Explainable Approaches to Machine Learning

Final Report, S1 2024

Prediction Project on BLEVE

Unit Code : COMP6013

Name : Syed Muhammad Ahmed Zaidi

Student ID: 20972008

Table of Contents

1) Introduction

1.1) Project Overview:

With an increase in globalization around the world, transportation of resources from one place to has become imminent. This involves the movement of hazardous materials like Liquefied Petroleum Gas (LPG) which can be a serious risk any society. The process of the Boiling Liquid Expanding Vapour Explosions (BLEVEs) can result in a huge explosions causing great impact on the human lives as well as the infrastructure around if not handled properly. Therefore, to minimize the risk, data is collected to carry out a complex prediction about the repercussion of such an incident with obstacles around the explosion. It requires a data-driven machine learning model predict the peak pressures that can be generated with such BLEVEs related incidents. Therefore, a scenario is created to where a BLEVE related tasks take place in a rectangular tank in a three dimensional environment. The walls of the tank will act as an obstacle in order to record the pressure created by the blast waves. What makes this even more complex is that due to these walls, there will be reflections and deflections making accurate predication much more difficult to achieve.

1.2) Objectives and Purpose:

To address this risk towards the society, a dataset is considered that contains numerous physical measurements related to BLEVE explosions. This would contains information about the tank dimensions, the obstacle dimensions, a series of temperatures, pressures sensors and much more. The goal of the project is to run multiple machine learning models that are able to most accurately predict the peak pressure at different sensor points around the obstacle. It requires to go through different stages in reaching this which involves, data pre-processing that encompasses of multiple techniques to clean the data. For example, data type conversion, feature scaling, feature engineering, feature selection etc. This would prepare the data for further use when creating a model, which is the next step. This stage of model development will go through multiple techniques to create machine learning algorithms that would allow selection of the most effective model. Once completed, the model will further be evaluated using interpretation techniques to find local and global insights. This comprehensive report will communicate all rationale behind the choice of selection and also give detailed knowledge about the results obtained. Finally, the last section will reflect about the learning and experiences that were encountered throughout and also how these can be improved in future projects.

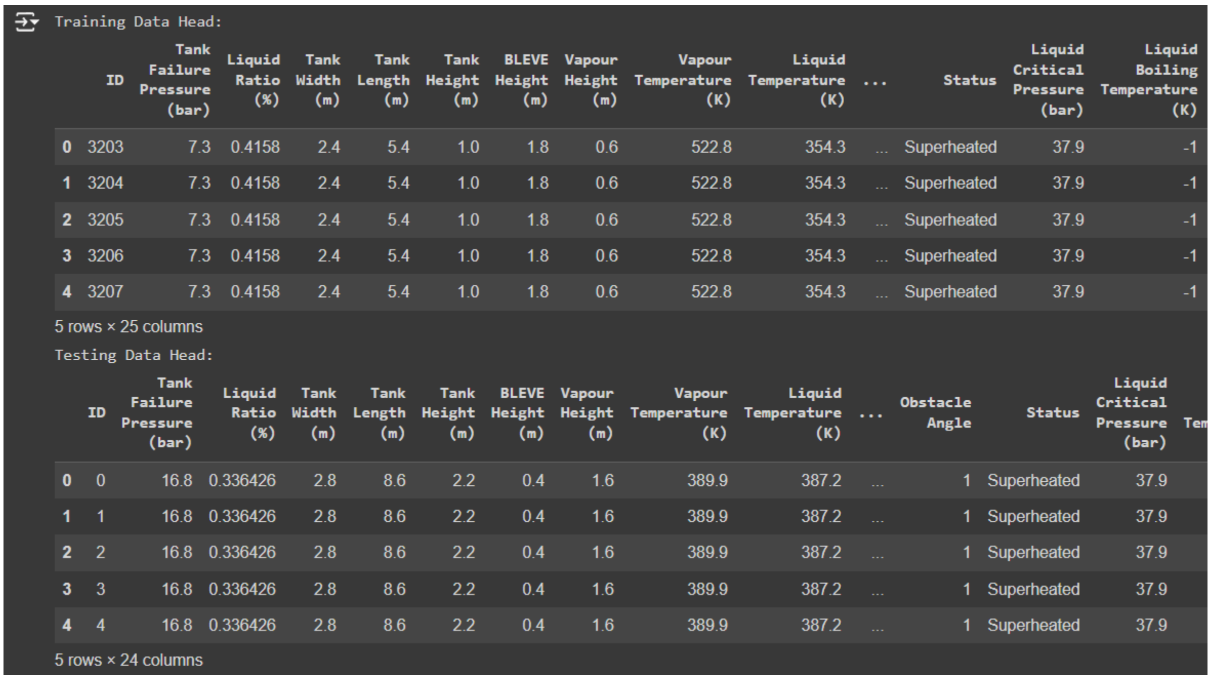
2) Data Preprocessing

2.1) Data Loading

The first step for this project was to load the data within the Google Colab for it to be understood. I took a different approach in loading the data from Google Drive rather than using them from local file system. This is because I did not wanted to make it inconvenient for the examiner to have to load the data themselves before running it. By uploading both files to drive, I was able to generate a link for each that will allow direct downloading into the colab environment, making the complete process automated. Thus, this way will allow anyone with the code to be able to run it on any computer in any location and it will automatically fetch the file and store it in the variable names of train_data and test_data.

2.2) Understanding Data

Next step was to understand what the data includes and the structure, content, and quality it provides. I used multiple commands to view and explore deeply which includes the .head command for both files showing the first 5 rows of each of them, giving an overview and a quick inspection of the features and values that are included.



Training Data Head:

	ID	Tank Failure Pressure (bar)	Liquid Ratio (%)	Tank Width (m)	Tank Length (m)	Tank Height (m)	BLEVE Height (m)	Vapour Height (m)	Vapour Temperature (K)	Liquid Temperature (K)	...	Status	Liquid Critical Pressure (bar)	Liquid Boiling Temperature (K)
0	3203	7.3	0.4158	2.4	5.4	1.0	1.8	0.6	522.8	354.3	...	Superheated	37.9	-1
1	3204	7.3	0.4158	2.4	5.4	1.0	1.8	0.6	522.8	354.3	...	Superheated	37.9	-1
2	3205	7.3	0.4158	2.4	5.4	1.0	1.8	0.6	522.8	354.3	...	Superheated	37.9	-1
3	3206	7.3	0.4158	2.4	5.4	1.0	1.8	0.6	522.8	354.3	...	Superheated	37.9	-1
4	3207	7.3	0.4158	2.4	5.4	1.0	1.8	0.6	522.8	354.3	...	Superheated	37.9	-1

5 rows × 25 columns

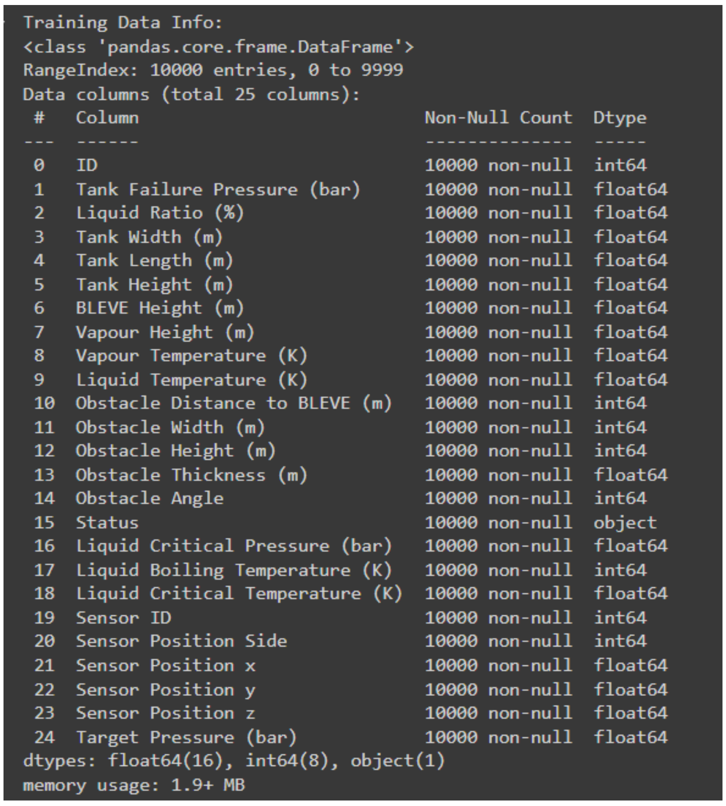
Testing Data Head:

	ID	Tank Failure Pressure (bar)	Liquid Ratio (%)	Tank Width (m)	Tank Length (m)	Tank Height (m)	BLEVE Height (m)	Vapour Height (m)	Vapour Temperature (K)	Liquid Temperature (K)	...	Obstacle Angle	Status	Liquid Critical Pressure (bar)	Tem
0	0	16.8	0.336426	2.8	8.6	2.2	0.4	1.6	389.9	387.2	...	1	Superheated	37.9	
1	1	16.8	0.336426	2.8	8.6	2.2	0.4	1.6	389.9	387.2	...	1	Superheated	37.9	
2	2	16.8	0.336426	2.8	8.6	2.2	0.4	1.6	389.9	387.2	...	1	Superheated	37.9	
3	3	16.8	0.336426	2.8	8.6	2.2	0.4	1.6	389.9	387.2	...	1	Superheated	37.9	
4	4	16.8	0.336426	2.8	8.6	2.2	0.4	1.6	389.9	387.2	...	1	Superheated	37.9	

5 rows × 24 columns

Understanding the Top 5 Rows using .head

Next the .shape command was used on both to find further details about the rows and columns included. It further clarified the size and the dimensionality of each of the features included. Observations were derived about the total count for data features and their associated data points, and its type.



```
Training Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   ID                             10000 non-null  int64
 1   Tank Failure Pressure (bar)    10000 non-null  float64
 2   Liquid Ratio (%)               10000 non-null  float64
 3   Tank Width (m)                 10000 non-null  float64
 4   Tank Length (m)                10000 non-null  float64
 5   Tank Height (m)                10000 non-null  float64
 6   BLEVE Height (m)               10000 non-null  float64
 7   Vapour Height (m)              10000 non-null  float64
 8   Vapour Temperature (K)         10000 non-null  float64
 9   Liquid Temperature (K)         10000 non-null  float64
 10  Obstacle Distance to BLEVE (m) 10000 non-null  int64
 11  Obstacle Width (m)             10000 non-null  int64
 12  Obstacle Height (m)            10000 non-null  int64
 13  Obstacle Thickness (m)         10000 non-null  float64
 14  Obstacle Angle                 10000 non-null  int64
 15  Status                         10000 non-null  object
 16  Liquid Critical Pressure (bar) 10000 non-null  float64
 17  Liquid Boiling Temperature (K) 10000 non-null  int64
 18  Liquid Critical Temperature (K) 10000 non-null  float64
 19  Sensor ID                      10000 non-null  int64
 20  Sensor Position Side           10000 non-null  int64
 21  Sensor Position x              10000 non-null  float64
 22  Sensor Position y              10000 non-null  float64
 23  Sensor Position z              10000 non-null  float64
 24  Target Pressure (bar)          10000 non-null  float64
dtypes: float64(16), int64(8), object(1)
memory usage: 1.9+ MB
```

Understanding the size and dimensionality using .shape

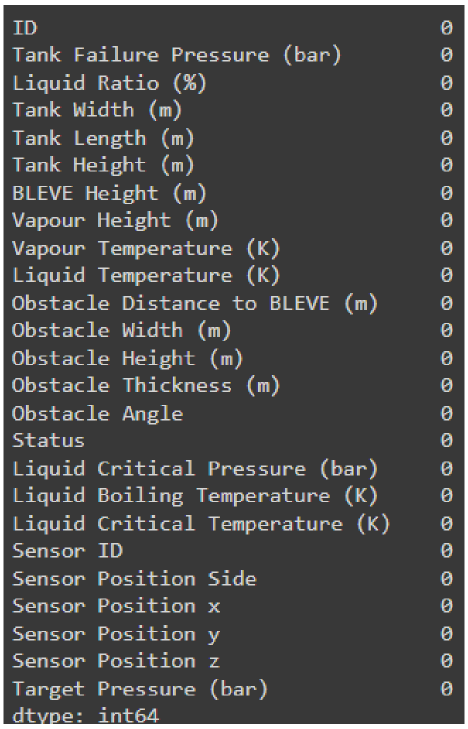
To further get deep insights about each of the feature's statistical summary, the command .describe() was used to understand information like mean, standard deviation, minimum, and maximum values. It helps in identifying potential outliers and recognizing the distribution of data.

2.3) Data Cleaning:

After the deep dive into the data and understanding the basic structure and features within, I moved forward to the process of cleaning the data. This is a critical step as it identifies and errors that need correcting or if there are any inconsistencies or inaccuracies that can be resolved at the earliest. This process ensure that the data is reliable enough so that the machine learning models can give their best possible results.

