1 Introduction and Background

In the last several decades, the need for the implementation of blast-resistant design in major construction projects has been recognized by observing numerous terrorist attacks on civilian structures (Ullah et al., 2017) and accidental explosions such as the 2020 Beirut explosion (Rigby, et al., 2020) and the 2015 Tianjin explosion. At present, various government, federal, and defense agencies in the United States enforce blast design guidelines for buildings with exceptional threat, vulnerability, and impact characteristics. The Unified Facilities Criteria (UFC), U.S. Army Corps of Engineers Protective Design Center Technical Report (PDC-TR), the Interagency Security Committee (ISC), U.S General Services Administration (GSA), U.S Department of Veterans Affairs Physical Security and Resiliency Design Manual (PSRDM) are among the main design procedures and construction techniques available to ensure blast hardening and provide protection for personnel and valuable equipment. Although these empirical and semi-empirical based design manuals are ideal for simple building configurations, the prediction of blast loads in complex geometries is typically carried out with Computational Fluid Dynamics (CFD) solvers (Remennikov and Rose, 2007). CFD simulations are usually performed by experienced computational modelers and are incredibly time-consuming. Consequently, CFD models cannot be used for interactive design and do not allow the engineer to consider more than a few alternative setups (Flood et al., 2009).

The combination of high-fidelity model simulations, experimental data, or both with machine learning techniques has been shown to provide fast and reliable results for non-ideal blast loading configurations. This is of great importance, as it allows engineers to

optimize the design or retrofit of buildings in terms of blast-mitigation performance and cost-effectiveness in a matter of minutes, as opposed to days (Flood et al., 2009).

One of the earliest uses of machine learning methods in the field of protective design is related to the prediction of blast loading on a building behind a solid barrier. Data from experimental measurements (Remennikov and Rose, 2007), DYSMAS simulations (Bewick et al., 2011), and PureWall models (Flood et al., 2009) were used to train Artificial Neural Networks (ANNs) (Zou et al., 2008) to predict a variety of blast properties (e.g., peak overpressure, peak impulse, positive phase duration, arrival time) behind a barrier. Results for peak overpressure prediction for relatively larger datasets were promising, with R^2 score values greater or equal to 0.996.

Other studies have demonstrated the use of machine learning methods for predicting the response of reinforced concrete slabs (Almustafa and Nehdi, 2020) and steel plates (Neto et al., 2020) exposed to blast loading. Data retrieved from experimental programs and numerical models were used to train and test linear regression, Support Vector Machine (SVM) (Hearst et al., 1998), and tree-based models to predict the maximum displacement of reinforced concrete slabs subjected to blast loading. The Random Forest (RF) algorithm (Ho, 1998) exhibited superior performance with R^2 score of 0.92. Moreover, neural networks were used to predict the mechanical response of mild steel plates subjected to localized blast loading. Experimental data obtained from literature (Jacob, et al., 2004), combined with Finite Element Analysis (FEA) results were used to train and test ANNs. Most predictions were reported to have an absolute error of less than 10%.

Further applications of neural networks include the prediction of peak impulse in a confined internal environment (Dennis et al., 2021). Data collected from Apollo Blastsimulator was used to develop ANNs with R^2 score predictions of 0.968 on the unseen testing data. Among the highlights of this study is the implementation of L2 regularization and dropout to prevent overfitting and improve the generalizability of ANN models.

In a recent study (Almustafa and Nehdi, 2022), tree-based methods such as Gradient Boosted Decision Trees (GBDT) (Friedman, 2001) and RF algorithms were used to predict the maximum displacement of reinforced concrete columns exposed to blast loading. Despite the use of a blended dataset (experimental, numerical, and analytical data) retrieved from existing literature, the GBDT achieved high performance with R^2 score of 0.974.

Transfer learning (Bozinovski and Fulgosi, 1976) is the reuse of a pre-trained model on a new problem and is becoming the go-to way of working with deep learning models as it reduces training time and tackles issues related to the lack of data. This approach has been applied to a recent study (Pannell et al., 2022a), where previous knowledge learned when modeling spherical charges are transferred to make predictions for near-field peak specific impulse when modeling cylindrical charges.

Hyperparameter tuning is very crucial, as it controls the overall behavior of the machine learning model. It can sometimes also be very time-consuming; therefore, reporting optimum parameters can help other researchers in their efforts for hyperparameter tuning. Present-day studies (Pannell et al., 2022b) have successfully reported this information. In this study, a Physics-Guided Neural Network (PGNN)

incorporating a physics-based regularization term outperformed a traditional neural network in predicting near-field blast loading. The dataset used to train and test the models was generated using the APOLLO Blastsimulator.

Although literature demonstrates the capability of machine learning methods to deliver fast and accurate predictions, notable shortcomings have been identified that the present study aims to address. For example, most existing studies have focused on using neural networks for blast load prediction; and systematic studies that compare the performance of various machine learning models are limited. Despite neural networks becoming incredibly popular in recent years for their ability to accurately model complex data, their interpretation and inference remain challenging (Borisov, et al., 2022). ANNs are composed of many interconnected processing nodes, and it is difficult to understand how the node weights result in the predicted output. Furthermore, data preparation for neural network models requires careful attention. If data is not scaled correctly, it can lead to suboptimal results. This is not the case for tree-based methods, as they are insensitive to the variance in data. Additionally, tree-based methods often require less hyperparameter tuning, and some studies (Shwartz-Ziv and Armon 2022) even suggest that gradient boosting models outperform current deep models on tabular data.

For small and medium-sized data sets (less than 1M samples), ANNs tend to overfit, and simpler machine learning models may perform better (Borisov, et al., 2022) (Haykin, 2009) (Kavzoglu and Mather, 2003). Since collecting a sufficient number of reliable blast data points remains a significant challenge, it is essential to compare the performance of neural networks to other traditional regression models. Many studies to date have used relatively small datasets, some a combination of experimental, blast

simulation, and empirical modeling data. This in turn has affected the overall performance of the trained machine learning models, especially neural network models.

To address the identified deficiencies, in the present study, a CFD blast simulation program is used to measure blast loads on a protruded architectural structure, resulting in a sample size of over 250,000 data points. This façade configuration which is favorable among architects, cannot be analyzed using current blast design manuals and requires CFD analysis. Due to the high sample size, fully-connected neural networks are trained and tested while incorporating the latest data preprocessing, regularization, and hyperparameter turning methods. Neural networks are known for their high predictive power but are less interpretable than conventional regression methods. For that reason, the performance of neural networks is compared to various linear and nonlinear regression methods such as simple linear regression, polynomial regression, RF, GBDT, and Extreme Gradient Boosted Decision Trees (XGBoost) (Chen and Guestrin, 2016).

2 Data Collection Using ProSAir

The three-dimensional blast simulation program ProSAir, a compressible CFD package, was used to collect maximum overpressure and impulse values at various locations of the protruded structural façade. This package has been validated against experimental data and is widely used for blast load modeling (Alterman et al., 2019; Castedo et al., 2019; Remennikov and Rose, 2005). Detailed information on the underlying computational equations and assumptions for ProSAir can be found on the Cranfield University website (ProSAir computational blast loading tool, 2022).

