if $\lambda = 0$: The ridge regression line is similar to cost-function of linear regression and if λ is too large: The function smooths out excessively and cause underfitting[**?**]. The model is run using a function `linear_model.Ridge` from scikit-learn[21] library.

2. **Lasso Regression**
   The lasso regression is similar to concept of ridge regression. The difference is instead of taking square, the absolute value of co-efficients(θ) are considered. The cost function used for lasso regression is,
   $$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(X^i) - y^i)^2 + \lambda\sum_{j=1}^{n}|\theta_j|\right) \tag{2.7}$$

Similar to Ridge regression, when $\lambda = 0$, the above equation behaves as linear regression cost-function by calculating only squared-residuals and if λ is increated to larger extent, the co-effcients values can lead to zero because increased value of λ can smooth out excessively and results in zero intercepts leading to underfitting. So, some features with less importance(zero-intercepts) are ignored for the estimation of output in Lasso regression.Thus, Lasso regression technique not just reduces overfitting condition but also does feature selection. The model is run using a function `linear_model.Lasso` from scikit-learn[21] library.

2.5.2 Decision Tree

The decision tress are classified based on the problem definition specified. For instance, if the problem defined is to predict mapping function to map input variables **X** to continuous variable **y** is classed as regression decision trees, wheres as to map to discrete output variable **y** is classed as classification decision Tree. In this algorithm, the data or samples are split into two or more sub samples based on significant splitting decisions.
The structure and basic terminologies used in decision tree are shown in the below figure.

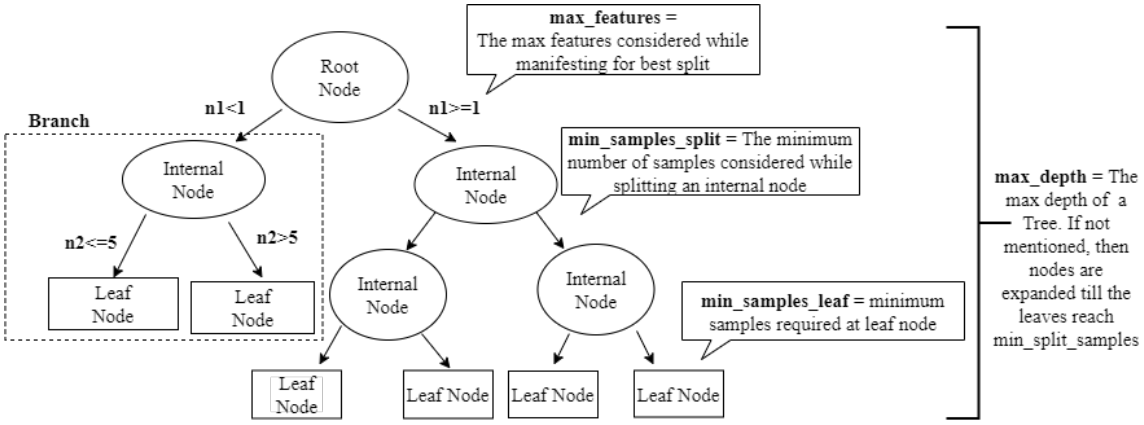


Figure 2.9: Decision Tree algorithm structure

1. **Root Node**
   This is the starting node of the tree which exemplify the entire sample and gets further splits into two or more internal/sub nodes.
2. **Internal Node**
   The internal node illustrates the distinguishing characteristics of data set and branches further into sub-nodes depicting decision criterion.
3. **Leaf Node:** This is the last node which do not split further resulting in giving final outcome in discrete value or average values of observations in that decision tree/region.

Overfitting is the common problem faced from decision tree models when considered continuous numerical variables. So, this problem can be overcome by limiting the

hyper -parameters values using hyper-parameter tuning and cross-validation techniques. The key hyper-parameters which significantly define the decision trees are:

1. **min_samples_split:** This parameter represents the minimum number of samples/observations to be considered while splitting at each node. The higher values could result in underfitting and vice versa. So, the optimum value is chosen where the model is generalised well.
2. **min_samples_leaf:** This parameter represents minimum number of samples/observations required at leaf node. At leaf nodes, the mean of observations value are stored.
3. **max_depth:** This parameter tells the maximum depth of a tree. As the depth is higher, the model learns specifically well to the samples selected.
4. **max_features:** The number of features to be considered for best split from the node.
5. **max_leaf_nodes:** This parameter represents the number of nodes in a tree. The leaf nodes are created substantially as the depth of the tree increases.

2.5.2.1 Regression Tree

Regression tree is a supervised learning method which solves multivariate regression difficulties.When the data to be trained is skewed or non-linearly distributed, the straight line(Linear regression) intuitively produces wrong predictions and leads to contradictory mapping functions from **X** to **y**. Therefore, the regression tree algorithms generates a tree considering all features observations with logical decision technique during splitting at each node.

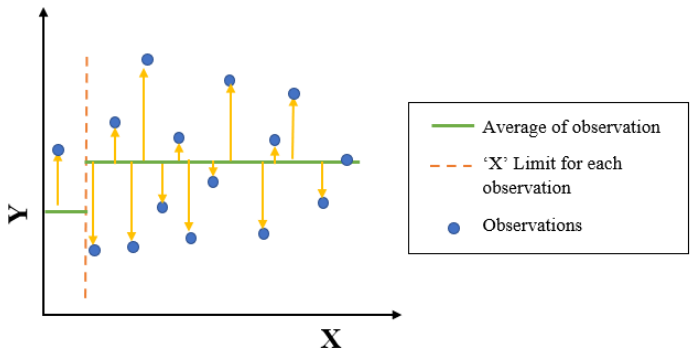


Figure 2.10: Recursive partitioning algorithm

In regression tress, the values obtained at leaf nodes during training are the mean value of observations for that particular modelled decision.Therefore, when unseen data is swamped in that decision tree flow, the predictions are performed with the mean value. This algorithm performs with the initiation by using recursive partitioning algorithm [32]. During splitting at each node of a tree,the algorithm chooses the best variable and its decision limit value by least square error method or least residual error. The Figure 2.10 illustrates the recursive partitioning technique

through which decision tree splits the nodes on all the features of the data set and then selects the appropriate variable and its limit resulting with least residual error. The least squared residual/variance is calculated using the below equation,

$$\text{Sum of squared residual error} = \frac{1}{N}\sum_{i=1}^{N}(y - \bar{y})^2 \tag{2.8}$$

where, Y is observation values(actual values), $\bar{Y}$ is mean of observation values as shown in Figure 2.10, N is the number of samples and least residual error is selected as shown by red box in Figure 2.11.Thus, a fully grown tree is built though splitting process and continued up to the limit of hyper-parameters values specified by the user.

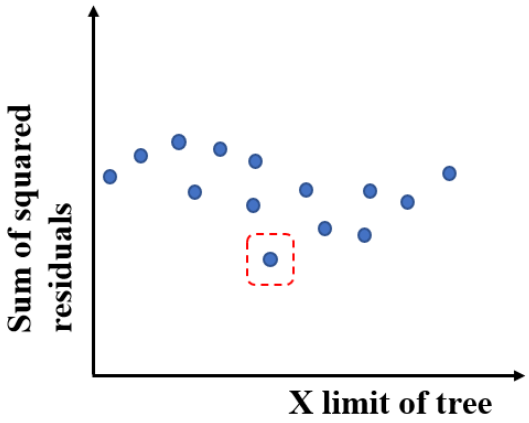


Figure 2.11: Sum of squared residual plot at every X limit for Tree

2.5.3 Ensemble Methods

Ensemble, as the name, refers to grouping. Ensemble methods refer to combine multiple machine learning methods for classification and regression problems to achieve better predictions with a trade-off between bias-variance errors. To elucidate, ensemble methods are capable of reducing prediction errors and generalize well with the model. The commonly used methods to implement ensemble techniques are Boosting and Bagging. This thesis involves implementation of these two techniques to build a good model for the injury risk predictions by generalising well with the FE-simulation data.

1. **Bagging**: Bagging is a technique which combines several decision tress(regression or classification based on problem definition) predictions on random samples of a data set.

(a) **Random forest**

The Random forest is a bagging technique which models several regression trees using bootstrap[1][27] data sets to overcome high variance/overfitting condition on the training data. In regression problem, this algorithm takes the average of all the regression tree outputs to generalise the predictions. The working flow of random forest algorithm is illustrated as follows,

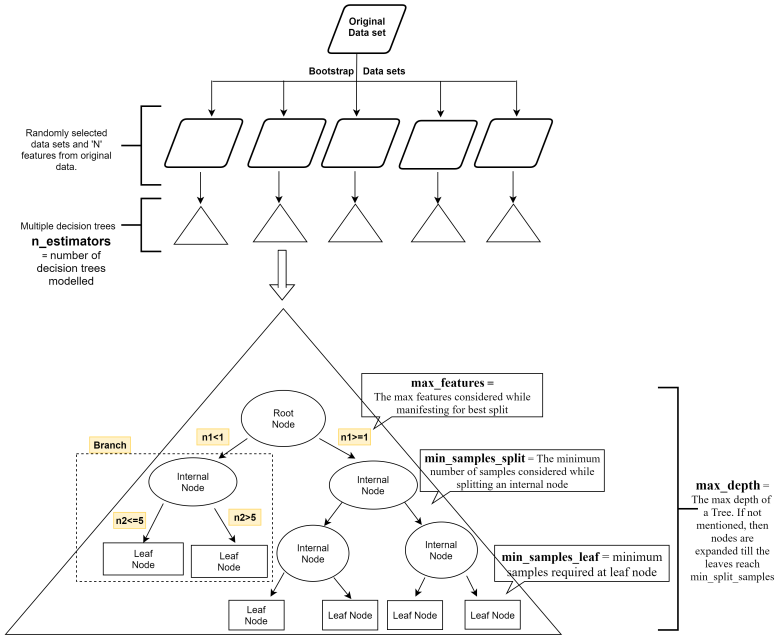


Figure 2.12: Random Forest Regression algorithm workflow

The steps followed in the random forest regression are as follows,

i. Consider the number of samples in the data set is M, then random few samples from M sets of data are selected with indeed repetitive data sets.

ii. Assuming if there are N number of features in M data sets, then random n<N features are selected at each node. The splitting at each node is performed as per the regression tree algorithm as discussed above.

iii. Several decision trees are modeled using different data samples and features and the unseen data is predicted by averaging the predictions from trees modeled.

The random forest algorithm has hyper-parameters same as decision tree with additional *n_estimators* parameter which represents the number of

[1]Bootstrap is a method which selects random samples from data set with replacement(enabling to appear same sample more than once)

regression trees to be modelled. These parameters contribute predominantly in performance of an algorithm in generating good predictions.

2. **Boosting**

Boosting is an ensemble technique that enhances weak-learners into a strong-learner[29] for predictive modeling problems. This technique applies to both classification and regression problems. A weak learner is a decision tree, also known as a base learner to generate a strong rule[30]. So, multiple weak learners(decision trees) are built for the same training data sets in each iteration. And combines each learner to form a strong rule and predictions from the multiple decision trees are aggregated for a regression problem.

(a) **Gradient Boost**

Gradient boost is a boosting technique and developed from decision tree models. The tress are added sequentially based on the residuals from the previous tress and fit until residual errors are minimized. This algorithm uses a gradient-descent algorithm, so the name Gradient boosting and the model is fit until the gradient is minimized. The below figure explains the overview of an algorithm,

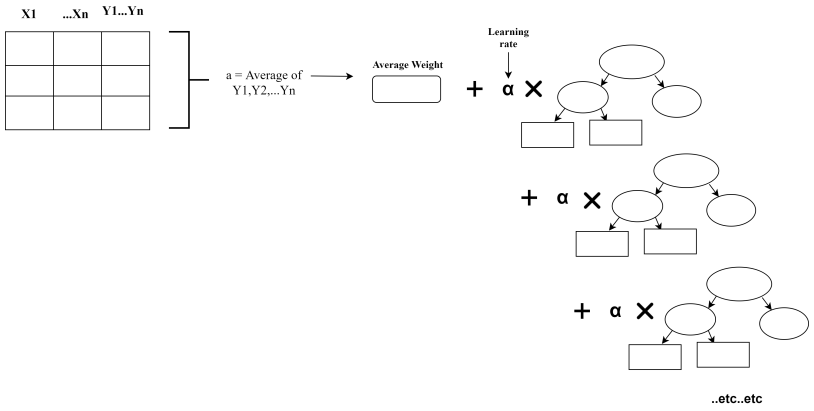


Figure 2.13: Gradient Boost Regression algorithm workflow

The steps involved in GradientBoost regression algorithm for predicting **y** from training data set as follows,

i. Firstly, the average of **y** variable values is calculated as shown in Figure 2.13, considering that all values of **y** variable is equal to mean.

ii. The gradient boost algorithm builds a tree based on errors i.e. Residual Error $(y_i - \bar{y})$ from previous tree. And prediction is performed for every data set by considering average of **y** variable values calculated at first and decision tree with multiplication factor α or learning rate as in Figure 2.13. This learning rate solves the problem of low bias

and high variance issue by scaling a tree such that taking small steps results in better predictions with test data set i.e Low variance.

iii. The Residuals are calculated for each training set by taking the difference between observed weight calculated from step1 and latest predictions from the tree built. If the residuals obtained are smaller than previous, then acclaimed for proceeding in the right direction.

iv. The new tree is built with the new residuals and combined with the previous tree and average value calculated at the initial step. All the trees are scaled with appropriate learning rate and add everything together. The predictions are made with the training data set and residuals are calculated as previous step. The tress are built till the residual error has reached minimum and stable.

The final outcome of the model is represented mathematically as follows,

$$F(x) = h_0(x) + \alpha_1 h_1(x) + ... + \alpha_n h_n(x) \tag{2.9}$$

$$F(x) = \sum_{i=1}^{n} \alpha_i h_i(x) \tag{2.10}$$

where, $h(x)$ = cost function and α = learning rate The hyper-parameters used for gradient boosting regressor are similar to Random Forest regressor with additional α which is called as learning rate.

(b) **XG Boost**

The XG Boost is similar to the Gradient boosting algorithm but uses a unique regression tree. This algorithm was designed to be used with large, complicated or unstructured datasets on classification and regression modelling problems. Like in Gradient Boosting algorithm, the residuals are calculated first and Unique XGBoost trees are built on those values. The steps involved while building XGBoost trees for regression are,

i. Firstly, the similarity scores are calculated for Root node and leafs. The root node contains all the residuals and consider threshold ('X' limit) to split the node into two groups containing residuals. And later, Gain is calculated to the root node to determine the splitting of data with optimal threshold. The similarity and Gain are calculated as follows,

$$Similarity\_score = \frac{Sum of residuals squared}{Number of residuals + \lambda} \tag{2.11}$$

$$Gain = Left\_leaf_{similarity} + Right\_leaf_{similarity} - Root\_node_{similarity} \tag{2.12}$$

ii. The tree is pruned[2] by calculating the difference between Gain values and a user defined complexity parameter, γ(Gamma).
   - The tree is not pruned, if the difference is positive
   - The tree is pruned, if the difference is negative

iii. Lastly, the output values are calculated for the remaining leafs after tree is built.

$$Output\_value = \frac{Sum of residual squared}{Number of residuals + \lambda} \tag{2.13}$$

Here, λ is a regularisation parameter and when $\lambda > 0$, results in more pruning, therby shrinking the similarity scores, and results in smaller output values in the leaves. So, the complete tree is built considering the hyper-parameters namely λ, γ and learning rate η , which is same as α in Gradient boosting algorithm.

(c) **Histogram-based Gradient Boosting Regression Tree**

Histogram-based gradient boosting technique is an ensemble method built based on gradient boosting regressor algorithm. Gradient boosting performs well for structured predictive modeling problems. Even though the gradient boosting algorithm performs well in practice, the training process slows down because trees need to be created and added sequentially.

Histogram Gradient boosting regressor method improves the training process by decreasing the number of values by discretizing output variables especially for regression problems into fixed groups. The discretization method converts the output variables which are of different distribution(skewed data set) to a continuous probability distribution (normal distribution) and eventually reduces the number of essential values for each feature from thousands of down to hundreds of or tens of. Therefore, the decision trees are built on discrete or a continuous probability distribution dataset instead of discrete values in the training set.

The histogram represents the discretization of data which are used in the construction of trees, so it is intuitive to refer gradient boost algorithms as histogram-based gradient boosting regression tree. Hyper-parameters used are similar to gradient boosting regression with an important additional parameter named `l2_regularisation`[21], which helps in generalizing the model to overcome overfitting.

[2]Pruning is a method to overcome overfitting a regression tree to training data. This is achieved by removing some of the leaf nodes and replace the split with a leaf that is average of large number of Y variable values. Leaf nodes are removed until the minimization in the cot-function has reached on the test set

2.5.4 Gaussian Process Regression (GPR)

Gaussian process is a distribution over the function $f(X)$ defined by a mean value $\mu(X)$ and a covariance function $\Sigma(X, y)$ [31]. The GPR can be represented in a equation as in 2.14 where GP is the defined GPR.

$$f(X) \sim GP(\mu(X), \Sigma(X, y)) \tag{2.14}$$

Gaussian process distributions over a defied function can be used like regression models. Gaussian process defined by the kernel function can be used to create a distribution of the given data. This distribution can then be used to predict values of output **y** based on the given input **X** [34].
A covariance function is an essential part of the Gaussian process model as it determines the continuity properties of predictions. Covariance functions are also known as kernels. Some examples of kernels used are,

1. **Rational Quadratic**
   Rational Quadratic kernel is similar to the addition of several squared exponential kernels of various length scales, which determines the range of the Gaussian process. The parameter α in the equation of the kernel regulates the relative weight of the variations. The Gaussian process for using this kernel varies very smoothly in the ranges defined.[35]. in the equation X is the inputs and y is the outputs. σ is the variance, l is the length scale, bigger the value of l^2 less complex the fit. α is the scale mixture.$d()$ function is the euclidean distance.
   $$k(X, y) = \sigma^2 \left(1 + \frac{d(X, y)^2}{2\alpha l^2}\right)^{-\alpha} \tag{2.15}$$
2. **Exponential**
   This kernel raises a scalar input p to the base kernel as represented in the above equation.
   $$k_{exp}(X, y) = k(X, y)^p \tag{2.16}$$

2.6 Finite Element Analysis

Finite Element Analysis (FEA) is the representation of a physical occurrence such as a crash scenario using numerical technique called Finite Element Method (FEM). Using FEA, complex situations such as in crash are being studied in detail and can be used for the development of safety systems.

2.6.1 Human Body Models for Injury assessment

Human Body Models (HBMs) are numerical tools used to simulate occupant responses in crash scenarios. To develop safety systems it is important to understand the injury mechanisms. Injury criteria are used to develop safety systems in vehicles. As compared to physical crash test dummies, HBM is sensitive to different loading directions similar to a physical occupant. HBM's can represent different

occupant sizes, gender, anthropometry. HBM's are essential tools for virtual testing and verification within traffic safety research to improve and evaluate real life safety systems.[10]

2.6.1.1 Hybrid III M50 Fast Model

Hybrid III M50 Fast Model is a semi-deformable FE model. The name fast represents that the model is inexpensive due to the presence of fewer elements. The model used in this project represents a 50^{th} percentile male dummy, 175.26 cm tall, and has a mass of approximately 78 kg. Due to the less computation time, this model was used to run simulations to generate data to train ML models.

2.6.1.2 SAFER HBM v9

The model used in this study is Version 9 of the SAFER HBM representing the 50^{th} percentile male. This model contains increased anatomical detail and improved biofidelity of material models representing the human occupant in more detail. Version 9 has an improved generic ribcage representing a human male [11]. SAFER HBM v9 is used to study injury and injury mechanisms in greater detail compared to the Hybrid III M50 Fast Model.

2.6.2 Injury criteria and Injury risk

Traumatic injuries occur when the mechanical tolerance limits of the body are exceeded. Injuries are generally believed to result from excess strain induced by direct or indirect loading. Injury criteria are the probability of trauma to mechanical parameters which can be measured using instrumented crash test dummies or cadavers, or evaluated in a numerical model. Without injury criteria, the severity of trauma in crash reconstruction cannot be evaluated [14].

Injury risk is the measure of the likelihood of an occupant to be injured during a crash. It is defined in terms of probability or percentage as the number of crashes divided by the size of the population [41].

Head Injury Criteria (HIC) is the measure of the likelihood of head injury occurring due to head impact during a crash. It calculated using the head acceleration measured at the center of mass of the crash test dummy or a FE HBM. In the equation to calculate the HIC, equation 2.17 $t_1 and t_2$ are the initial and final time used to calculate HIC.

$$HIC = max\left[\frac{1}{t_2 - t_1}\int_{t_1}^{t_2} a(t)dt)\right]^{2.5}(t_2 - t_1) \tag{2.17}$$

2.6.3 Application of FE in ML

The crash reconstruction parameters influence the injury to the occupant. These crash reconstruction parameters are used as input parameters and the injury criteria, injury risk, and injury matrices are the outputs. An ML model generalizes the

relation between a set of input and its output. To develop a surrogate of the FE-HBM to predict injury, reconstruction parameters are the inputs and the injury predictions are the outputs. A well established, generalized relation between the two is set up using various ML algorithms.

2.6.4 Softwares Used

In this thesis, LS-DYNA (R11.1) was used to numerically solve the model. The parameter sampling by distributing the parameters in a design space was done using `dynakit`, a tool developed in this thesis to structure the design process in a design space. `dynakit` is developed in python(3.8.5) scripting tool and this library is available in package installer for Python. The post processing of the simulations was done in LS-Post processor, `Lasso-python`(1.5.0) python library, and META post processor (v 20.1.1).

3

Surrogate Models for Dummy Responses

In this chapter, surrogates to a human body model dummy were evaluated by exploring ML pipeline comprising data preparation, model validation, and optimization for HIC15, maximum chest deflection, maximum Head acceleration, and maximum Neck force during a crash. Further, the number of simulations were evaluated for accurate surrogates for two data sets generated from the Hybrid III M50 fast model.

3.1 Methods

Surrogate models were built using the data generated from FE simulations. The data was analyzed and prepared to train the ML models. Later, the models were tuned and optimized to get accurate predictions and finally evaluated using error metrics. The workflow is explained in detail in the below sections.

3.1.1 Generate Data

The data was generated using a fast hybrid III model for its short computational run time. The model used in this study represents the 50^{th} percentile male population and is in a seated position at the driver's end. Various factors influence injuries during a crash, and these factors are referred to as FE parameters. Two categories of data were generated based on FE parameters to investigate its influence on ML model performance.

1. **Restraint Parameters**
   This consists of parameters influencing the restraints in the vehicle to protect the occupant. The parameters included are belt pre-tensioner trigger time, max belt pre-tensioner force, seatbelt load limiting force, airbag peak pressure, airbag pressure leakage rate scale factor, airbag trigger time, and steering column compressive force.
2. **Crash parameters**
   This consists of parameters influencing the crash situation. The parameters included are delta velocity, angle of impact during a crash (PDOF_H), and crash pulse duration which is the time period of the acceleration pulse acting on the vehicle. A shorter pulse duration results in higher impulse force on the occupant.

Injury criteria and kinematics related to injury outputs are generated from simulation in LS DYNA. Four outputs were selected to evaluate the ML models,

1. **HIC15**

   HIC15 is the measure of HIC over 15 ms. Lower HIC15 values may result in minor concussions and the higher values may result in severe head injuries.

2. **Maximum head acceleration**

   Maximum head acceleration is the peak acceleration measured at the center of mass of the head.

3. **Maximum chest deflection**

   Maximum chest deflection is a measure of rib cage compression during a crash. In the fast hybrid III model, a spring damper is designed near the thoracic area which captures this measure. Higher chest deflection might result in fractured ribs and might result in injuries to internal organs.

4. **Maximum Neck force**

   Maximum Neck force is the measure of the maximum resultant force acting on the neck. It is recorded approximately near the C3 and C4 Cervical bones of the spine in the fast hybrid III model. A high value of neck force corresponds to the possibility of injuries to the neck.

There are many variations in FE parameters during a crash. To develop a surrogate model, the data to train the model should be distributed in a design space that envelops all the FE parameter variations. The study conducted by Iraeus J and Lindquist M [12] presented ranges of FE parameter for 90^{th} percentile of real-life crashes. These ranges were used to generate data to evaluate the ML model in this study. The values in Table 3.1 were used to uniformly distribute FE parameter values in a design space.

Table 3.1: Parameter ranges for the uniform distribution for sampling and generation of data

Parameters	Unit	**Minimum**	**Maximum**
Pretensioner trigger time	[ms]	5	15
max pull-force of pretensioner	[KN]	0.99	2.5
Load limiter force	[KN]	2.25	5.85
Airbag pressure	[KPa]	111	170
Airbag leakage Scale factor	[-]	0.5	1.5
Airbag trigger	[ms]	16	23
steering column compressive force	[KN]	3	6.6
Delta Velocity	[Km/h]	40	60
Pulse duration	[ms]	77.2	142
Angle of impact	[degree]	-10	10

3.1.2 FE Simulations

Design of Experiment (DOE) was performed for the FE parameters to generate training data to be uniformly distributed in the parameter range space. DOE was conducted on the FE input parameters using Latin hypercube sampling with uniform distribution. A workflow was developed for parametric simulations and packaged as a python library called ***dynakit***. The parametric key files were solved using LS DYNA. Maximum head acceleration, maximum neck force, maximum chest deflection, and the HIC15 values were post-processed using META post to evaluate the ML model. Finally, the post-processed data was assembled with the FE parameters into a single data set.

3.1.3 Exploratory Data Analysis

The output data generated from FE simulations could be noisy, skewed, or have some outlier points. Statistically, outliers are those points that do not belong to the majority, but in FE simulations some meaningful data points may not belong to the majority group due to the combination of input parameters causing odd data. Such data points are recognized and analyzed.

Figures in appendix A.1 and A.2 are used to check the data for any kind of abnormalities like skewness. Since the data used during the study was generated in parameter space with uniform distribution, outputs were evenly spread. The skewness of the data was checked using a KDE plot which represents the distribution of the data as seen in Figure 3.1 and Figures 3.2. The skewness value between $[-1, 1]$ [39] of the training data considered to be acceptable for ML.

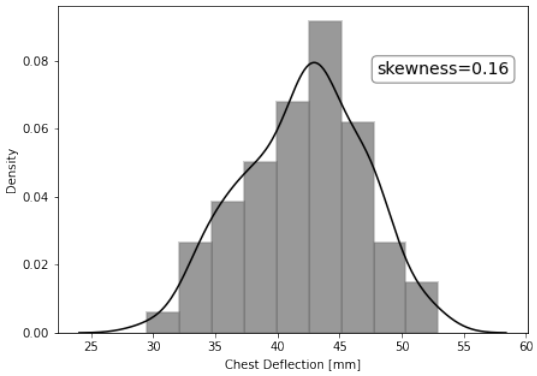


Figure 3.1: Distribution of the dummy chest deflection response in the simulations where 7 parameters in the restraint system were varied.

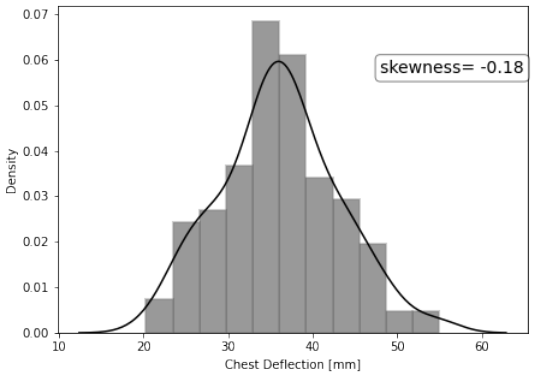


Figure 3.2: Distribution of the dummy chest deflection response in the simulations where 10 parameters in the crash and restraint system were varied.