2.3.1) Missing Values:

Both files were thoroughly explored to find missing values. The intention was to fill them up using different methods so that all null values can be treated. However, both files were had no empty slots as all data instances had something included. The method used was to sum up the null values by the command 'isnull().sum()'



```
ID                              0
Tank Failure Pressure (bar)     0
Liquid Ratio (%)                0
Tank Width (m)                  0
Tank Length (m)                 0
Tank Height (m)                 0
BLEVE Height (m)                0
Vapour Height (m)               0
Vapour Temperature (K)          0
Liquid Temperature (K)          0
Obstacle Distance to BLEVE (m)  0
Obstacle Width (m)              0
Obstacle Height (m)             0
Obstacle Thickness (m)          0
Obstacle Angle                  0
Status                          0
Liquid Critical Pressure (bar)  0
Liquid Boiling Temperature (K)  0
Liquid Critical Temperature (K) 0
Sensor ID                       0
Sensor Position Side            0
Sensor Position x               0
Sensor Position y               0
Sensor Position z               0
Target Pressure (bar)           0
dtype: int64
```

Quantity of Missing Values by features

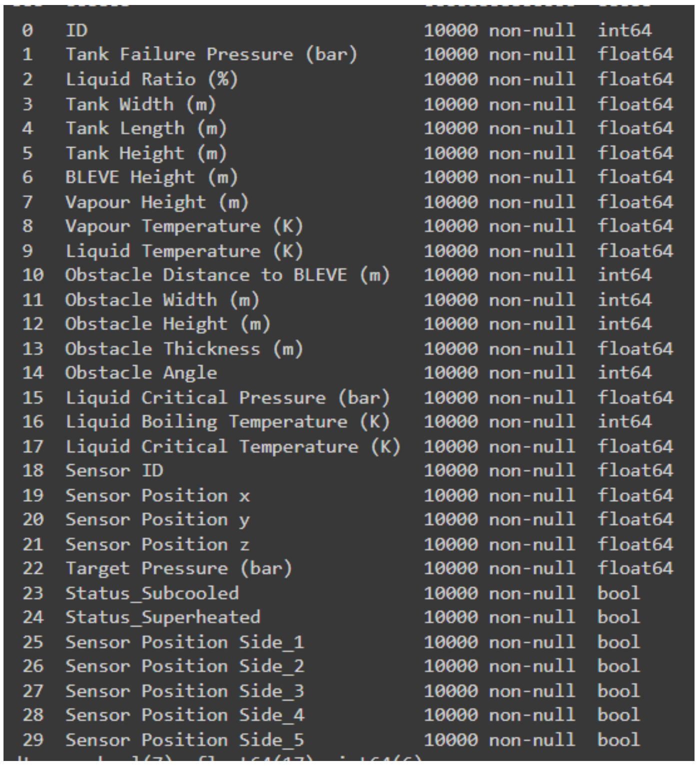
2.3.2) Duplicate Values:

Next, I moved towards the search for duplicate values. Duplicate records can introduce inaccuracies and bias if not handled properly. This ultimately effects the models performance as the results are far from the actual values. In order to find them, the command train_data.duplicated().sum() was used on both files. However, the results indicated that there were 0 rows that were having a duplicate record. This ensures the integrity of the data being robust and ready for next steps.

2.3.3) Data Type Conversion

As machine learning algorithms perform best with numerical input, I tried to find all the categorical variables found within the data. The discovery using .dtypes command showed 3 features having categorical data types. These included 'Status', 'Sensor ID', 'Sensor Position Side. The ‘Status’ feature indicated only two possible values which were either ‘subcooled’ or ‘superheated’. Similarly, the feature 'Sensor Position Side’ shows the wall side where the sensor was positioned in 3 dimensional environment, There were a total of 5 unique values. Therefore for these two I decided to use One hot Encoding method. This technique allows categorical variables to convert into a format that allocates a special row to each of its unique values and uses binary of 0 and 1 to show whether that is available for that particular row or not.

Another feature called the “Sensor ID” had a total of 26 Unique Values. Hence, based on this information I did not use the one hot encoding method on this since it would have created too many columns. This would have made the dataset a sparse one with majority of data points being 0. This would have caused computational complexity making the program taking more time to run and also negatively impacting the end results from prediction models. Therefore, for this particular feature I used the mean encoding method that allows average outcome of each category into the model's training process, which can help learn associations between specific sensors and their typical pressure levels. Below is a snapshot of the features and their corresponding data type after carrying out data type conversion.



```
0   ID                              10000 non-null  int64
1   Tank Failure Pressure (bar)     10000 non-null  float64
2   Liquid Ratio (%)                10000 non-null  float64
3   Tank Width (m)                  10000 non-null  float64
4   Tank Length (m)                 10000 non-null  float64
5   Tank Height (m)                 10000 non-null  float64
6   BLEVE Height (m)                10000 non-null  float64
7   Vapour Height (m)               10000 non-null  float64
8   Vapour Temperature (K)          10000 non-null  float64
9   Liquid Temperature (K)          10000 non-null  float64
10  Obstacle Distance to BLEVE (m)  10000 non-null  int64
11  Obstacle Width (m)              10000 non-null  int64
12  Obstacle Height (m)             10000 non-null  int64
13  Obstacle Thickness (m)          10000 non-null  float64
14  Obstacle Angle                  10000 non-null  int64
15  Liquid Critical Pressure (bar)  10000 non-null  float64
16  Liquid Boiling Temperature (K)  10000 non-null  int64
17  Liquid Critical Temperature (K) 10000 non-null  float64
18  Sensor ID                       10000 non-null  float64
19  Sensor Position x               10000 non-null  float64
20  Sensor Position y               10000 non-null  float64
21  Sensor Position z               10000 non-null  float64
22  Target Pressure (bar)           10000 non-null  float64
23  Status_Subcooled                10000 non-null  bool
24  Status_Superheated              10000 non-null  bool
25  Sensor Position Side_1          10000 non-null  bool
26  Sensor Position Side_2          10000 non-null  bool
27  Sensor Position Side_3          10000 non-null  bool
28  Sensor Position Side_4          10000 non-null  bool
29  Sensor Position Side_5          10000 non-null  bool
```

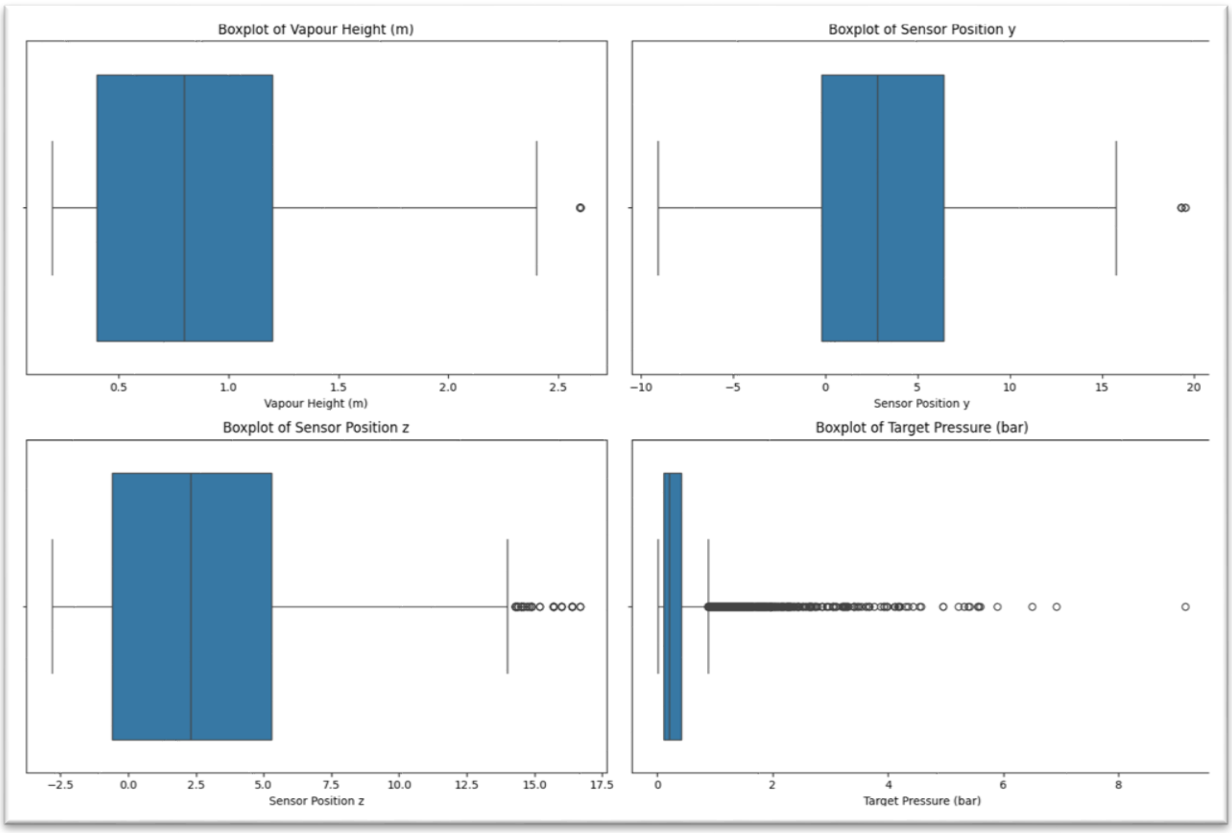
Features of Test_data after Data Type Conversion

2.4) Normalization

Once the data cleaning was all complete, the next step included a normalization of the data. It is another important step which involves scaling the features so that as a whole it can be ensured that they are contributing equally to the model predictions and improves the convergence speed of the intended algorithm uses

2.4.1) Outlier Analysis

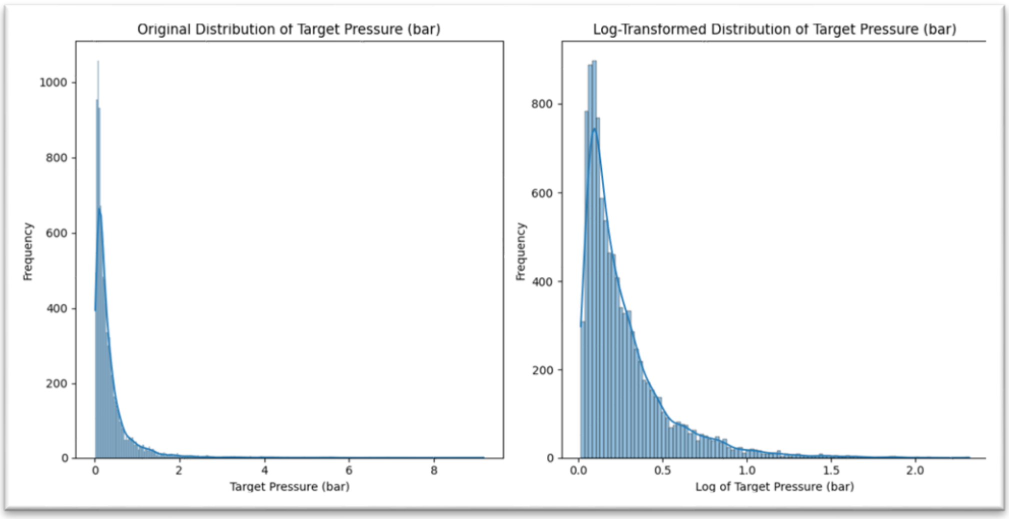
In order to find which features to normalize, I did the outlier analysis to find out the features that have observations within the data that differ significantly from others. This will allow to understand why certain features are skewed in a particular direction as it can cause misleading results when trying to train the data. For this I found the Z scores for all numerical columns as it measures the standard deviations of a data point from its mean. From this I was able to find 4 different features having multiple data points that are more than the z scores of 0. Below is an attached screenshot of boxplot showing the features and their corresponding outlier values.



Outlier Analysis

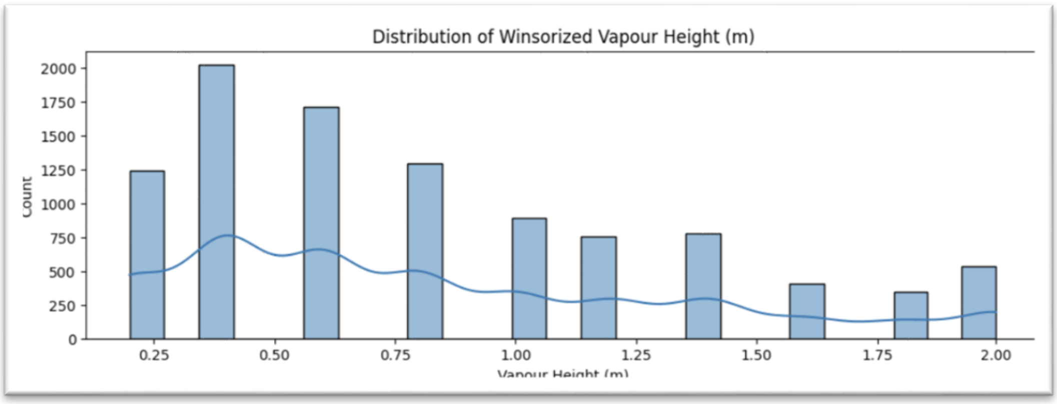
2.4.2) Feature Scaling

The sensor position Y and Z indicate specific locations that could be physically higher or lower than most other sensors. This can be a part of the data and normalizing on treating outliers may cause disruption in actual data hence I decided to keep them as it is. However, for Target Pressure bar, the extreme values can significantly affect the model's performance therefore I normalized them by applying a log transformation to reduce the skewness and lessen the effect of extreme values. The visualization below shows the original distribution of Target Pressure (Bar) found in the original data and also the log-transformed distribution.