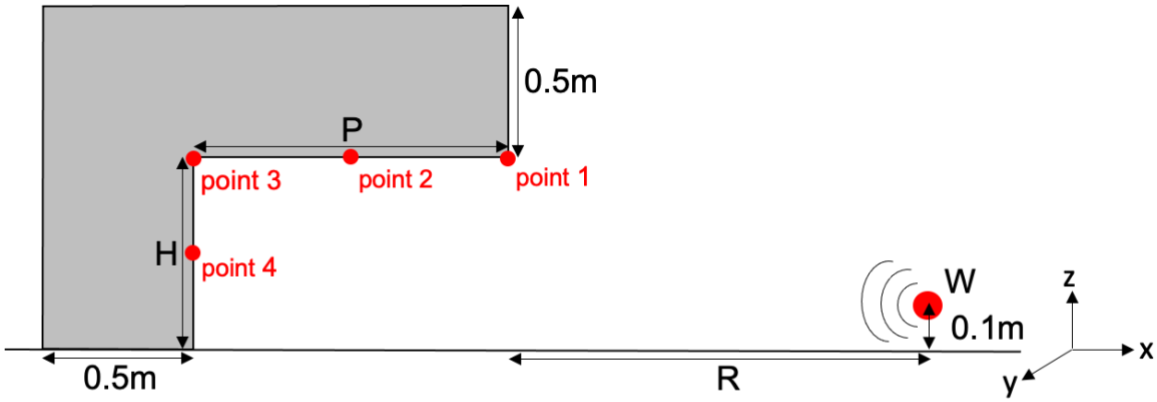
Figure 1 depicts a schematic of the protruded structural façade with protrusion length (P), height (H), explosive weight (W), and standoff distance (R). To minimize

computation time while maintaining accuracy, a small-scale structural configuration has been adopted for this study. Furthermore, to reduce the number of training samples required to develop a reliable machine learning algorithm, the façade thickness and its out-of-plane length (L) have been set to fixed values of 0.5m and 2m, respectively. The spherical explosive charge is also detonated at a constant height of 0.1m above ground. Values for P, H, and W can be found in Figure 1. Standoff is chosen to correspond with a scaled distance range of 3.17 to 7.93 $m/kg^{1/3}$ (8 to 20 $ft/lbs^{1/3}$). Target points are specified as a grid of 10x10 evenly spaced points along surfaces PxL and HxL, yielding a total of 200 target points per test.

ProSAir solves 1D, 2D, and 3D forms of the Euler equations to maximize the computational efficiency of the solution to a 3D problem. The 1D (spherical) component models the blast front until it encounters its first boundary (ground). The result at that time is mapped to a 2D space, and the problem continues in 2D (radially symmetrical) until the next boundary is encountered, with the problem then mapped to the entire 3D domain. The 2D and 3D space cell sizes were chosen based on the convergence of results and are equal to 0.005 m and 0.01 m, respectively. Results of the cell size sensitivity study in the 3D domain for points 1 and 3 can be found in Figure 2. Additionally, 1D space cell size is calculated based on the charge weight (see Table 1). The 3D domain is set to 0.02 m larger than the structure in each direction, ensuring the accuracy of results while maintaining computational speed. This program does not require any material properties, and buildings are modeled using solid cells, as opposed to air. A summary of user inputs for the ProSAir program is provided in Table 1. It must be noted that ProSAir models the

charge as a “bubble” of ideal gas equivalent to a TNT explosion, which poses limitations for detonations or blast-fireball interactions in the near field.

The process of data collection was automated using Python and Microsoft Excel. A total of 1287 successful simulations were run, each yielding 204 independent variables and 2 output variables. Alternatively, the Branching Algorithm (Dennis et al. 2022) can be used to identify unique inputs from initial conditions and the problem domain.



- P(m) = 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
- H(m) = 0.4, 0.5, 0.6
- W(kg) = 0.006, 0.008, 0.01, 0.02, 0.03, 0.05
- R(m) corresponding to scaled distance of 3.17-7.93 m/kg$^{1/3}$

Figure 1- Schematic of protruded façade and the range of input parameters used for ProSAir simulations at mid-length (y= 1m)

Table 1- Summary of inputs for ProSAir simulations

General Setting	
Switch time (s)	$0.0012*W^{1/3}$
Spherical Geometry	
Charge mass (kg)	W
Problem time (s)	1
CFL number	0.5
Cell size (m)	$(3W/4\pi*1600)^{1/3}$
Domain size (m)	0.1
Cylindrical Geometry	
Problem time (s)	1
CFL number	0.5
Cell size (m)	0.005
Height of blast (m)	0.1
Domain size (m)	r= R, h= R+0.2
Boundary types	r: stop, h: stop
3D Geometry	
Maximum problem time	Large enough to ensure maximum impulse is reached
CFL number	0.5
Cell size (m)	0.01
Domain (m)	x_{min}= -0.02, x_{max} = 0.5+P+0.02, y_{min}= -0.02, y_{max}= 2.02, z_{min}= 0, z_{max}= 0.5+H+0.02
Boundaries	x_{min}: transmissive, x_{max}: transmissive, y_{min}: transmissive, y_{max}: transmissive, z_{min}: reflective, z_{max}: transmissive
Origin of remap	x= 0.5+P+R, y= 1, z= 0
Obstacle 1	x_{min}= 0, x_{max}= 0.5, y_{min}= 0, y_{max}= 2, z_{min}= 0, z_{max}= 0.5+H
Obstacle 2	x_{min}= 0.5, x_{max}= 0.5+P, y_{min}= 0, y_{max}= 2, z_{min}= H, z_{max}= 0.5+H
Target point grid 1	x_{min}= 0.5001, x_{max}= 0.5+P, y_{min}= 0, y_{max}= 1, z_{min}= z_{max}= H-0.0001
Target point grid 2	x_{min}=0.5001, x_{max}= 0.5001, y_{min}= 0, y_{max}= 1, z_{min}= 0, z_{max}= H-0.0001

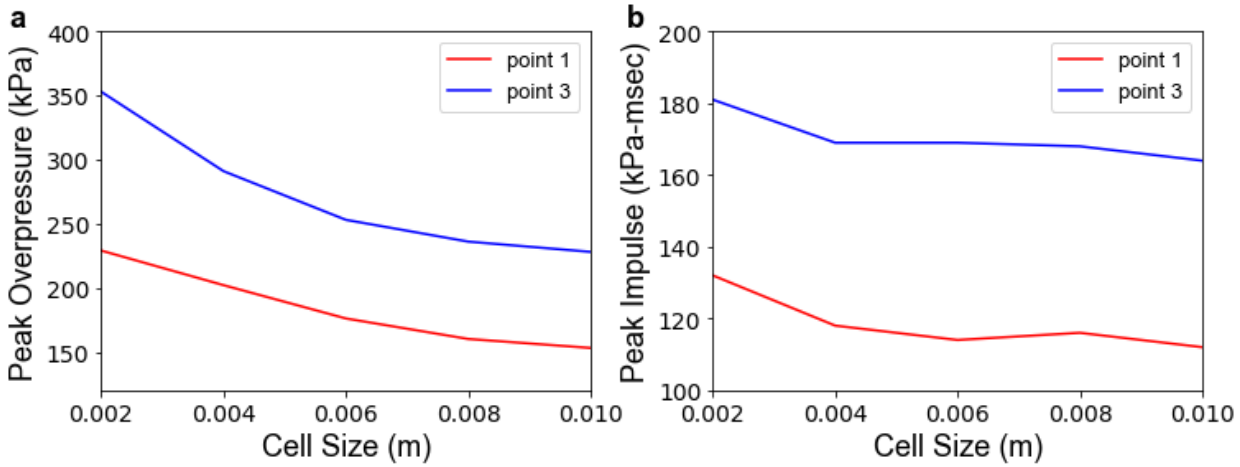


Figure 2- Peak overpressure (a) and peak impulse (b) at points 1 and 3 as a function of 3D domain cell size for P=0.5m, H=0.5m, W=0.02kg, and R=1.5m.

3 Machine Learning Algorithms

Initially, data is randomly split into 80%, which is used for training/validation, and 20%, which is the test set (Figure *3*). The test set represents unseen data that will be used to evaluate the machine learning model once the model has been trained. This randomized assignment of data ensures that the statistical properties of the data subsets are close to each other and thus represent the same statistical population. The 80/20 split ratio is chosen such that the model's ability to learn and its ability to generalize is maximized. Seven features (P, H, W, R, and target point coordinates x, y, z) and one output variable (peak overpressure or impulse) are used to train, validate, and test each model.

Regularization is adopted throughout the study to further improve the model's generalizability by reducing variance (the difference between the error rate of the training set and testing set). High variance, also known as overfitting, occurs when a model learns the detail and noise in the training set to the extent that it negatively impacts the model's performance on new data. Regularization can be achieved by modifying the loss function (e.g., L1 regularization, L2 regularization), modifying the data sampling technique (e.g., K-fold cross-validation), or modifying the training algorithm (e.g., dropout). Details regarding various forms of regularization are provided in the following sections.