3.1.4 Build a Machine Learning model

The analyzed data was divided into a test set and train set using a module from sklearn [21]. 20% of the data was used for testing and the rest was used to train using k-Fold CV. Figure 3.3 was used to evaluate the even split of the data into test and train. 10 fold training was implemented to train the model using 108 samples. The training and the testing errors were evaluated using Mean absolute error (MAE). MAE was used instead of root mean square error (RMSE) because RMSE penalizes the error more than MAE. A single wrong prediction can get the mean error to increase more compared to MAE. So, to evaluate the performance of prediction, MAE is a better criterion.

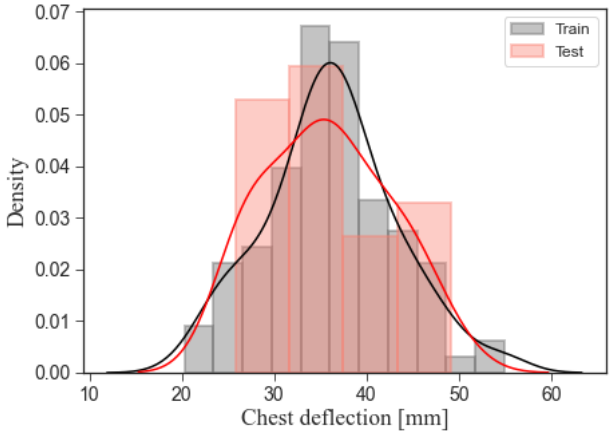


Figure 3.3: Distribution of the chest deflection values when split into test and train. The values are plotted against the probability density of the value.

ML models listed in Table 3.2 were used to train and prediction accuracy of each was compared. The hyperparameters of each model were tuned using random search and grid search to find the most suitable model for the selected outputs. The final tuned models are tabulated in Appendix section A.3.

Table 3.2: library in python from which the ML model is used and its version

ML Method	Python package	Version
Lasso regressor	sklearn.linear_model.Lasso	scikit-learn version 0.23.2
Ridge regressor	sklearn.linear_model.Ridge	scikit-learn version 0.23.2
Decision Tree regressor	sklearn.tree.DecisionTreeRegressor	scikit-learn version 0.23.2
Random Forest regressor	sklearn.ensemble.RandomForestRegressor	scikit-learn version 0.23.2
Gradient boost regressor	sklearn.ensemble.GradientBoostingRegressor	scikit-learn version 0.23.2
Histogram based gradient boost regressor	sklearn.ensemble.HistGradientBoostingRegressor	scikit-learn version 0.23.2
XGBoost regressor	xgboost.XGBRegressor	xgboost version 1.4.2
Gaussian process regressor	sklearn.gaussian_process.GaussianProcessRegressor	scikit-learn version 0.23.2

The model performances were compared using bar plots of test and train MAE. Residual plots of the test set were used to evaluate the predictions of the test set

for an ML model. ML model was trained with various training sizes and tested, the outputs were plotted to determine the most suitable model for the number of training samples available.

3.1.5 Numerical Noise estimation

Computational models like FE models while simulating may have numerical noise because of round-off errors in numerical solvers, nonlinear contacts defined in the model, control settings for the solver, etc. So, this numerical noise may possibly affect the results accuracy of surrogates. It is necessary to estimate this value in order to draw any conclusions regarding the performance of the ML model.

To estimate the value of numerical noise, the variation in output for small percentage change in input has to be evaluated. The change should not be too big as it might capture the true data. Too small change will fail to capture the noise. For the study $\pm 0.5\%$ change is selected. The input FE parameters are uniformly distributed within $\pm 0.5\%$ change from the mean. Total of 60 samples were simulated and post processed. The average difference of the values from the mean was calculated and used to evaluate prediction results for different models.

3.2 Result

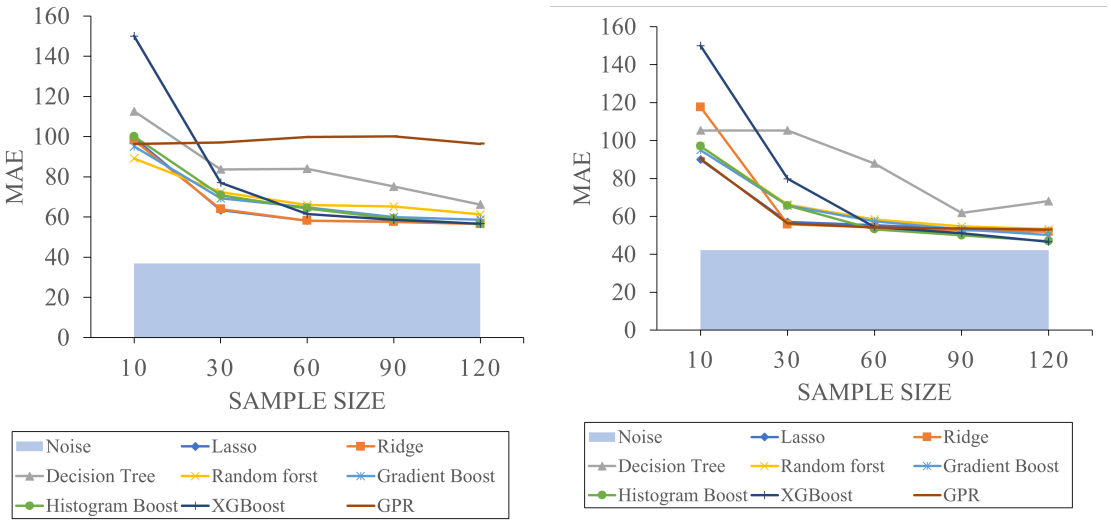
The performances of the ML models for two different sets of simulation data in predicting outputs are illustrated. Both the data sets have a total of 130 samples. Before comparing the prediction accuracy of the models, the magnitude of numerical noise was estimated. The range to evaluate the noise was selected for HIC15 values between 300 to 400, as these values had relatively higher prediction residuals. The noise was estimated based on data distribution plots in Appendix A.1 where the mean of all the outputs considered to be the true value. The mean difference of the values from the true value is calculated and listed in Table 3.3.

Table 3.3: Numerical Noise estimates for the generated samples

Measure	**Unit**	**Numerical Noise Estimate**	
		Restraint FE parameters	**Crash and Restraint FE parameters**
HIC15	$[-]$	36.927	42.182
Maximum Chest Deflection	$[mm]$	0.749	0.673
Maximum Head Acceleration	$[m/s^2]$	0.055	0.063
Maximum Neck Force	$[KN]$	0.069	0.068

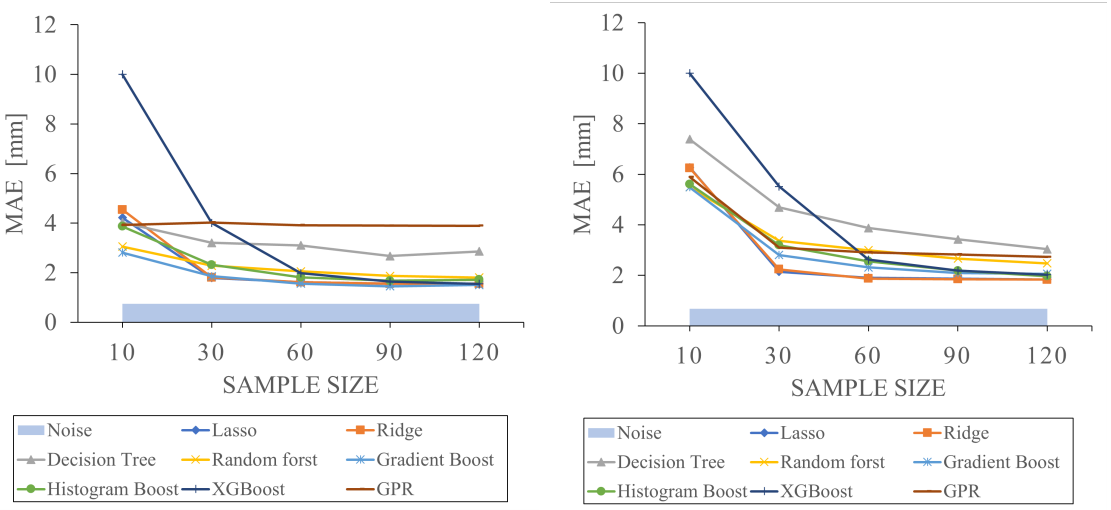
Test MAE curves indicating the MAE of predictions for different training sample can be used to select suitable models. Random forest regressor performed better than the other models in predicting the values of HIC15 when trained for 10 samples of data with 7 input FE parameters as seen in Figure 3.4a. With the increase in samples, linear models namely Lasso and Ridge perform better until 90 training

samples. Further increasing the samples to about 120, tree-based boosting methods outperform in predicting accurate results. Similarly, When the input FE parameters in the data are increased to 10, resulted in giving similar results as shown in Figure 3.4b.



(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure 3.4: Test set MAE for HIC surrogate models with respect to sample size

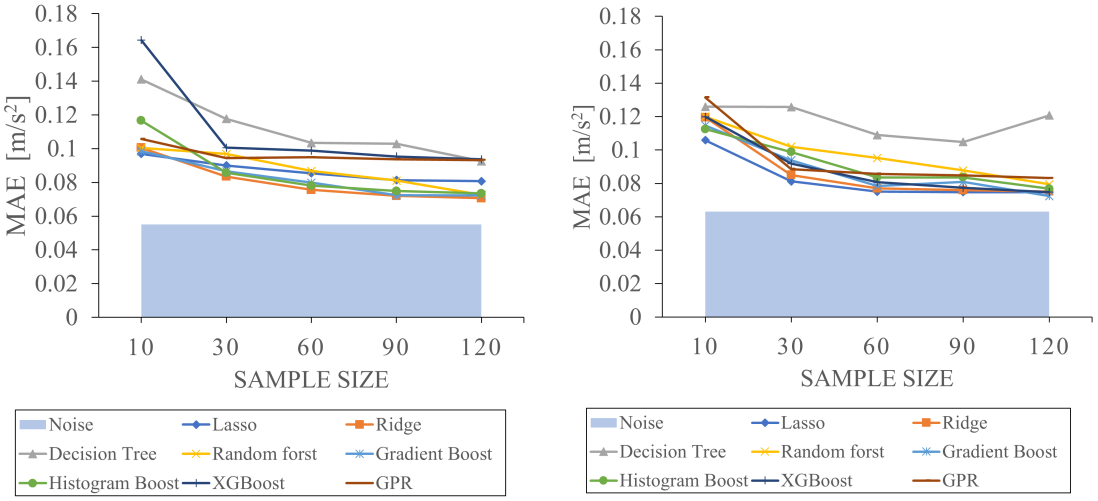


(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure 3.5: Test set MAE for maximum chest deflection surrogate models with respect to sample size

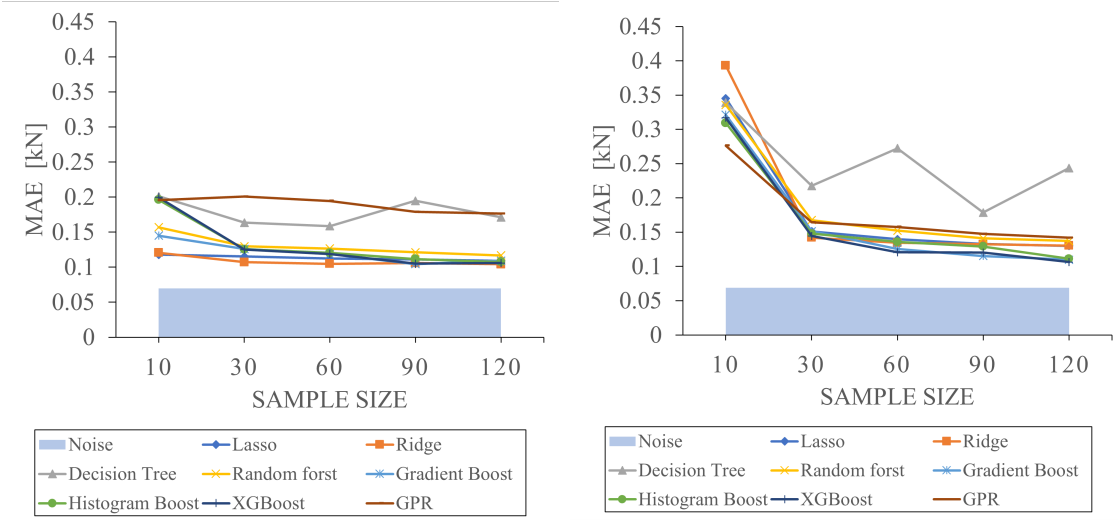
The test MAE reduced with increase in sample size for all the output measures. However, for Maximum chest deflection (Figure 3.5) , XGBoost does not predict good results for fewer training samples. When up to 90 training samples are available, linear models are better to predict maximum head acceleration (Figure 3.6).

Tree-based boosting methods predict maximum neck force (Figure 3.7) better than the other models when 90 - 120 training samples are available. These curves can be used for selecting suitable ML models for all outputs based on the training data available and the accuracy required.



(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure 3.6: Test set MAE for maximum head acceleration surrogate models with respect to sample size



(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure 3.7: Test set MAE for maximum neck force surrogate models with respect to sample size

The final tuned model as presented in Appendix Table A.8 was used to predict

HIC15 values. The training and testing MAE for the ML models considered were near to numerical noise estimate, except for decision tree model which resulted with high test error as shown in Figure 3.8. Among the ML models considered, Tree-based boosting methods like Gradient boost, XGBoost, and Histogram-based gradient boosting outperform the other in predicting HIC15 values. When compared with other boosting methods, the training error for XGBoost was proximate to the estimated noise.

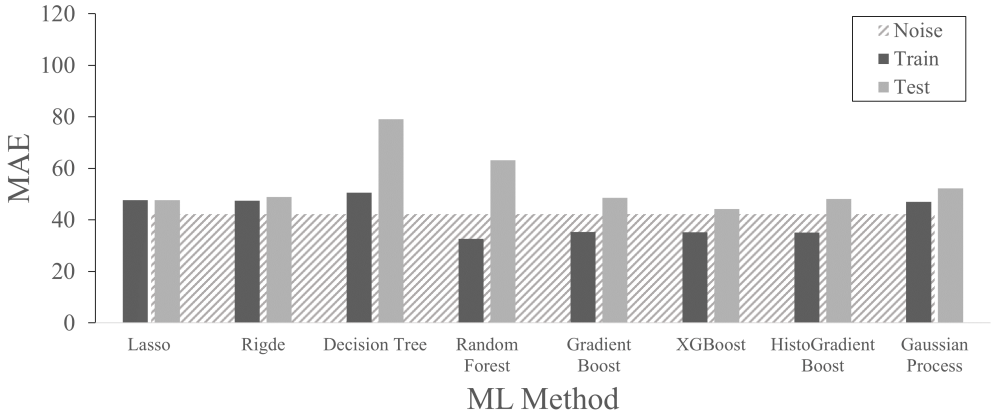


Figure 3.8: Model comparisons for test and train MAE of predictions using bar graph of HIC15 with noise estimate for crash and restraint parameters

The learning curve for XGBoost in Figure 3.9 shows the comparison of the training and cross-validation errors. The training error is within one standard deviation value of MAE for testing. It implies that the model would not be highly biased or have high variance.

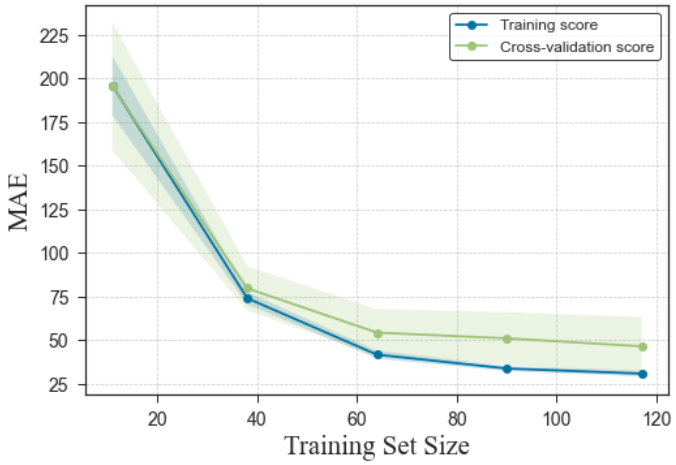


Figure 3.9: Learning Curve for XGBoost to predict HIC15 for data with Crash and Restraint FE parameters

Prediction accuracy of ML model was evaluated using residuals. The residuals for the test set are plotted against the predicted values as shown in Figure 3.10. The

average value of the residuals is around 42.182. The residuals for testing are not high compared to the test residuals, which indicates the model does not overfit. One bad prediction around 270 influenced the MAE of testing. Increasing the number of training samples will improve this.

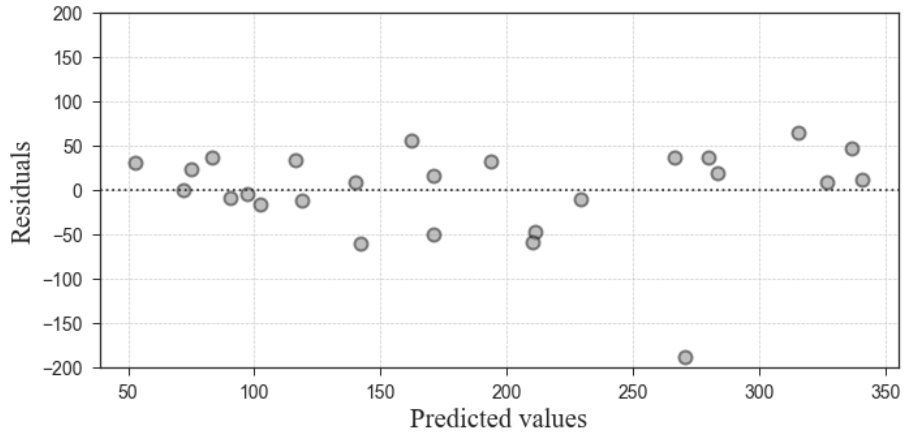


Figure 3.10: Residual plot for XGBoost to predict HIC15 for crash and restraint parameters

The presented workflow for selecting and evaluating the model was applied to other outputs too. The comparison graph, the learning curve for the most suitable model, and the residual plot that illustrates the prediction performance for the other output measures are provided in Appendix A.2 for the two data sets.

3.3 Discussion

ML model performance in predicting output measures depends on many factors, including the number of input variables, the size of training data, type of distribution, the level of numerical noise, and the algorithm type. The selection of the ML model based on the availability of training samples and the required prediction accuracy for two categories of data sets was evaluated with test MAE plots.

Since a uniform sampling distribution was applied to a defined range, the data generated had no skewness. In all evaluations, the output measures were evenly distributed, thus no transformations were necessary. This has also been verified, as the transformation implementation did not improve the prediction accuracy of all evaluated models.

3.3.1 Estimating the number of simulations needed to obtain accurate models

ML models (linear and nonlinear) were extensively tuned, trained, and tested to achieve accurate predictions. The suitability of a model in predicting an output

measure is affected by the number of training samples and the number of FE parameters. Sometimes, high complex models are not suited to predict outputs compared to simpler models. Complex models (tree-based models and GPR) typically have more effective hyperparameters than simple models (linear models). This because models undergo overfitting by learning well with training data rather than generalizing. Lasso and Ridge, when trained with 30 - 90 samples, performed consistently better than other models in predicting all outputs.

HIC15 predictions showed nearly equal MAE for test and train for all the models except Decision Tree in Figure 3.8, indicating that these models do not overfit or underfit. All the training MAEs are near the estimated noise, which implies the models do not learn too much noise. As the Decision Tree model builds a single tree, it might overfit the training data. Random Forest, which has the option to increase the number of trees, *n_estimators* can be used to decrease overfitting. But the testing MAE is considerably more compared to the training error. In tree-based boosting, the MAE is reduced by using hyperparameters such as regularisation parameters to overcome overfitting. As a result boosting methods like Gradient boost, XGBoost, and Histogram-based gradient boost have lower prediction MAE for testing compared to the others. Among the three methods, XGBoost especially has the lowest test error. This indicates that the model predicts the HIC15 values generalizing the data without learning the numerical noise. To further evaluate the XGBoost model, the learning curve and residual plot are studied. As we see in the learning curve in Figure 3.9 the training error is within the standard deviation of the cross-validation error. The curve also indicates the model has less variance and bias, implying the model does not underfit or overfit.

Residuals in Figure 3.10 illustrate that the residuals are not consistently high. In the figure, the residual for one of the predictions is very high, this can be reduced by training the model with more samples. From the learning curve, by increasing the number of training samples, the training and cross-validation MAE can be decreased. However, the model might end up learning the noise in the data as the training error with the current sample size is at the limit of estimated noise. Similarly, the model selection procedure and inferences can be followed for all output measures.

Tree-based models and GPR are prone to overfitting with fewer samples because the models learn the training set too well and fail to generalize. These models are not suitable to predict HBM output measures for small training sizes. And for larger training samples, preferably from 105 to 120, tree-based boosting methods are consistently good in predicting near the limit of all output measures.

3.3.2 Numerical Noise

The numerical noise in the FE simulation data needs to be estimated before comparing the prediction MAE of various ML models. A variation of $\pm 0.5\%$ in input FE parameter from the mean values resulted in 17.5% variation in HIC15 output, this variation is referred as numerical noise. Numerical noise was estimated by changing the input FE parameters by a small amount and checked for the variation in the output. The small amount is not standard, as if this is too big might end up

considering true values as noise. But, if the amount to change is too small might fail to capture the noise. ±0.5% change was used for the study, as this value might not be true, so it has referred to as estimates. The noise estimated in was varied by ±0.5% around the values of FE parameters resulting HIC15 value between 300-400. Varying it around other values can have a different estimate so the estimated values are only an approximate measure. For a more accurate value for estimating, the numerical noise should be calculated for all the measures separately.

3.4 Conclusion

The prediction accuracy of different ML models with an increase in sample size was evaluated using test error curves for each output. In general, test MAE of all ML models showed a decreasing trend with an increase in training sample size. Model is defined as accurate when the change in error percentage is not significant with an increase in training samples, i.e the accuracy saturates over increase in training samples as the model has reached the error limit. The sample size required to attain accuracy was different for each output. For maximum chest deflection, a minimum of 60 training samples was required to get accurate results. With an increase in samples size from 60 to 108, the percentage change in test MAE curve slope is about 0.75%. Decreasing the training sample size from 60 to 30 increases the MAE by 2%. Similarly, the minimum number of samples required to get accurate ML models for each output are listed in Table 3.4.

Table 3.4: Change in percentage of test curve slope with respect to sample size

Output Measure	Number of training samples	% Change in MAE with further increase in samples
Maximum chest deflection	60	0.75
HIC15	90	4.90
Maximum head acceleration	60	0.01
Maximum neck force	30	0.02

Among the evaluated models, the suitable models in predicting outputs observed to tree-based boosting methods. The suitable model for each output measure is summarized in Table 3.5. In the process of model evaluation, numerical noise found to be an important factor for selecting ML models. The numerical noise in the training data has to be estimated to make better inferences of the results.

Table 3.5: Suitable ML models for outputs

Output Measure	Restraint parameters	Crash and restraint parameters
Maximum chest deflection	Histogram based gradient boostin	Gradient boosting
HIC15	Gradient boosting	XGBoost
Maximum head acceleration	Histogram based gradient boostin	XGBoost
Maximum neck force	Histogram based gradient boostin	Gradient boosting

4

Surrogate Model for Human Body Model

Surrogate models to SAFER HBM are built in this chapter. ML pipeline were implemented to determine how many simulations are required to predict accurate outputs. The ML pipeline include data preparation, model validation, and optimization. The models were trained for outputs namely, HIC15, risk of rib fracture, maximum head acceleration, maximum chest deflection, maximum pelvis acceleration, and strains on all 24 ribs.

4.1 Available Data

The study used data from Iraeus J., and Pipkorn [11], which used SAFER HBM v9 FE model to evaluate occupant response. The data had 1000 crashes reconstructed using the parameters which are sampled based on distributions derived from real-life crashes. Latin hypercube sampling was used to sample the data. Surrogate models are built based on input parameters and post-processed outputs, as shown in Table 4.1.

Surrogate models were training for two kinds of output measures, HBM responses, and injury metrics. Responses include maximum head acceleration, maximum pelvis acceleration, the peak rib strains for all the ribs, and the chest compression value. Injury metrics HIC15 and the risk percentage of two or more rib fractures for a 45 years occupant were computed using a Poisson-binominal distribution function of the maximum rib strains [42].

From the available data, the output variables selected as mentioned in Table 4.1 are extracted and combined into a single data set. Approximately 0.5% pf the simulations had an error termination due to numerical issues during solving. These samples were recognized and excluded from further studies. Linear regression models like lasso and ridge, tree-based regression models like the Random Forest, Gradient boost, XGBoost, Histogram-based gradient boosting, and Gaussian process regressor was used to develop ML models for all the selected outputs. The performance of the different models was compared to find the best-suited model for each of the outputs.

Table 4.1: FE parameters in the available data which was used as the input to develop the ML and the outputs based on kinetics like max head acceleration, max pelvis acceleration, Dmax and the rib strains. Injury criteria HIC15 and rib fracture risk

FE Reconstruction parameters	**Postprocessed outputs**
Friction between occupant and airbag	HIC15
Friction between occupant and seatbelt	Dmax
Seatbelt loadlimiter force	Max pelvis acceleration
Delta Velocity (in km/h)	Max Head acceleration
Pulse duration (in ms)	Risk of 2+rib fracture 45yrs+
Horizontal pulse angle (PDOF_H)	24 Max rib strains
Airbag peak pressure (in kPa/100)	
Radii of driver airbag	
EV1 to EV3 (defines the pulse shape)	
Trig time for driver airbag (in ms)	
Friction between occupant and instrument panel	
Linear stiffness of knee impact area (in kN/100mm def)	
Seatbelt pretension force (in kN)	
Deformation force for lower part of rim (in kN)	
Friction between occupant and seat	
Slipring friction	
Steering wheel angle (in degrees)	
Steering column axial collapse force (in kN)	
Steering wheel longitudinal adjustment (in mm)	
Yaw scale factor	

4.2 Methods

A similar procedure as that used in chapter 3 was used in developing the surrogate model. The data structure was evaluated as the first step of the process. ML models do not perform well with skewed training data, so any abnormalities in the data were checked. The output measures in the available data were skewed as shown in Table 4.2 and Appendix Table B.5. Before training the model, transformations are used to correct the skewness.

Table 4.2: Skewness of the output data and the transforms used to correct it.

Parameter	Initial skewness	Final Skewness	Transformer
Dmax	0.468	[-]	[-]
Max pelvis acceleration	1.315	0.000	quantile
Risk of rib fracture	3.533	0.534	logit
HIC 15	6.646	-0.002	quantile
Max head acceleration	2.095	0.000	quantile

ML models listed in Table 3.2 were used to train surrogate models and compare the performances. 20% of the 1000 samples was used as the test set was later used to evaluate the model performance. The remaining 795 samples were trained using

the 10-fold training technique. The ML models were tuned using random search and grid search, and the final tuned models for all output measures are presented in Table B.10.