Distritubuion of Target Pressure (bar) before and after Log Normalization

For vapour height, I decided to winsorize them because there was not too many values that may affect the data too much. Hence applying any other normalization technique wouldn't be appropriate however cutting them down would be a better option. Below is a snapshot of the distribution after the winsorization was completed.



Vapour Height Distribution after Winsorization

2.5) Feature Engineering

There are many features already in the training data that we can use to create a robust model, however, there were still gaps that could be filled by engineering new features by using pre-existing ones. This allows leveraging domain knowledge to create features that better represent the underlying patterns in the data. Hence, by creating new features that involves ratios, and combined metrics there is a chance of improving the models ability to make even more accurate predictions. For this particular reason my approach was to find and implement as many relevant ideas I can find to create further features and then find through correlation and other feature selection methods if they are worth keeping to train models. Below are the new features engineered

Tank Dimension Ratios: The shape of the tank plays a critical role in how pressure is built and released during the explosion scenario. Hence to capture the geometrics of this this feature was engineer to capture the aspect ratio of the tank. Ratios of height to width and length to width can describe the shape of the tank. This will help better understand how the shape of the tank plays the role in pressure dynamics. To calculate this we can perform this for both datasets :

$$Tank\ Height\ to\ Width\ Ratio = Tank\ Height\ (m)/Tank\ Width\ (m)$$

$$Tank\ Length\ to\ Width\ Ratio = Tank\ Length\ (m)/Tank\ Width\ (m)$$

Relative Positioning: With this feature we can find the relative distance of the obstacle to the BLEVE event that while in relation to the tank dimensions. This shows us how close the obstacle is to the tank and how it can influence the pressure of the explosion. To calculate this we can use the following:

$$Relative\ Obstacle\ Distance = Obstacle\ Distance\ to\ BLEVE\ (m)/Tank\ Width\ (m)$$

Combined Temperature Features: Another feature that could allow us to find deep insights into the volatility and stability of the tank's content can be seen through the difference between the vapour and liquid temperatures. A larger different can indicate a higher volatility and vice versa. To calculate this I used this formula:

$$Temperature\ Difference = Vapour\ Temperature\ (K) - Liquid\ Temperature\ (K)$$

Scaled Pressure: As a feature I wanted to use a scaled version of the tank failure pressure by the liquid critical pressure. This would provide a normalized version of the risk or potential impact. This can tell us how close to tank is to the failure point under different pressure conditions. For this the Formula used is :

Scaled Tank Failure Pressure=Tank Failure Pressure (bar)/Liquid Critical Pressure (bar)

Tank Base Area: Just to experiment, I also tried to calculate the base area of the tank as it could indicate how much liquid is in direct contact with the base of the tank. This can affect the heat transfer and pressure development. This is achieved with this basic formula

Tank Base Area (m^2)=Tank Width (m)×Tank Length (m)

Tanks Surface Area: Very similar to the above feature engineer, I tried to calculate the surface area of the tank as it can imply more exposure and external conditions. It may influence how the pressure is build up and then released. The formula used is :

Tank Surface Area (m^2)= 2 × (Tank Width (m) × Tank Length (m) + Tank Height (m) × Tank Length (m) + Tank Height (m) × Tank Width (m))

Combined Gas and Liquid Properties: A combination of Gas and Liquid was intended as the ratio of vapor to liquid height within the tank can provide insights into the state of the mixture. It would help in understanding the balance between gas and liquid found within the tank. The formula is below:

Vapour to Liquid Ratio=Vapour Height (m)/Tank Height (m)

Normalized Obstacle Proximity: Comparing the obstacle distance to the overall size of the tank setup might give a better sense of how constrained the space is, potentially affecting the dynamics of the explosion. This will give deep insights into the relative positioning of obstacles in relation to the tank. The formula used is

Normalized Obstacle Distance=Obstacle Distance to BLEVE (m)/(Tank Width (m)+Tank Length (m))

Average Temperature: An average of the vapor and liquid temperatures might simplify the model without losing critical thermal information. It will provide a combined measure of an average overall temperature found in the tank.

Average Temperature (K)=(Vapour Temperature (K)+Liquid Temperature (K))/2

Obstacle Impact Factor: Creating a feature that combines obstacle dimensions with its distance could provide a proxy for how much the obstacle might affect the blast wave propagation. Hence, the size and the proximity of the obstacles will be considered to give a detailed measure of its results.

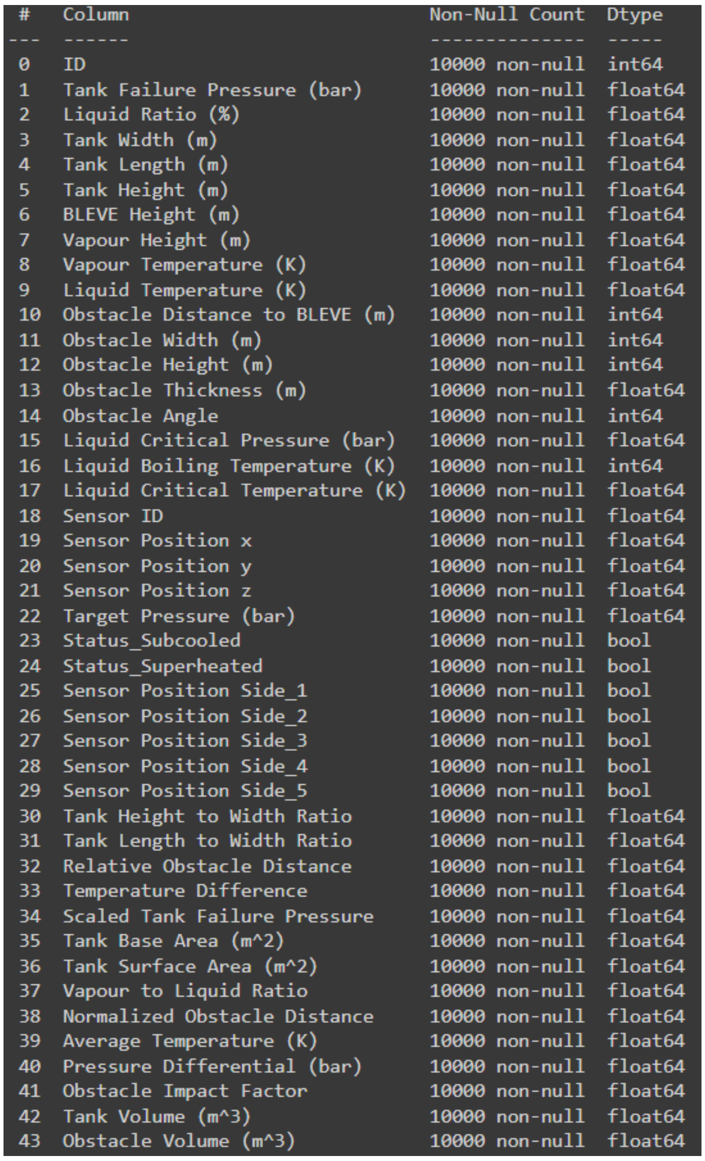
Obstacle Impact Factor=(Obstacle Width (m)×Obstacle Height (m))/Obstacle Distance to BLEVE (m)

Volume of the Tank and Obstacle: Volume can also be a good indicator for insights about the space the tank as well as the obstacle occupy. This volume can tell us what are the areas in general that are impacted the most when the blast takes place. The formulas for each of them are below:

Tank Volume (m^3)=Tank Width (m)×Tank Length (m)×Tank Height (m)

Obstacle Volume (m^3)=Obstacle Width (m)×Obstacle Height (m)×Obstacle Thickness (m)

Once all the features were engineered, the total number of trainable features were then increased to 43 including the target pressure bar. A snapshot of all of them is attached below.



```
#   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
0   ID                             10000 non-null  int64
1   Tank Failure Pressure (bar)    10000 non-null  float64
2   Liquid Ratio (%)               10000 non-null  float64
3   Tank Width (m)                 10000 non-null  float64
4   Tank Length (m)                10000 non-null  float64
5   Tank Height (m)                10000 non-null  float64
6   BLEVE Height (m)               10000 non-null  float64
7   Vapour Height (m)              10000 non-null  float64
8   Vapour Temperature (K)         10000 non-null  float64
9   Liquid Temperature (K)         10000 non-null  float64
10  Obstacle Distance to BLEVE (m) 10000 non-null  int64
11  Obstacle Width (m)             10000 non-null  int64
12  Obstacle Height (m)            10000 non-null  int64
13  Obstacle Thickness (m)         10000 non-null  float64
14  Obstacle Angle                 10000 non-null  int64
15  Liquid Critical Pressure (bar) 10000 non-null  float64
16  Liquid Boiling Temperature (K) 10000 non-null  int64
17  Liquid Critical Temperature (K) 10000 non-null  float64
18  Sensor ID                      10000 non-null  float64
19  Sensor Position x              10000 non-null  float64
20  Sensor Position y              10000 non-null  float64
21  Sensor Position z              10000 non-null  float64
22  Target Pressure (bar)          10000 non-null  float64
23  Status_Subcooled               10000 non-null  bool
24  Status_Superheated             10000 non-null  bool
25  Sensor Position Side_1         10000 non-null  bool
26  Sensor Position Side_2         10000 non-null  bool
27  Sensor Position Side_3         10000 non-null  bool
28  Sensor Position Side_4         10000 non-null  bool
29  Sensor Position Side_5         10000 non-null  bool
30  Tank Height to Width Ratio     10000 non-null  float64
31  Tank Length to Width Ratio     10000 non-null  float64
32  Relative Obstacle Distance     10000 non-null  float64
33  Temperature Difference         10000 non-null  float64
34  Scaled Tank Failure Pressure   10000 non-null  float64
35  Tank Base Area (m^2)           10000 non-null  float64
36  Tank Surface Area (m^2)        10000 non-null  float64
37  Vapour to Liquid Ratio         10000 non-null  float64
38  Normalized Obstacle Distance   10000 non-null  float64
39  Average Temperature (K)        10000 non-null  float64
40  Pressure Differential (bar)    10000 non-null  float64
41  Obstacle Impact Factor         10000 non-null  float64
42  Tank Volume (m^3)              10000 non-null  float64
43  Obstacle Volume (m^3)          10000 non-null  float64
```

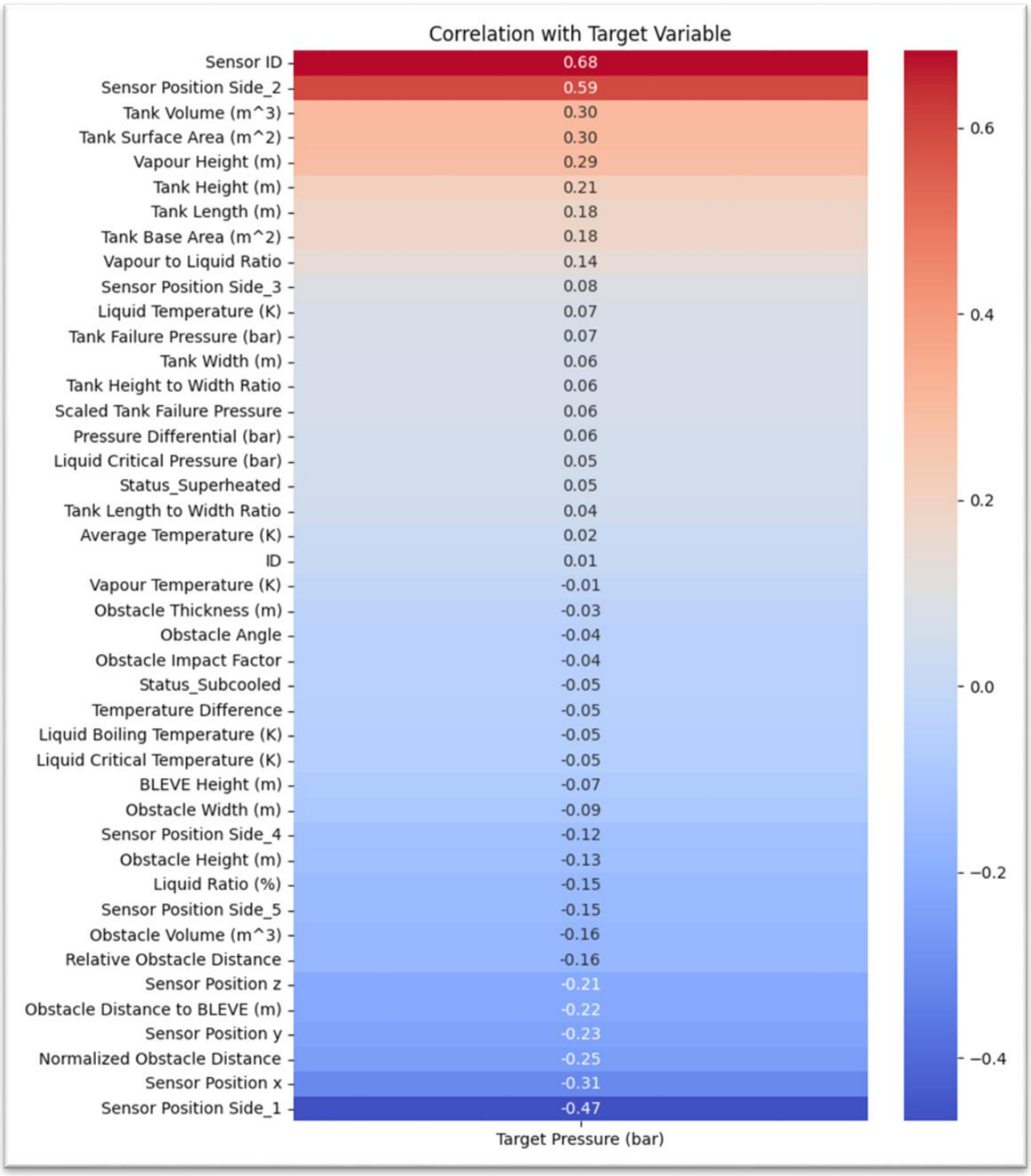
All trainable Features after Feature Engineering

2.6) Feature Selection

Once new features were engineered, the next step involved the selection. This requires picking the most relevant features from the dataset that can contribute more towards a better prediction result from different models. All features that are redundant or only add noise will be removed. The selection was done through multiple considerations.