This study uses randomized cross-validation to tune the hyperparameters for each model. The randomized parameter optimization approach implements a randomized search over a distribution of possible parameter values. The range of parameters of interest is specified using a Python dictionary, and a total of 15 parameter combinations are investigated. A 5-fold cross-validation technique is chosen to tackle overfitting. In 5-fold cross-validation, the train/validation data is divided into 5 equal parts, and at every

split, the model is trained on 4 parts and validated on one part. The training process is repeated 5 times, each time withholding a different set for validation. Once the optimum hyperparameters are determined, the model is evaluated on the test set using two different scoring metrics: Mean Squared Error (MSE) and R^2 score.

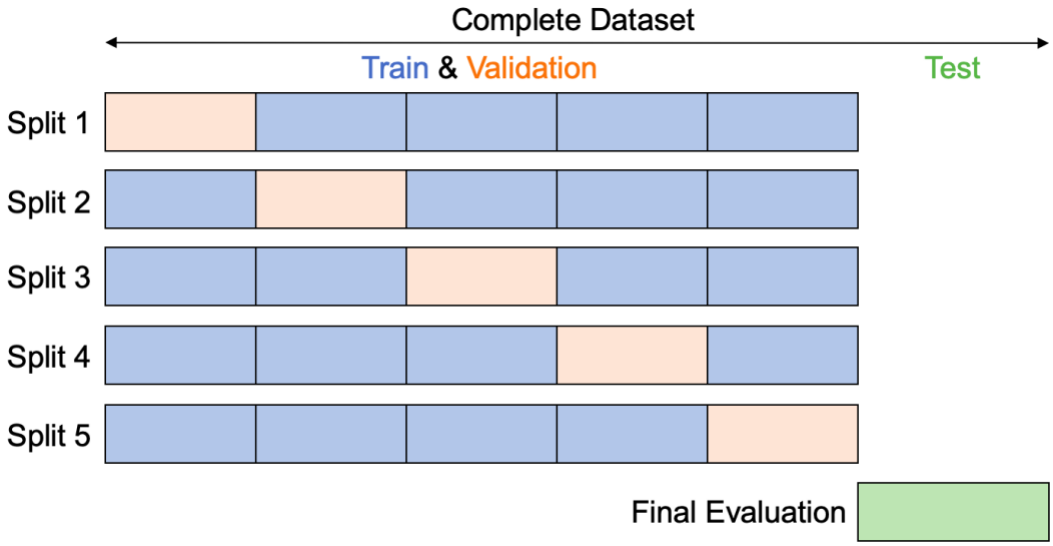


Figure 3- Schematic of 5-fold cross-validation

3.1 Linear Regression

In simple linear regression, the least-squares method is used to calculate the best-fitting curve for the observed data by minimizing the sum of squares of the vertical deviations from each data point to the curve. Since gradient descent-based algorithms are sensitive to the scale of features, various normalization (e.g., MinMaxScaler and MaxAbsScaler) and standardization (e.g., StandardScaler, PowerTransformer, QuantileTransformer, RobustScaler) techniques are applied to simple linear regression, polynomial regression, and neural networks. Tree-based algorithms on the other hand are relatively insensitive to the scale of features; therefore, feature scaling is not used for RF, GBDT, and XGBoost. Regularization is applied to the linear algorithms to favor a simpler prediction and avoid overfitting. The two main regularization methods are Ridge regression (McDonald, 2009)

and Lasso regression (Ranstam and Cook, 2018). L2 regularization (Ridge regression) is ideal when there is high collinearity among features or when the number of features is greater than the number of samples, encouraging fewer model parameters. L1 regularization (Lasso Regression) is ideal when there is high multicollinearity among features and is widely used for feature selection, encouraging sparse model parameters.

To account for the nonlinear relationship between dependent and independent variables, the predictive performance of polynomial regression is also evaluated. The optimum degree of the polynomial is a trade-off between bias and variance, meaning that as the model complexity grows, the bias reduces, and the variance increases. This can be seen as the divergence point of the MSE vs. polynomial degree graph for the validation and test sets. The polynomial model is chosen such that it performs well on both validation and unseen data sets.

3.2 Tree-Based Models

Tree-based models use a series of 'if-then' rules to generate predictions from one or more decision trees. A single decision tree is fast and highly interpretable but prone to overfitting. To overcome this issue, several decision trees are trained in parallel (i.e., bagging) or sequentially (i.e., boosting), creating a model that is more flexible and less data sensitive (see Figure *4*). This technique is referred to as 'Ensemble Learning'. RF and GBDT are the two most popular ensemble methods. In RF, a parallel combination of decision trees is used, meaning that each tree is trained on a random subset of the same training data (bootstrapped sample), and the results from all trees are averaged. As for GBDT, multiple decision trees are trained sequentially, with each tree slightly improving the mistakes of the previous one. It is called gradient boosting because it uses a gradient

descent algorithm (Ruder, 2016) to minimize the loss when adding new decision trees. Due to the sequential connection between models, boosting algorithms are usually slow to learn but also highly accurate. XGBoost improves upon the computation time of gradient boosted trees by adopting a combination of software and hardware optimization techniques. Furthermore, XGBoost penalizes more complex models through additional regularization parameters to improve models' generalization capabilities.

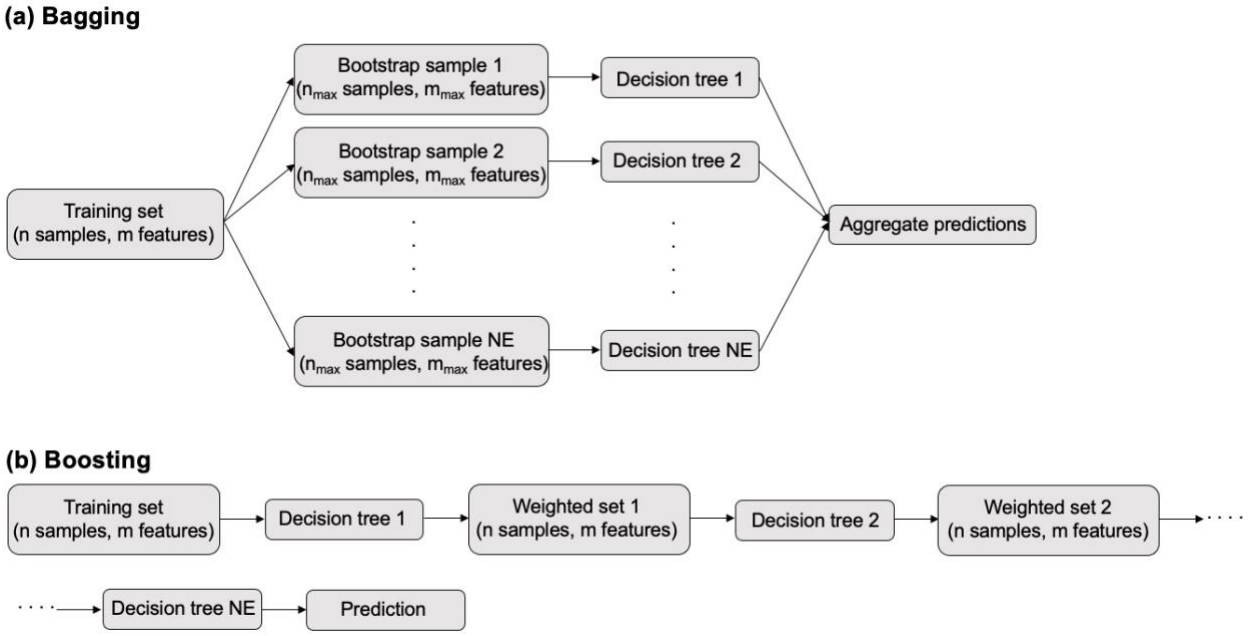


Figure 4- Schematic of (a) bagging vs. (b) boosting technique. In this schematic, n, n_{max}, m_{max}, and NE refer to the number of data points in the training set, the number of data points in each bootstrap sample, the number of features to consider at each split, and the number of estimators, respectively.