MAE for the test set, which compares the predictions of the ML models, is evaluated for different training sample sizes to find the number of simulations needed to train accurate models. The performance of the best-suited model was further evaluated using learning curves and residual plots for all measures.

According to the studies in chapter 3, simulation data will have some numerical noise during FE simulations. To draw more meaningful conclusions from the results, the magnitude of numerical noise was estimated. 64 simulations were performed with input FE parameters changed by $\pm 0.5\%$ from their mean values. The mean variation of the results from the 64 simulations were used as the numerical noise estimate.

4.3 Results

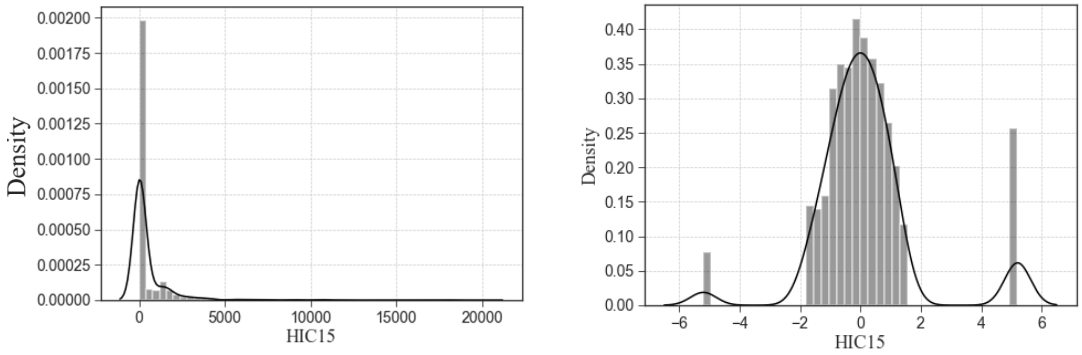
In this section, we evaluate the performance of ML models in predicting HBM responses and injury metrics for different training samples sizes to find the most suitable model for a given training size. The used data had 21 input FE parameters and 1000 samples based on distributions extracted from real-life crashes. A total of six simulations failed to complete because of error terminations. These were excluded from further analysis. The magnitude of the numerical noise was estimated for all the outputs from the distribution of output measures and is tabulated in Table 4.3. The plots of the distribution are included in Appendix B.1.

Table 4.3: Estimated numerical noise for SAFER HBM

Measure	**Units**	**Numerical Noise Estimate**
HIC15	$[-]$	6.443
Rib fracture risk probability	$[-]$	0.027
Maximum Chest Deflection	$[mm]$	0.294
Maximum Head Acceleration	$[g]$	6.009
Maximum Pelvis Acceleration	$[g]$	1.002
1^{st} Left Rib strain	$[\%]$	0.173

The available data was skewed, therefore appropriate transformation methods, such as logarithmic, logit, and quantile, were implemented to correct it. The HIC15 data had very high values that would have been caused due to situations not occurring in real-life crashes, but rather a theoretical case. To inspect this extremity, the peak value of HIC15 in the available data was used i.e. 20008. The crash animation was analyzed to find the reason for the values being over 20000 and observed that the forces acting on the occupant were not physical. Poor parameter combinations like high crash pulse, low load limiter force, late trigger time of the airbag, rigid wind screen, and high relative velocity caused this condition. From an injury point

of view, AIS 3+ injuries are severe and difficult to medicate. By referring to the HIC15 risk curves as a function of AIS level determined by the study from Hayes and Kuppa (2004) [40], values of HIC15 above 2000 results in 100% possibility of AIS3+ injury. So, the HIC15 data is limited to 2000. After limiting the maximum values to 2000, the data had positive skewness (right-skewed). Quantile transformation was used to convert skewed data to a normal distribution as shown in Table 4.2. Figure 4.1a represents the initial data distribution and Figure 4.1b represents the distribution after transformation. A similar procedure was followed to analyze and correct skewness in the data for all output measures.

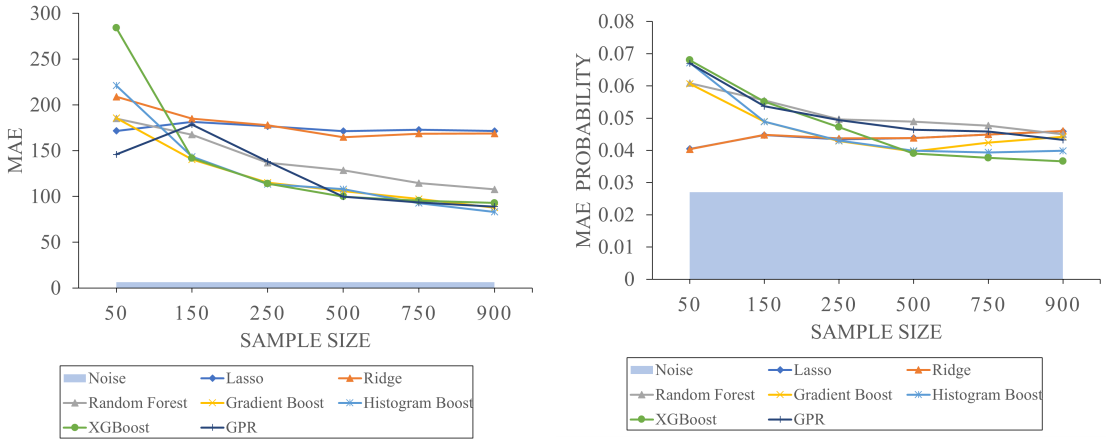


(a) Distribution of raw data from FE simulations

(b) Distribution of data after using quantile transformation

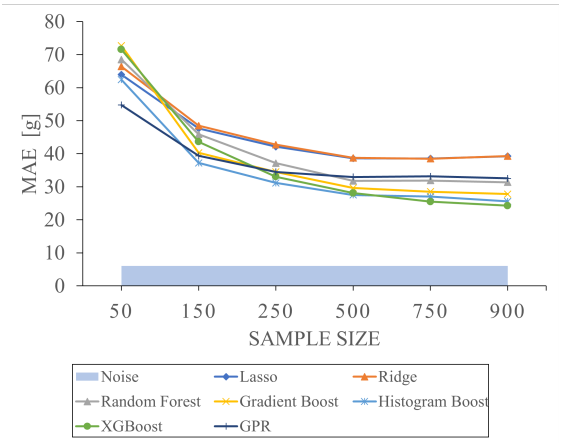
Figure 4.1: Data preparation for HIC15 raw data

Test MAE curves for the output measures were used to determine the number of training samples needed to get models with a specified accuracy. A suitable model was selected using these curves based on the available training size.

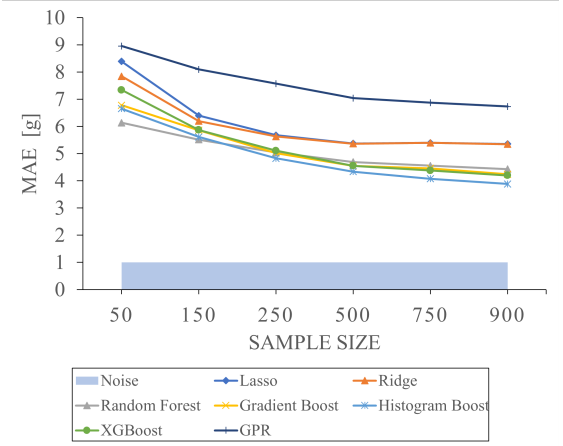


(a) HIC15

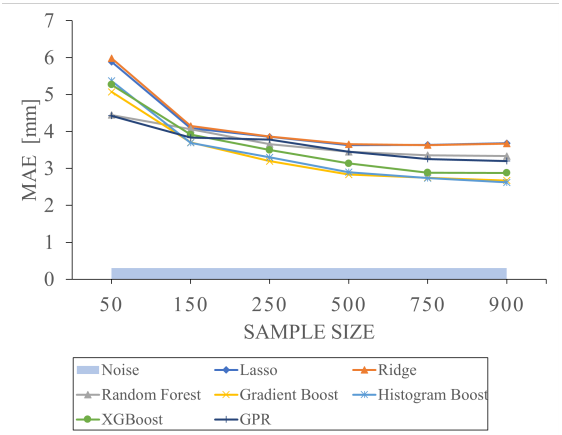
(b) Rib fracture risk



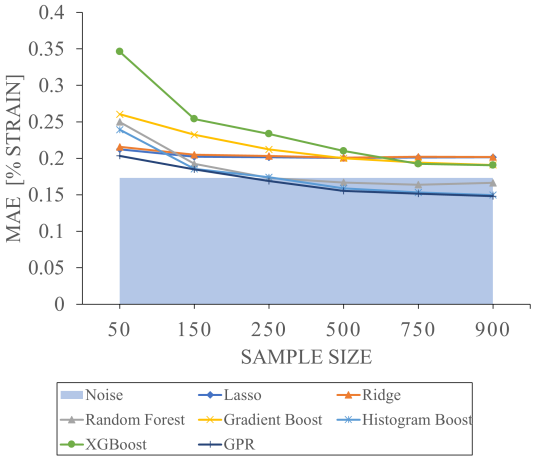
(c) Maximum head acceleration



(d) Maximum pelvis acceleration



(e) Maximum chest deflection



(f) 1^{st} Left Rib strain

Figure 4.2: Test MAE curves for model comparison when trained with different size of training data

When 795 training samples are available, the most suitable model to predict rib fracture risk was determined from Figure 4.2b. Histogram-based gradient boost and XGBoost outperform other models to predict rib fracture risk. To verify the test error curve, the test and training MAE for all the models are compared in Figure 4.3. Model performance is evaluated based on residual distribution and MAE for the test and training sets. The residuals and the MAE for all the models are tabulated in Table 4.4.

Table 4.4: Comparison of MAE and residuals for probability of more than 2 ribs fracture for 45 years and older occupants of different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	0.044	0.052	-0.003	0.131	-0.014	0.149
Ridge	0.044	0.053	-0.003	0.131	-0.014	0.149
Random Forest	0.032	0.046	0.010	0.106	-0.006	0.154
Gradient Boost	0.030	0.050	-0.003	0.095	-0.023	0.159
XG Boost	0.026	0.040	0.000	0.088	-0.016	0.141
Histogram Based Gradient Boost	0.033	0.040	0.006	0.108	-0.008	0.125
Gaussian Process Regressor	0.014	0.031	0.001	0.057	-0.011	0.100

Linear models such as Lasso and Ridge have high values for both test and train sets. On the other hand, Random Forest and Gradient boost have the training error on the limit of estimated numerical noise, but the test MAE is big. XGBoost, Histogram-based gradient boost and GPR predicts the risk values better than the other models. But the training MAE for XGBoost and GPR are below the estimated noise limit. Histogram-based gradient boost has the train MAE more than the noise. Also, Histogram-based gradient boost has less mean residuals for test set from Table 4.4 compared to the other models. Based on all these factors learning curve is plotted to further analyze the model.

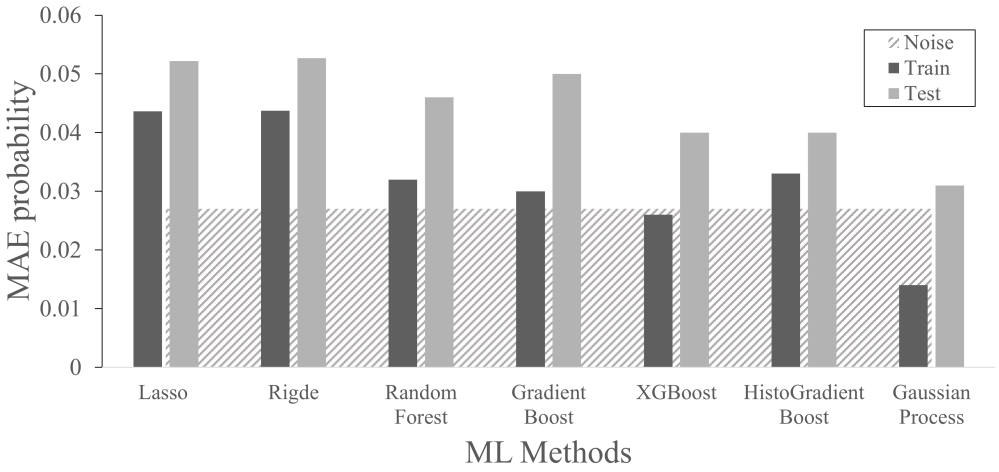


Figure 4.3: Model comparisons of MAE for probability of 2+ rib fracture

The learning curve for Histogram-based gradient boost predicting the value of rib fracture risk is plotted as in Figure 4.4 to check the model behavior. The training error is within the range of the standard deviation of the cross-validation error, implying that the model does not overfit the data. Both the training and the cross-validation MAE saturates toward the end of the plot. Further details on model prediction for test data are interpreted using the residual plot.

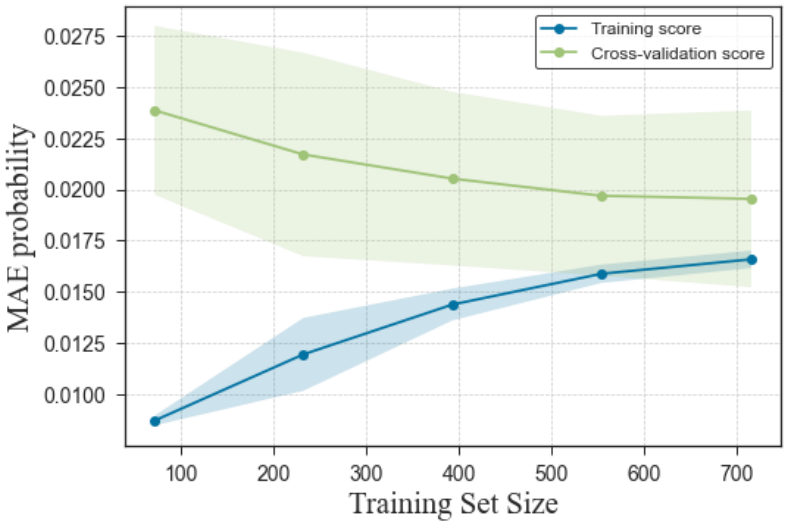
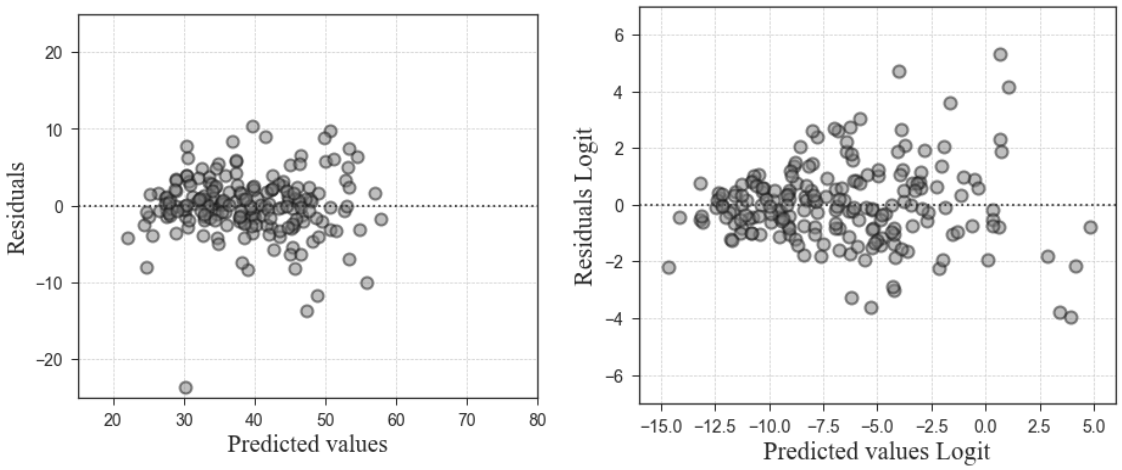


Figure 4.4: Learning curve for rib fracture risk values using histogram based gradient boosting

The residual plot for predicted logit values of risk probabilities, Figure 4.5b shows the residuals for test set are close to zero for values between -12.5 to 2. Higher values has higher residuals. In comparison, the residual plot for the maximum chest deflection Figure 4.5a, the residuals are more scattered near the zero line indicating that the model predictions are accurate. So, using this residual plot the model performance was tested.



(a) Maximum Chest Deflection **(b)** Rib fracture risk

Figure 4.5: Plot for the Test residuals vs the predicted values

From the above results, the test MAE curves give information on the training size required by the ML model with a given accuracy to predict the rib fracture risk.

Similar information for all the output measures was inferred from Figure 4.2. Similarly, results for HIC15, maximum chest deflection, maximum pelvis acceleration, maximum head acceleration, and all the ribs are presented in the Appendix B.2.

4.4 Discussion

The performance of ML models is limited to certain factors such as the number of samples, data distributions, and parameters while evaluating models in this particular study.

4.4.1 Data Analysis

The data was skewed because of skewness in the real life data. The data was based on 90^{th} percentile of real world crashes, which had more data on low severity cases. When the model was trained using the skewed data the model prediction accuracy was bad. The average prediction MAE was 31.144 higher when the training data was not transformed. This is represented in Figure 4.6. The transformation for HIC15 was done using quantile transformation as shown in Figure 4.1. The selection of the type of transformer is based on trial and error, as seen in Figure 4.6, log transformer works better for less sample size. The type of transformer used to correct each type for data is unique and determining them is dependent on the requirement of the surrogate.

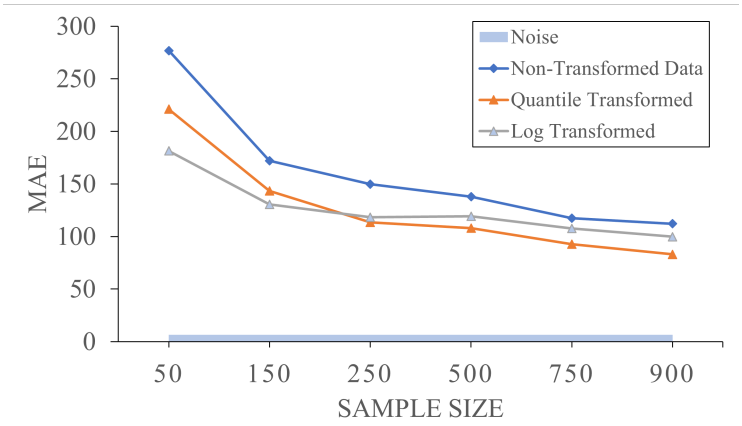


Figure 4.6: HIC15 prediction test error curves for Histogram based gradient boost

Along with correcting the skewness, the data outliers have to be analyzed and corrected. Like limiting the value of HIC15 to 2000, the risk of rib fracture had some data values indicating the risk to be 100%. The maximum possible value for risk can be 99.99% as it cannot be said with 100% certainty at any value that the rib can fracture during a crash. These data points were neglected to continue the study. Similar steps were followed to improve the skewness and analyzed the data outliers for all the output measures as it is different for each of them. The results for these are presented in Appendix B.2. Therefore, the data structure was improved and used to train the models.

The data used was sampled using Latin Hypercube sampling. When sampling the parameters for 1000 simulations, the data had many simulations resulting in close to zero risk. Since the data was skewed, it would not be an ideal sampling strategy for training ML models. Sampling methods as followed in chapter 3 can be followed to generate data. Because uniform distribution generates training data evenly distributed in the parameter value space.

4.4.2 Estimating the number of simulations needed for accurate models

Linear models, Lasso and Ridge predict better results compared to others for fewer training samples. For output measures like the risk of rib fracture and rib strains, linear models are more suitable when about 100 training samples are available. For the other measures, tree-based gradient boosting performs better for this size of training samples. GPR works consistently better than the other models to predict the rib strains for all the 24 ribs for all the training sizes.

When training samples more than 250 is available, tree-based boosting methods perform better than other models for all HBM responses as seen in Figure 4.2. However, XGBoost has a very high testing error when trained with small training samples. This is due to the possibility of overfitting when XGBoost is used for data with more input parameters and fewer samples.

Test MAE curve was used to find the suitable model for the risk of rib fracture with 795 training samples. From the curve in Figure 4.2b, tree-based boosting methods like XGBoost and Histogram-based gradient boost performs better than other evaluated models. To further examine the results from the test MAE curve, the test and train prediction MAE for the models trained with 795 samples is compared with noise estimate as in Figure 4.3. Linear models are not suitable in predicting the risk as both test and train MAE for prediction is high.

Tree-based boosting methods outperform other models in predicting the risk of rib fracture. An improvement in the performance of tree-based boosting models is because of the influence of effective hyper-parameters that includes generalization parameters such as λ, α,γ, learning rate, and in addition to tree-based building parameters. These parameters help in generalizing the data better. GPR has low test and train MAE compared to the other models. However, the training error is considerably less compared to the test error and is much lower than the limit of estimated numerical noise. This may be due to overfitting as it learns too much from the training data. For these reasons, we do not consider GPR as the better suitable model to predict risk despite having a lower test MAE. Evaluating the tree-based boosting methods, gradient boosting has test higher test MAE compared to the other two models. XGBoost and Histogram-based gradient boost both have similar test MAE. But, training MAE for XGBoost is lower than the limit of estimated noise. This might indicate that XGBoost has learned some of the noise in the training data. As Histogram-based gradient boost has test MAE above the estimated numerical

noise and the mean residuals are less compared to the others, it is more suitable in predicting the rib fracture risk.

Further, Histogram-based gradient boost was evaluated using learning curve, Figure 4.4. From the learning curve, it is seen that increasing the training samples with similar distribution cannot improve the prediction MAE, as the cross-validation and training curve looks to be saturating. By increasing the samples from 600 to 700 the validation error in decreased by only 0.05%. To improve the model further, residual plots are evaluated, Figure 4.5b and Figure 4.7.

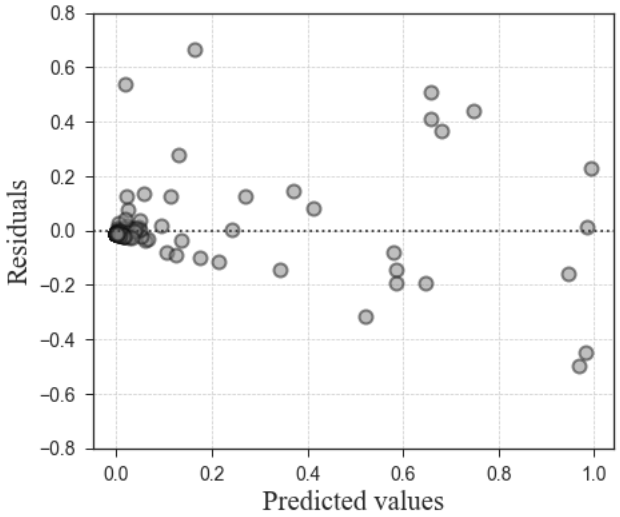


Figure 4.7: Rib fracture risk probability residual plot using inverse logit transform on the 4.5b plot.

The residuals for the logit prediction and the inversed values are compared to check the model performance. The residuals for the predicted values of rib fracture risk are plotted as in Figure 4.7. Figure 4.5b shows the logit transformed risk value used to train the model.real life crashes have lot of data for low riskes so the model is well trained for it. Similar trend in the residual plot for inverse transformed was also seen, Figure 4.7. The residuals for the predictions values less than 0.1 are low. But the prediction values between 0.2 to 1 have large values of residuals. As seen in the residual plots, higher values of risk have higher residuals. This is because of a lack of training data for the model for higher risk measures. This model can be improved further if the model is trained with more data for higher risk values. Similar inferences for all the output measures can be done using the results from Appendix B.2.

4.4.3 Numerical Noise

Numerical noise estimate for the output measures namely HIC15, chest deflection, maximum head acceleration, and pelvis acceleration were much lower compared to the test prediction MAE. But for the risk of rib fracture and rib strains for all the ribs, the test MAE was almost at the limit of numerical noise. The estimation was done by varying the 21 input FE parameters by ±0.5% from the mean. But varying around another point might have given other noise estimates. For example, for the HBM simulations the PDOF was varied about -27deg, a near side oblique crash, where the occupant head more or less misses the airbag. Varying around other PDOFs might produce other noise estimates. Varying the value from the mean would not have influenced outputs like HIC15, chest deflection, etc. As there was no significant influence of the inputs on the outputs, the noise might not have been captured for these measures. In order to obtain more accurate noise estimates, the range of the noise measurement must be chosen based on its influence on specific outputs.

4.5 Conclusion

Depending on the output, a different number of simulations are needed to learn from the data. Increasing the samples generally decreases the prediction MAE. Prediction accuracy for risk of rib fracture using 250 samples is about 0.43. Further increase in training samples decreases the MAE by only 0.001%. But decreasing the training samples to 150, the error increases by 0.04%. 250 is the minimum number of training samples required to obtain good predictions, further increasing the samples decreases the prediction MAE by a very small amount. Similarly, the minimum training samples for all the outputs are listed in Table 4.5.

Table 4.5: Minimum training samples for outputs

Output Measure	Number of training samples	% Change in MAE with further increase in samples
Rib fracture risk	250	0.001
HIC15	250	5.380
Maximum chest deflection	250	0.110
Maximum head acceleration	500	0.250
Maximum pelvis acceleration	500	0.040
Maximum rib strains	250	0.004

To improve the prediction accuracy for skewed outputs, transformations are required. Data from FE simulations have numerical noise, estimating its magnitude should be included in the workflow to evaluate ML models. Tree-based boosting methods worked better to predict the HBM responses and injury metrics. Suitable models for outputs are tabulated in Table4.6.

Table 4.6: suitable models for the outputs

FE output	**Final model**
Maximum chest deflection	Gradient boosting
HIC15	XG boosting
Maximum head acceleration	XG boosting
Maximum pelvis acceleration	XG boosting
2+ fractured rib risk for 45 years	Histogram based gradient boosting
Rib strain (Left rib1)	Gaussian process regressor
Rib strain (Right rib 12)	Histogram based gradient boosting

5
Conclusion

The aim of the thesis was to evaluate the use of ML models in building surrogates to predict HBM responses and injury metrics. Surrogates were developed using machine learning to FE human body models using data generated from the fast hybrid III model and SAFER HBM. A ML pipeline was developed to train, tune and test the surrogate model. The data analysis and model evaluation in this thesis was done in jupyter notebook, these are available in GitHub[1].

The first objective to fulfill the aim was to evaluate the number of simulations required to build accurate machine learning models for a given set of FE parameters. Surrogate models were built for three types of data set. Fast hybrid III had two types of data, 130 samples with only restraint FE parameters and another 130 samples with crash and restraint FE parameters. The SAFER HBM data had 1000 samples with 21 FE parameters. To get accurate models for fast hybrid III, about 60 training samples are needed. Whereas, outputs from SAFER HBM required a minimum of 250 training samples. The number of samples required depends on the parameterization of the crashes for the HBM simulations. In the study, HBM needed more training samples because it had more FE input parameters and was sampled for a broad range of values compared to fast hybrid III simulation data. Because the data had fewer parameters and sampling was restricted to a smaller range.

The second objective involved implementing a general ML pipeline consisting of data preparation, model validation, and optimization. As a part of data preparation for HBM, outputs needed to be transformed to have a better distribution, so that the ML models performs well. In addition, the presence of numerical noise in the data generated from FE simulations is required to be estimated and considered to be a part of the ML pipeline to predict HBM responses.

For surrogate models of fast hybrid III, linear models are more suitable for training sizes less than 60. However, for more training samples up to 120 tree-based boosting methods perform better. The tree-based boosting algorithms are suitable for surrogates to predict HBM responses and injury metrics for all training sizes examined. However, GPR works better as a surrogate to predict rib strains than tree-based boosting methods.