2.6.1) Correlation

First was to find the correlation between all the features against the target variable which is the Target Pressure (Bar). This helps find which of them is strongly related and will have more impact due to change while also indicating those that are not that much related. This helps in reducing the dimensionality so that the models can run smoothly without considering information within the data that is not that relevant or impactful for prediction results. Below is a snapshot of the correlation values of all the features against the target variable.

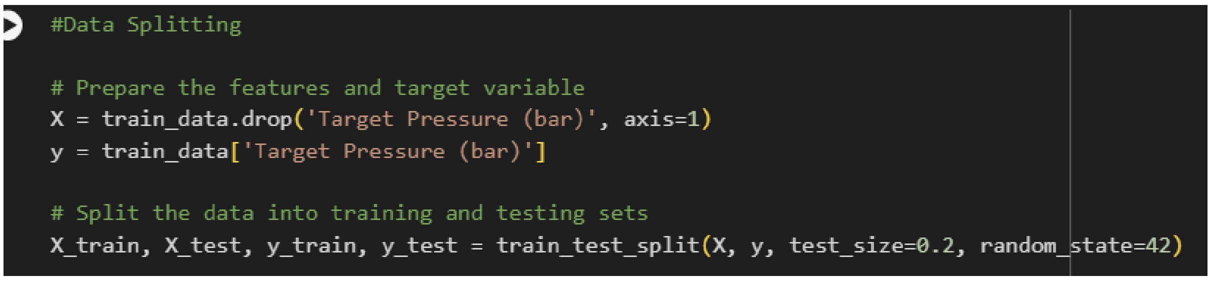


Correlation Matrix for all Trainable Features

It shows that the highest positive correlation was with the Sensor ID as it being 0.68. This suggests that the position of the sensor within the tanks plays a vital role in how the target pressure bar is recorded. The Tank Volume, which is an engineer feature also shows a high 0.30 positive relationship with Pressure that implies that the larger the volume of the tank the larger will be the overall pressure. Apart from this, there were some features identified to having little or no correlation. For example, Average Temperature which is another engineer experiment showing the that the average temperature within the tank does not really effect the pressure being created. The ones at the bottom in Blue color shows that these feature area adversely affecting the target. These are significant feature as well since they are changing how the pressure bar works. Apart from Sensor position side, the Normalize Obstacle Distance shows a -0.25 correlation. This suggests that the closer the obstacles are means the high the pressure will be.

2.6.2) Data Splitting

Before going further into model creation or Feature selection, I wanted to first get the data splitting completed. For this the data was already split into Training and Test data. However, it was important to divide them further into test and validation sets. This allows the models to train on one half of the model and then test their performance on a validation set to check if the performance is satisfactory. If only it is, it is then sent out into the real world where it finds the prediction with unseen data set. For example the test data that is provided. There, firstly the Target Pressure bar is separated and is stored as y. Then the data is split into 80% and 20%. One for training and one for testing the validation. The random_state is set to 42 which ensures reproducibility of the split. A snapshot of the code for splitting is attached below:



```
#Data Splitting

# Prepare the features and target variable
X = train_data.drop('Target Pressure (bar)', axis=1)
y = train_data['Target Pressure (bar)']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

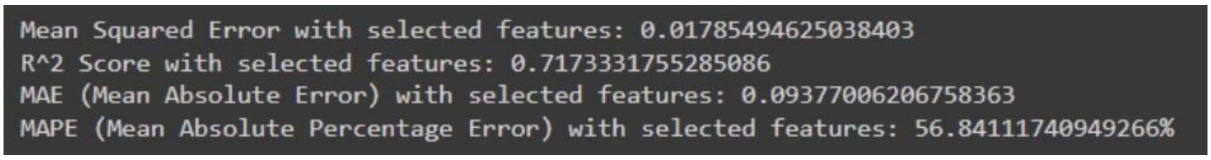
2.6.3) Feature Selection Using Lasso

Once the data was split, there were different methods considered to select the best features. However, I have used Lasso to select the features because it tend to be the most relevant ones for prediction, as Lasso's regularization penalty encourages sparsity by shrinking less important features' coefficients to zero by using cross-validation, LassoCV can find the optimal regularization strength (alpha) that minimizes prediction error on unseen data, leading to better generalization. With this the best and selected features were stored in the variable called x_train_selected and c_test_selected. These were then set as the default features that will be used by all the upcoming models.

3) Model Development

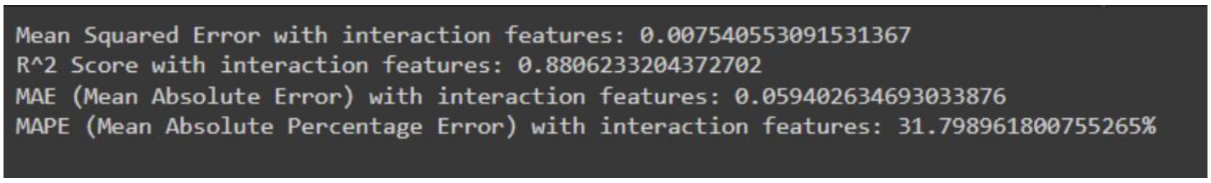
3.1) Model 1 : Linear Regression:

This being one of the most widely used machine learning model for linear problems, was incorporated as the first model in the prediction analysis. This being simple yet a powerful statistical tool used to find relationship between the dependent and independent variables. In this context of predicting peak pressure from BLEVE occasions, it is a very useful model due to its simplicity, reproduction and also easy performance analysis. I used the Linear Regression class from the sklearn library. The features used within this model were selected from the previous method of Lasso. Hence, the model was trained and used the validation to find its performance. I used multiple metrics to understand how it performed. From the early visit I got a R^2 of 0.71 with MAPE being 56.84.



```
Mean Squared Error with selected features: 0.01785494625038403
R^2 Score with selected features: 0.7173331755285086
MAE (Mean Absolute Error) with selected features: 0.09377006206758363
MAPE (Mean Absolute Percentage Error) with selected features: 56.84111740949266%
```

To improve this further I tested multiple techniques to carry out hyperparameter tuning. This for a linear regression model is fairly easy however I chose to use the Polynomial Feature class with a degree of 2 and interaction_only set is True. This would allow to capture the relationships between different features that were not fully captured when considering them individually. This was aimed to improve the performance of the prediction results by combing the features. The results were improved immensely as the R^2 was now touching 88%, whereas the MAPE felt down to 31. Below shows a screenshots of the improved results.

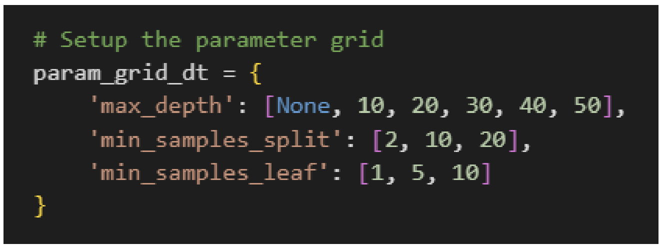


```
Mean Squared Error with interaction features: 0.007540553091531367
R^2 Score with interaction features: 0.8806233204372702
MAE (Mean Absolute Error) with interaction features: 0.059402634693033876
MAPE (Mean Absolute Percentage Error) with interaction features: 31.798961800755265%
```

To reproduce these results, others can use the code while selecting different features to have a differentiated output. The degree can be increased in the hyperparameters for polynomial features to find deeper connections between the features to get even more deeper insights.

3.2) Model 2 : Decision Tree and Random Forrest

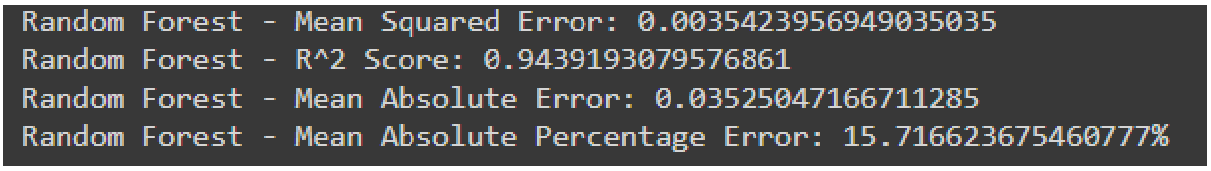
Another widely used model for regression task is Decision Tree. It requires the data to be splitted up into subsets based on certain values and input features. The splits are made in a way that maximizes the separation between different outcomes. This model is great when it comes to handling complex relationships between different features. This was the exact requirement by the data set related to BLEVE scenarios. Therefore, a model was created using the Grid Search to choose from different parameters involving the max depth, min_samples_split and also the min_samples_leaf. A snapshot of the used parameter grid is shown below.



```
# Setup the parameter grid
param_grid_dt = {
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 10, 20],
    'min_samples_leaf': [1, 5, 10]
}
```

The results showed impressive results with the best parameters {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2}. I was able to get great R^2 of 0.86 and also a Mape much lesser than the Linear Regression Model of 20.54.

In order to improve this model, I went deeper to create a Random Forrest which is an ensemble learning method that builds multiple decision trees and merges their outputs to improve accuracy and control overfitting. This allows a reduction in the overall variance and also a robust prediction result. This was imported from the sklearn library. Based on the parameters of n_estimators being 100 and the random_state being 42. An impressive model was created gibing better results than the general Decision Tree used before. The results are shown below.



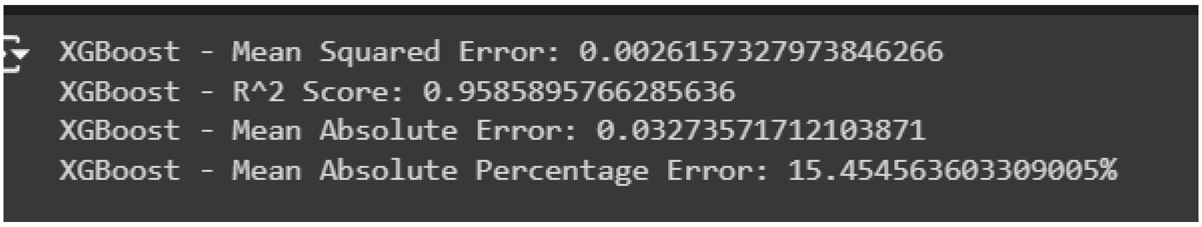
```
Random Forest - Mean Squared Error: 0.0035423956949035035
Random Forest - R^2 Score: 0.9439193079576861
Random Forest - Mean Absolute Error: 0.03525047166711285
Random Forest - Mean Absolute Percentage Error: 15.716623675460777%
```

The code is reproducible by others and does allow a lot of room for adjustments. I used the selected features from LASSO however, another method of feature selection can be used. The hyperparamters of random state can also the parameter grid for decision tree can be adjusted to find further insights on how the model can perform.

3.3) Model 3 : XGBoost

XGboost which is also known as the extreme Gradient Boosting model, is an advanced and powerful model that allows gradient boosting framework. This model is generally known for its high performance and accurate prediction results. With its built-in regularization techniques, the model was a good choice to be used for the BLEVE dataset as it will be able to capture complex relationships and give improved results. Within the code the model was imported with the name XGBRegressor from the xgboost library.

The hyperparameters used within the model to set the learning task as the regression analysis was set to 'reg:squarederror'. Furthermore, the n_estimators was set to 100 and the seeds reproducibility was set to 42. With this the results achieved were surprisingly good as the Mean Squared Error (MSE): 0.00262, R² Score: 0.9558, and Mean Absolute Error (MAE): 0.0327.