This study compares the predictive performance of RF, GBDT, and XGBoost methods on the test set to that of linear methods. Randomized cross-validation (5-fold) is used for determining the optimum hyperparameters. The Negative Mean Absolute Error (NMAE) is used as the scoring argument, where values closer to zero represent less prediction error by the model.

The complete list of hyperparameters for RF, GBDT, and XGboost models can be found online on the Scikit-learn website (Scikit-learn, 2022). Among the main parameters for RF include number of estimators (total number of trees), criterion (function to measure the quality of a split), maximum depth (maximum depth of each tree), minimum samples split **(**minimum number of samples required to split an internal node), minimum samples leaf (minimum number of samples required to be at a leaf node), maximum leaf nodes (maximum number of leaf nodes), maximum features **(**number of random features to consider at each split), and maximum samples (size of each bootstrapped sample). It is safe to assume that the bootstrapped sample is the same size as the training set because we are randomly selecting, which will cause 1/3 of the sample, on average to be different each time.

RF incorporates regularization but not in the form of a penalty to the cost function, as seen in linear regression. Instead, by limiting maximum tree depth, node size, and minimum information gain during node split, the variance within the model is controlled.

The GBDT model considers most hyperparameters used in RF. Additional parameters include loss (loss function to be optimized) and learning rate (learning rate between consecutive trees). Finally, XGBoost implements regularized boosting through alpha (L1 regularization term on weights), lambda (L2 regularization term on weights), gamma (minimum loss reduction required to make a further partition on a leaf node), and minimum child weight (minimum sum of instance weight needed to split an internal node) parameters.

3.3 Artificial Neural Networks

In prediction problems involving large datasets, ANNs tend to outperform all other methods, and the fully connected feedforward ones are the most widely used architecture. This specific network consists of layers that are fully connected such that every neuron in one layer is connected to every neuron in the next layer. The main hyperparameters for ANNs are network architecture (number of hidden layers and neurons per layer), feature scaling, weight initialization, activation function, optimizer, loss function, learning rate, batch size, number of epochs, and dropout. In this study, two fully connected networks are developed for predicting peak overpressure and peak impulse.

Data normalization of features is used to obtain a mean close to zero and accelerate convergence. The Rectified Linear Activation Unit (ReLU) activation function (Nwankpa et al., 2018) is chosen for the hidden layers as it solves the problem of vanishing gradients during backpropagation. Since this is a regression problem, linear activation is chosen for the output layer. The Adam optimizer (Kingma and Ba, 2015), which tends to perform better for most problems, is used in this study. MSE is taken as the loss function for the Adam optimizer. The models are constructed using the Keras TensorFlow Python library. Although the Scikit-learn library also offers MLP regressor models, TensorFlow is a much more flexible library in terms of constructing neural networks.

The remaining hyperparameters, such as the number of neurons in each hidden layer, weight initialization method, learning rate, batch size, number of epochs, and dropout rate are tuned using randomized 5-fold cross-validation. For each hidden layer, neurons ranging from 10 to 100 are investigated. In addition, various weight initialization

methods are considered (i.e., uniform, normal, He uniform (He et al., 2015), He normal, Glorot uniform (Glorot and Bengio, 2010), Glorot normal) to mitigate the chances of exploding or vanishing gradients, and consequently improve convergence. Learning rates of 0.001, 0.002, and 0.005, followed by batch size values of 100, 500, and 1000 and the number of epochs of 20, 25, 30, and 35 are also evaluated. To avoid any potential overfitting, dropout of 0, 0.1, and 0.2 are taken into consideration as the regularization strategy. All remaining hyperparameters are set to their default values which can be found on the TensorFlow website (TensorFlow, 2022).

4 Results and Discussion

4.1 Exploratory Data Analysis

The peak overpressure and peak impulse have mean values of 62.9 kPa and 38.3 kPa-msec, and standard deviations of 48.8 kPa and 26.6 kPa-msec, respectively. Additional information regarding the distribution of data can be found in the histogram plots of Figure 5. In both cases, the data follows a fairly lognormal probability distribution. Similar trends have also been reported for peak incident pressure, peak impulse, and time of positive phase duration model errors from explosive field trials (Stewart et al., 2020). Since the data was collected using the ProSAir program, no data point was excluded from this study, and the complete dataset was used for training/testing various machine learning models.

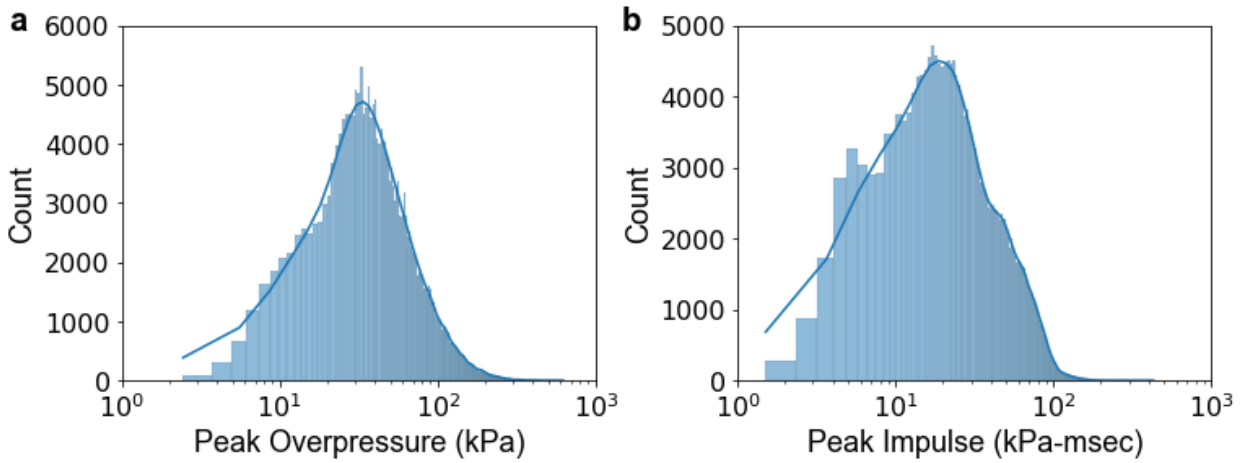


Figure 5- Histogram plots of peak overpressure (a) and peak impulse (b).

To better understand the effect of protrusion length and building height on the results, the overpressure and impulse values are plotted as a function of simulation time for points 1, 2, 3, and 4 in Figure 6. The location of these points is provided in Figure 1. It can be seen that progression inwards from point 1 to point 3 has resulted in an increase in peak impulse. Also, point 4 shows almost the same peak impulse as point 3. This indicates that the protrusion length may have a more significant impact on peak impulse than the structure height. This finding is later compared to GBDT feature importance studies in section 4.3.2.

Among all points, point 3 is the most critical in terms of peak overpressure and impulse. The 2D colormap scatter plots for this point as a function of scaled protrusion length, scaled height, and standoff are provided in Figure 7. Standoff valuesof 1, 1.2, 1.4, and 1.6m are the most common among all simulations and therefore chosen to demonstrate the effect of threat standoff on blast parameters. Results show that smaller scaled height and protrusion lengths yield higher peak overpressure and impulse values. As expected, with an increase in standoff, both peak overpressure and impulse values are reduced.