[1] `https://github.com/yash-n-p/FE_HBM_ML`

The important outcomes obtained from this study were,

1. For modelling skewed data it was important to first transform the data. The quantile transformer generally worked good for the the HBM reponses and Logit transformer worked better for the injury risks data from SAFER HBM v9.
2. Numerical noise estimation for the simulation outputs should be considered to be a part of the ML workflow.
3. Test MAE curves were used to compare the performance of ML models for each output. These curves were used to choose suitable ML models based on the available size of training data.
4. Tree-based boosting methods were suitable in predicting for most of the HBM responses and injury metrices followed by GPR, Random Forest, linear models and Decision tree.

6
Future Work

HBM data used in the study had 21 FE parameters with a wide range of values. Due to this, the ML models required a large number of training samples to predict accurate outputs. The presence of many FE parameters also affects ML performance due to multicollinearity and overfitting. To overcome this, a further study can be carried to evaluate ML models for a well-defined crash case i.e. a specific crash in a specific car with only a few design parameters for the restraints. In this way, accuracy improvements can be evaluated when the ML model is trained with fewer training samples of data having fewer FE parameters.

Linear, ensemble methods, and GPR were used in the study to explore its performance in predicting HBM response and injury metrics. Implementing stacking algorithms in building surrogate models would be an interesting study for future research. The method combines learning strategies from multiple machine learning algorithms to improve the prediction accuracy of the surrogate. Using this method, we can develop a generalized surrogate to predict HBM responses and injury criteria.

Bibliography

[1] Road safety facts — association for safe international road travel [Internet]. Asirt.org. 2018. Available from: link

[2] Accident statistics 2019: number of fatalities at a low level - only one group with alarming numbers . The Limited Times. 2020 Available from:link

[3] Jakobsson L. Active human body models for virtual occupant response: step 3. 2018.

[4] Kumar P, Sinha K, Nere NK, Shin Y, Ho R, Mlinar LB, et al. A machine learning framework for computationally expensive transient models. Sci Rep. 2020;10(1):11492.

[5] Li, Y., & Fan, W. (david). (2019). Pedestrian injury severities in pedestrian-vehicle crashes and the partial proportional odds logit model: Accounting for age difference. Transportation Research Record, 2673(5), 731–746.

[6] Assi, K., Rahman, S. M., Mansoor, U., & Ratrout, N. (2020). Predicting crash injury severity with machine learning algorithm synergized with clustering technique: A promising protocol. International Journal of Environmental Research and Public Health, 17(15), 5497.

[7] Perez-Rapela D, Forman J, Huddleston S, Crandall J. Methodology for vehicle safety development and assessment accounting for occupant response variability to human and non-human factors. Computer Methods in Biomechanics and Biomedical Engineering. 2020;:1-16.

[8] Joodaki H, Gepner B, Kerrigan J. Leveraging machine learning for predicting human body model response in restraint design simulations. Computer Methods in Biomechanics and Biomedical Engineering. 2020;:1-15.

[9] Industrialisation of a Finite Element Active Human Body Model for Vehicle Crash Simulations | SAFER – Vehicle and Traffic Safety Centre at Chalmers 2021. Available from: link

[10] Brolin K, Östh J, Mendoza-Vazquez M. HUMAN BODY MODELING FOR APPLIED TRAFFIC SAFETY. In p. 13–15.

[11] Iraeus J, Pipkorn B. Development and validation of a generic finite element ribcage to be used for strain-based fracture prediction. In Proceedings of the IRCOBI Conference 2019 (pp. 193-210).

[12] Iraeus J. Stochastic Finite Element Simulations of Real Life Frontal Crashes. With emphasis on chest injury mechanisms in near-side oblique loading conditions. 2015.

[13] Iraeus J, Lindquist M. Development and validation of a generic finite element vehicle buck model for the analysis of driver rib fractures in real life nearside oblique frontal crashes. Accident Analysis & Prevention. 2016 Oct 1;95:42-56.

[14] Simms C, Wood D. Pedestrian and cyclist impact: a biomechanical perspective. Springer Science & Business Media; 2009.

[15] Helsinki.fi. Data mining: machine learning, statistics, and databases. Available from: link

[16] Raschka S, Mirjalili V. Python machine learning: Machine learning and deep learning with python, scikit-learn, and TensorFlow 2, 3rd edition. 3rd ed. Birmingham, England: Packt Publishing; 2019. Available from: link

[17] Brownlee J. Regression metrics for machine learning. Machine learning mastery.com. 2021. Available from: link

[18] Brownlee J. What is data preparation in a machine learning project. Machine learning mastery.com. 2020. Available from: link

[19] Brownlee J. A Gentle Introduction to k-fold Cross-Validation. Machinelearning mastery.com. 2018. Available from:link

[20] Brownlee J. Overfitting and underfitting with machine learning algorithms. Machine learning mastery.com. 2016. Available from: link

[21] Pedregosa et al. Scikit-learn: Machine Learning in Python„ JMLR 12, pp. 2825-2830, 2011

[22] sabita-ai. Machine learning algorithms. Analyticsvidhya.com. 2021. Available from: link

[23] Hallén R. A study of gradient-based algorithms. 2017 Available from: link

[24] Alpaydin E. Introduction to Machine Learning. 4th ed. London, England: MIT Press; 2020.

[25] de Rooij M, Pratiwi BC, Fokkema M, Dusseldorp E, Kelderman H. The Early Roots of Statistical Learning in the Psychometric Literature: A review and two new results. arXiv [stat.ME]. 2019. Available from:link

[26] Gupta P. Regularization in Machine Learning [Internet]. Towards Data Science. 2017. Available from:link

[27] Singh K, Xie M. Bootstrap: a statistical method. Unpublished manuscript, Rutgers University, USA. Retrieved from http://www. stat. rutgers. edu/home-/mxie/RCPapers/bootstrap. pdf. 2008:1-4.

[28] Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. Journal of statistical software. 2010;33(1):1.

[29] Zhou Z-H. Ensemble methods: Foundations and algorithms. Caithness, UK: Whittles Publishing; 2012.

[30] Vidhya A. Tree based algorithms [Internet]. Analyticsvidhya.com. 2016 [cited 2021 Jun 7]. Available from: link

[31] Sammut C, Webb GI. Encyclopedia of machine learning. Springer Science & Business Media;

[32] Rokach L, Maimon OZ. Data mining with decision trees: Theory and applications (2nd edition). 2nd ed. Singapore, Singapore: World Scientific Publishing; 2014.

[33] Brownlee J. XGBoost for regression. Machinelearningmastery.com. 2021 [cited 2021 Jun 7]. Available from: link

[34] Gaussian processes (1/3) - From scratch. (n.d.). Retrieved June 22, 2021, from Github.io website: link

[35] Kernel Cookbook. Toronto.edu.. Available from: link

[36] Jordan J. Evaluating a machine learning model. Jeremyjordan.me. Jeremy Jordan; 2017. Available from: link

[37] Frey BB, editor. The SAGE encyclopedia of educational research, measurement, and evaluation. Sage Publications; 2018.

[38] Waskom M. seaborn: statistical data visualization. J Open Source Softw. 2021;6(60):3021. link

[39] Ben. What should be the value of skewness is acceptable? [Internet]. Mvorganizing.org. [cited 2021 Jun 30]. Available from:link

[40] Kuppa S. Injury criteria for side impact dummies. Washington, DC: National Transportation Biomechanics Research Center, National Highway Saftey Administration, US DOT; 2004.

[41] Merlin LA, Guerra E, Dumbaugh E. Crash risk, crash exposure, and the built environment: A conceptual review. Accid Anal Prev. 2020;134(105244):105244.

[42] Forman JL, Kent RW, Mroz K, Pipkorn B, Bostrom O, Segui-Gomez M. Predicting rib fracture risk with whole-body finite element models: development and preliminary evaluation of a probabilistic analytical framework. InAnnals of Advances in Automotive Medicine/Annual Scientific Conference 2012 Oct (Vol. 56, p. 109). Association for the Advancement of Automotive Medicine.

A

Dummy Results

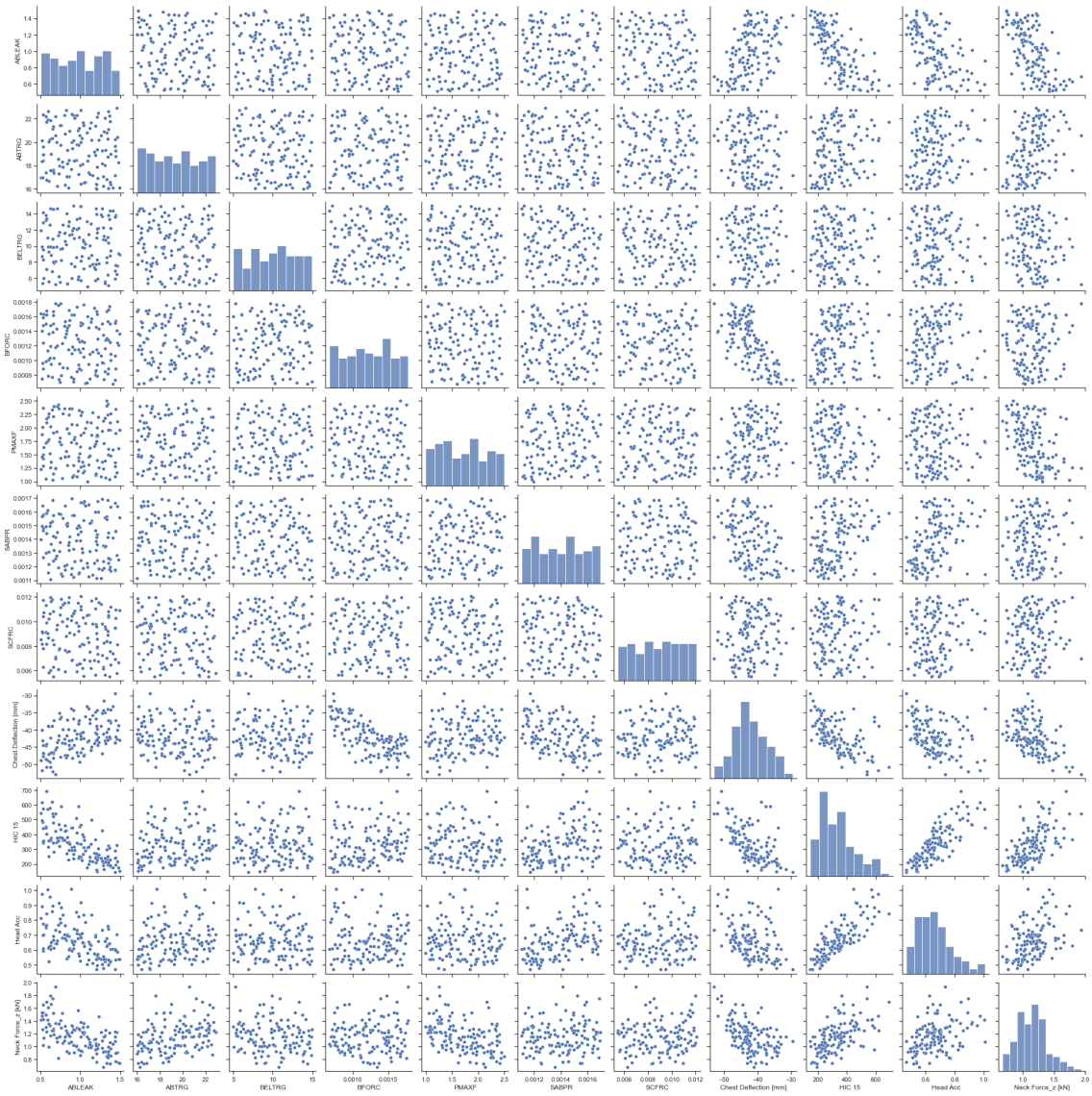


Figure A.1: Plot describing the interactions of HIC15, chest deflection, head acceleration and neck force and the input parameter, i.e. the restraint parameters to study about the structure of the data.

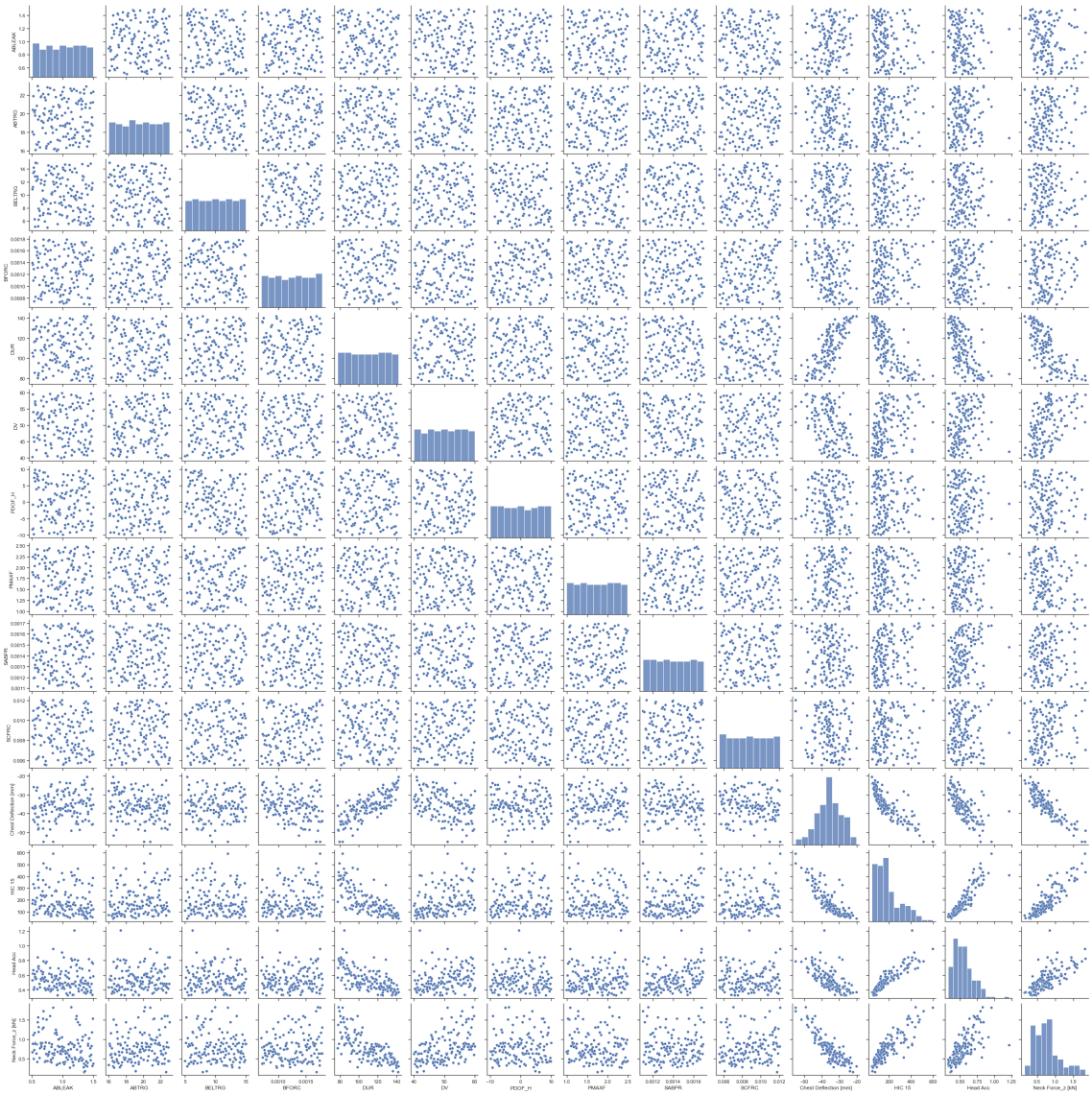
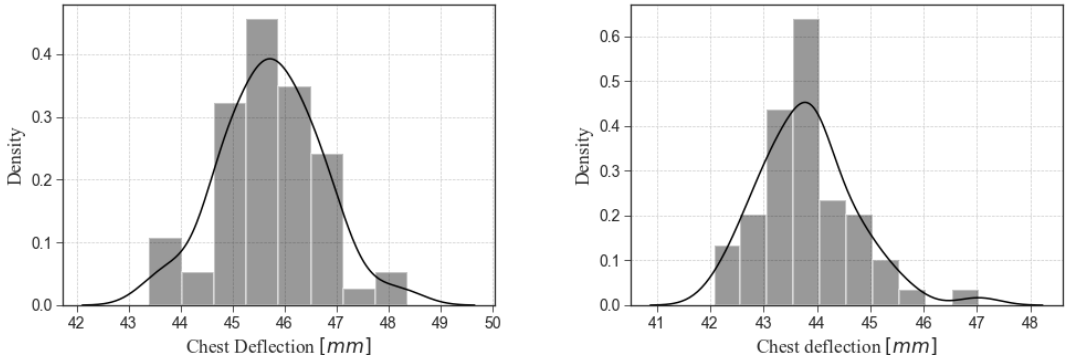


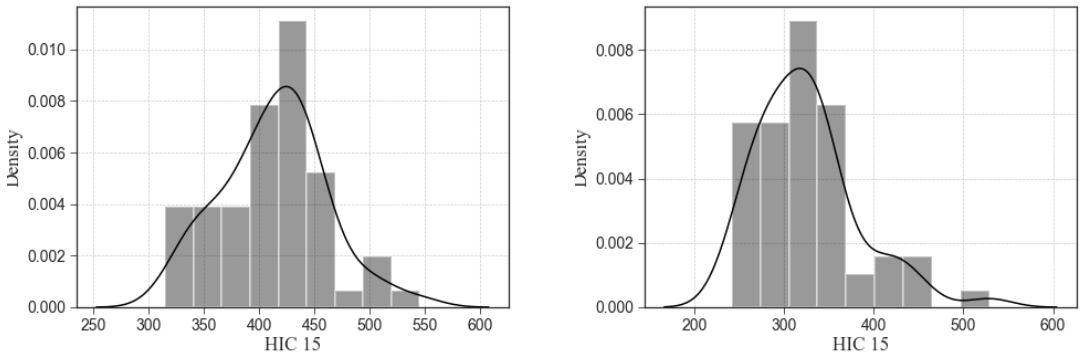
Figure A.2: Plot describing the interactions of HIC15, chest deflection, head acceleration and neck force and the input parameter, i.e. crash and restraint parameters to study about the structure of the data.

A.1 Numerical Noise



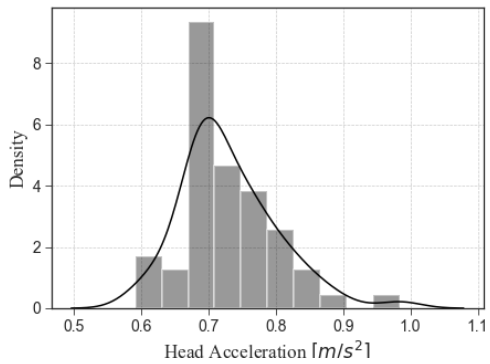
(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure A.3: Distribution of maximum chest deflection $[mm]$ values for 1% change in inputs

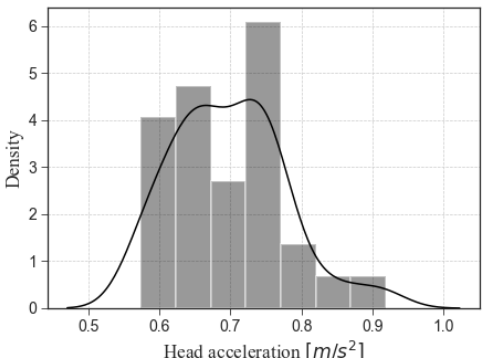


(a) Restraint parameters **(b)** Crash and Restraint parameters

Figure A.4: Distribution of HIC15 values for 1% change in inputs

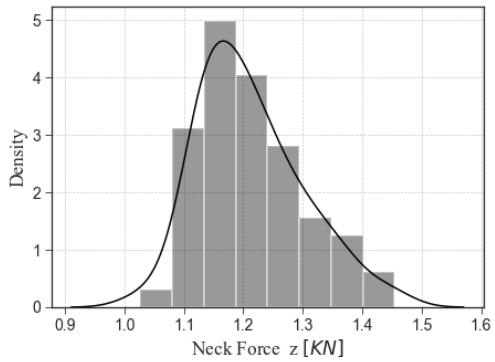


(a) Restraint parameters

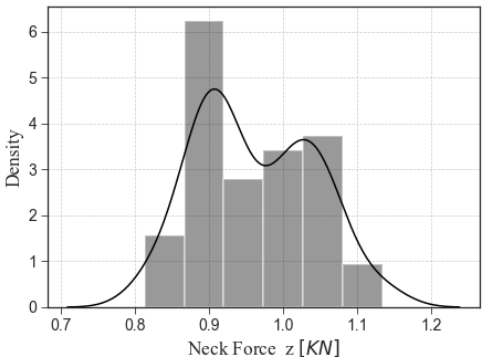


(b) Crash and Restraint parameters

Figure A.5: Distribution of maximum head acceleration $[m/s^2]$ values for 1% change in inputs



(a) Restraint parameters

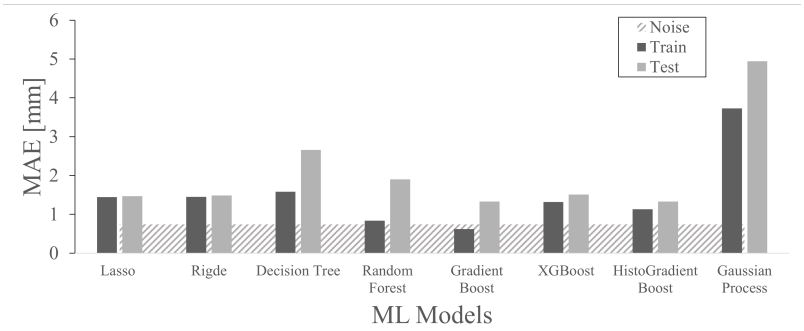


(b) Crash and Restraint parameters

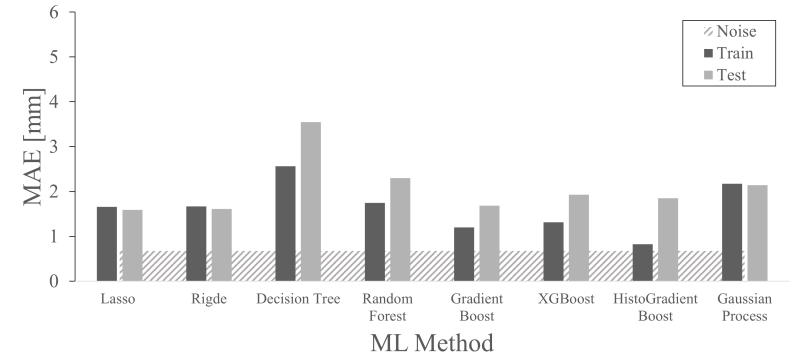
Figure A.6: Distribution of maximum neck force $[KN]$ values for 1% change in inputs

A.2 Learning curves and Model comparisons

A.2.1 Chest deflection

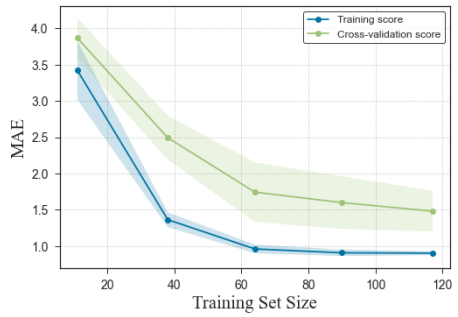


(a) Restraint parameters

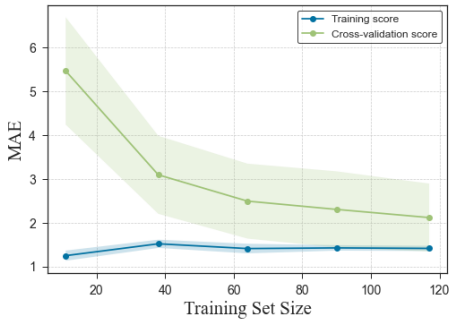


(b) Crash and Restraint parameters

Figure A.7: Model comparisons for test and train MAE of predictions using bar graph of maximum chest deflection with noise estimate

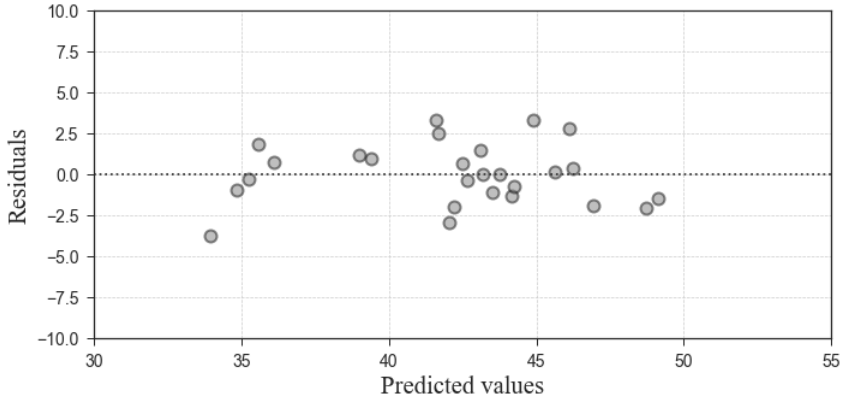


(a) Restraint parameters-Histogram based gradient boost

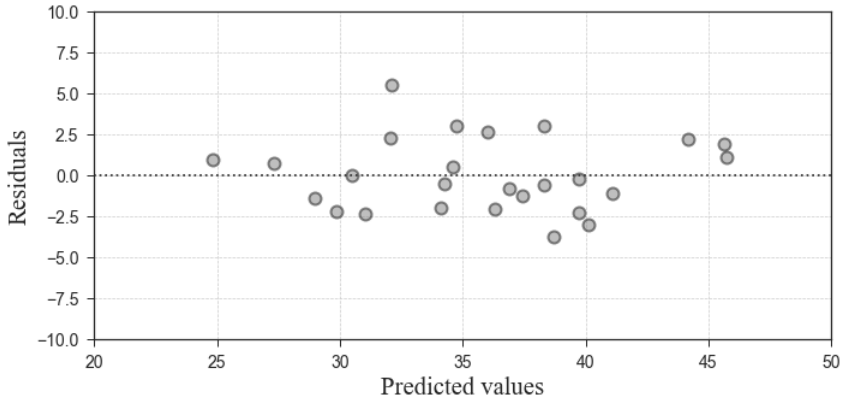


(b) Crash and Restraint parameters-Gradient Boost

Figure A.8: Learning Curves for the most suitable model to predict chest deflection



(a) Restraint parameters-Histogram based gradient boost



(b) Crash and Restraint parameters-Gradient Boost

Figure A.9: Residual plot for the most suitable model to predict chest deflection

A.2.2 HIC15

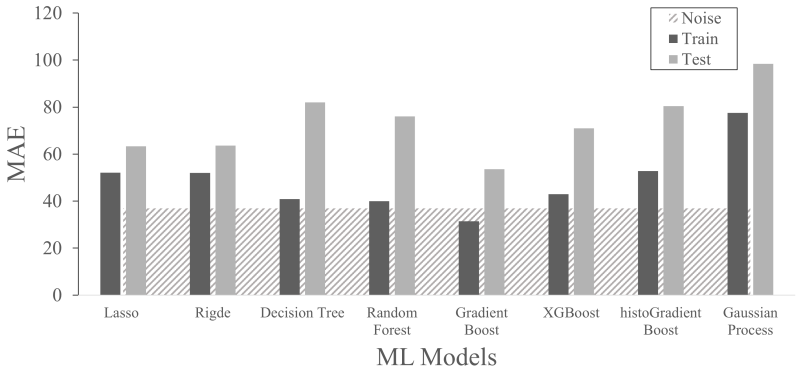


Figure A.10: Model comparisons for test and train MAE of predictions using bar graph of HIC15 with noise estimate for restraint parameters