```
XGBoost - Mean Squared Error: 0.0026157327973846266
XGBoost - R^2 Score: 0.9585895766285636
XGBoost - Mean Absolute Error: 0.03273571712103871
XGBoost - Mean Absolute Percentage Error: 15.454563603309005%
```

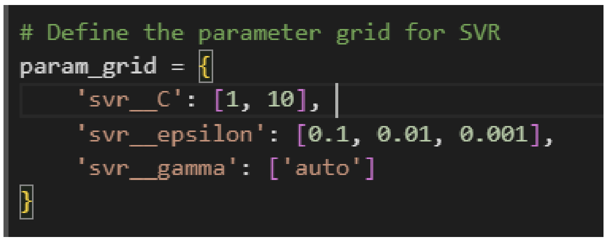
Mean Absolute Percentage Error (MAPE): 15.45% There was a grid search set with further hyperparameters to tune however, the code was taking way too long with not much improvement in the results hence was kept simple.

The reproducibility of this model is very simple since not many hyperparameters were used in the final output of the code. The code does allow immense opportunity for changes that can further be experimented to see if it can improve the results. There can be inclusion of further hyperparameters which can involve the learning rate, the max depth, subsamples and also the gamma. The Cross Validation technique can be further added to go through a range of hyperparameters to see which ones are best for the most accurate results. Lastly, once again changes to the above sections like engineering new features and using them within the model can also help in finding a better result when reproducing.

3.4) Model 4 : XGBoost

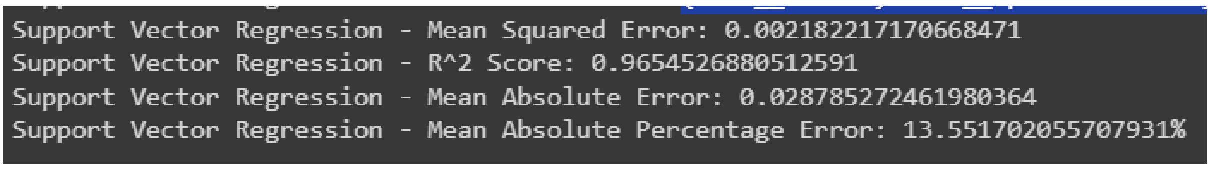
Support Vector Regression (SVR) is also a well renowned model for usage in regression prediction task. This machine learning model finds a function considering specified margins which is extracted from observed values. It does were effectively when the data set has high dimensionality and in the current scenario, this model was ideal one to predict the peak pressure values for BLEVE scenarios as it runs well with non-linear relationships as found by the polynomial analysis done in the Linear Regression. The model was called from the sklearn library and then multiple additions were made to make it more robust in giving the results.

The hyperparameters used within the model were cross validated using the grid-search method. There were multiple parameters set in the grid which involved the scr_C that was put in between the range of 1 to 10. The svr_epsilon which is the epsilon-tube within which no penalty is associated in the training loss function, with values [0.1, 0.01, 0.001]. Furthermore, the svr_gamma that was set between the kernel coefficients of 'rbf', 'poly', and 'sigmoid'. A screenshot of the param_grid is shown below.



```
# Define the parameter grid for SVR
param_grid = {
    'svr__C': [1, 10],
    'svr__epsilon': [0.1, 0.01, 0.001],
    'svr__gamma': ['auto']
}
```

There were other hyperparameter also tuned to get more variations of the results however, this model in particular was taking several minutes to run. The results did had slight changes however not enough to change the hyperparameters included. The best ones selected were {'svr__C': 10, 'svr__epsilon': 0.01, 'svr__gamma': 'auto'} and this was able to get an exceptional result with a high R² Score of 0.9654 and a reduced Mape of 13.55%.



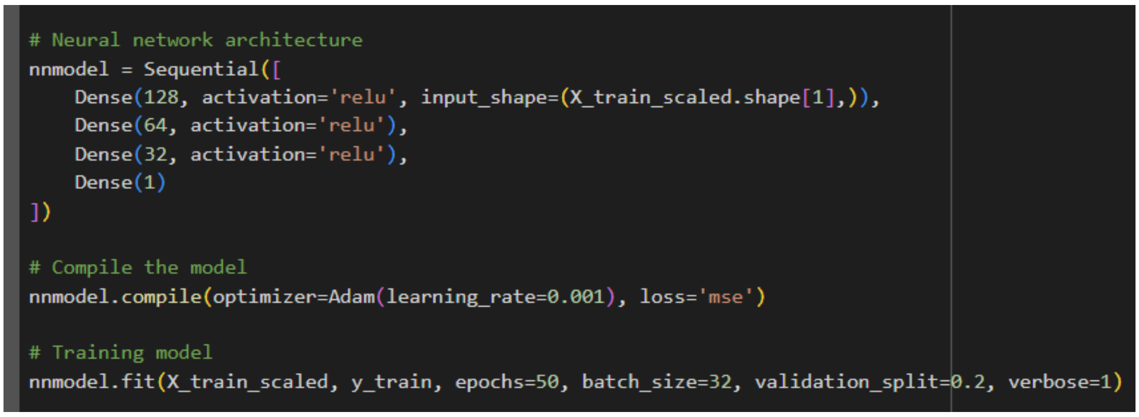
```
Support Vector Regression - Mean Squared Error: 0.002182217170668471
Support Vector Regression - R^2 Score: 0.9654526880512591
Support Vector Regression - Mean Absolute Error: 0.028785272461980364
Support Vector Regression - Mean Absolute Percentage Error: 13.551702055707931%
```

The code once again is highly repeatable as it does not take too much computer power nor is it too complex. There are other changes to degree, kernel, shrinking and cache_size that were experimented but excluded from the final code however, anyone else in search to improve it further can always include it.

3.5) Model 5 : Neural Networks

This model is an advanced tool that uses high computational power to find prediction results. It works similar to a human brain with interconnected nodes that recognize patterns and allow the model to make accurate predictions. These neurons are particularly efficient when capturing non-linear relationships due to their layered structure. In the context of finding peak pressure prediction for BLEVE scenarios, the neural network is implemented using TensorFlow and Kares Libraries. It was a complex model that had multiple dense layers with Relu Activation functions that were optimized using the Adam optimizer.

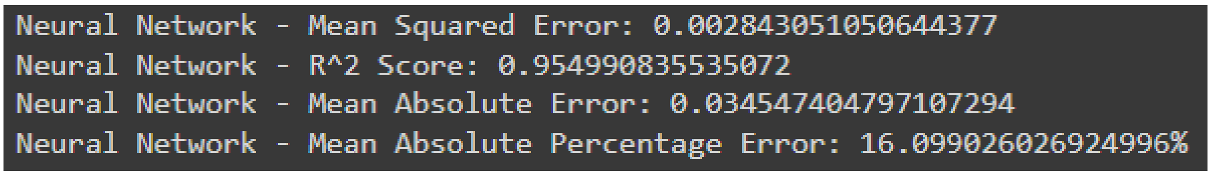
In order to train the model, the same selected features where used and were scaled using the standardscaler. The hidden layers were set densly with 128,64 and 32 neurons respectively. They were using the ReLU activation functions to find the target variable. The hyperparameters were set to the learning rate of 0.001 with Adam optimizer. The batch size was experimented however was set to 32. Number of Epochs were set as 50 which allowed to go through the dataset multiple times including the backward and forward passes. A screenshot of these hyperparameter and the following result is shown below.



```
# Neural network architecture
nnmodel = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

# Compile the model
nnmodel.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Training model
nnmodel.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=1)
```



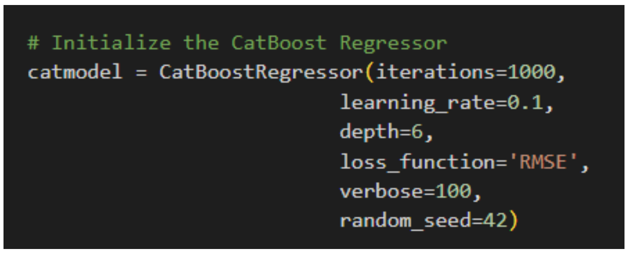
```
Neural Network - Mean Squared Error: 0.002843051050644377
Neural Network - R^2 Score: 0.954990835535072
Neural Network - Mean Absolute Error: 0.034547404797107294
Neural Network - Mean Absolute Percentage Error: 16.099026026924996%
```

The results showed an impressive performance as the model was able to capture complex patterns within the data. The output had a highly reduced MAPE and also the R^2 0.95 suggesting the results were not that overfitted. These can be further experimented when reproducing with the change in number of layers. The learning rates can also be adjusted however this will result in higher run time and the code may even get stuck due to complex computation happening in the background.

3.6) Model 6 : CatBoost

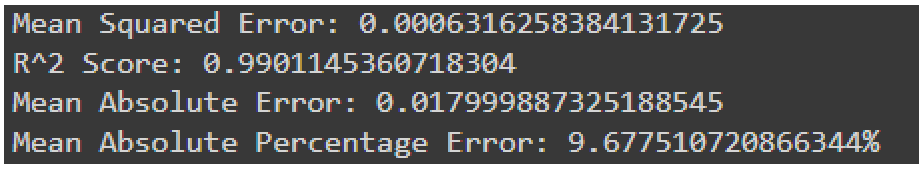
Similar to XGBoost, this is also a gradient boosting algorithm. Upon research, I was able to understand that this model is famous for its robustness in results and also its feature to reduce overfitting while giving precise results. Considering the results seen from the last few models, I wanted to experiment something that can get high results however does take into account the issue of overfitting in the background. Therefore, by using the same feature set given by LASSO, the model was implemented.

The Hyperparameters were experimented in numerous ways as the iterations were set to 1000 to reduce time for running. However, the learning rate was kept 0.1, which higher than what was set in neural network but as this model was taking lesser time to run I wanted to keep a balance. The depth is set to 6 to control the complexity of the model and prevent it do overfitting. The loss_function is RMSE to minimize Root mean squared error and random seed of 42 for consistent results.



```
# Initialize the CatBoost Regressor
catmodel = CatBoostRegressor(iterations=1000,
                             learning_rate=0.1,
                             depth=6,
                             loss_function='RMSE',
                             verbose=100,
                             random_seed=42)
```

The Grid-search for cross validation was crated to find the best parameters however, due to the extreme run times and computational power required, it was set to a fixed numbers. Even then the model was able to provide the best results out all the 6 options considered.