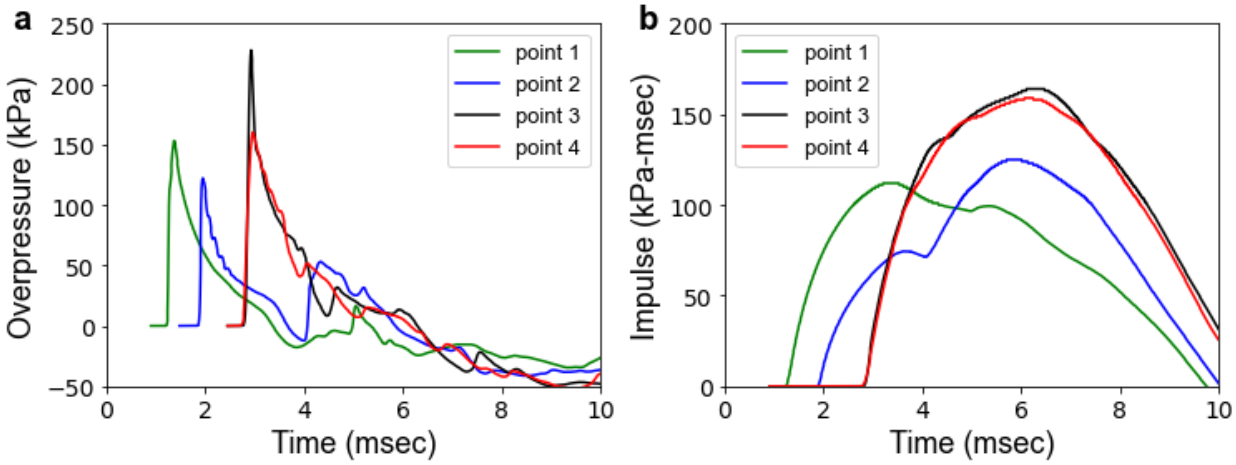


Figure 6- Overpressure (a) and Impulse (b) at points 1, 2, 3, and 4 as a function of time for P=0.5m, H=0.5m, W=0.02kg, and R=1.5m.

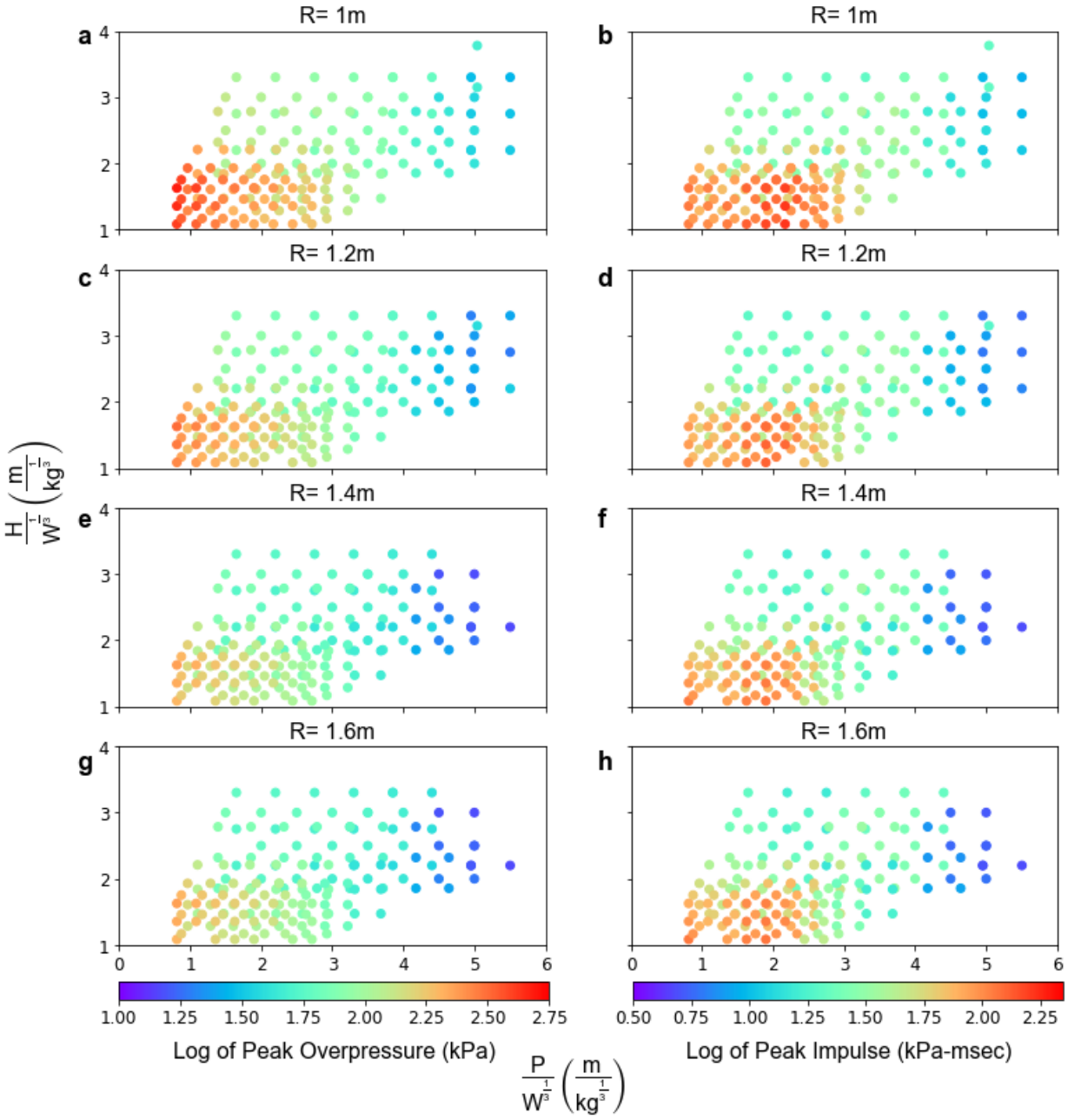


Figure 7- Scatter plot of peak overpressure (a, c, e, g) and peak impulse (b, d, f, h) at point 3, as a function of scaled protrusion length, scaled height, and standoff

4.2 Linear Regression

The performance of machine learning models for peak overpressure and peak impulse is provided in Table 2. Among the various data preprocessing techniques used for simple linear regression, the Quantile Transformer-Uniform yielded the best results,

8 Table 2- Summary of prediction error for various machine learning models

Model	Optimum Hyperparameters	Peak Overpressure				Peak Impulse			
		MSE_Valid[a]	R^2_Valid	MSE_Test[a]	R^2_Test	MSE_Valid[b]	R^2_Valid	MSE_Test[b]	R^2_Test
Simple Linear Regression	Standardization technique: Quantile Transformer-Uniform	721.96	0.697	703.99	0.702	321.76	0.547	314.93	0.556
Polynomial Regression	Standardization technique: Quantile Transformer-Uniform	22.51	0.990	19.27	0.992	8.02	0.989	6.33	0.991
RF	No. of estimators = 800 Min. sample split = 6 Min. sample leaf = 3 Max. depth = 20	33.52	0.986	25.64	0.989	2.51	0.996	1.92	0.997
GBDT	No. of estimators = 500 Min. sample split = 15 Max. depth = 10 Learning rate = 0.2	5.46	0.998	3.82	0.998	0.51	0.999	0.34	1.000
XGBoost	No. of estimators = 500 Min. child weight = 3 Max. depth = 10 Learning rate = 0.2 Alpha Regularization = 0.1 Gamma Regularization = 0.3 Lambda Regularization = 1	5.30	0.998	3.69	0.998	0.53	0.999	0.35	1.000
ANN	No. of neurons in layer 1 = 100 No. of neurons in layer 2 = 100 No. of neurons in layer 3 = 100 No. of neurons in layer 4 = 50 No. of neurons in layer 5 = 50 Dropout = 0 Learning rate = 0.001 Weight initialization = Glorot uniform Epochs = 25 Batch size = 500	10.76	0.995	8.60	0.996	1.67	0.998	1.32	0.998

[a] The units of MSE for peak overpressure is $(kPa)^2$

[b] The units of MSE for peak impulse is $(kPa\text{-}msec)^2$

9

and was subsequently used for polynomial regression. This technique transforms variables to have a uniform probability distribution; thus, decreasing the impact of outliers, if any.

The application of L1 and L2 regularizations did not improve the results. This is because the number of samples is significantly greater than the number of features. Therefore, a more complex model will more likely yield better predictions. Furthermore, feature selection performed by Lasso regression revealed that no features are deemed unimportant.