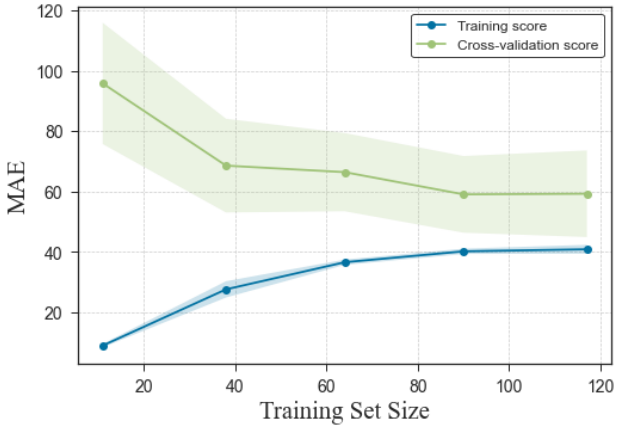


Figure A.11: Learning Curve for the most suitable model to predict HIC15 for restraint parameter

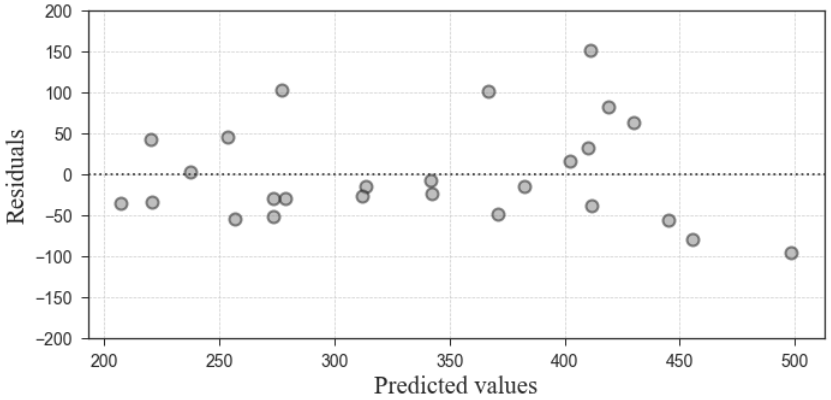
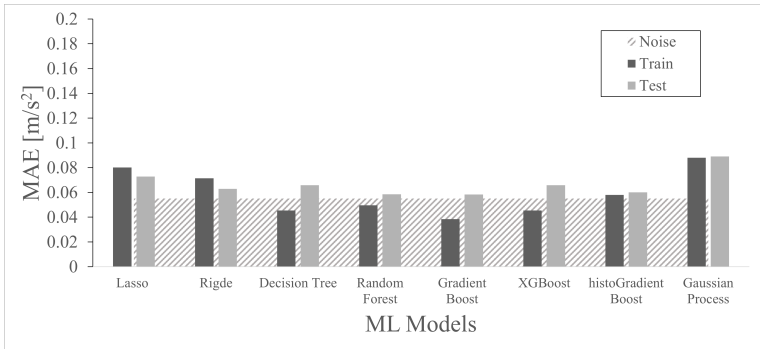
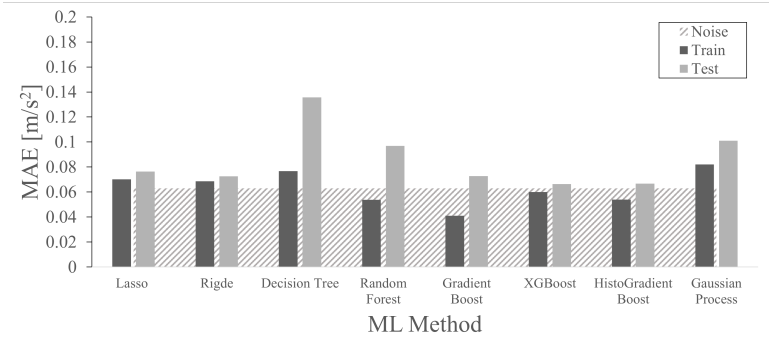


Figure A.12: Residual plot for the most suitable model to predict HIC15 for restraint parameter

A.2.3 Maximum head acceleration

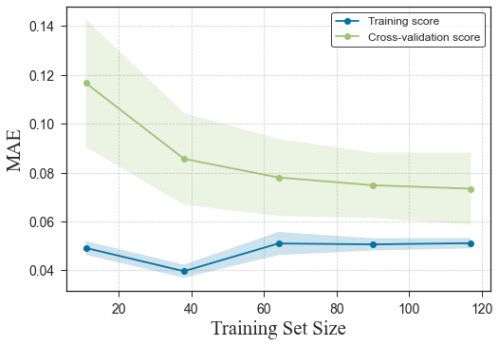


(a) Restraint parameters

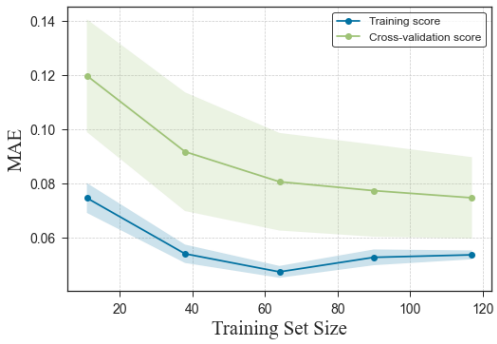


(b) Crash and Restraint parameters

Figure A.13: Model comparisons for test and train MAE of predictions using bar graph of maximum head acceleration with noise estimate

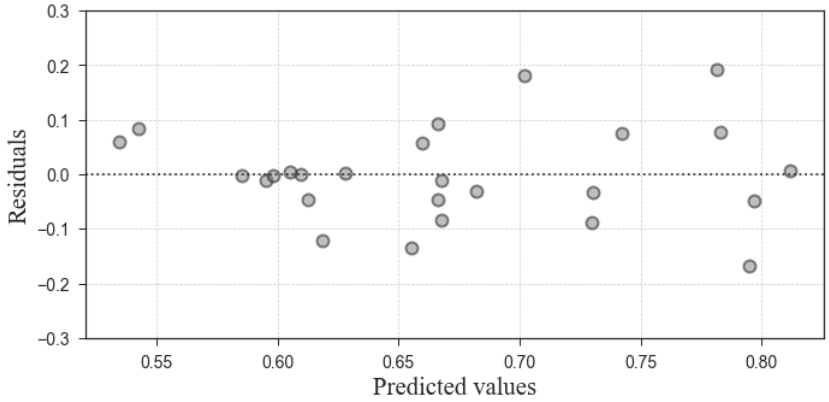


(a) Restraint parameters-Histogram based gradient boost

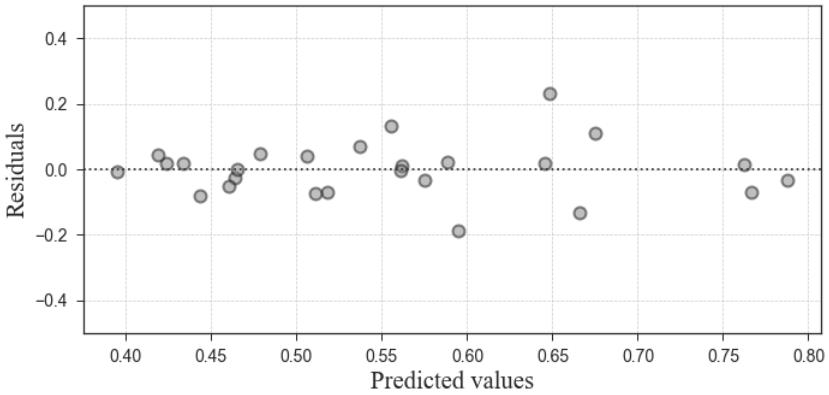


(b) Crash and Restraint parameters-XGBoost

Figure A.14: Learning Curves for the most suitable model to predict maximum head acceleration



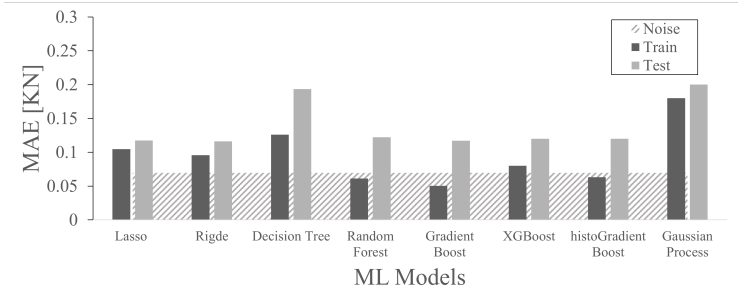
(a) Restraint parameters-Histogram based gradient boost



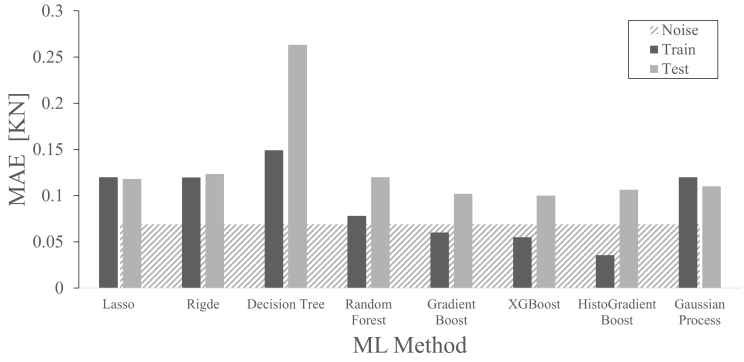
(b) Crash and Restraint parameters-XGBoost

Figure A.15: Residual plot for the most suitable model to predict maximum head acceleration

A.2.4 Maximum neck force

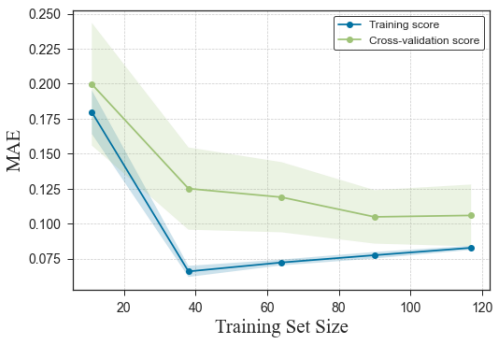


(a) Restraint parameters

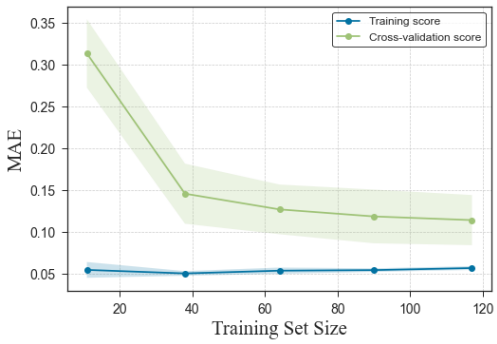


(b) Crash and Restraint parameters

Figure A.16: Model comparisons for test and train MAE of predictions using bar graph of maximum neck force with noise estimate

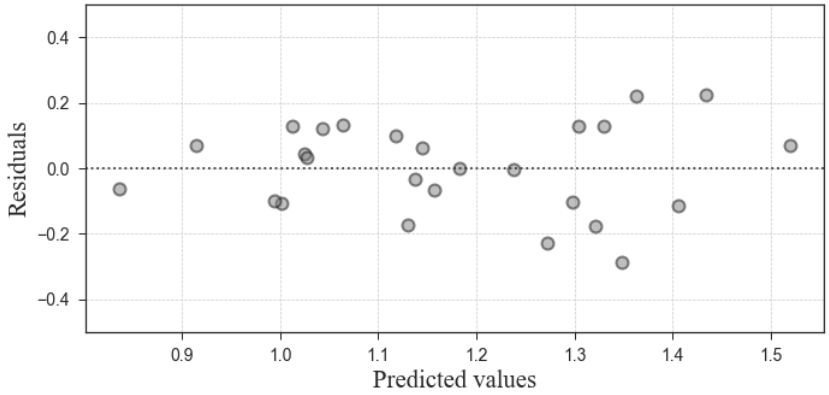


(a) Restraint parameters-XGBoost

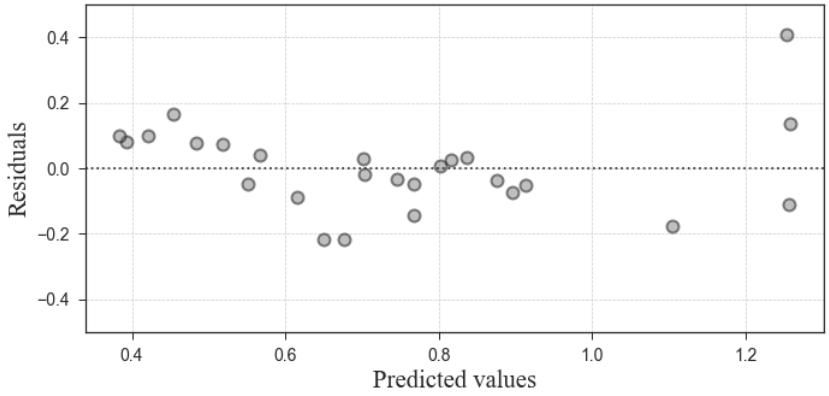


(b) Crash and Restraint parameters-Gradient boosting

Figure A.17: Learning Curves for the most suitable model to predict maximum neck force



(a) Restraint parameters-XGBoost



(b) Crash and Restraint parameters-Gradient boosting

Figure A.18: Residual plot for the most suitable model to predict maximum neck force

A.3 Tuned Models

Table A.1: Tuned Hyperparameters of ML models to predict Chest deflection for the Restraint parameters

ML Model	Tuned Hyperparameters
Lasso	alpha = 1e-06,selection='random',max_iter = 2000
Ridge	alpha = 1e-07,max_iter = 4000
Decision Tree	max_depth=20, max_features=0.5,min_samples_leaf=0.03, min_samples_split=0.085, criterion= 'mse', max_leaf_nodes = 13
Random Forest	max_depth=5, min_samples_leaf=0.001, min_samples_split=0.004,criterion='mse', bootstrap=True,max_features='auto',n_estimators=15,oob_score=False,warm_start=False
Gradient Boost	subsample=0.5, n_estimators=100, min_samples_split=0.1, min_samples_leaf=0.07, max_features='auto', max_depth=10, learning_rate=0.09,criterion= 'mse',warm_start=True
XG Boost	base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.915, gamma=11, gpu_id=-1,importance_type='gain', interaction_constraints='',learning_rate=0.2, max_delta_step=12, max_depth=4,min_child_weight=20.5, monotone_constraints='()', n_estimators=110, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.85,tree_method='exact', validate_parameters=1, verbosity=None
Histogram based Gradient Boost	l2_regularization=0.41, learning_rate=0.4, max_depth=1, max_iter=70, max_leaf_nodes=6, min_samples_leaf=16, scoring='neg_root_mean_squared_error'
Gaussian Process Regressor	kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5) n_restarts_optimizer=4, alpha=4, normalize_y=True

Table A.2: Tuned Hyperparameters of ML models to predict Neck force for restraint parameters

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha = 0.01,selection='random',max_iter = 2700)
Ridge	Ridge(alpha = 0.1,max_iter = 4000)
Decision Tree	DecisionTreeRegressor(max_depth=5, max_features=0.5, min_samples_leaf=0.01, min_samples_split=0.09,criterion= 'mse',max_leaf_nodes = 7)
Random Forest	RandomForestRegressor(max_depth=10, min_samples_leaf=0.02, min_samples_split=0.04, criterion='mse',bootstrap=True,max_features='auto',warm_start=False,n_estimators=65, oob_score=False)
Gradient boost	GradientBoostingRegressor(max_depth=4, min_samples_leaf=0.06, min_samples_split=0.1, criterion='mse',subsample=0.5,max_features=0.5,warm_start=True,n_estimators=80,learning_rate=0.08)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=0.41, learning_rate=0.25,max_depth=2, max_iter=55, max_leaf_nodes=4,min_samples_leaf=9,scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.925, gamma=0.04, gpu_id=-1,importance_type='gain', interaction_constraints='',learning_rate=0.1, max_delta_step=0, max_depth=1,min_child_weight=5, monotone_constraints='()',n_estimators=70, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.5,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.3: Tuned Hyperparameters of ML models to predict head acceleration using restraint parameters

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha = 0.0012222222222222222,selection='random',max_iter = 2100)
Ridge	Ridge(alpha = 1e-07,max_iter = 4000)
Decision Tree	DecisionTreeRegressor(max_depth=32, max_features=0.5, min_samples_leaf=0.025, min_samples_split=0.06,criterion= 'mse',max_leaf_nodes = 17)
Random Forest	RandomForestRegressor(max_depth=20, min_samples_leaf=0.01, min_samples_split=0.04,criterion='mse',bootstrap=False,max_features='sqrt', n_estimators=10, oob_score=False,warm_start=True)
Gradient boost	GradientBoostingRegressor(max_depth=30, min_samples_leaf=0.06, min_samples_split=0.1,criterion='mse',max_features=0.5,n_estimators=80, learning_rate=0.03, warm_start=False)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=10, learning_rate=0.02,max_depth=3, max_iter=50, max_leaf_nodes=4,min_samples_leaf=15,scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.915, gamma=11, gpu_id=-1,importance_type='gain', interaction_constraints='',learning_rate=0.2, max_delta_step=12, max_depth=4, min_child_weight=20.5, monotone_constraints='()',n_estimators=110, n_jobs=12, num_parallel_tree=1, random_state=0,reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.85,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.4: Tuned Hyperparameters of ML models to predict HIC15 using restraint parameters

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha=1e-07, selection='random',max_iter=100000)
Ridge	Ridge(alpha = 1e-07,max_iter = 4000)
Decision Tree	DecisionTreeRegressor(max_depth=5, max_features='auto',min_samples_leaf=0.055, min_samples_split=0.05,criterion= 'mae',max_leaf_nodes = 30)
Random Forest	RandomForestRegressor(max_depth=4, min_samples_leaf=0.001, min_samples_split=0.09,criterion='mse',bootstrap=True,max_features='auto', n_estimators=25, oob_score=True,warm_start=True)
Gradient boost	GradientBoostingRegressor(max_depth=2, min_samples_leaf=0.06, min_samples_split=0.02,criterion='mae',max_features=0.5,n_estimators=120, learning_rate=0.03, warm_start=False)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=0.41, learning_rate=0.07,max_depth=2, max_iter=70, max_leaf_nodes=4,min_samples_leaf=9,scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.95, gamma=0.03, gpu_id=-1, importance_type='gain', interaction_constraints='',learning_rate=0.075, max_delta_step=0, max_depth=3,min_child_weight=11, monotone_constraints='()',n_estimators=65, n_jobs=12, num_parallel_tree=1, random_state=0,reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.5,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.5: Tuned Hyperparameters of ML models to predict chest deflection for crash and restraint

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha=2.1181818181818184e-05, warm_start=True)
Ridge	Ridge(alpha=1e-06)
Decision Tree	DecisionTreeRegressor(criterion='mae', max_depth=5, max_features='auto', max_leaf_nodes=7, min_samples_leaf=0.009,min_samples_split=0.05)
Random Forest	RandomForestRegressor(max_depth=15, min_samples_leaf=4, min_samples_split=6, criterion='mse',bootstrap=True,max_features='auto',warm_start=False,n_estimators=40, oob_score=True)
Gradient boost	# mod=GradientBoostingRegressor(learning_rate=0.05, loss='huber', max_features='auto', min_samples_leaf=0.03, min_samples_split=0.03,n_estimators=75, subsample=0.4,max_depth=2)
Histogram based gradient boost	mod=HistGradientBoostingRegressor(l2_regularization=0.4, learning_rate=0.3, max_depth=2, max_iter=50, max_leaf_nodes=4,min_samples_leaf=15, scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.875, gamma=10.5, gpu_id=-1, importance_type='gain', interaction_constraints='',learning_rate=0.15, max_delta_step=13, max_depth=3,min_child_weight=20.8, monotone_constraints='()', n_estimators=110, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.85,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.6: Tuned Hyperparameters of ML models to predict neck force for crash and restraint

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha=1e-06, selection='random')
Ridge	Ridge(alpha=1.1090909090909092e-05, solver='svd')
Decision Tree	DecisionTreeRegressor(criterion='friedman_mse', max_depth=20, max_features=0.5, max_leaf_nodes=5, min_samples_leaf=0.05,min_samples_split=0.1)
Random Forest	RandomForestRegressor(max_depth=17, min_samples_leaf=3, min_samples_split=6,criterion='mse',bootstrap=True,max_features='auto', warm_start=False,n_estimators=60, oob_score=False)
Gradient boost	GradientBoostingRegressor(learning_rate=0.05, loss='huber', max_features='auto', min_samples_leaf=0.04, min_samples_split=0.04,n_estimators=77, subsample=0.45)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=0.4, learning_rate=0.25, max_depth=2, max_iter=58, max_leaf_nodes=5,min_samples_leaf=9, scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.925, gamma=0.02, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.1, max_delta_step=0, max_depth=2, min_child_weight=5, monotone_constraints='()', n_estimators=80, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.5,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.7: Tuned Hyperparameters of ML models to predict Head acceleration for crash and restraint

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha=0.0006064545454545455, normalize=True, selection='random',warm_start=True)
Ridge	Ridge(alpha=0.000989909090909091, normalize=True, solver='saga')
Decision Tree	DecisionTreeRegressor(criterion='mae', max_depth=20, max_features=0.5, max_leaf_nodes=6, min_samples_leaf=0.01,min_samples_split=0.09)
Random Forest	RandomForestRegressor(max_depth=23, min_samples_leaf=5, min_samples_split=6,criterion='mse',bootstrap=True,max_features='auto',n_estimators=50, oob_score=False,warm_start=True)
Gradient boost	GradientBoostingRegressor(criterion='mse', learning_rate=0.055, loss='huber',max_depth=4, max_features='auto',min_samples_leaf=0.04, min_samples_split=0.04,n_estimators=76, subsample=0.4)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=0.4, learning_rate=0.3, max_depth=1, max_iter=50, max_leaf_nodes=5,min_samples_leaf=5, scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.925, gamma=0.02, gpu_id=-1, importance_type='gain', interaction_constraints='',learning_rate=0.15, max_delta_step=0, max_depth=1, min_child_weight=5, monotone_constraints='()', n_estimators=60, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.5, tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

Table A.8: Tuned Hyperparameters of ML models to predict HIC15 for crash and restraint

ML Model	Tuned Hyperparameters
Lasso	Lasso(alpha=0.00045509090909090917, selection='random')
Ridge	Ridge(alpha=1e-06)
Decision Tree	DecisionTreeRegressor(criterion='mae', max_depth=25, max_features=0.5, max_leaf_nodes=6, min_samples_leaf=0.02,min_samples_split=0.06)
Random Forest	RandomForestRegressor(max_depth=20, min_samples_leaf=3, min_samples_split=6,criterion='mse',bootstrap=True,max_features='auto', n_estimators=40, oob_score=False,warm_start=False)
Gradient boost	GradientBoostingRegressor(learning_rate=0.029, loss='huber', max_depth=4,max_features='auto', min_samples_leaf=0.03, min_samples_split=0.03, n_estimators=79,subsample=0.35)
Histogram based gradient boost	HistGradientBoostingRegressor(l2_regularization=0.58, learning_rate=0.21, max_depth=3, max_iter=50, max_leaf_nodes=3, min_samples_leaf=7,scoring='neg_root_mean_squared_error')
XGBoost	xg.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.975, gamma=0.03, gpu_id=-1, importance_type='gain', interaction_constraints='',learning_rate=0.075, max_delta_step=0, max_depth=2,min_child_weight=10, monotone_constraints='()', n_estimators=70, n_jobs=12, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1,scale_pos_weight=1,subsample=0.5,tree_method='exact', validate_parameters=1, verbosity=None)
Gaussian process Regressor	GaussianProcessRegressor(kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5),n_restarts_optimizer=1, normalize_y=True,random_state=0, alpha=0.4)

B

HBM Results

B.1 Numerical Noise

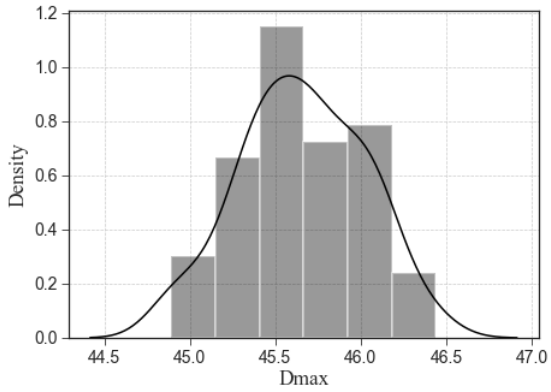


Figure B.1: Distribution of chest deflection [mm] values

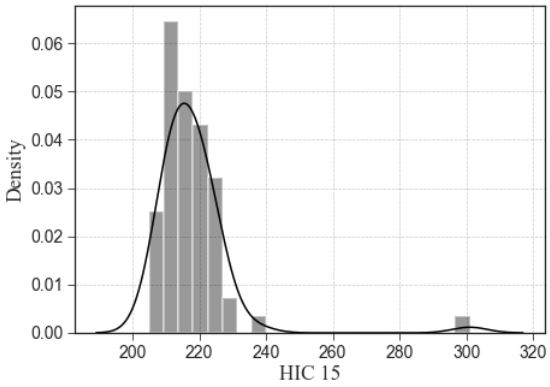


Figure B.2: Distribution of HIC15 values

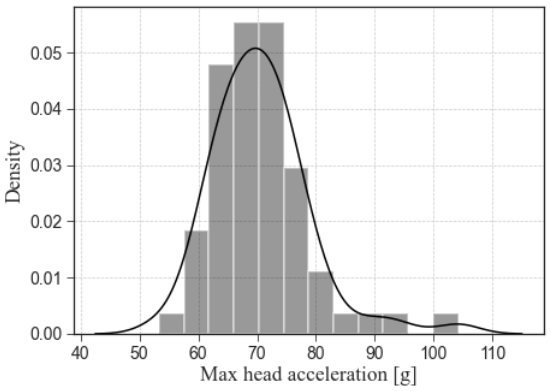


Figure B.3: Distribution of maximum head acceleration[g] values

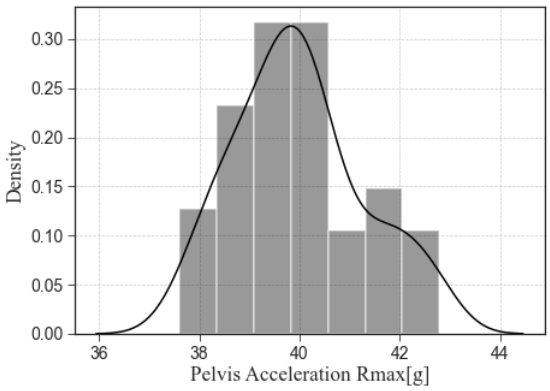


Figure B.4: Distribution of maximum pelvis acceleration [g] values

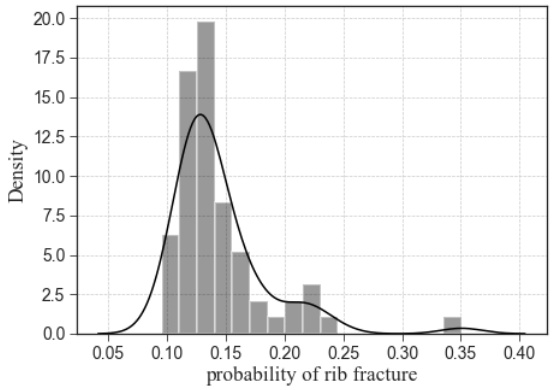


Figure B.5: Distribution of probability of two or more rib fractures for 45 years and older occupants