```
Mean Squared Error: 0.0006316258384131725
R^2 Score: 0.9901145360718304
Mean Absolute Error: 0.017999887325188545
Mean Absolute Percentage Error: 9.677510720866344%
```

These exceptional results show a high learning of the training data with R^6 of 0.99 however, this model known for its reduction in overfitting was able to generate the lowest MAPE of only 9.67. Upon considering, with more computational power and no limit to run time, I believe this model can perform even better if we introduce more hyperparameters when trying to reproduce. These may mean introducing the l2_leaf_reg as the regularization to prevent the overfitting even further and also consider the border counts for setting a limit to the splits in numerical features.

3.7) Model Comparisons

After applying the above mentioned models to the training data set and validating it on the test data set, there were many comparisons made before choosing the best one. The metrics that were used included the MSE, R Squared, MAE and MAPE. Below is a through discussion of what each shows in the context of the BLEVE project and which ones came out to be the top 3 in each metric.

Starting from the MSE (Mean Squared Error) which captures the average of the squares of the errors. This means that it finds the difference between the estimated value and the actual values and takes an average. Based on this, the best model to have the reduced error was the CatBoost, which had only 0.00036 followed by Random Forest 0.00354 and finally the XGBoost with the value 0.00262.

Secondly, the metric used to evaluate the performance of models was the R-squared (R^2). This actually measures the proportion of variance for a dependent variable (Target Pressure Bar) that is explained by the independent variables (Selected Features). The higher it is the more robust the results are. However, it does also mean that increased value can reflect overfitting issues. With the results, Catboost once again was topping the scale with an extremely high 0.99 result. Followed by Random forest of 0.94 and then once again XGboost with the value of 0.95.

Next metric used was the Mean Absolute error (MAE). This averages out the magnitude of the errors in a set of predictions without considering their directions. It is the average of the test sample considering the differences between the actual and the prediction of each instance while keeping the weights equal. In this context as well, the same results are repeated with XGboost topping the chart with values of 0.017, Random Forest getting the second spot with 0.035 and XGBoost following with 0.032

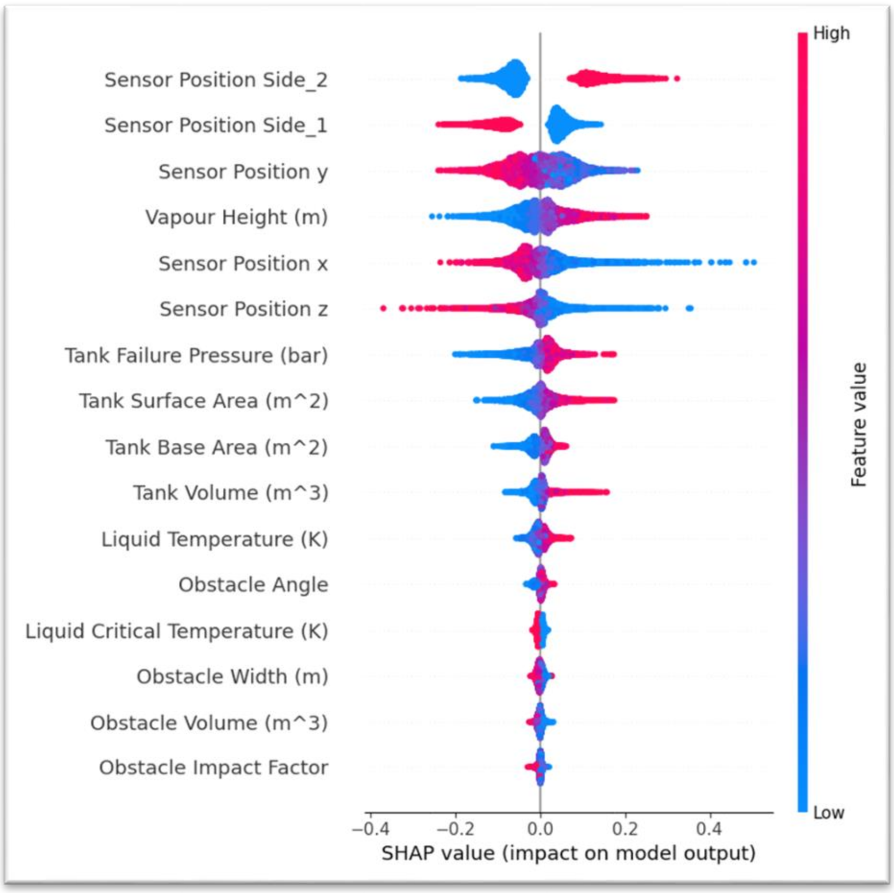
Last metric, which was highly considered while making changes to hyperparameters is the MAPE also known as the Mean Absolute Percentage Error. This measures the size of the error in percentage term. The lesser it is means the better the performance of the model. Therefore considering the results, CatBoost was able to score the lowest with only 9.67% followed by Random Forest and XGboost that scored around 15%. The same metric was also used in a kaggle competition to find the lowest score comparing it with a dataset that was uploaded on the leader board section. The results were evidently showing the same output that Catboost performed the best as it was able to score the least value of 25.045. All other predictions from other models were ranging between 50 to 26 even after tuning the hyperparameters multiple times.

With the comparison above, there was enough evidence to select Catboost performing the best among all the other models that I had applied to this dataset. It showed superior performance in all the metrics and even on the kaggle competition. It reflects how robust the results were and also it being known for treating the overfitting issue, it sets perfectly to be showing the most precise predictions. Once selected, multiple attempts were made to create a Grid-search with other hyperparamters. However, the results were only taking more time to come with predictions showing worse marks on kaggle as well as the metrics discussed above. Hence, the simple approach was taken to set specified values that are performing the best.

4) Interpretation Of Selected Model : CatBoost

4.1) Global Interpretation

4.1.1) Feature Effect Plot

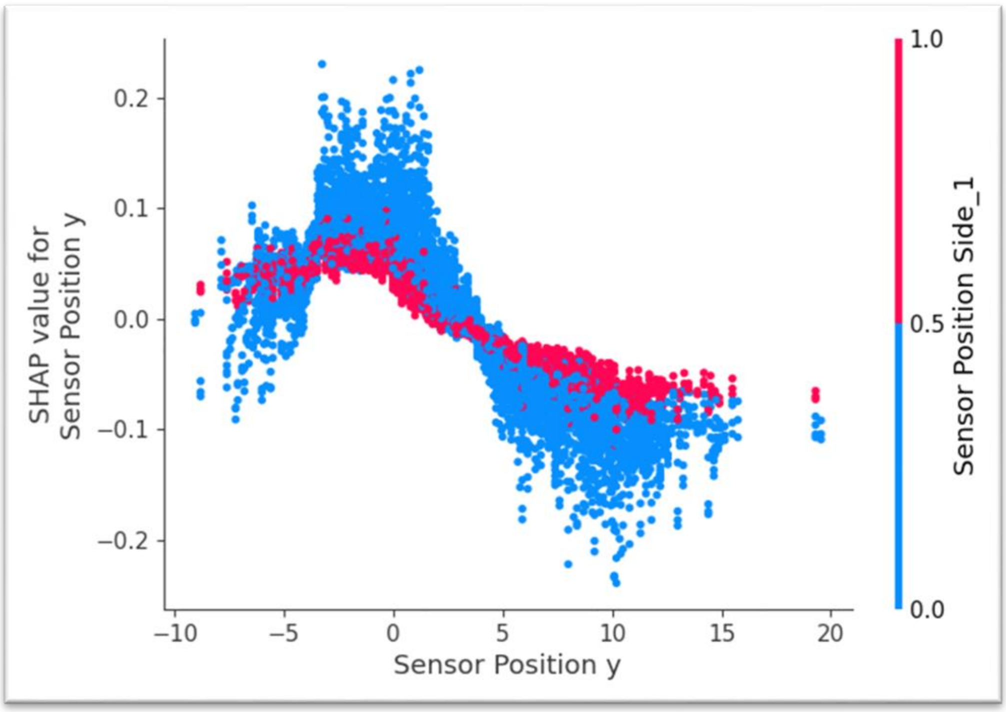


The feature effect graph above shows the contribution of each feature to the prediction of a model. The ones at the top are having the highest influence on the prediction compared to ones at the bottom. Sensor Position Side_2 and Sensor Position Side_1: These features appear to be highly influential, showing a mix of positive and significant negative effects. This suggests that the sensor positions on these sides of the obstacle have a variable but often substantial impact on the model's predictions. In other words, positions of the sensors likely affect how they capture pressure waves, impacted by factors like angle and distance to the explosion. Sensor Position y, x, z are also some of them that rank higher. These features, representing the 3D coordinates of sensors, have varied impacts. Their SHAP distributions are mixed, indicating the complex spatial dynamics in how pressure waves propagate and are captured by sensors. vapour Height: This feature shows a moderate influence with a tendency to increase the prediction as the height increases. In the context of BLEVEs, higher vapour columns within a tank could contribute to more intense pressure outputs upon explosion.