Polynomial regression results included in Table 2 show significant reductions in prediction error compared to simple linear regression, implying that data has high complexity. The decrease in MSE with increasing polynomial degree is shown in Figure 8. Considering the large volume of data and computational limitations, polynomial regression was carried out up to the 9^{th} degree. With increasing model complexity, both validation and test set errors continue to decrease. This is because the underlying phenomenon truly follows a high-degree polynomial. Also, since a computer simulation has generated the data, both validation and test sets have similar data distribution.

Another interesting observation made in Figure 8 is the higher values of MSE for the average validation set compared to the test set. This is due to the smaller sample size for each validation set versus the test set. The intent of the validation set is to determine optimum hyperparameters, while the size of the test set should be large enough to minimize prediction error.

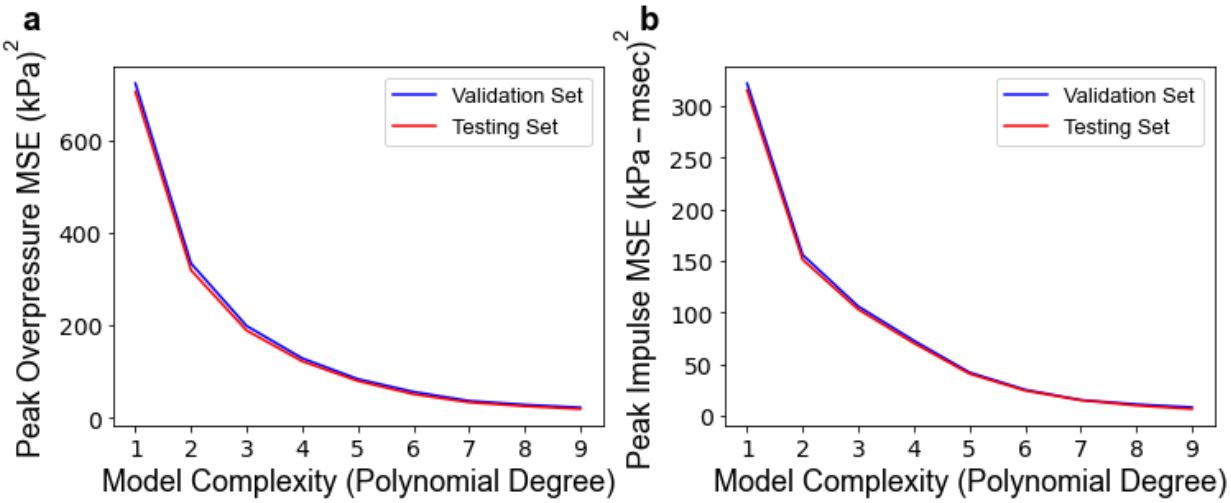


Figure 8- MSE as a function of polynomial degree for (a) peak overpressure and (b) peak impulse

4.3 Tree-Based Models

4.3.1 Random Forest

Random Forest offers fairly similar results to that of 9th degree polynomial regression (see Table 2). While peak impulse results are improved compared to 9th degree polynomial regression, the peak overpressure prediction errors are slightly higher. The optimum hyperparameters for each RF model can also be found in Table 2. The number of estimators is the most important hyperparameter in an RF model which represents the number of trees in the model. Generally, greater number of estimators results in higher accuracy at the cost of slower training speed. The optimum number of random features considered at each split is equal to the default value (total number of features). This is possibly due to the fact that the number of features in the dataset is small. Overall, increasing the number of features tends to decrease bias, as there is a better chance that good features are included in each tree. However, this may lead to increased variance and training time.

Maximum depth is defined as the longest path between the root and leaf node. The deeper the tree, the more splits it has and the more information about the data it takes into account. The high value for maximum depth, together with the low values for minimum sample split and minimum sample leaf (compared to the size of the dataset), reveal relatively complex trees within the forest. The remaining hyperparameters in the Scikit-learn Python library are set to their default values, as any modifications to these parameters did not offer improvements to the results.

4.3.2 Gradient-Boosted Decision Trees

The optimal hyperparameters and performance of the GBDT model are provided in Table 2. As expected, the MSE and R^2 score of GBDT models has significantly improved compared to RF. GBDT models usually perform better than RF, mainly by reducing bias. Boosting is a method that converts weak learners (high bias, low variance) into strong learners. Therefore, GBDT is more prone to overfitting. On the other hand, RF uses fully grown trees (low bias, high variance) and tackles the error reduction task by reducing variance. This translates to lower maximum depth and greater min sample split values for GBDT models, as reported in Table 2.

To better understand the accuracy of randomized search cross-validation in correctly identifying the optimum hyperparameters, the MSE of the average validation set as a function of four primary hyperparameters in the GBDT models is plotted in Figure 9. For each plot, the remaining hyperparameters are set to the values reported in Table 2. Results show that the number of estimators, maximum depth, and learning rate have the most influence on MSE, and both output variables follow a similar trend with regards to changes in the hyperparameters.

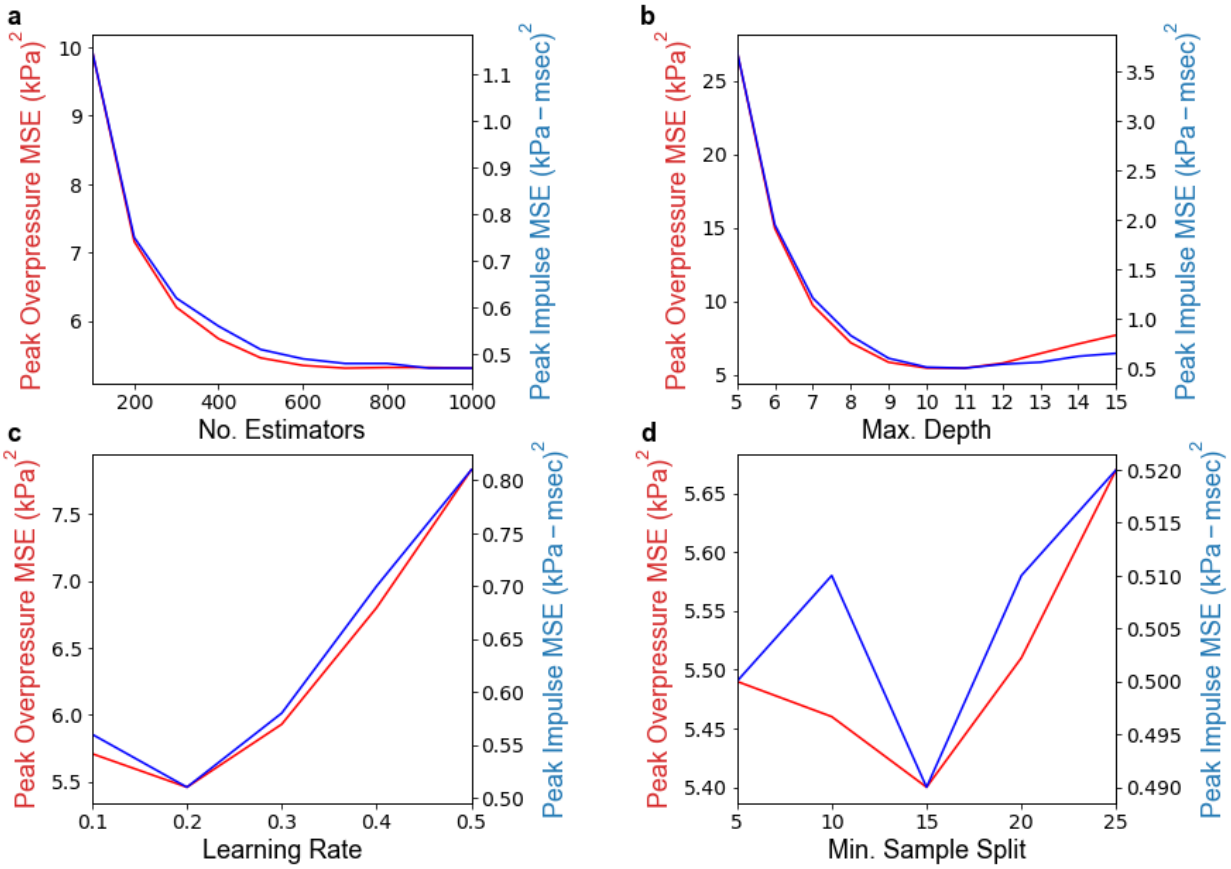


Figure 9- MSE of the validation set as a function of (a) number of estimators, (b) maximum tree depth (c) learning rate, (d) minimum sample splits for peak overpressure and peak impulse in GBDT models

Although randomized cross-validation has been applied to the peak overpressure dataset, using the identified optimum parameters for predicting peak impulse is also acceptable. In fact, separate hyperparameter tuning for peak impulse will only result in slight reductions in peak impulse MSE. Since random sets of hyperparameters have been chosen in each iteration of the hyperparameter tuning process, the identified optimum values are very close to the absolute optimum values. This level of accuracy is acceptable for the purposes of the current study. However, for cases where MSE values are more sensitive to the changes of hyperparameters or when higher levels of accuracy are desired, it is recommended to run grid search cross-validation on a narrower search

space. This technique creates a grid over the search space and evaluates the model for all possible hyperparameters within the space.