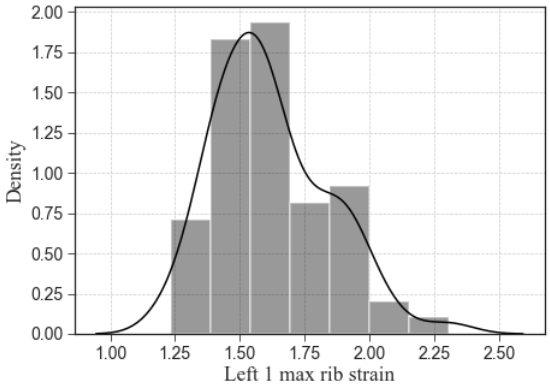


Figure B.6: Distribution of maximum strains for the left rib 1

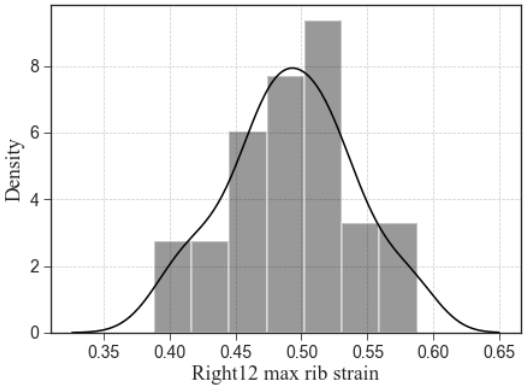


Figure B.7: Distribution of strains in the 12th right rib

B.2 Learning curves and model comparisons

B.2.1 chest deflection

Table B.1: Comparison of MAE and residuals for chest deflection (Dmax)[mm] of different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	3.530	3.210	0.017	5.120	0.203	4.789
Ridge	3.500	3.190	0.011	5.116	0.208	4.810
Random Forest	2.100	2.990	0.064	3.076	-0.176	4.189
Gradient Boost	1.700	2.700	0.179	2.986	0.038	3.946
XG Boost	1.500	2.670	-0.050	2.198	-0.227	3.977
Histogram Based Gradient Boost	1.300	2.500	0.013	1.987	0.102	3.754
Gaussian Process Regressor	1.800	2.600	0.020	2.466	-0.327	4.504

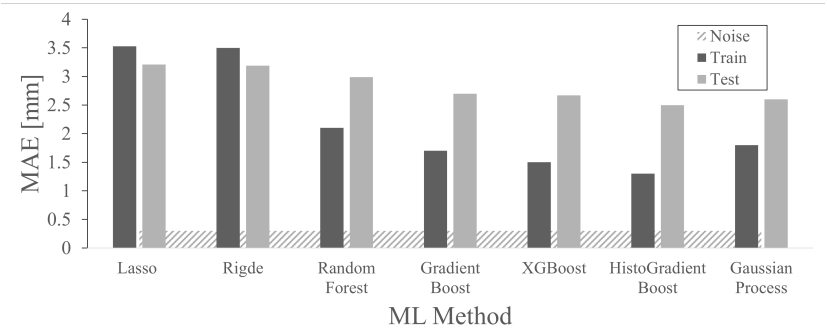


Figure B.8: Model comparisons for MAE of predictions using bar graph of Chest deflection [mm]

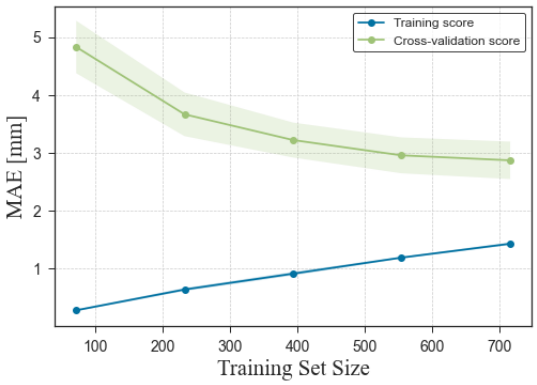


Figure B.9: Learning curve for chest deflection values using gradient boosting

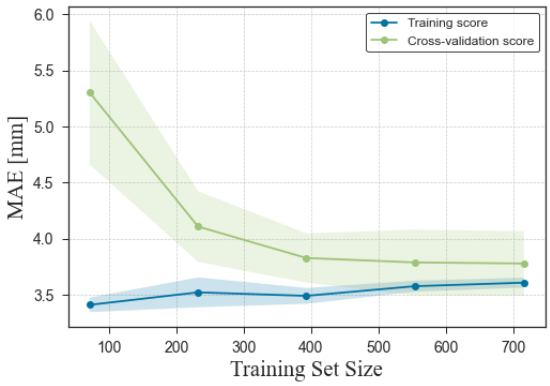


Figure B.10: Learning curve for lasso

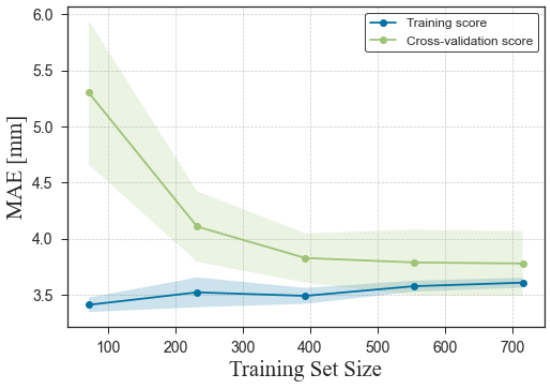


Figure B.11: Learning curve for Ridge.

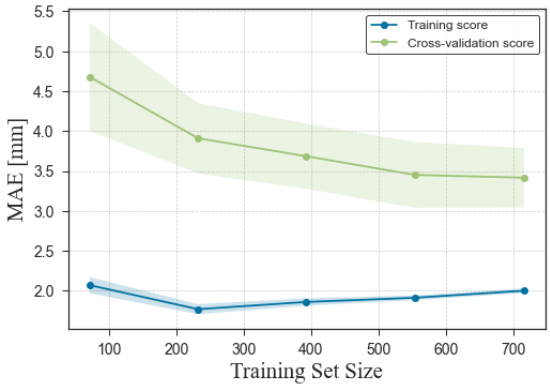


Figure B.12: Learning curve for Random Forest

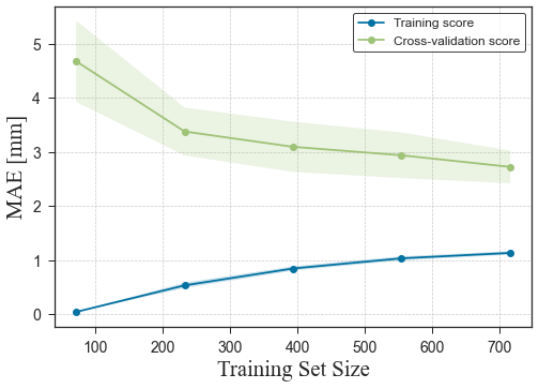


Figure B.13: Learning curve for Histogram based gradient boost.

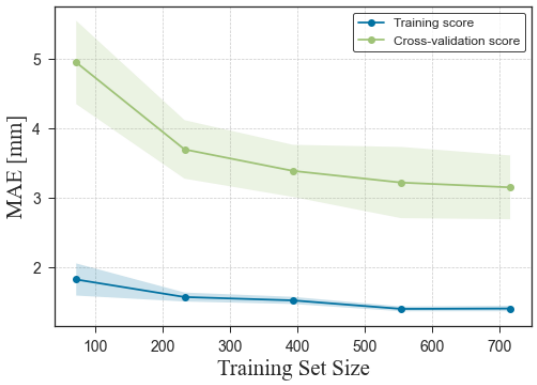


Figure B.14: Learning curve for XG Boost

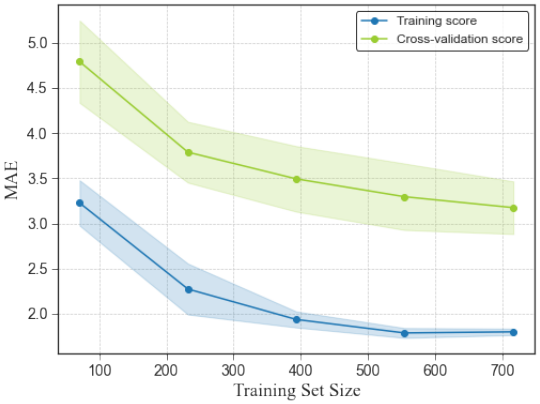


Figure B.15: Learning curve for Gaussian process regressor.

B.2.2 HIC15

Table B.2: Comparison of MAE and residuals for HIC15 of different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	175.637	158.006	166.840	424.066	151.762	401.607
Ridge	172.333	180.902	-1.548	432.506	-40.280	452.316
Random Forest	63.785	89.037	49.670	183.340	51.151	256.439
Gradient Boost	41.531	93.435	18.240	125.647	31.575	251.253
XG Boost	56.387	73.440	35.188	159.833	41.329	217.314
Histogram Based Gradient Boost	40.101	74.271	25.067	118.629	38.778	215.323
Gaussian Process Regressor	45.042	88.272	17.321	129.355	-2.485	250.895

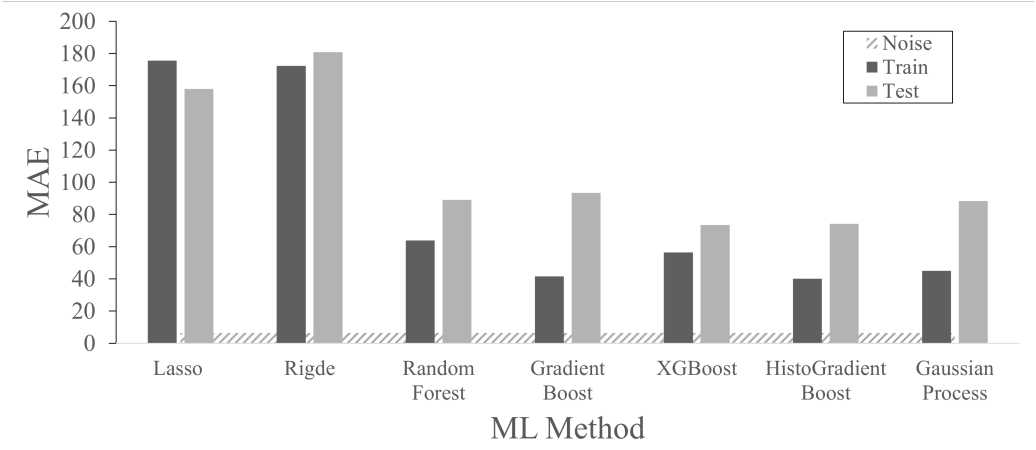


Figure B.16: Model comparisons for MAE in HIC15 predictions

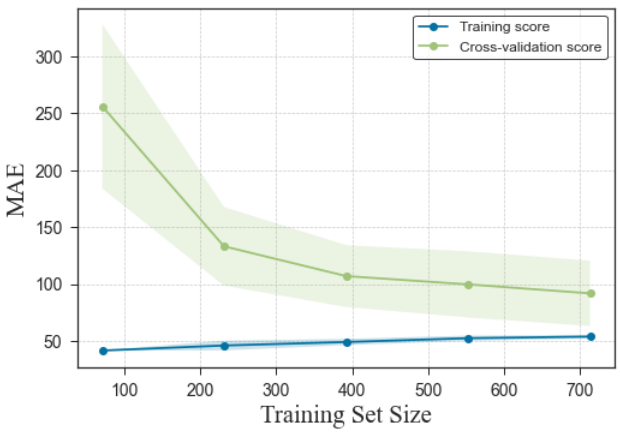


Figure B.17: Learning curve for HIC15 values using XG boosting

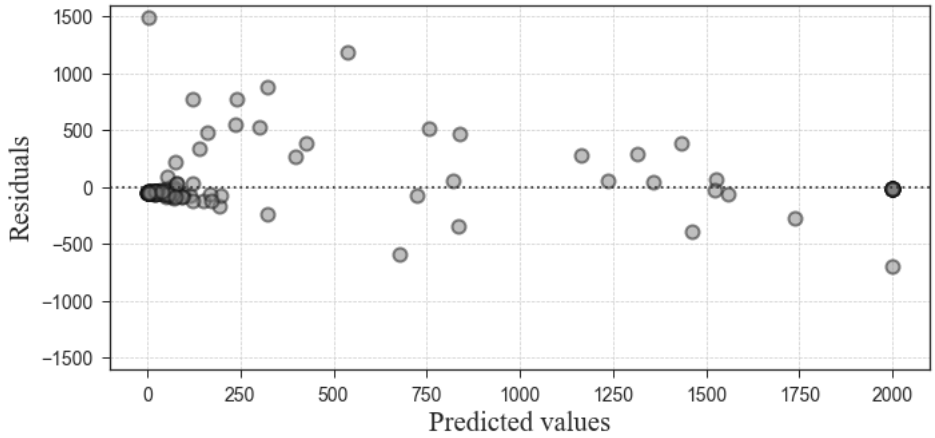


Figure B.18: Residual plots of the prediction and the training using XG boost for HIC15 values

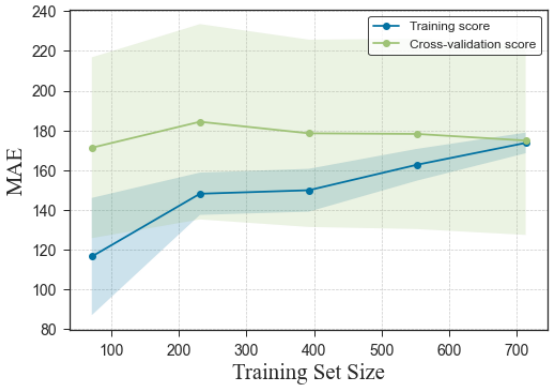


Figure B.19: Learning curve for lasso

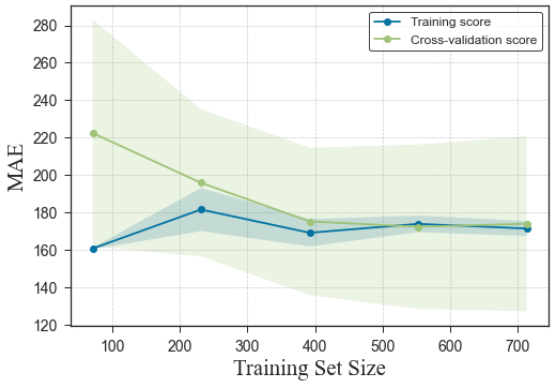


Figure B.20: Learning curve for Ridge.

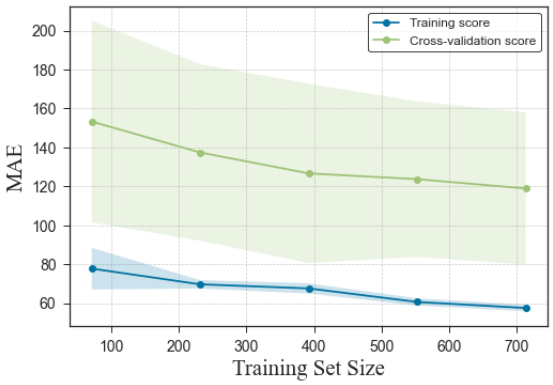


Figure B.21: Learning curve for Random Forest

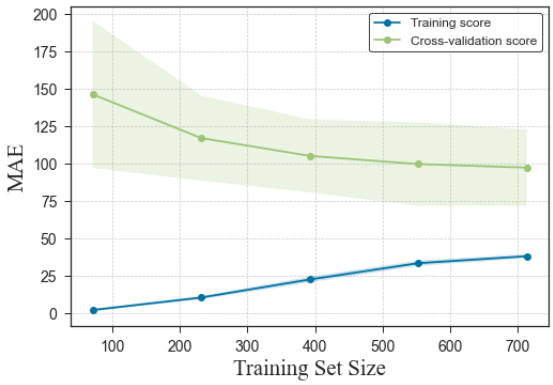


Figure B.22: Learning curve for Histogram based gradient boost.

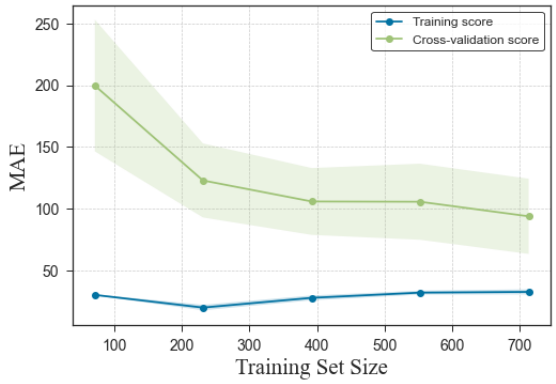


Figure B.23: Learning curve for XG Boost

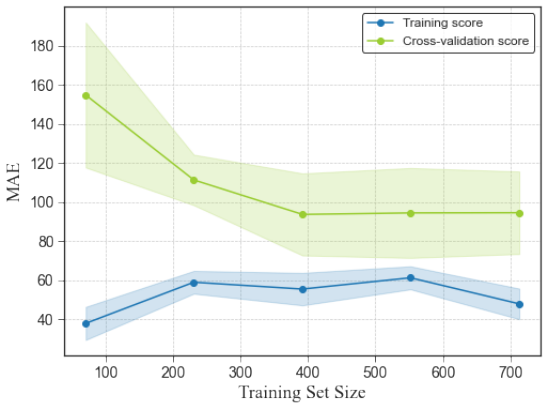


Figure B.24: Learning curve for Gaussian process regressor.

B.2.3 Maximum head acceleration

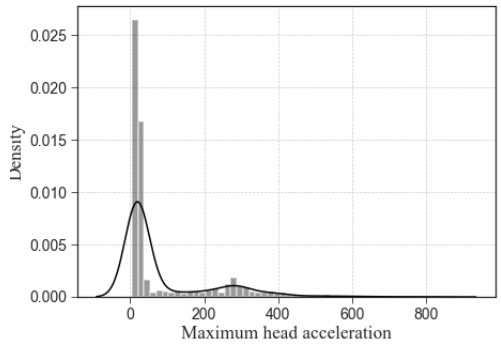


Figure B.25: Post processed head acceleration data distribution

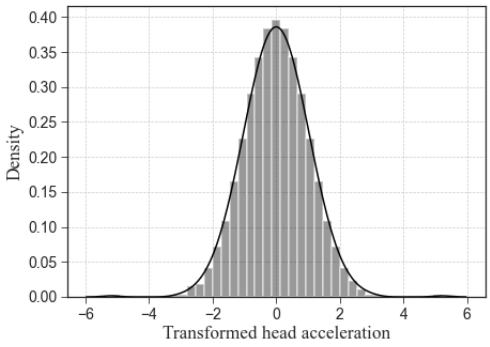


Figure B.26: Distribution of transformed head acceleration

Table B.3: Comparison of MAE and residuals for max head acceleration [g] with different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	38.102	38.840	7.287	85.955	7.634	85.149
Ridge	38.110	38.894	7.180	85.989	7.257	85.070
Random Forest	24.771	22.511	7.357	55.606	-4.121	52.902
Gradient Boost	22.728	20.114	1.535	49.452	-7.428	55.027
XG Boost	18.174	18.439	4.052	44.493	-2.694	46.269
Histogram Based Gradient Boost	17.353	19.592	3.309	41.752	-3.062	48.614
Gaussian Process Regressor	13.161	17.014	0.804	33.672	-2.091	39.485

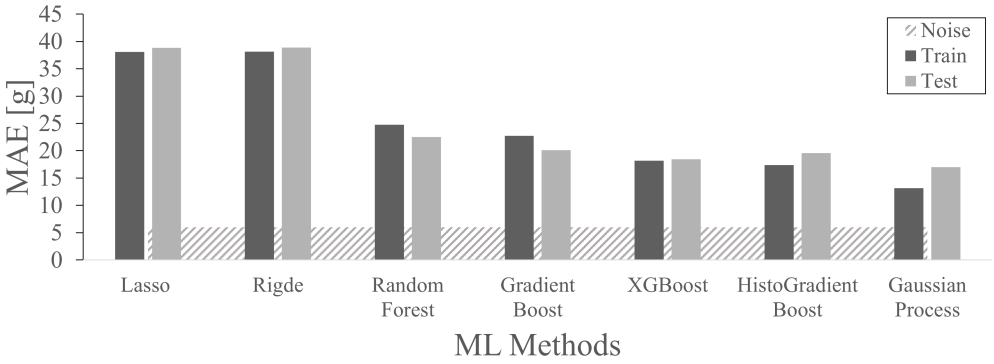


Figure B.27: Bar graph for maximum head acceleration Vs MAE comparing the different selected ML models

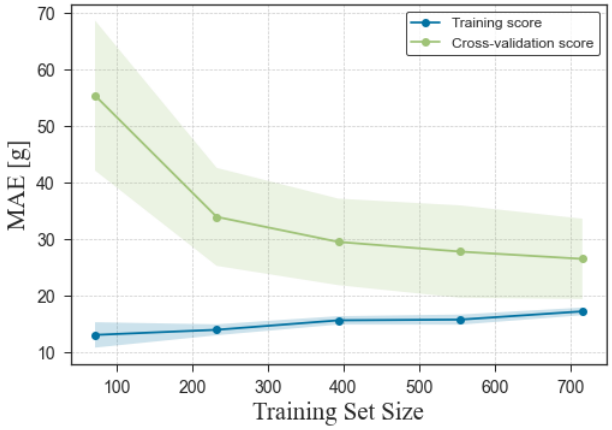


Figure B.28: Learning curve for maximum head acceleration [g] values using XGBoost

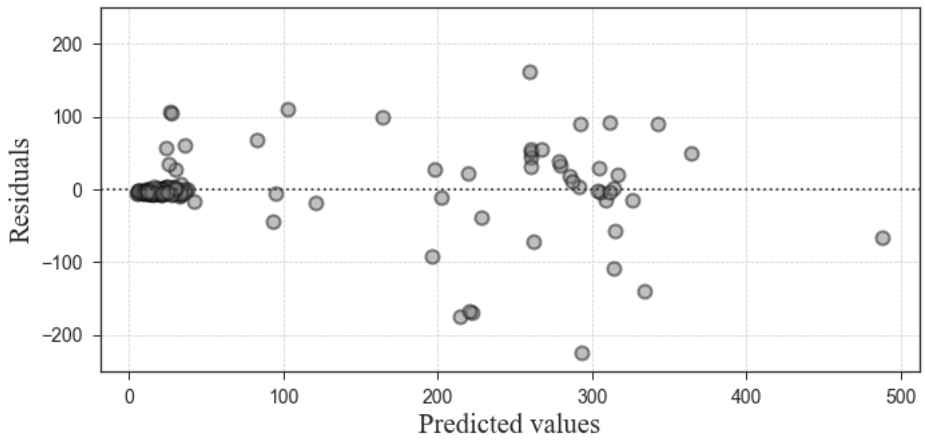


Figure B.29: Residual plots of the prediction and the training using XGBoost for max head acceleration

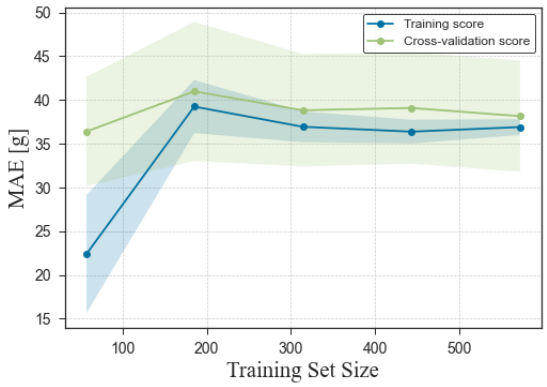


Figure B.30: Learning curve for lasso

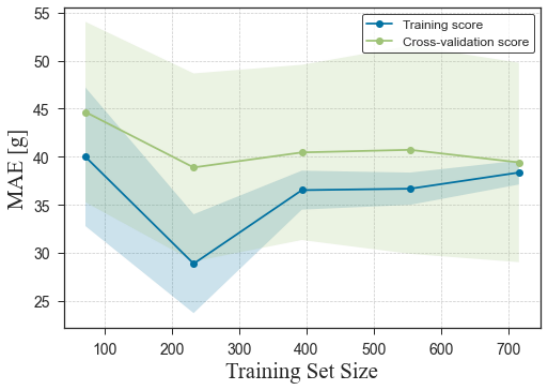


Figure B.31: Learning curve for Ridge.

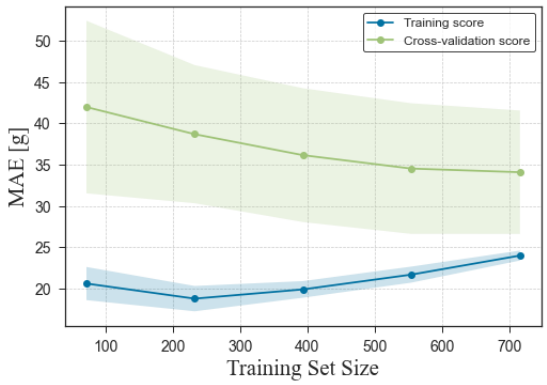


Figure B.32: Learning curve for Random Forest

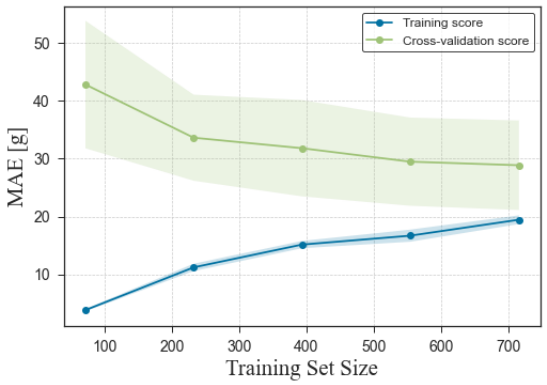


Figure B.33: Learning curve for gradient boost.

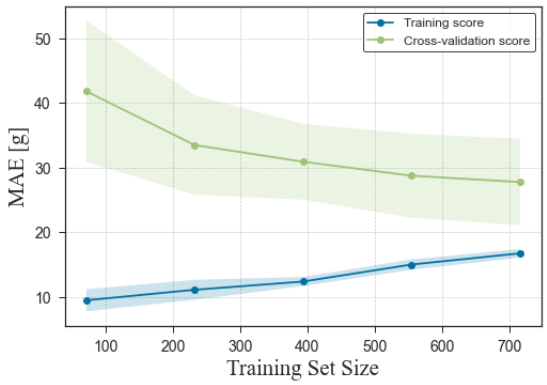


Figure B.34: Learning curve for histogram based gradient Boost

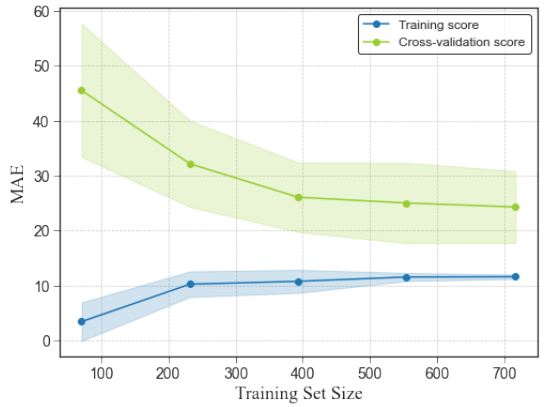


Figure B.35: Learning curve for Gaussian process regressor.