Other insights realized from this can be that the models should be more dependent on the feature engineering that is related to the sensor position and sides. Also Features like Obstacle Volume and Impact Factor show minimal influence and could potentially be omitted to simplify the model.

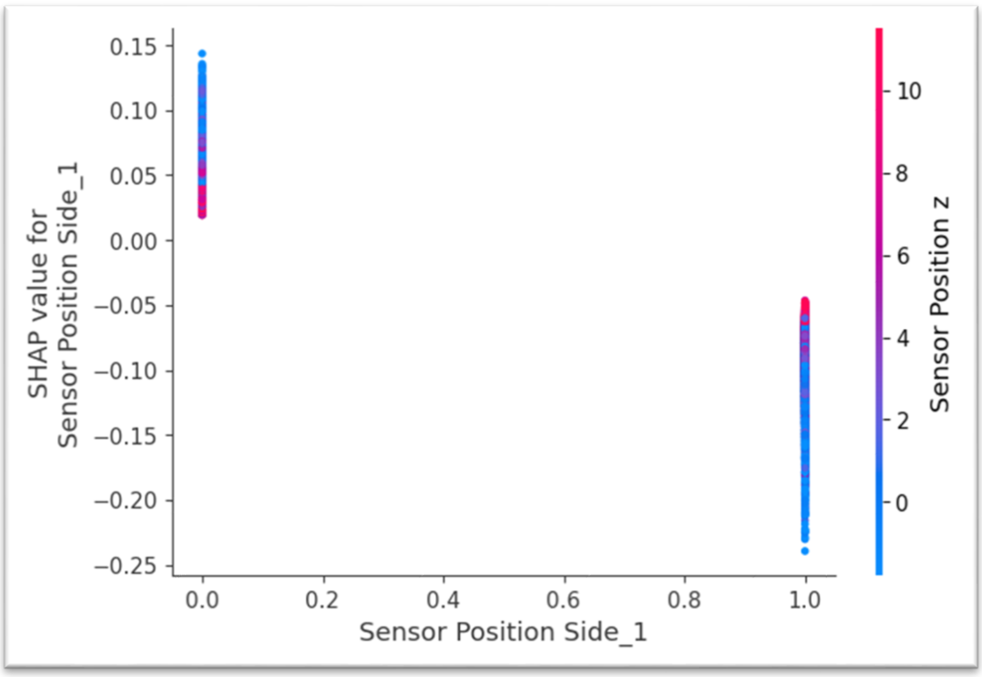
4.1.2) Partial Dependency Plots

Sensor Position y with Sensor Position Side 1



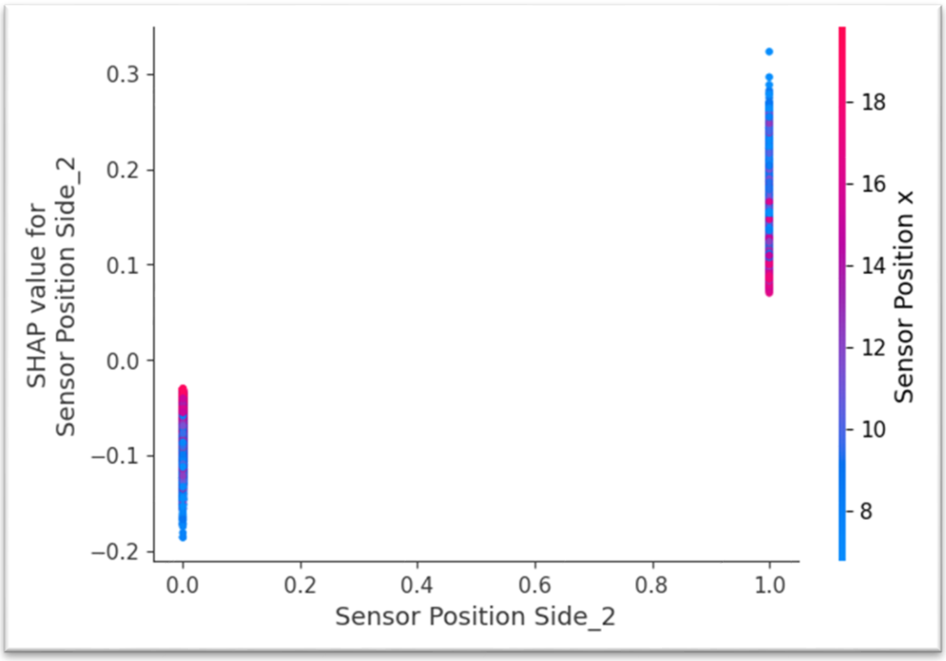
There is a notable trend where SHAP values are generally positive when the sensor position y is between -10 and 0, suggesting that as sensors are positioned more negatively along the y-axis, they tend to increase the predicted pressure. As the position moves towards the positive side from 0 to around 15, the influence on the model output shifts from positive to negative, indicating that sensors located further along the positive y-axis might be in positions that experience lower pressure. This feature's impact varies significantly across its range, highlighting its importance in predicting the outcome

Sensor Position Side 1 with sensor position Z



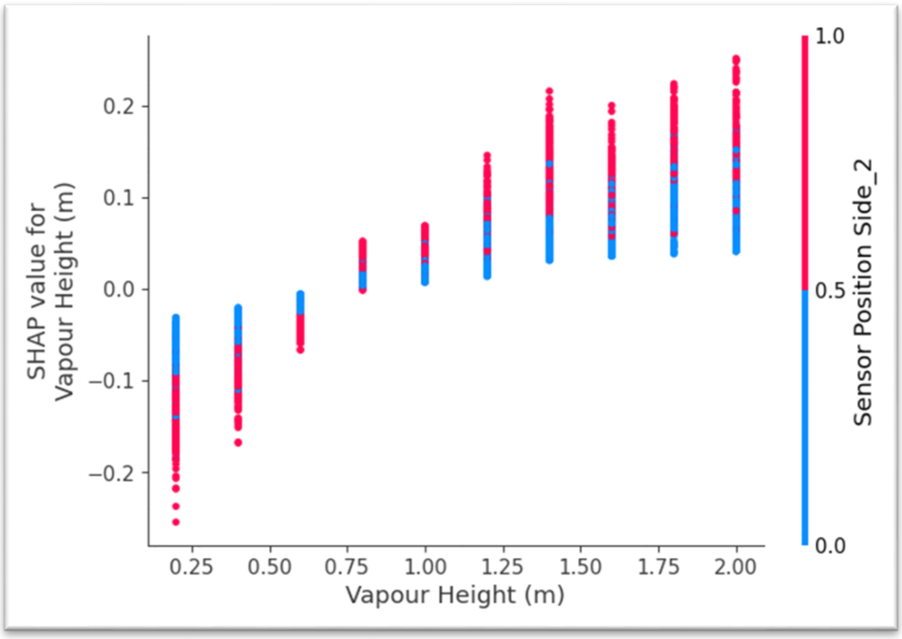
The SHAP values for Sensor Position Side_1 show only a small range of variation, predominantly hovering near zero across different positions. The plot is color-coded with Sensor Position z, indicating how the vertical position of the sensor influences the effect of Sensor Position Side_1. Despite the vertical variation, the overall impact of Sensor Position Side_1 is minimal. This suggests that the exact side of the sensor (whether left or right) does not significantly influence the predicted pressure outcomes compared to other positional factors. However, when combined with other factors this can allow better results of prediction.

Sensor Position Side 2 with Sensor Position X



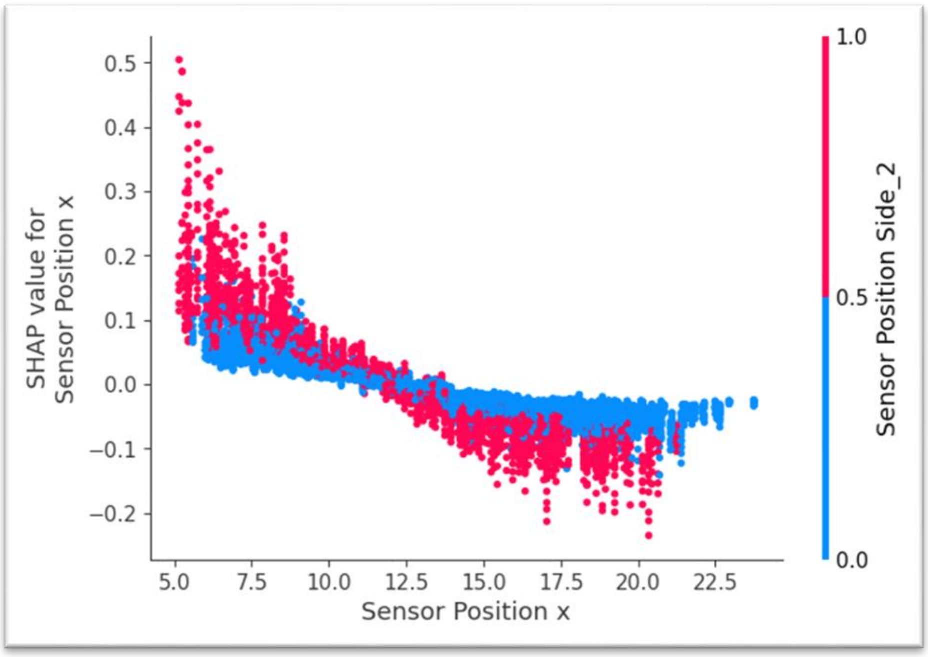
Similar to Side_1, Side_2 shows minimal variation in SHAP values. The impact of the sensor position on Side_2 is consistently low (close to zero), with occasional increases or decreases. This consistent pattern implies that the sensor's side does not play a significant role in affecting the model's pressure prediction but combining it with other factors like the position x can allow changes in predicted values.

Vapour Height with Sensor Position side 2



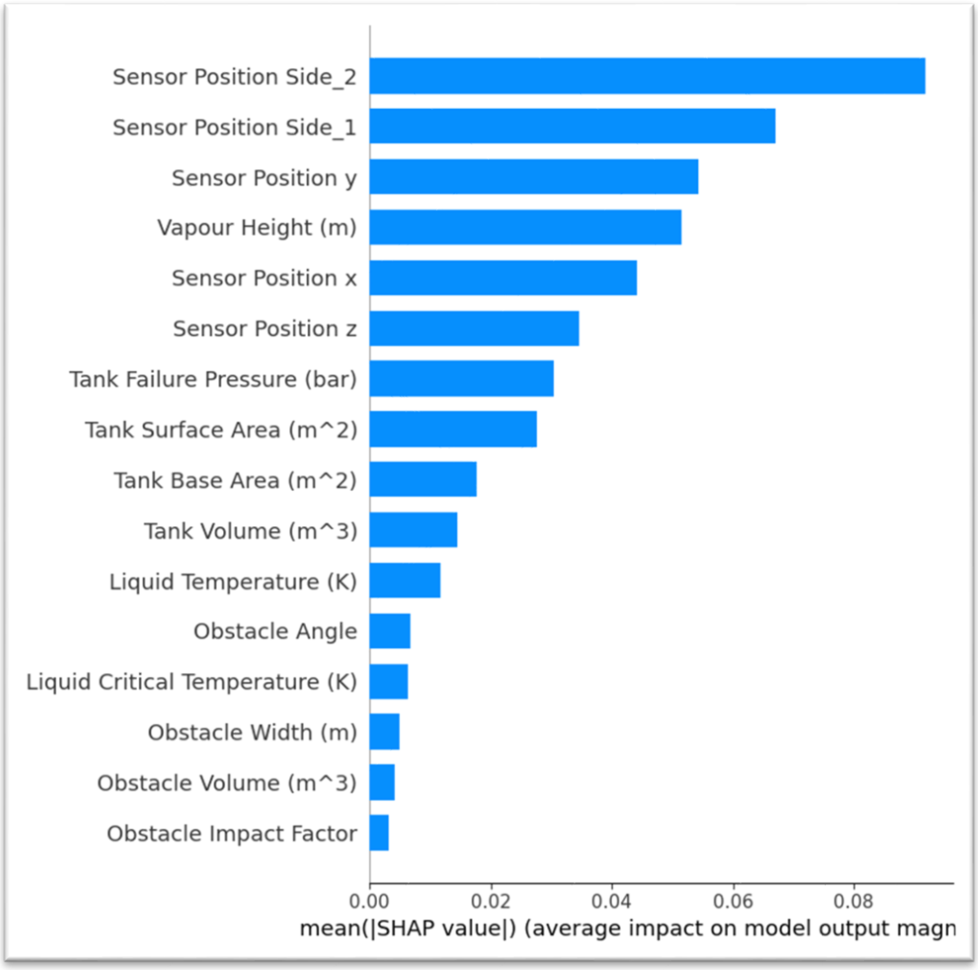
The SHAP values for Vapour Height show a pattern where certain discrete heights have a consistent impact on the model’s predictions. Heights at the lower end (0.25 m) and higher end (2.0 m) appear to have more positive impacts, suggesting these conditions might be associated with stronger or more hazardous pressure levels. The intermediate heights show variable but generally less impactful effects, which may be due to the different behaviour of vapor at these levels affecting the explosion's outcome.

Sensor Position x with Sensor Position side 2



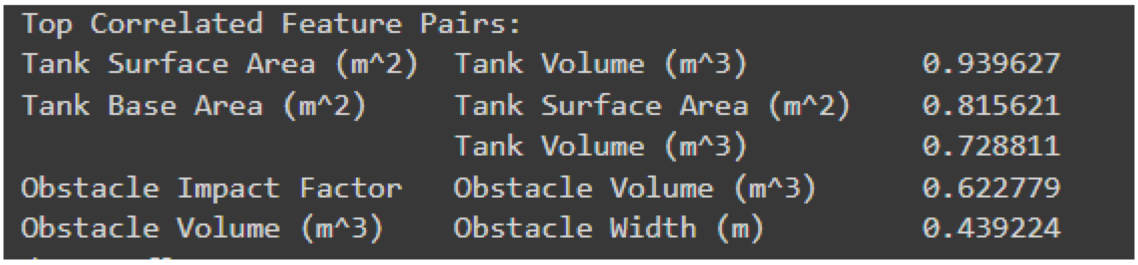
This plot reveals a downward trend in SHAP values as the sensor position x increases. When sensors are closer to the origin (lower x values), they tend to have a higher, positive impact on the predicted pressure, likely indicating proximity to the explosion source. As x increases, indicating distance from the source, the influence on the pressure prediction decreases, potentially due to geometric structure of the blast wave.

4.1.3) Feature Importance Plot



Majority of details in this graph are similar to the feature effect plot. On the x axis it shows the mean absolute values of SHAP which measures the average impact of each feature. On the Y axis there is the list of features that are shown ascendingly according to importance. The positioning of sensors on Side_2 significantly affects the model's pressure predictions, suggesting that this side of the sensor array captures critical dynamics of the blast wave. lso highly influential, this feature's positioning vertically influences the predictions significantly. Changes in the y-coordinate of sensors are likely associated with how the sensors capture the pressure changes. Vapour Height: The height of the vapour column in the tank also plays a crucial role in determining the predicted pressures. This suggests that vapour dynamics, possibly including the volume and characteristics of the vapour, are critical to understanding the intensity and spread of the blast. Tank-related Features (Tank Failure Pressure, Tank Surface Area, Tank Base Area, Tank Volume): These features, while having some impact, are less influential. They provide a context for the tank’s structural integrity and physical characteristics but are secondary to the direct sensor readings.

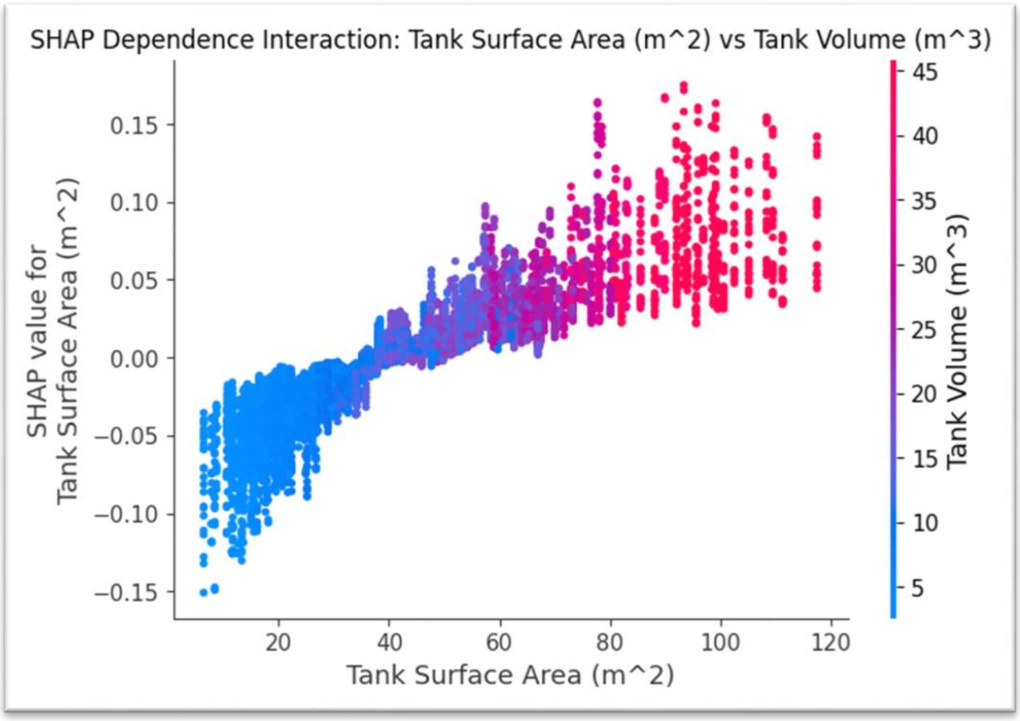
4.1.4) Feature Interaction Plots