The permutation feature importance scores for GBDT models are shown in Figure 10. The permutation feature importance is defined as the decrease in a model score when a single feature value is randomly shuffled (Breiman, 2001). In other words, the score indicates how influential each feature is in constructing the boosted decision tree.

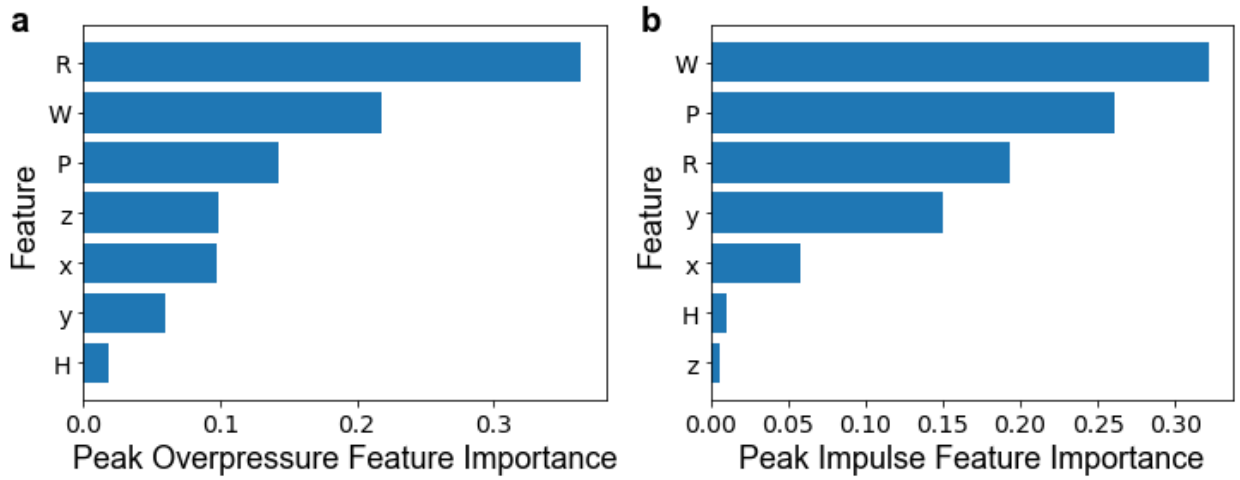


Figure 10- Permutation Feature Importance scores for (a) peak overpressure and (b) peak impulse

The protrusion length yields a significantly higher score than height for both target variables, which is in agreement with the results of section 4.1. However, the highest importance score is related to standoff and charge weight for peak overpressure and peak impulse, respectively. The feature importance score was determined by averaging the score over five repetitions of the optimum GBDT model.

4.3.3 Extreme Gradient Boosting

The MSE and R^2 score values for XGBoost models are provided in Table 2. The optimum hyperparameters were determined by tuning maximum depth, learning rate (Eta), minimum child weight, alpha, lambda, and gamma regularization parameters.

The gamma parameter is the minimum loss reduction to create a new tree-split, where a larger gamma regularizes the model by growing shallower trees. Hence, the small gamma values and high maximum depth reported in Table 2 are in agreement with each other.

Minimum child weight refers to the minimum sum of instance weight (hessian) needed for an internal node to split. This is a secondary regularization parameter when compared to gamma. The low values of minimum child weight reported in Table 2 further demonstrate the complexity of individual trees.

The L1 and L2 regularization on leaf weights are implemented using the alpha and lambda regularization parameters, respectively. L1 regularization punishes less-predictive features and encourages sparsity, while L2 is used to further punish large leaf weights. The default values for alpha and lambda parameters are equal to 0 and 1, respectively. Results show that L1 and L2 regularization has little to no effect on the results. In fact, XGBoost has provided similar results in terms of prediction error to GBDT.

4.4 Artificial Neural Networks

Results for ANN models are reported in Table 2. The optimum network architecture consists of 5 hidden layers with 100, 100, 100, 50, and 50 neurons in each consecutive layer. In selecting the network architecture, multiple layers are shown to perform better than one, and the number of layers is increased until no substantial improvement in

performance is observed. Similarly, use of number of neurons greater than 100 per layer did not improve the results. Regularization in the form of dropout is equal to 0 and the optimum weight initialization follows a uniform distribution. Additional information regarding hyperparameter tuning can be found in Table 2.

Since learning in ANNs is an iterative process, the entire dataset (divided into batches) needs to be passed into the neural network multiple times to properly adjust the model's weights. Each complete pass of the dataset is defined as one epoch. Figure 11 displays the reduction in MSE as a function of the number of epochs, also known as the learning curve. Separate graphs are plotted for learning rates of 0.001, 0.002, and 0.005, and all remaining parameters are kept constant at values reported in Table 2. As can be seen, the MSE for the validation set plateaus at the optimum number of epochs, and therefore the model has suitably fit the dataset. Models for both variables perform well using learning rate of 0.001 sec, batch size of 500 after 25 epochs.

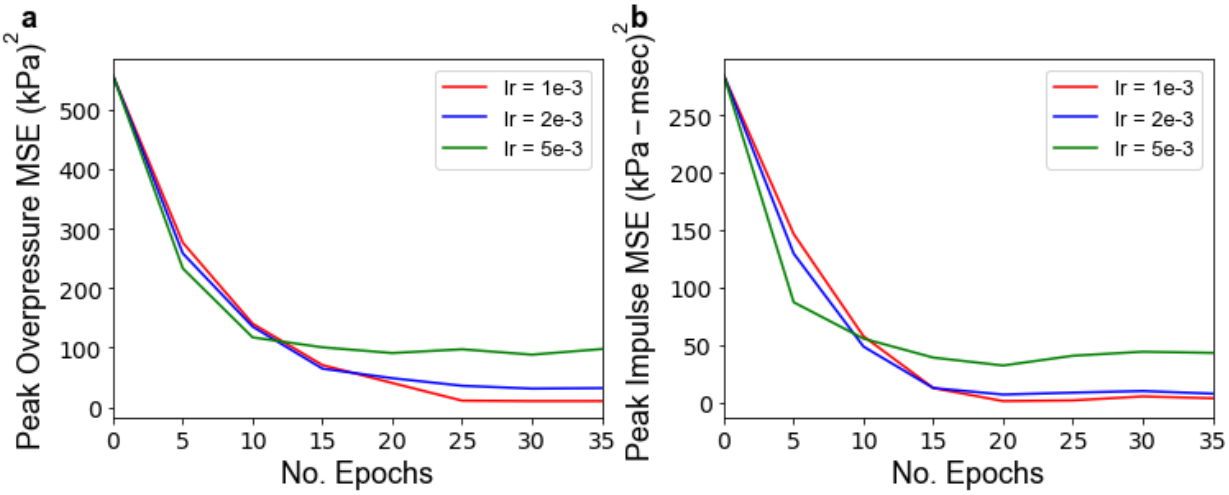


Figure 11- MSE of the validation set as a function of learning rate and number of epochs for (a) peak overpressure, (b) peak impulse in ANN models

Results in Table 2 show that although ANNs have performed better than RF and polynomial regression, gradient boosting algorithms such as GBDT and XGBoost have outperformed ANNs both in terms of MSE and R^2 score. Recent studies (Grinsztajn L et al., 2022) have also shown tree-based models to easily yield good predictions for tabular data, with much less computational cost.