B.2.4 Maximum pelvis acceleration

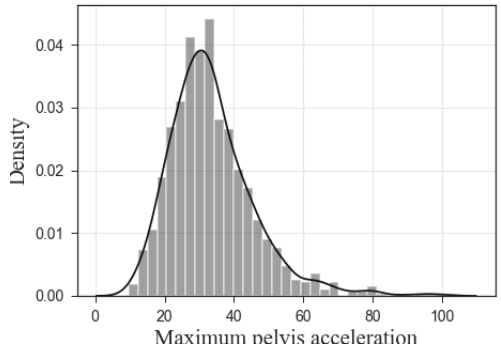


Figure B.36: Distribution of post processed pelvis acceleration data

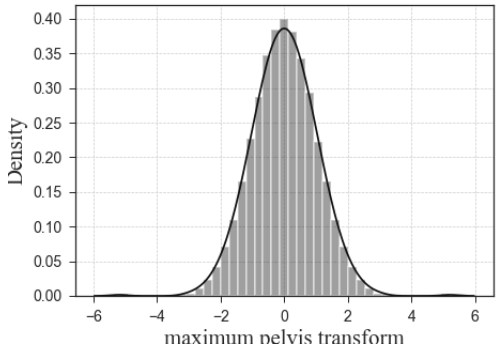


Figure B.37: Distribution transformed head acceleration

Table B.4: Comparison of MAE and residuals for maximum pelvis acceleration [g] for different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	5.240	5.320	0.520	7.552	0.330	7.741
Ridge	5.270	5.060	0.487	7.729	0.362	7.098
Random Forest	3.520	3.940	0.665	5.546	0.231	5.392
Gradient Boost	3.200	3.800	0.539	5.223	0.356	5.130
XG Boost	1.900	3.700	0.533	4.911	0.380	5.360
Histogram Based Gradient Boost	2.200	3.700	0.418	4.097	0.984	5.025
Gaussian Process Regressor	5.810	5.650	0.749	7.903	0.532	7.116

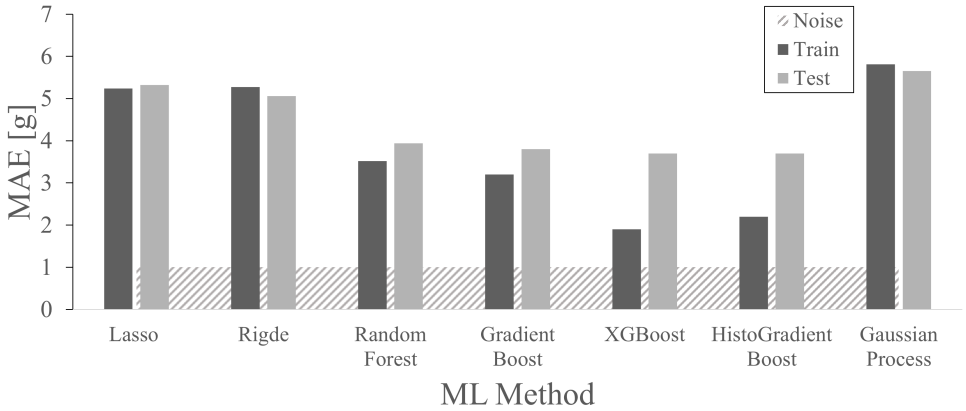


Figure B.38: Model comparisons for MAE of maximum pelvis acceleration

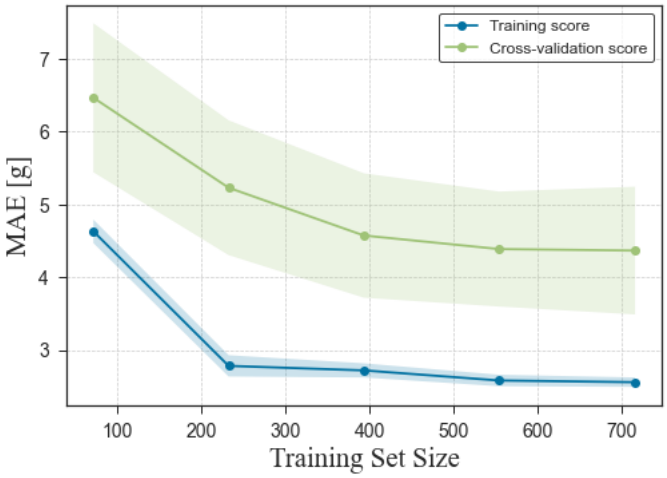


Figure B.39: Learning curve for maximum pelvis acceleration values using XG-Boost

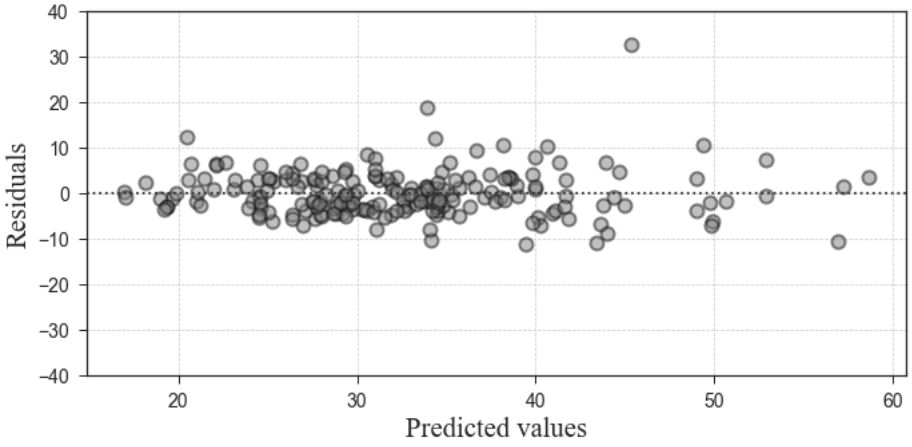


Figure B.40: Residual plots of the prediction and the training using XG boost for max pelvis acceleration

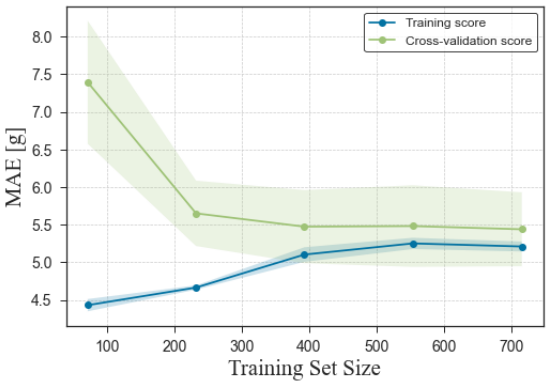


Figure B.41: Learning curve for lasso

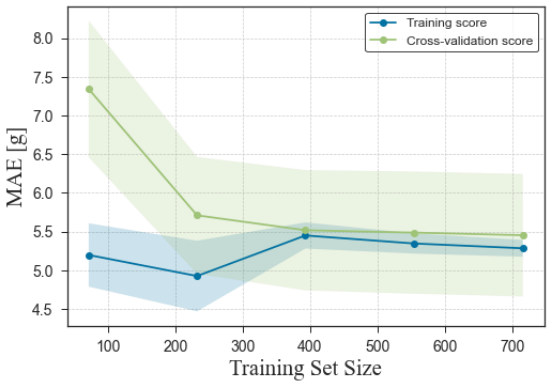


Figure B.42: Learning curve for Ridge.

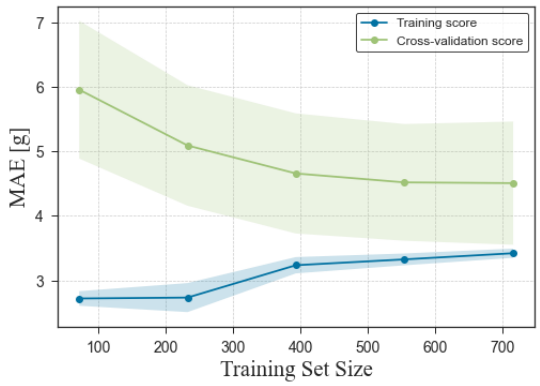


Figure B.43: Learning curve for Random Forest

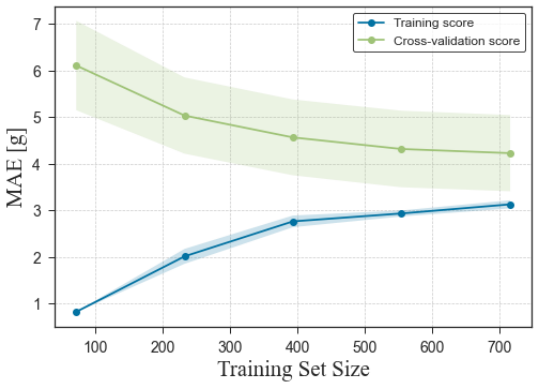


Figure B.44: Learning curve for gradient boost.

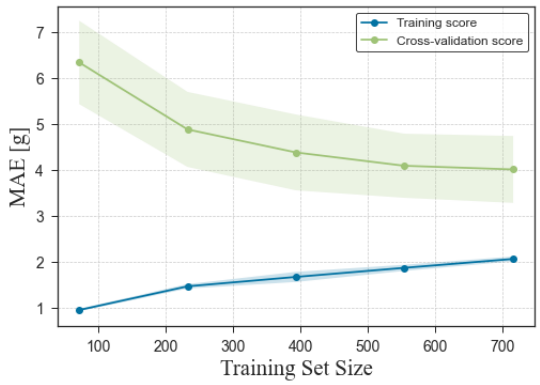


Figure B.45: Learning curve for histogram based gradient Boost

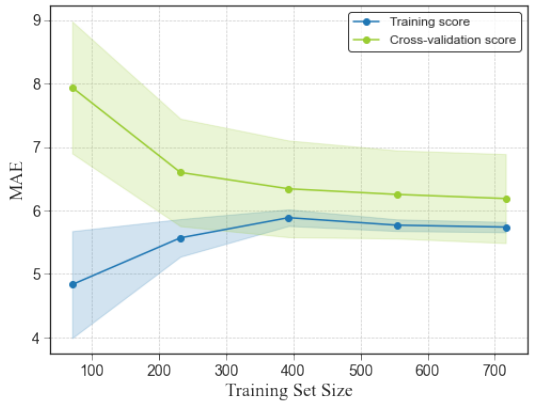


Figure B.46: Learning curve for Gaussian process regressor.

B.2.5 Rib fracture risk

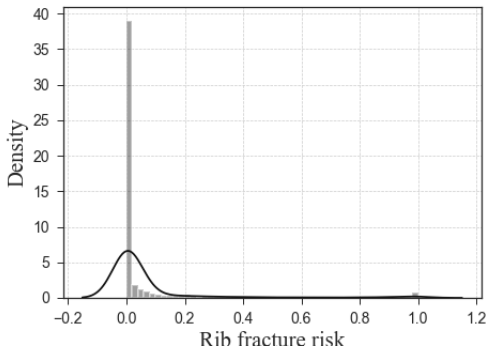


Figure B.47: Distribution of post processed Rib risk

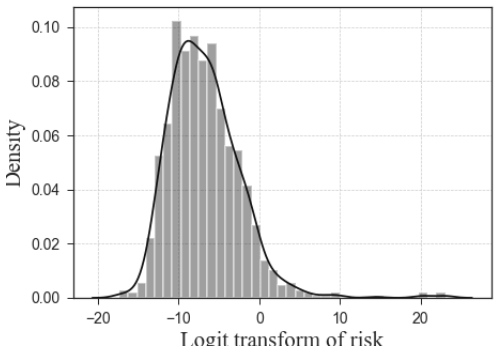


Figure B.48: Distribution transformed rib risk

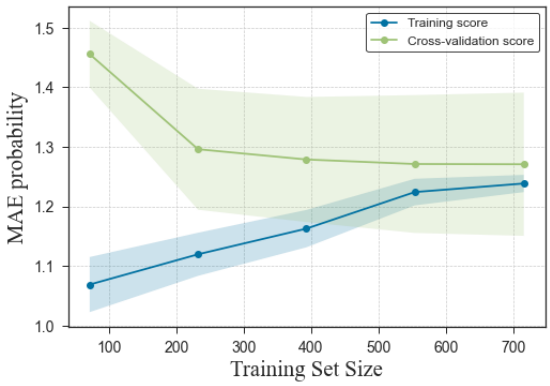


Figure B.49: Learning curve for lasso

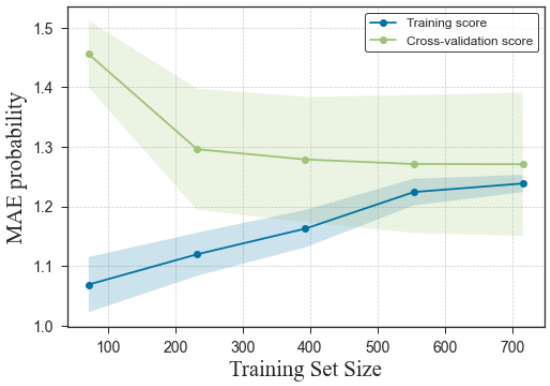


Figure B.50: Learning curve for Ridge.

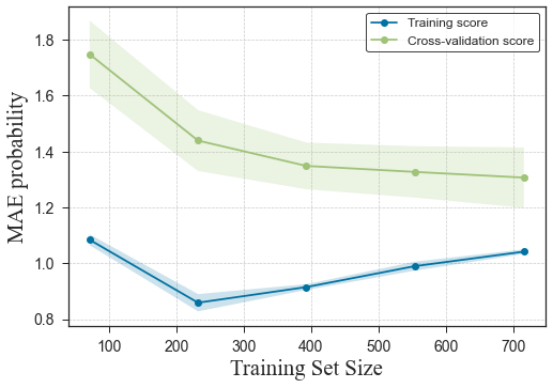


Figure B.51: Learning curve for Random Forest

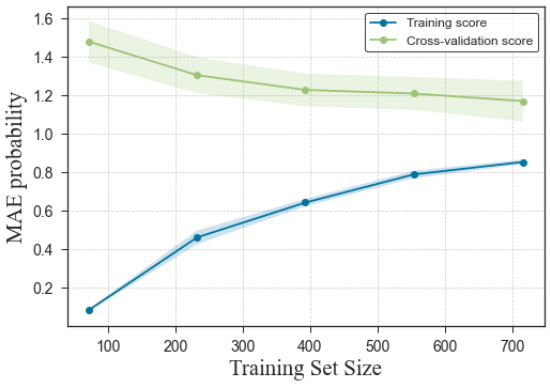


Figure B.52: Learning curve for gradient boost.

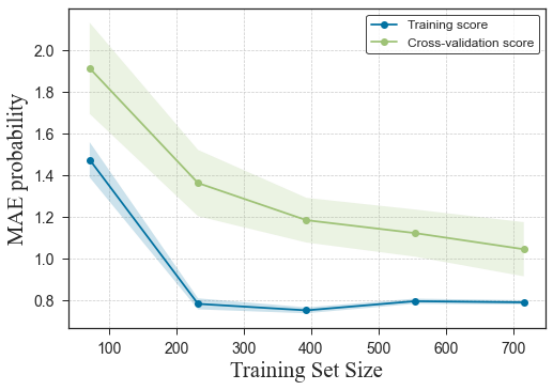


Figure B.53: Learning curve for XGBoost

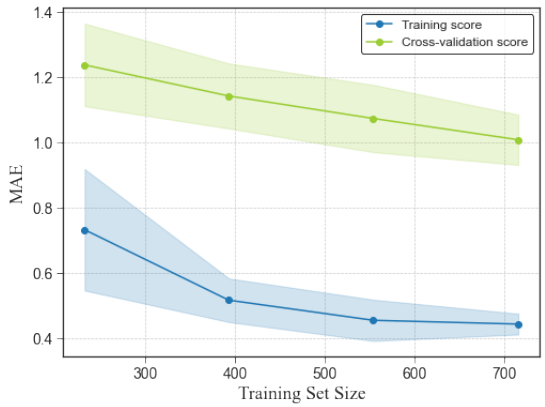


Figure B.54: Learning curve for Gaussian process regressor.

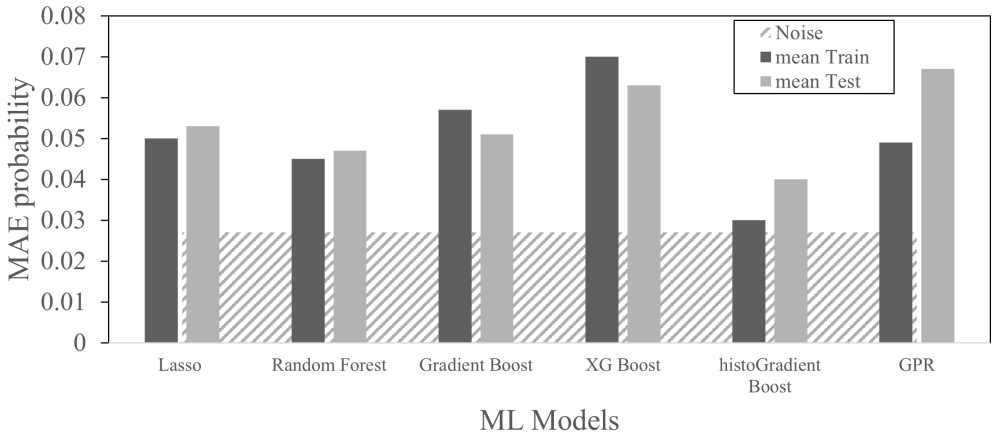


Figure B.55: Rib risk evaluated with quantile transformation

B.3 Rib strains results

Table B.5: Skewness of the maximum strains data of the 24 ribs, the corrected value and the transform used.

Parameter	Initial skewness	Final Skewness	Transformer
Left Rib 1	2.444	0.863	log
Left Rib 2	6.456	1.505	log
Left Rib 3	3.390	0.688	log
Left Rib 4	2.356	0.506	log
Left Rib 5	6.685	0.874	log
Left Rib 6	5.715	0.704	log
Left Rib 7	7.290	1.311	log
Left Rib 8	3.388	1.311	log
Left Rib 9	3.158	1.349	log
Left Rib 10	4.119	0.918	log
Left Rib 11	3.175	0.603	log
Left Rib 12	3.215	0.639	log
Right Rib 1	2.666	0.492	log
Right Rib 2	6.000	0.877	log
Right Rib 3	6.449	0.388	log
Right Rib 4	5.744	0.427	log
Right Rib 5	7.446	0.561	log
Right Rib 6	3.909	0.407	log
Right Rib 7	1.983	0.139	log
Right Rib 8	1.982	0.263	log
Right Rib 9	2.970	0.303	log
Right Rib 10	6.119	1.014	log
Right Rib 11	5.914	0.835	log
Right Rib 12	8.903	2.040	log

B.3.0.1 1^{st} Left Rib

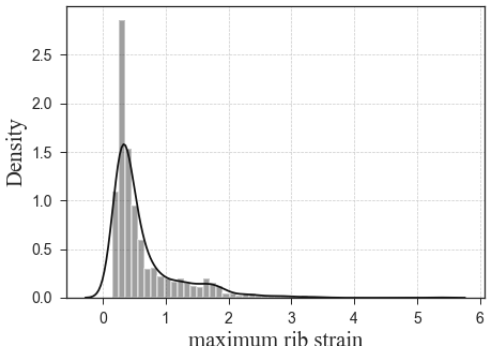


Figure B.56: Distribution of post processed rib strains for the first left rib

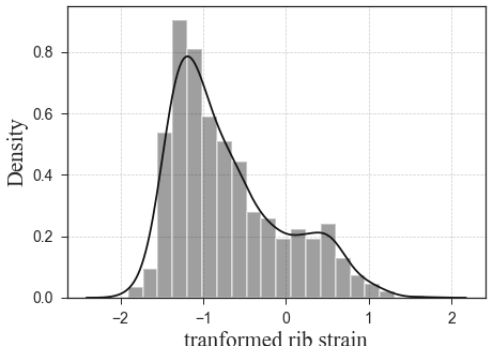


Figure B.57: Distribution transformed rib strains for the first left rib

Table B.6: Comparison of MAE and residuals for maximum rib strains of the 1st left rib applied for different ML models

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	0.198	0.192	0.049	0.380	0.012	0.333
Ridge	0.191	0.196	0.049	0.380	0.012	0.330
Random Forest	0.148	0.146	0.053	0.327	0.024	0.246
Gradient Boost	0.168	0.167	0.091	0.349	0.024	0.246
XG Boost	0.287	0.272	0.053	0.472	0.031	0.380
Histogram Based Gradient Boost	0.121	0.134	0.055	0.272	0.037	0.238
Gaussian Process Regressor	0.079	0.107	0.025	0.169	0.011	0.186

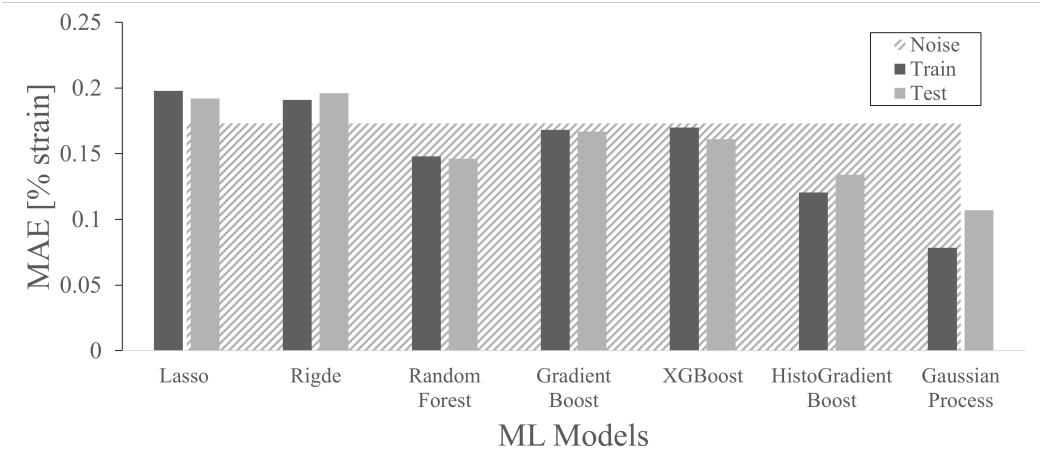


Figure B.58: Model comparisons for HIC15

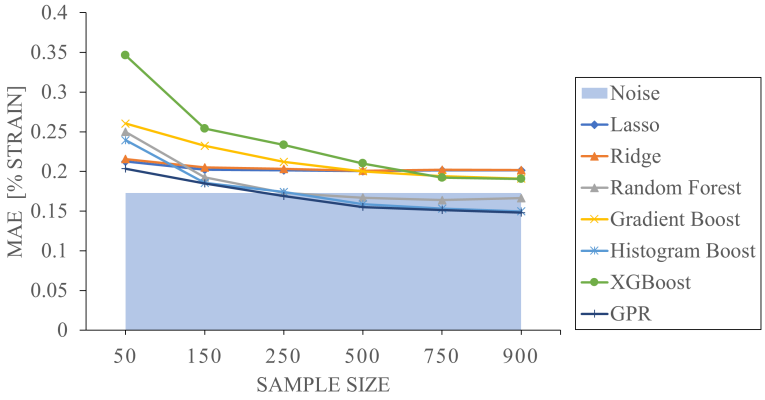


Figure B.59: Plot for MAE of predictions of the test set for different sample sizes for the ML models used

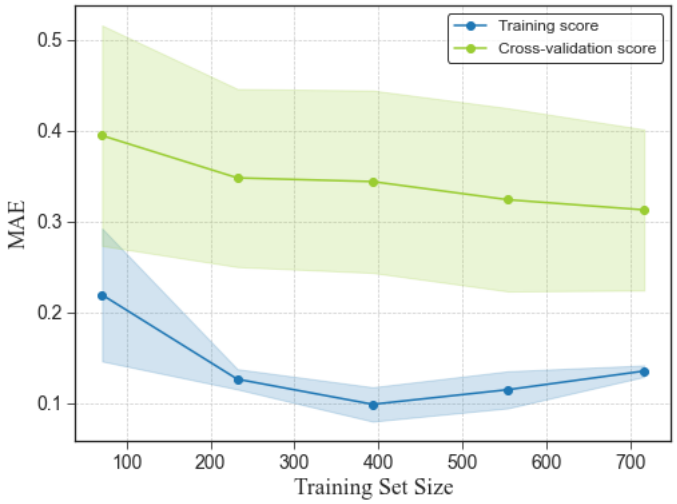


Figure B.60: Learning curve for rib strain of the 1st left rib values using GPR

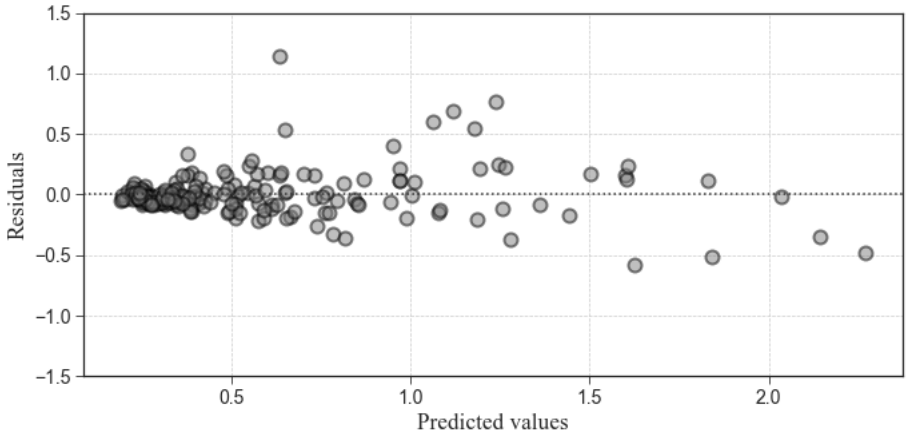


Figure B.61: Residual plots of the prediction and the training for strains of 1st left rib

B.3.0.2 12^{th} Right Rib

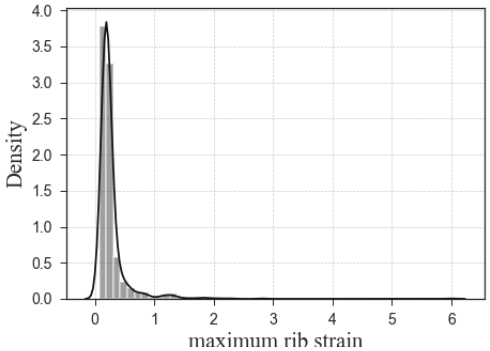


Figure B.62: Distribution of post processed rib strains for the first left rib

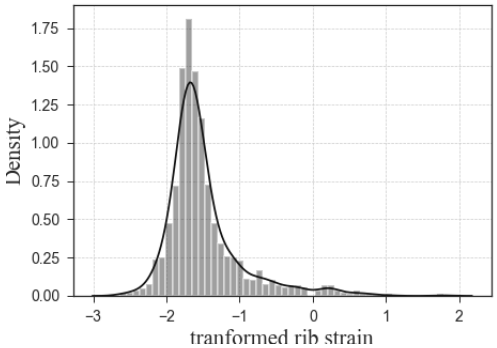


Figure B.63: Distribution transformed rib strains for the first left rib

ML Method	MAE		Residual			
	Train	Test	Train		Test	
			mean	std	mean	std
Lasso	0.086	0.078	0.032	0.268	0.006	0.193
Ridge	0.088	0.069	0.030	0.271	0.005	0.139
Random Forest	0.053	0.052	0.023	0.200	0.005	0.116
Gradient Boost	0.064	0.061	0.025	0.237	0.012	0.127
XG Boost	0.054	0.050	0.017	0.217	0.000	0.108
Histogram Based Gradient Boost	0.037	0.046	0.015	0.158	0.006	0.110
Gaussian Process Regressor	0.038	0.056	0.017	0.217	0.000	0.108