```
Top Correlated Feature Pairs:
Tank Surface Area (m^2)  Tank Volume (m^3)        0.939627
Tank Base Area (m^2)     Tank Surface Area (m^2)  0.815621
                         Tank Volume (m^3)        0.728811
Obstacle Impact Factor   Obstacle Volume (m^3)    0.622779
Obstacle Volume (m^3)    Obstacle Width (m)       0.439224
```

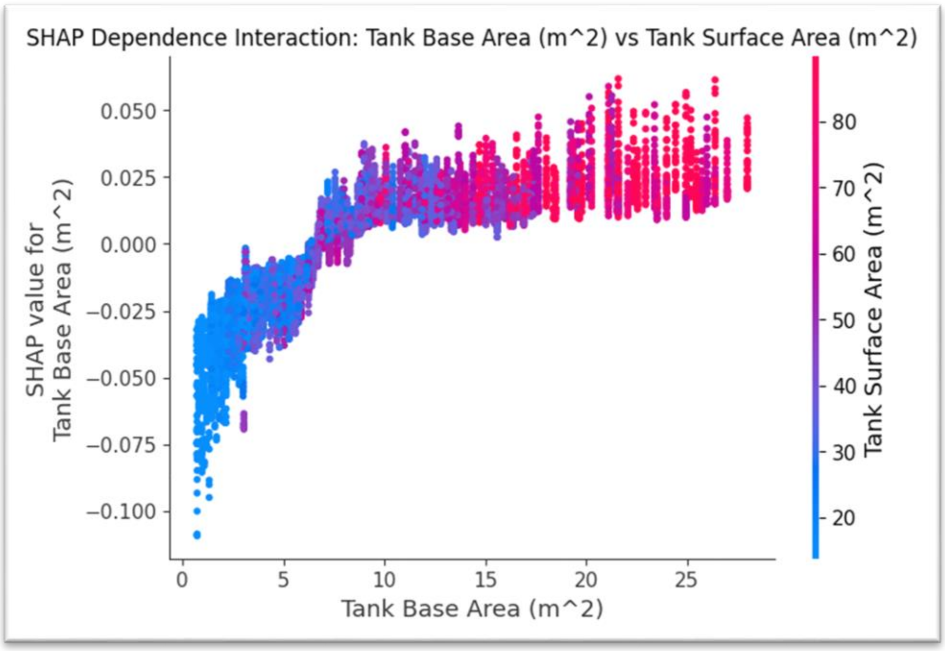
Based on the requirements, above shown screenshot shows the top most correlated features and their respective correlation at the front.

Tank Surface Area with Tank Volume



General Trend: As the tank surface area increases, the SHAP values increase from negative to positive, suggesting that larger tanks tend to increase the predicted pressure impact of a BLEVE. Volume Impact: The color gradient shows that for smaller tank volumes (blue), the SHAP values are generally lower (more negative). As the tank volume increases (moving towards red), the SHAP values also increase, becoming more positive. Critical Insight: Larger tanks with greater surface area and volume contribute to higher predicted pressures, possibly due to the larger quantity of materials capable of contributing to the explosion, and the greater surface area affecting the dynamics of heat and pressure release.

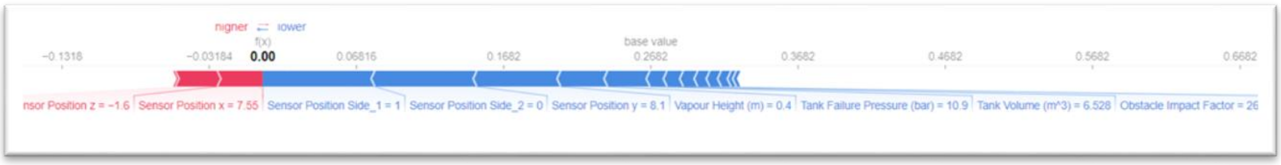
Tank base Area with Surface Area



General Trend: As the tank base area increases, SHAP values show a positive slope transitioning from strongly negative to mildly positive, suggesting that larger bases are associated with higher predicted pressures. Surface Area Impact: Initially, for smaller tank bases (left side of the plot), SHAP values are negative across various surface areas, with lower values for smaller surface areas. As the base area increases, the impact on the model's prediction turns less negative and eventually positive, particularly noticeable in tanks with larger surface areas (represented by red dots). Critical Insight: This interaction indicates that tanks with larger bases and surface areas have a more significant impact on predicted pressures. This could be due to increased stability and capacity of the tank, influencing how the tank fails and releases pressure in the event of a BLEVE.

4.2) Local Interpretation

4.2.1) SHAP Force Plot for Lowest Prediction



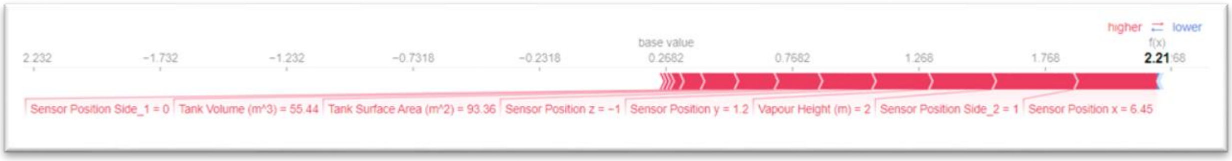
Instance ID : 8133

Prediction value: 0.00

Base value: 0.2682

Feature Contributions: Most of the features (represented by blue bars) are pushing the prediction lower from the base value. Notably, "Sensor Position Side_1 and 2 are pushing the results down. There are other important features like sensor position y and vapour height as well that are lowering the overall prediction from the base. On the other hand just two are pushing it forward significantly which include "Sensor Position Z = -1.6" and sensor position x = 7.55.

4.2.2) SHAP Force Plot for Highest Prediction



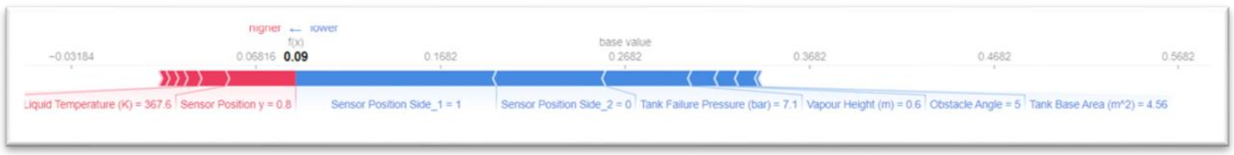
Instance ID: 10110

Prediction value: 2.21

Base value: 0.2682

Feature Contributions: In contrast to the first plot, most features here (red bars) push the prediction higher than the base value. Significant contributors include "Tank Volume (m^3) = 55.44", "Tank Surface Area (m^2) = 93.36", and "Sensor Position Y = 1.2". All other like vapor height, position side 2 and position x of the sensors are also contributing even higher to bring the predication which has caused it to be the highest. This indicates a higher risk or event likelihood according to the model's learning

4.2.3) SHAP Force Plot for Largest Error



Instance ID: 9913

Prediction value: 0.09

Base value: 0.2682

Feature Contributions: This plot shows a mix of features contributing in both directions. The predication going off from the base value is majority due to the sensor position 1 pushing it back significantly along with position 2. Whereas the position y and liquid temperature (k) fighting back to keep it high.

Analysing the SHAP force plots collectively provides several critical insights into the behaviour of the predictive model and the underlying dynamics of blast pressure generated by BLEVEs in a controlled setting. Sensor positioning, both vertically and horizontally, significantly affects the detected pressures, indicating the importance of strategic sensor placement for accurate monitoring and risk assessment. Larger tank volumes and surface areas are strongly associated with higher pressures, suggesting that the physical characteristics of the storage container play a critical role in the dynamics of blast propagation. Additionally, the prediction errors highlighted by the SHAP plot for the largest error emphasize the need for robust modelling that accurately captures the complexities of lower tank pressures and vapour heights, crucial for improving safety

5) Self-Reflection

5.1) Understanding about the data and model:

At the start of the project, the description of what the project is about was very daunting. Since I have no background to physics or even anything close to science, the information made little or no sense. It was when I did a background research of what BLEVE scenario really are, did the data start making sense. I went t through every feature in detail to understand what it represents and also how is it linked to what we are trying to predict. In my exploratory data analysis (EDA), I utilized multiple techniques to visualize and summarize what the data was representing. By using histograms, box plots, and other statistical tools, I was able to understand deeper insights of how each feature is connected to the others. Techniques like correlation and Lasso analysis, allowed me to see which features are having greater impact on the target variable and also the ones that need lesser focus.

Once the data was fully understood, processed and features were selected, I moved to the model development phase. I did some research on the internet plus used some of the techniques taught in the practical sessions to understand which models would be highly applicable to this context. I tried to start with the easy ones including linear regression and decision trees. Then moved on to the ones that I was using for the first time, for example Neural Networks and Catboost. I was able to understand that every model has their own way of predicting values. Each has their underlying assumptions, strengths and weaknesses and they should be comprehended before carrying out a complete implementation. Every model has its own hyperparameters and it should be played with to get better results. I balance should be achieved in how much every model requires in terms of the computational needs and the run time it asks for. Comparison between models will also be applicable only if they are being assessed given the same data with the same performance metrics.

Overall, my understanding of the data and modelling techniques deepened with the time I spent on it. There were errors are rectifications that allowed me to understand the internals of how the model is running and producing results.

5.2) Lessons and challenges:

The project itself was a long ride that had multiple challenges however, each of them paved way to a lesson learnt. In the early stage, I learnt the critical importance of having a good quality data. There were categorical variables present, some outliers and also some features that were adding no value to the model. These needed to be understood and treated before moving forward to the next stages. For example, there were challenges with model not performing well due to the existence of outliers within multiple features. By normalizing the target pressure bar and winsorizing the vapour height, the effect on the results was highly improved. I learnt how the finest details can actually be a turning point in order to achieve accurate predictions.

Feature Engineering was another important learning within the project. I firstly attempted to go on with only a new features that I found interesting to impact results. However, the more thought I gave to features, the more connections I was able to find which can potentially lead to further improvements in outcomes. A significant breakthrough came with Tank Dimension Ratios, volume and scaled pressure. These engineer features were able to capture insightful relationship between dependent and independent variables allowing techniques like lasso and correlation to capture their powers. Hence, the lesson was to always contemplate deeply on the data provided since it may have hidden gems within it which ones brought out can allow achievements in terms of precise predictions.

The project also did present some technical and computational challenges. Training complex models like neural networks and catboost, or using Grid-search for a wider parameter range required significant computational power and memory. The process was at times very slow and even had multiple crashes. However, I learnt a way around this by using GPU acceleration to speed up the process. It allowed to do multiple iterations to see if the models improve if changes were made.

Another mental challenge I faced during the project was the pressure I felt by seeing the results on the kaggle leader board. Even though it was specifically mentioned that it is not a competition, having to upload your best shot and still ranking low always gave the sense of doing something wrong which others might be doing right. The competition at the same time did give a lot of motivation to go back, make changes, improve and then try again. Lesson learnt was that it’s always good to have a healthy competition between your peers. I personally might have submitted my results of MAPE 50 thinking its good enough; however, seeing fellow students putting effort and improving everyday allowed me to work on my models until I was able to get as close to the top 10 as possible.

5.3) Reflection on Model Interpretation:

Once the model was selected there were two ways the model was interpreted, Global and Local. In Global interpretation techniques, multiple visualizations were used to understand an overview of how the model works across the entire dataset. It allowed us to understand which variables had the most significant impact on predicting the peak pressures. For instance the feature importance analysis revealed that the tank volume, height and scaled pressure were consistently among the features effecting results. Another useful way I found was to create partial dependency plots while considering two variables at the same time. This allowed to understand the relationship with each other at the same time how they are impacting the target pressure bar.

On the other hand Local interpretation, focuses on infidel instances of the prediction results. Understanding the zoomed version of how prediction is being done at individual level helped better understand the process of how the results are being generated. Applying techniques like SHAP allowed to create visualizations that broke down the impact each variable was having on a particular instance. This level of granularity was of immense significance to understand how the chosen model was reasoning and identifying potential biases and inconsistencies while making predictions.

Having get done with model interpretation in such detail played a vital role in enhancing the transparency and accountability of the Catboost. It not only gave deep understanding of how the particular model is working right now but also paved way towards the thought of how this can be used to improve it further. For example features that were consistently showing more impact through both global and local interpretation can be given more importance when making changes with the aim of making the model more robust.

5.4) Future Directions:

If there is an opportunity to work on this project further, there are many things that can be done to experiment getting better prediction results. Starting from the data itself, there is always a possibility of including more scenarios for BLEVE that would allow making models much more robust since they will have more instances to support a particular effect of the explosion. Apart from that the data can also be further analysed to engineer more features that can find even deeper insights between the feature currently included.

For the models there are also multiple changes we can do. This can include to test even more machine learning algorithms that are available in the market. This may mean to research deeper and experiment with models apart from the 6 already used.

The already created models can also be worked on to find if they can perform better than they currently are. This may require to find more hyperparameter or defining larger ranges for cross validation searches so that the best parameters can be sought. This may mean the models asking for more computational power and run time however, a deeper dive can help recognize the true potential of the models that are already used.

Interms of interpretability, I would certainly like to look further in Local interpretation specifically to understand how changing values effect the prediction. As of now according to the requirement of the assignment, I looked into high the features and impacting individual instances, however knowing how changing each feature while keeping other constant, impact the prediction results would be a great way to delve into the intricacies of selected model

5.5) Concluding Thoughts:

In conclusion, this project has been a bitter sweet journey that had plenty of challenges but allowed myself to learn and reflect deeply of how machine learning actually works. The project taught me that a problem may be of a very different nature that I may hardly have any information about, however interest, commitment and knowledge about machine learning can allow predictions to be made that can solve issues of the real world. This project, that seemed to be something impossible to do at first, has left me with valuable skills that can be used as a career pathway and also to engage with everyday problems. As I reflect on the accomplishments and lessons of this project, I am left inspired to continue exploring into real world issues that can be solved with machine learning capabilities to create a positive impact on the society we live in. I believe the assignment did complete justice in teaching detailed knowledge about how machine learning works and how it can be interpreteded to understand what it actually means. My suggestion to improve the assignment would be to give multiple project topics rather than just one. This would allow students to pick from what they are most interested in. This will help them to put in more effort as they will be able to link it with their prior interest and knowledge.

REFERENCES

Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., & Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research*, *2*(3), 87-93.

Li, Q., Wang, Y., Li, L., Hao, H., Wang, R., & Li, J. (2023). Prediction of BLEVE loads on structures using machine learning and CFD. *Process Safety and Environmental Protection*, *171*, 914-925.

Molnar, C., Casalicchio, G., & Bischl, B. (2020). Interpretable machine learning–a brief history, state-of-the-art and challenges. Joint European conference on machine learning and knowledge discovery in databases,

Slack, D., Friedler, S. A., Scheidegger, C., & Roy, C. D. (2019). Assessing the local interpretability of machine learning models. *arXiv preprint arXiv:1902.03501*.

Vellido, A., Martín-Guerrero, J. D., & Lisboa, P. J. (2012). Making machine learning models interpretable. ESANN,