5 Conclusions

Technical challenges attributed to CFD modeling of complex structural geometries against blast loading have prompted researchers to investigate alternative/complementary techniques to traditional CFD simulations. The development of computational mechanics software that uses machine learning tools may significantly accelerate processing time, but the efficacy of such tools must be investigated.

The present study evaluates various machine learning methods (linear regression, tree-based, and neural networks) for predicting blast loading on a protruded architectural structure. A dataset comprising of over 250,000 data points is used to train, validate, and test the models. This is unique in its kind as a high number of data points are collected from a single source. Gradient boosting methods (GBDT and XGBoost) are incorporated into the study, which are among the most popular choices of algorithms due to their high predictive power.

The results demonstrate the potential for machine learning algorithms to revolutionize CFD analysis for blast loading. GBDT and XGBoost outperformed ANNs with R^2 scores of 0.998 and 1.000 for peak overpressure and peak impulse, respectively. XGBoost which is a more regularized form of the gradient boosting algorithm offered relatively similar results to GBDT. The number of estimators, maximum depth, and learning rate were among the most influential hyperparameters for gradient boosting models. Although randomized search cross-validation was applied to the peak overpressure dataset, the use of the same optimum hyperparameters for peak impulse resulted in very high prediction accuracy. Optimum hyperparameter selection using randomized search achieved relatively similar results to the traditional grid search approach, while significantly reducing computation time. Finally, the protrusion length was shown to have a significantly higher permutation feature importance score for GBDT models than structure height, meaning that the protrusion length was a more influential feature than structure height in the construction of GBDT models.

Future research may include evaluating a hybrid between gradient boosting methods with ANN's. One of the major challenges in machine learning is that there is never enough training data to tackle every machine learning problem. For that reason, further studies into the use of transfer learning in ANN's can be of great importance.

6 Acknowledgments

The authors wish to express their gratitude to Dr. Shaun Forth at Cranfield University for providing the ProSAir software, which has been used for data collection. The authors are also extremely thankful to Brian Katz and Shalva Marjanishvili at Integral Research Solutions Group for their invaluable insight and technical support.

7 Funding

The authors received no financial support for the research, authorship and/or publication of this article.

8 References

Almustafa MK, Nehdi ML (2020) Machine learning model for predicting structural response of RC slabs exposed to blast loading. *Engineering Structures,* 221: 111109.

Almustafa MK, Nehdi ML (2022) Machine learning model for predicting structural response of RC columns subjected to blast loading. *International Journal of Impact Engineering,* 162:104145.

Alterman D, Stewart MG, Netherton MD (2019) Probabilistic assessment of airblast variability and fatality risk estimation for explosive blasts in confined building spaces. *International Journal of Protective Structures,* 10(3): 306– 329.

Bewick B, Flood I, Chen Z (2011) A Neural-network model-based engineering tool for blast wall protection of structures. *International Journal of Protective Structures,* 2(2): 159-176.

Borisov V, Leemann T, Seßler K, et al. (2022) Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*.

Bozinovski S, Fulgosi A (1976) The influence of pattern similarity and transfer learning upon training of a base perceptron b2. *In Proceedings of Symposium Informatica*, 3: 121-126.

Breiman L (2001) Random forests. *Machine Learning,* 45: 5-32.

Castedo R, Reifarth C, Santos AP, et al. (2019) Application of grid convergence index to shock wave validated with LS-DYNA and ProsAir. *Ingeniería e Investigación,* 39(3): 20-26.

Chen T, Guestrin C (2016) *Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp.* 785-794.

Dennis AA, Pannell JJ, Smyl DJ, et al. (2021) Prediction of blast loading in an internal environment using artificial neural networks. *International Journal of Protective Structures,* 12(3): 287– 314.

Dennis AA, Smyl DJ, Stirling CG, Rigby SE (2022) A branching algorithm to reduce computational time of batch models: Application for blast analyses. *International Journal of Protective Structures*: 20414196221085720.

Flood I, Bewick BT, Dinan RJ, et al. (2009) Modeling blast wave propagation using artificial neural network methods. *Advanced Engineering Informatics,* 23(4): 418–423.

Friedman JH (2001) Greedy function approximation: A gradient boosting machine. *Annals of statistics*: 1189-1232.

Glorot X, Bengio Y (2010) *Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics*. pp. 249-256, JMLR Workshop and Conference Proceedings

Grinsztajn L, Edouard O, Gaël V (2022) Why do tree-based models still outperform deep learning on tabular data?. *arXiv preprint arXiv:2207.08815.*

Haykin SS (2009) *Neural networks and learning machines, 3/E*: Pearson Education India

Hearst MA. Dumais ST, Osuna E, et al. (1998) Support vector machines. *IEEE Intelligent Systems and their applications,* 13(4): 18-28.

He K, Zhang X, Ren S, et al. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. Proceedings of the IEEE international conference on computer vision.* pp. 1026-1034.

Ho TK (1998) The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence,* 20(8): 832-844.

Jacob N, Yuen SCK, Nurick GN, et al. (2004) Scaling aspects of quadrangular plates subjected to localised blast loads-experiments and predictions. *International Journal of Impact Engineering,* 30 (8-9): 1179–1208.

Kavzoglu T, Mather PM (2003) The use of backpropagating artificial neural networks in land cover classification. *International Journal of Remote Sensing,* 24(23): 4907-4938.

Kingma D, Ba J (2015) *Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980*.

McDonald GC (2009) Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics,* 1(1): 93-100.

Neto LB, Saleh M, Pickerd V, Yiannakopoulos G, et al. (2020) Rapid mechanical evaluation of quadrangular steel plates subjected to localised blast loadings. *International Journal of Impact Engineering,* 137: 103461.

Nwankpa CE, Ijomah W, Gachagan A, et al. (2018) Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378.*

Pannell JJ, Rigby SE, Panoutsos G (2022a) Application of transfer learning for the prediction of blast impulse. *International Journal of Protective Structures,* 0(0): 1-21.

Pannell JJ, Rigby SE, Panoutsos G (2022b) Physics-informed regularisation procedure in neural networks: An application in blast protection engineering. *International Journal of Protective Structures,* 13(3): 555-578.

ProSAir computational blast loading tool. Available at: www.cranfield.ac.uk/facilities/prosair-computational-blast-loading-tool (accessed on 10 October 2022)

Ranstam J, Cook JA (2018) LASSO regression. *Journal of British Surgery,* 105(10): 1348-1348.

Remennikov AM, Rose TA (2005) Modelling blast loads on buildings in complex city geometries. *Computers & Structures*, 83(27): 2197–2205.

Remennikov AM, Rose TA (2007) Predicting the effectiveness of blast wall barriers using neural networks. *International Journal of Impact Engineering*, 34(12): 1907-1923.

Rigby SE, Lodge TJ, Alotaibi S, et al. (2020) Preliminary yield estimation of the 2020 Beirut explosion using video footage from social media. *Shock Waves,* 30(6): 671-675.

Ruder S (2016) An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747 .*

Scikit-learn Machine Learning in Python. Available at: https://scikit-learn.org/stable (accessed on 10 October 2022)

Shwartz-Ziv R, Armon A (2022) Tabular data: Deep learning is not all you need. *Information Fusion,* 81: 84-90.

Stewart MG, Netherton MD, Baldacchino H (2020) Observed airblast variability and model error from repeatable explosive field trials. *International Journal of Protective Structures*, 11(2): 235-257.

TensorFlow. Available at: https://tensorflow.org (accessed on 10 October 2022)

Ullah A, Ahmad F, Jang HW, et al. (2017) Review of Analytical and Empirical Estimations for Incident Blast Pressure. *Structural Engineering,* 21(6): 2211-2225.

Zou J, Han Y, So SS (2008) Overview of artificial neural networks. *Artificial Neural Networks*: 14-22.