Table B.7: Comparison of MAE and residuals for maximum rib strains of the 12th right rib applied for different ML models

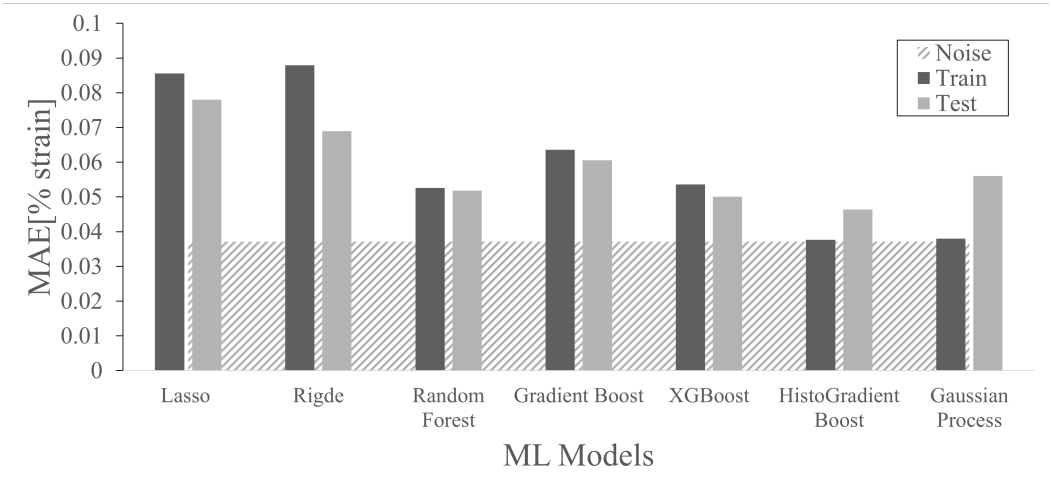


Figure B.64: Model comparisons for strains in 12th Right rib

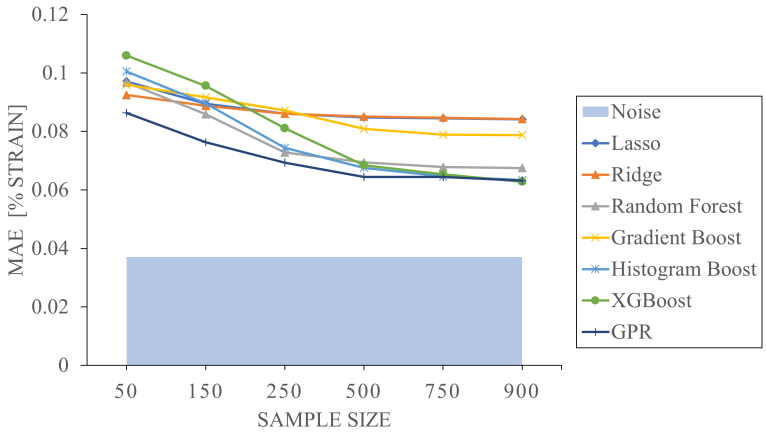


Figure B.65: Plot for MAE of predictions of the test set for different sample sizes for the ML models used

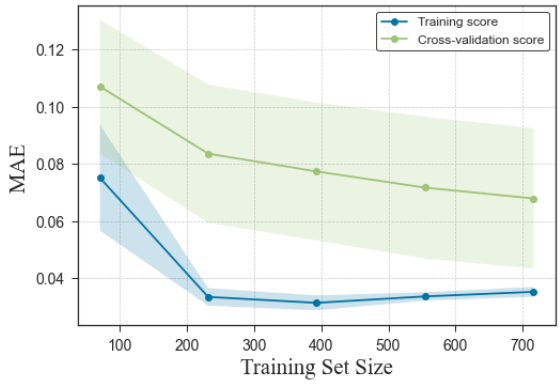


Figure B.66: Learning curve for rib strain of the 12^{th} right rib values using Histogram based gradient boost

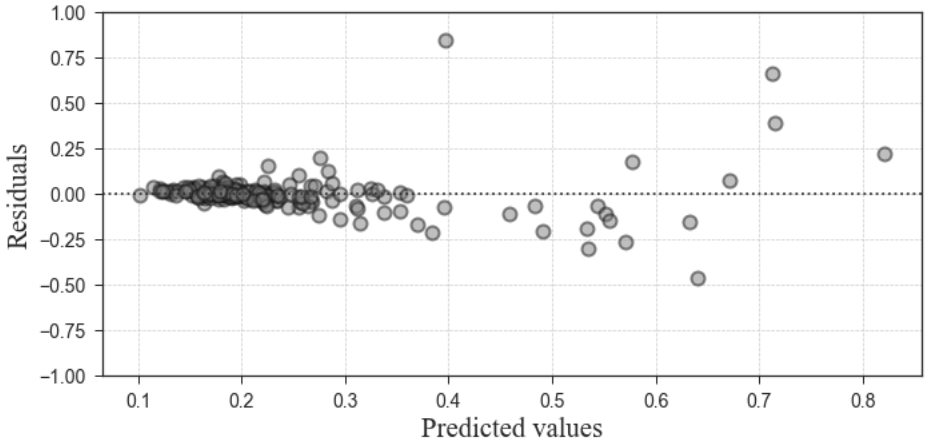


Figure B.67: Residual plots of the prediction and the training for the 2th right rib strains

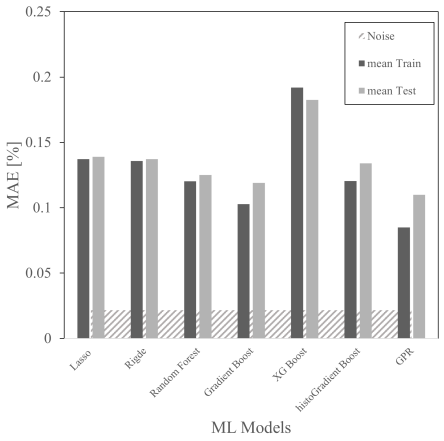


Figure B.68: Left rib 2 surrogates MAE

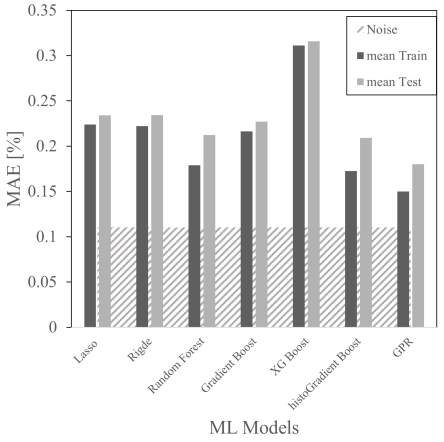


Figure B.69: Left rib 3 surrogates MAE

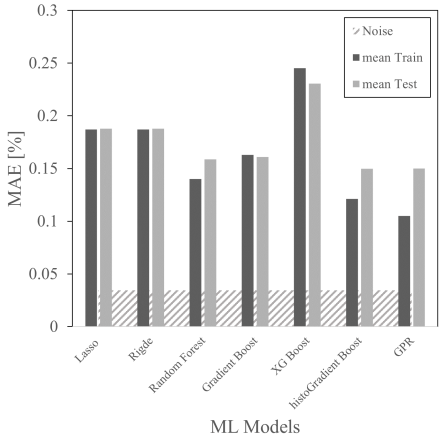


Figure B.70: Left rib 4 surrogates MAE

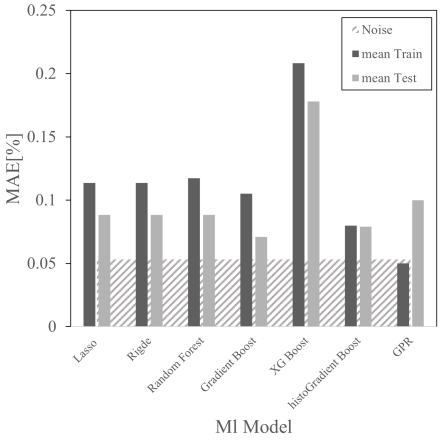


Figure B.71: Left rib 5 surrogates MAE

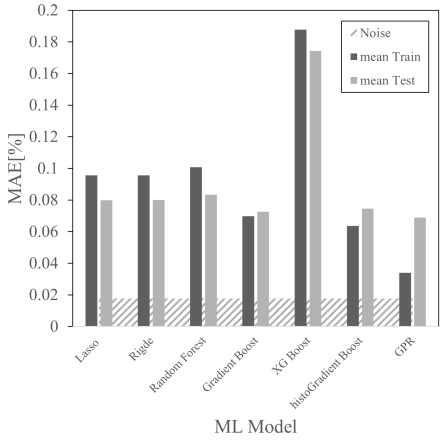


Figure B.72: Left rib 6 surrogates MAE

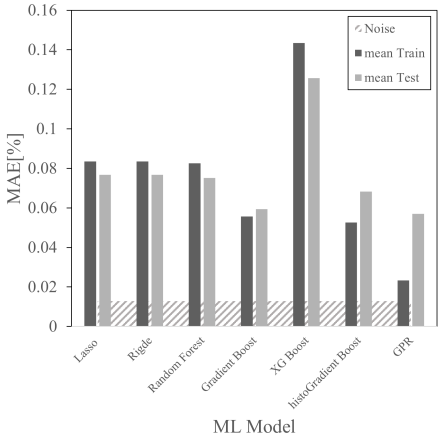


Figure B.73: Left rib 7 surrogates MAE

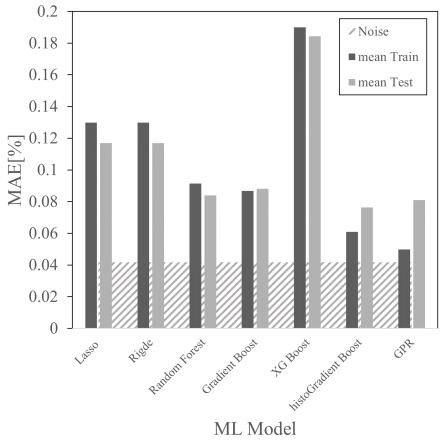


Figure B.74: Left rib 8 surrogates MAE

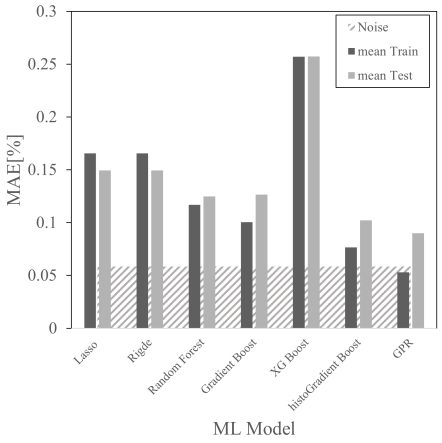


Figure B.75: Left rib 9 surrogates MAE

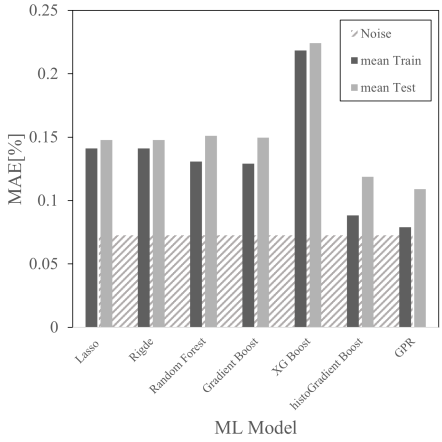


Figure B.76: Left rib 10 surrogates MAE

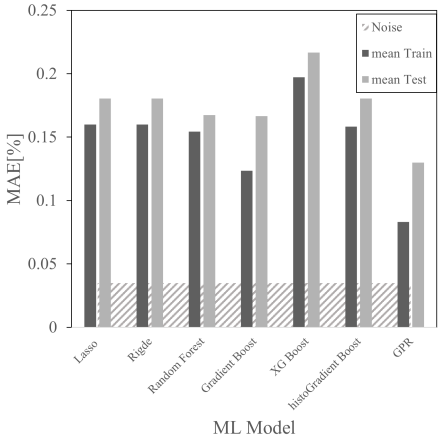


Figure B.77: Left rib 11 surrogates MAE

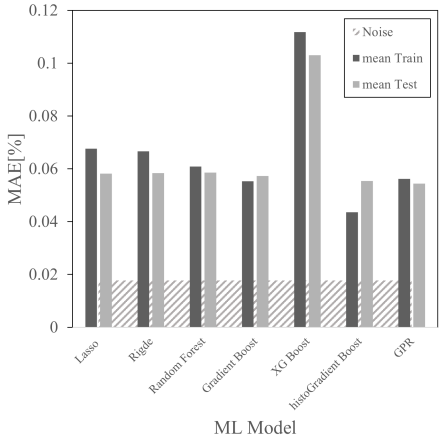


Figure B.78: Left rib 12 surrogates MAE

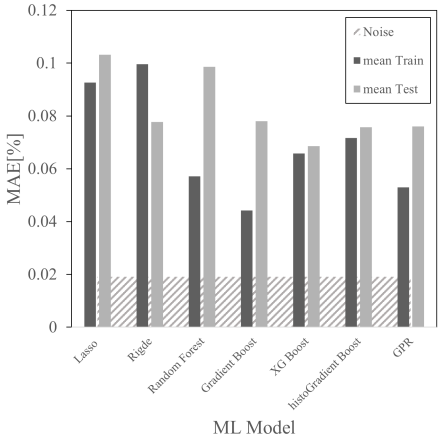


Figure B.79: Right rib 1 surrogates MAE

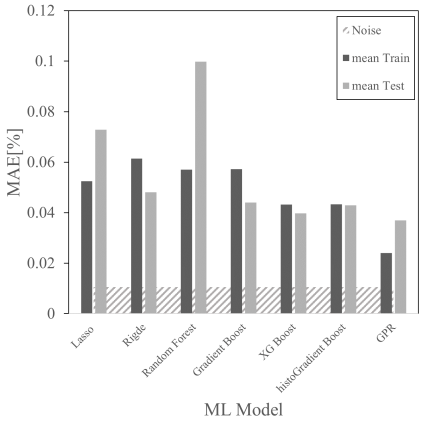


Figure B.80: Right rib 2 surrogates MAE

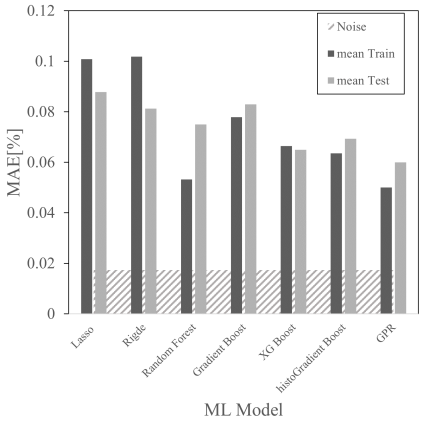


Figure B.81: Right rib 3 surrogates MAE

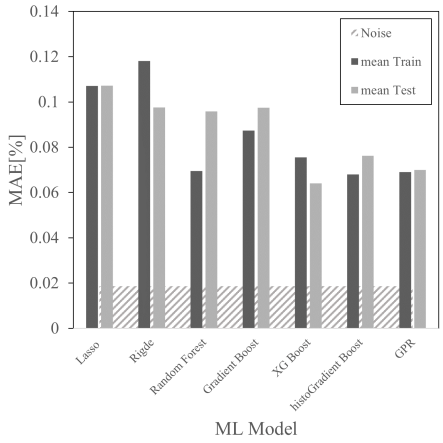


Figure B.82: Right rib 4 surrogates MAE

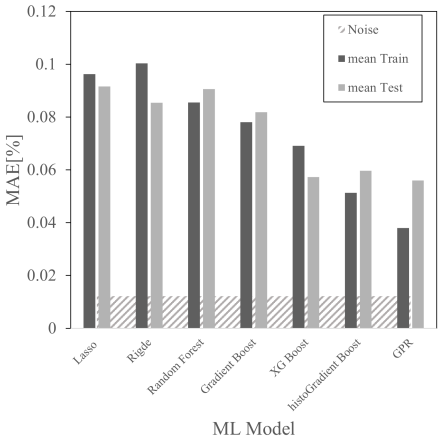


Figure B.83: Right rib 5 surrogates MAE

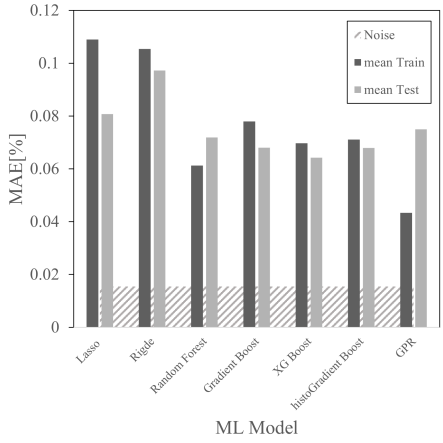


Figure B.84: Right rib 6 surrogates MAE

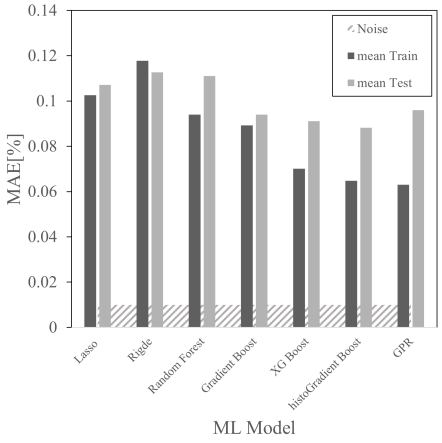


Figure B.85: Right rib 7 surrogates MAE

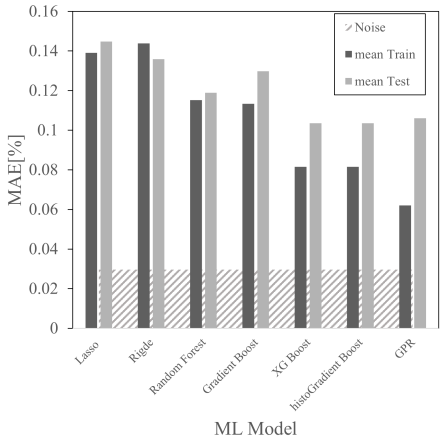


Figure B.86: Right rib 8 surrogates MAE

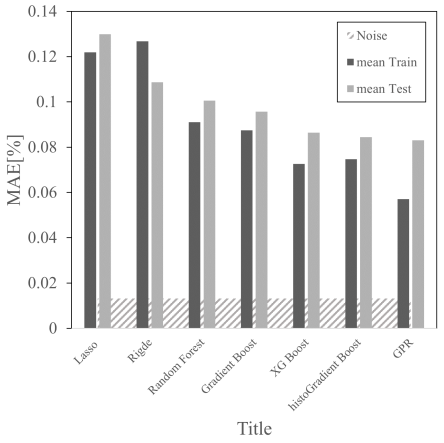


Figure B.87: Right rib 9 surrogates MAE

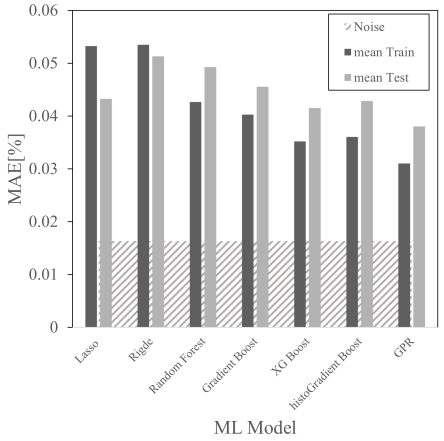


Figure B.88: Right rib 10 surrogates MAE

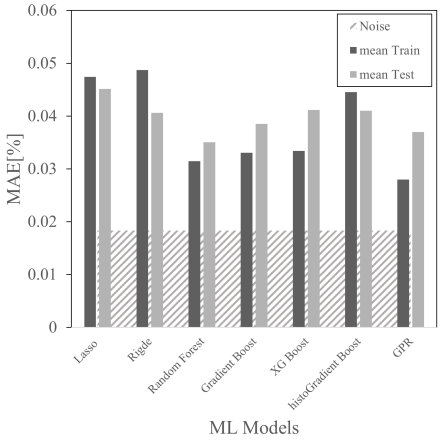


Figure B.89: Right rib 11 surrogates MAE

Table B.8: Mean absolute Error for the Left ribs using different methods

	Left Rib 1		Left Rib 2		Left Rib 3		Left Rib 4		Left Rib 5		Left Rib 6		Left Rib 7		Left Rib 8		Left Rib 9		Left Rib 10		Left Rib 11		Left Rib 12	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Lasso	0.198	0.192	0.137	0.139	0.224	0.234	0.187	0.188	0.114	0.088	0.096	0.080	0.083	0.077	0.130	0.117	0.166	0.149	0.141	0.148	0.160	0.180	0.068	0.058
Ridge	0.246	0.242	0.136	0.137	0.222	0.234	0.187	0.188	0.114	0.088	0.096	0.080	0.083	0.077	0.130	0.117	0.166	0.149	0.141	0.148	0.160	0.180	0.067	0.058
RandomForest	0.148	0.146	0.120	0.125	0.179	0.212	0.140	0.159	0.117	0.088	0.101	0.083	0.083	0.075	0.091	0.084	0.117	0.125	0.131	0.151	0.154	0.167	0.061	0.059
Gradient Boost	0.168	0.167	0.103	0.119	0.216	0.227	0.163	0.161	0.105	0.071	0.070	0.073	0.056	0.059	0.087	0.088	0.101	0.126	0.129	0.150	0.123	0.166	0.055	0.057
XG Boost	0.287	0.272	0.192	0.183	0.311	0.316	0.245	0.231	0.208	0.178	0.188	0.174	0.143	0.126	0.190	0.184	0.257	0.257	0.218	0.224	0.197	0.217	0.112	0.103
Histogram based Gradient Boost	0.121	0.134	0.121	0.134	0.173	0.209	0.121	0.150	0.080	0.079	0.064	0.075	0.053	0.068	0.061	0.076	0.077	0.102	0.088	0.119	0.158	0.181	0.044	0.055
Gaussian process Regressor	0.079	0.107	0.085	0.110	0.150	0.180	0.105	0.150	0.050	0.100	0.034	0.069	0.023	0.057	0.050	0.081	0.053	0.090	0.079	0.109	0.083	0.130	0.056	0.054

Table B.9: Mean absolute Error for the Right ribs using different methods

	Right Rib 1		Right Rib 2		Right Rib 3		Right Rib 4		Right Rib 5		Right Rib 6		Right Rib 7		Right Rib 8		Right Rib 9		Right Rib 10		Right Rib 11		Right Rib 12	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Lasso	0.093	0.103	0.052	0.073	0.101	0.088	0.107	0.107	0.096	0.092	0.109	0.081	0.103	0.107	0.139	0.145	0.122	0.130	0.053	0.043	0.047	0.045	0.086	0.078
Ridge	0.100	0.078	0.061	0.048	0.102	0.081	0.118	0.098	0.100	0.085	0.105	0.097	0.118	0.113	0.144	0.136	0.127	0.109	0.053	0.051	0.049	0.041	0.088	0.069
RandomForest	0.057	0.099	0.057	0.100	0.053	0.075	0.070	0.096	0.086	0.091	0.061	0.072	0.094	0.111	0.115	0.119	0.919	0.101	0.043	0.049	0.031	0.035	0.053	0.052
Gradient Boost	0.044	0.078	0.057	0.044	0.078	0.083	0.087	0.098	0.078	0.082	0.078	0.685	0.089	0.094	0.113	0.130	0.087	0.096	0.040	0.046	0.033	0.039	0.064	0.061
XG Boost	0.066	0.069	0.043	0.040	0.066	0.065	0.075	0.064	0.069	0.057	0.070	0.064	0.070	0.091	0.081	0.104	0.073	0.086	0.035	0.042	0.033	0.041	0.054	0.050
Histogram based Gradient Boost	0.072	0.076	0.043	0.043	0.064	0.069	0.068	0.076	0.051	0.060	0.071	0.068	0.065	0.088	0.081	0.104	0.075	0.084	0.036	0.043	0.045	0.041	0.038	0.046
Gaussian process Regressor	0.053	0.076	0.024	0.037	0.050	0.060	0.069	0.070	0.038	0.056	0.043	0.075	0.063	0.096	0.062	0.106	0.057	0.083	0.031	0.038	0.028	0.037	0.049	0.067

Table B.10: Finalised ML models for maximum chest deflection, HIC15, maximum head acceleration, maximum pelvis acceleration, rib fracture risk and rib strains.

Output	Final model
Maximum chest deflection	**GradientBoostingRegressor**(learning_rate=0.03, max_features='auto',min_samples_leaf=0.05, min_samples_split=0.1,n_estimators=300, subsample=0.9,max_depth=6)
HIC15	**XGBRegressor**(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.8, gamma=0.075, gpu_id=-1,importance_type='gain', interaction_constraints='', learning_rate=0.04, max_delta_step=0, max_depth=3,min_child_weight=15, monotone_constraints='()', n_estimators=350, n_jobs=12, num_parallel_tree=1,random_state=0,reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.8,tree_method='exact', validate_parameters=1, verbosity=None)
Maximum head acceleration	**XGBRegressor**(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.8, gamma=0.075, gpu_id=-1,importance_type='gain', interaction_constraints='', learning_rate=0.025, max_delta_step=0, max_depth=3,min_child_weight=15, monotone_constraints='()' ,n_estimators=300, n_jobs=12, num_parallel_tree=1, random_state=0,reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=0.9,tree_method='exact', validate_parameters=1, verbosity=None)
Maximum pelvis acceleration	**XGBRegressor**(base_score=0.5, booster='gbtree', colsample_bylevel=1,colsample_bynode=1, colsample_bytree=0.8, gamma=0.05, gpu_id=-1,importance_type='gain', interaction_constraints='', learning_rate=0.01, max_delta_step=0, max_depth=7,min_child_weight=15, monotone_constraints='()', n_estimators=380, n_jobs=12, num_parallel_tree=1, random_state=0,reg_alpha=0, reg_lambda=5, scale_pos_weight=1, subsample=0.8, tree_method='exact',validate_parameters=1, verbosity=None)
2+ fracture Rib risk 45yrs+	**HistGradientBoostingRegressor**(learning_rate=0.06, max_depth=1, max_iter=300,max_leaf_nodes=7, min_samples_leaf=9,scoring='neg_root_mean_squared_error')
Rib strain (Left rib1)	**GaussianProcessRegressor**(n_restarts_optimizer=4, normalize_y=True,random_state=0, kernel=RationalQuadratic(alpha=1, length_scale=1) ** 2 + WhiteKernel(noise_level=0.5))
Rib strain (Right rib12)	**HistGradientBoostingRegressor**(learning_rate=0.05, max_depth=3, max_iter=300, max_leaf_nodes=9, min_samples_leaf=7,scoring='neg_root_mean_squared_error',l2_regularization=15)

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY