COMP6013 Explinable Approaches to Machine Learning

Predicting Boiling Liquid Expanding Explosions (BLEVES) using Machine Learning Methods

Final Assessment

S1 2025 Computing, Curtin University

September 23, 2025

Contents

1 Assignment Pipeline

1.1 Introduction

Boiling Liquid Expanding Vapour Explosions (BLEVEs) are hazardous and extremely dangerous events that frequently occur during the transportation of Liquefied Petroleum Gas (LPG) [1]. When the liquid inside a tank exceeds its boiling temperature, the resulting pressure causes the tank to rupture, leading to an explosion [1][2]. Numerous human fatalities and significant destruction to infrastructure in the vicinity have been reported due to such explosions. Several studies have been conducted to analyse, monitor, and predict the occurrence of BLEVEs.

Recent advances in machine learning techniques have enabled the development of predictive models to analyse when and how such explosions might occur, as well as the factors contributing to them. Accordingly, this assignment performs a predictive analysis of the peak pressure generated by BLEVEs. An obstacle equipped with 27 sensors was used to detect the pressure from various perspectives, as illustrated in the Figure 1. The data was cleaned, processed, and fitted to various machine learning algorithms, ranging from simple linear regression to advanced XGBoost methods. Among all the models, the **CatBoost Regressor**, after hyperparameter tuning, performed the best, achieving a mean absolute percentage error of **0.12** on the validation set and **0.21523** on the testing set in Kaggle.

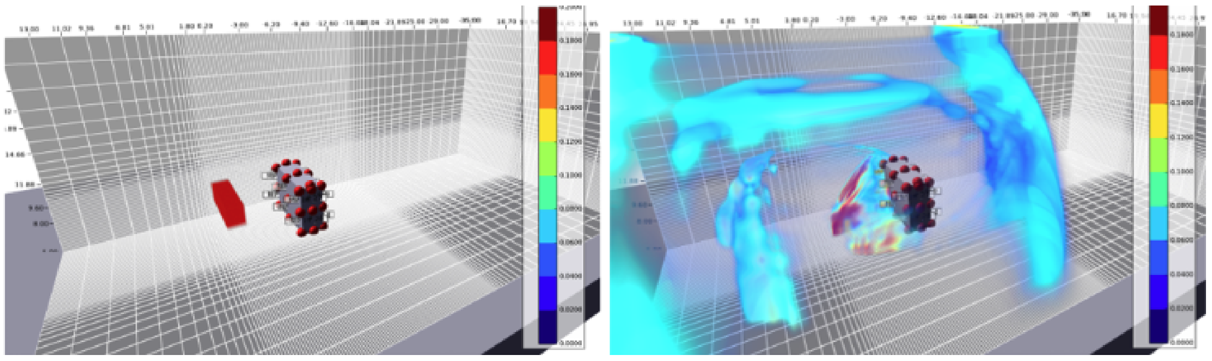


Figure 1: BLEVE blast with sensors fitted for data collection

1.2 Data Split

The data was split into following sets with ratio of (80% for training set, 20% validation set and separate test provided) as follows:

1. Training set - for training the data
2. Validation set - for evaluating the model
3. Test set - the separate test data for the final evaluation of the performance of the model on Kaggle.

Table 1 shows the distribution of data into training, validation, and test set.

1.3 Data Preprocessing

Table 2 shows the number of null values each feature contained. Initially, the null values were replaced with **average value**, but after fitting the model, there was not a significant

Dataset	**# of Data**
Training Data	7912
Validation Data	1978
Testing Data	3203

Table 1: Data distribution after feature selection.

improvement. Instead of replacing with average value, all the null values were removed from the dataset all together.

Feature	**Null Count**
Liquid Ratio	9
Tank Width (m)	7
Tank Length (m)	5
Tank Height (m)	8
BLEVE Height (m)	10
Vapour Height (m)	9
Vapour Temperature (K)	28
Liquid Temperature (K)	28
Obstacle Distance to BLEVE (m)	8
Obstacle Width (m)	8
Obstacle Height (m)	8
Obstacle Thickness (m)	8
Obstacle Angle	8
Status	8
Liquid Critical Pressure (bar)	28
Liquid Boiling Temperature (K)	27
Liquid Critical Temperature (K)	29
Sensor ID	8
Sensor Position Side	10
Sensor Position x	7
Sensor Position y	9
Sensor Position z	7
Target Pressure (bar)	6

Table 2: Number of null values for each feature before cleaning

Most of the features were highly skewed and had higher variance. For example, for the *Tank Failure Pressure (bar)*, the mean was **37.45** and the maximum was **4882.57**. This suggests higher variability and the presence of outliers. Instead of removing the outliers, outlier detection using interquartile range (IQR) was used. An upper bound was defined as the 75th percentile $+ 1.5\times$ interquartile range, whereas the **lower bound** was defined as the 25th percentile $- 1.5\times$ interquartile range [3] (refer to Appendix 3.1). Any values beyond the upper bound and lower bound were replaced with the upper bound and lower bound respectively. This is to ensure that outliers do not affect the model's performance.

The **Status** feature should contain only two values: **Superheated** and **Subcooled**, but the raw data contained many variations:*["Superheated", "Superheat", "Subcooled", "subcooled", "Saperheated", "Subcool", "Subcoled", "superheated"].* All of these categories were changed to either *superheated* or *subcooled* to denote the two correct categories. The categories were then encoded using the **one-hot** encoding method. Similarly, the feature **Sensor Position Side** was one-hot encoded.

Linear regression assumes that target output is normal distribution. The response variable (target pressure) is not normally distributed and skewed. To address this, the response variable is log transformed using `log1p`. Figure 2 shows the data distribution and after `log1p` transformation. This transformation effectively reduces skewness and produces a more symmetric distribution. The `log1p` is particularly suitable as it accommodates target pressure of 0 without undefined error (e.g., log(0)). Furthermore, monotonic log transformation preserves the relationship between the response variable and features.

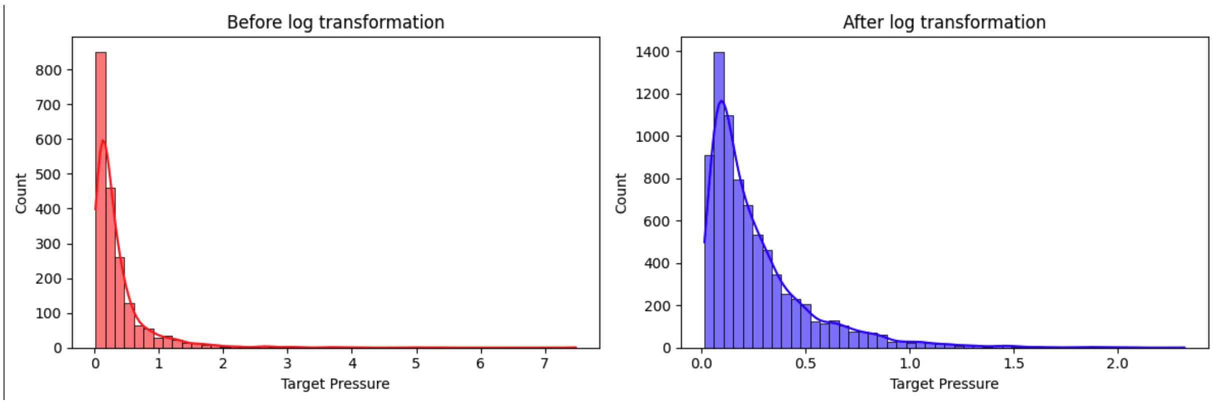
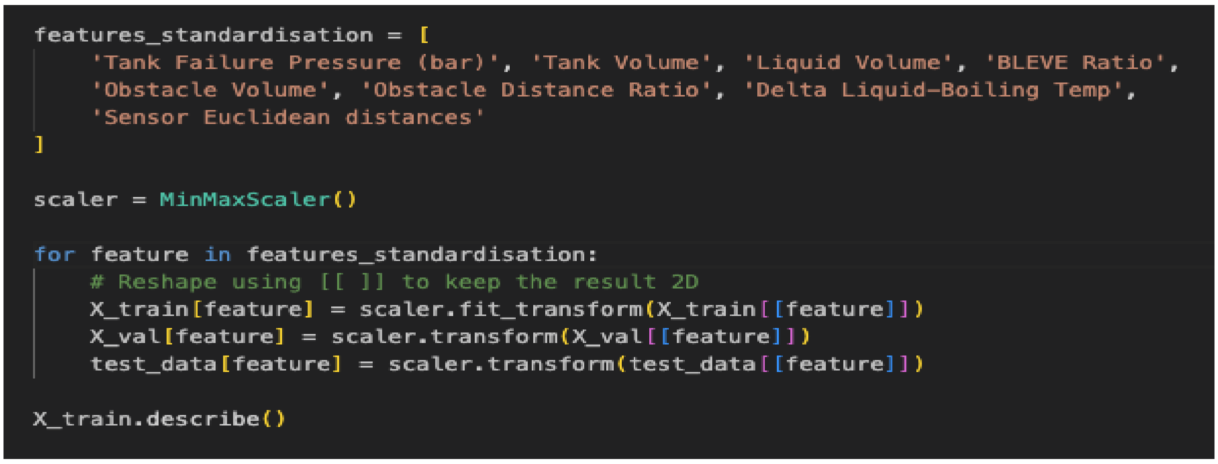


Figure 2: Log1p transformation

Not all the features were normalised, but only selected features, particularly ones that are highly skewed and not on a similar scale. *MinMaxScaler()* was used to scale the features, as standardisation assumes that the data is normally distributed. Feature scaling ensures that one feature with higher values does not dominate over other features and increase bias in the model's predictions. Figure 3 shows the features and normalisation method used.



```
features_standardisation = [
    'Tank Failure Pressure (bar)', 'Tank Volume', 'Liquid Volume', 'BLEVE Ratio',
    'Obstacle Volume', 'Obstacle Distance Ratio', 'Delta Liquid-Boiling Temp',
    'Sensor Euclidean distances'
]

scaler = MinMaxScaler()

for feature in features_standardisation:
    # Reshape using [[ ]] to keep the result 2D
    X_train[feature] = scaler.fit_transform(X_train[[feature]])
    X_val[feature] = scaler.transform(X_val[[feature]])
    test_data[feature] = scaler.transform(test_data[[feature]])

X_train.describe()
```

Figure 3: Normalisation of the features

Lasso regression was used for selecting the features from total of 31 features after feature engineering described in 1.4. Lasso does by reducing the coefficients of irrelevant features to zero. Figure 4 shows coefficient values for all selected features.

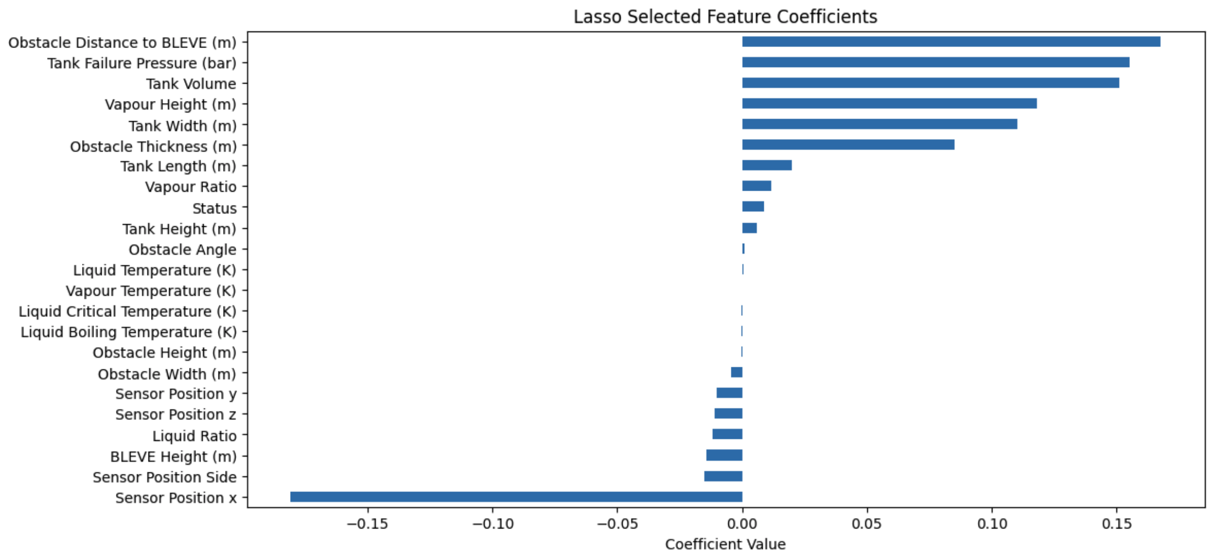


Figure 4: Lasso regression for feature selection

1.4 Feature Engineering

A total of 9 new features were created (augmented) using the available features. The features augmented and their descriptions are as follow:

1. **Tank Volume:** Calculated as length × width × height. It describes the volume of the container in which the liquid is stored.

2. **Liquid Volume:** Liquid ratio × tank volume. It is amount of liquid stored in the container.

3. **BLEVE Ratio:** BLEVE height divided by tank height to give a relative measure of BLEVE to tank height. A larger BLEVE ratio might indicate large explosion

4. **Obstacle Volume:** Width × height × thickness of the obstacle. If the obstacle volume is larger, the target pressure the obstacle detect might be higher.

5. **Vapour Ratio:** Calculated as Vapour height relative to the tank height.

6. **Obstacle Distance Ratio:** Relative distance from the BLEVE.

7. **Delta Liquid-Boiling Temp:** Difference between liquid temperature and its boiling temperature. It describe how close the liquid is to the boiling temperature.

8. **Critical Ratio:** Ratio of critical temperature to the liquid temperature. It describes how close is the liquid to the supercritical where it might behave like non-liquid.

9. **Sensor Euclidean Distance:** Distance based on sensor coordinates.

Lasso regression selected the following augmented features: **tank volume**, **vapour ratio** to be associated with target pressure 4. In addition, these two features were found to be affecting the model's output. As per the visualisation on the SHAP value, tank volume is the feature with the second strongest association with target pressure. As tank volume increases, so does the target pressure, as it is positively associated, and so is vapour ratio, although its impact is not as significant as the tank volume.

1.5 Model Selection

The following models were selected for analysis and evaluated for predicting the BLEVE target pressure:

1. **Linear Regression**: A baseline model for its simplicity and interpretability. It helps to understand the linear relationships between the response variable and predictor variables. It is the simplest model.

2. **Decision Tree Regressor**: The relationship between the target pressure and other factors is likely to be non-linear. The decision tree regressor is a simple model that can capture non-linear relationships and can be easily interpreted. A baseline model for non-linear relationships.

3. **Random Forest Regression**: It is an ensemble of decision trees and improves generalisability and reduces overfitting while still being simple and interpretable due to its tree structure. It also handles high-dimensional data better than a decision tree, making it suitable for BLEVE with 30 plus features.

4. **Support Vector Regressor (SVR)**: It is suitable for high-dimensional data and can capture both linear and non-linear relationships through kernel functions. A linear kernel captures linear relationships, a poly kernel captures non-linear relationships, and the radial basis function (RBF) can capture complex and non-linear relationships, thus making it extremely flexible and easier to use for BLEVE prediction.

5. **XGBoost**: A decision tree model is trained sequentially to minimise the loss function, allowing the model to learn from previous errors and improve accuracy over time. Consequently, it improves performance and efficiency for smaller datasets. It reduces overfitting and works well for tabular data, thus making it ideal for the BLEVE prediction task with a tabular dataset.

6. **CatBoost**: Another boosting method effective for datasets with categorical values. This model was ideal because most of the features were categorical despite being numerical, such as sensor position side, which had values from 1 to 5, and other features with only two binary values: liquid critical pressure (38 and 42), liquid boiling temperature (-40, 0), and liquid critical temperature (100, 150). As the data had a combination of both categorical and continuous values, it was ideal to use CatBoost.

1.6 Hyperparameter Tuning

All the models were fitted with and without hyperparameter tuning except linear regression. Two hyperparameter tuning methods were used depending on the number of hyperparameters. If the model contained many hyperparameters, then **RandomizedSearchCV** used instead **GridSearchCV**. For XGBoost and CatBoost, **RandomizedSearchCV** was used as there were many hyperparameters and training required significant computational resources. Table 3 shows hyper parameters tuned, search strategey, range, and optimal values found for each model

Model	**Hyperparameters**	**Search Strategy and Range**	**Optimal values**
Decision Tree	max_depth max_features min_samples_split min_samples_leaf	Grid Search 5-fold CV	max_depth: 20 max_features: None min_samples_leaf: 1 min_samples_split: 2
Random Forest	max_depth max_features n_estimators	Grid Search 5-fold CV	max_depth: 25 max_features: sqrt n_estimators: 55
Support Vector	C epsilon kernel	Grid Search 5-fold CV	kernel: rbf epsilon: 0.01 C: 10
XGBoost	learning_rate max_depth n_estimators subsample colsample_bytree gamma reg_alpha reg_lambda min_child_weight	RandomizedSearchCV 5-fold CV 100 iterations	subsample: 0.7 reg_lambda: 0.5 reg_alpha: 0.1 n_estimators: 400 min_child_weight: 5 max_depth: 9 learning_rate: 0.03 gamma: 0 colsample_bytree: 1.0
CatBoost	iterations learning_rate depth l2_leaf_reg bagging_temperature random_strength	RandomizedSearchCV 3-fold CV 20 iterations	iterations: 1000 learning_rate: 0.1 depth: 8 l2_leaf_reg: 9 bagging_temperature: 1 random_strength: 1

Table 3: Hyperparameter Tuning Summary for all the models

1.7 Model Comparison

All the models were evaluated on three metrics: R^2, **Mean Squared Error**, and **Mean Absolute Percentage Error**. R^2 indicates how well the model fits the data and how it explains the variability of the data. **MSE** is the average of the squared differences between predicted values and actual values. **MAPE** is the average percentage error relative to the actual values over all the observations/data. Table 4 and 5 shows the performance of each of this model pre and post hyperparameter tuning.

Model	$\mathbf{R^2}$	**MSE**	**MAPE**
Decision Tree	0.86	0.01	0.23
Random Forest	0.81	0.12	0.31
Support Vector (SVR)	0.19	0.05	0.86
XGBoost	0.96	0.00	0.17
CatBoost	0.93	0.00	0.20

Table 4: Model Performance Before Hyperparameter Tuning

Model	$\mathbf{R^2}$	**MSE**	**MAPE**
Decision Tree	0.85	0.01	0.22
Random Forest	0.89	0.01	0.25
Support Vector (SVR)	0.39	0.04	0.50
XGBoost	0.97	0.001	0.13
CatBoost	0.98	0.00	0.12

Table 5: Model Performance After Hyperparameter Tuning

Abbreviations:

- $\mathbf{R^2}$: Coefficient of Determination
- **MSE**: Mean Squared Error
- **MAPE**: Mean Absolute Percentage Error

Before hyperparameter tuning, **XGBoost** performed the best on the validation set with R^2=**0.96**, **MSE=0.00**, and **MAPE=0.17**. MSE suggests the model is likely overfitting.

After hyperparemter tuning, **CatBoost** regressor performed the best on the validation set with R^2=**0.98**, **MSE=0.00**, and **MAPE=0.12**.

Both these models were compared on the testing dataset in Kaggle. **CatBoost** outperformed **XGBoost** with **MAPE=0.215** to **MAPE=0.241**.

1.8 Model Interpretation

CatBoost regressor was selected for model interpretation as it performed the best on the validation set with **MAPE=0.12** despite the model likely to be overfitting over the models.

1.8.1 Global Interpretation

Shapley value is used to analyse how each feature contribute to the model's output. Figure 5 shows the SHAP value for each feature and how it impact's model output.

Of all the features, the top three that have the highest impact on the model output are **Sensor Position Side**, **Tank Volume**, and **Sensor Position Y**. In general, features related to sensor positions, including Sensor Position X, Y, and Z, tend to have the most significant influence on the model's output. For Sensor Position Side, the impact varies: extreme values (either very high or very low) tend to increase the model output, while values within a certain intermediate range have less impact. Tank Volume positively correlates with the output—higher values increase the predicted output, and lower values decrease it. In contrast, higher values of Sensor Position Y decrease the model's output, and lower values increase it.

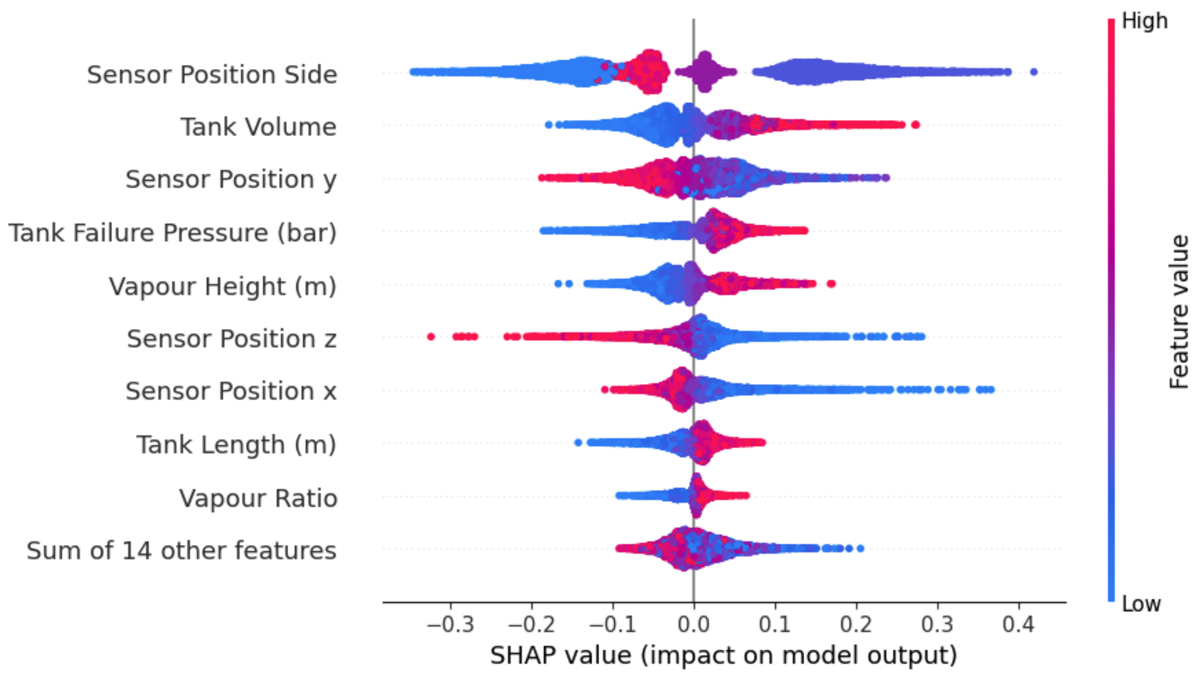


Figure 5: Shapley values for each feature

Figure 6 visualisation of importance of features.

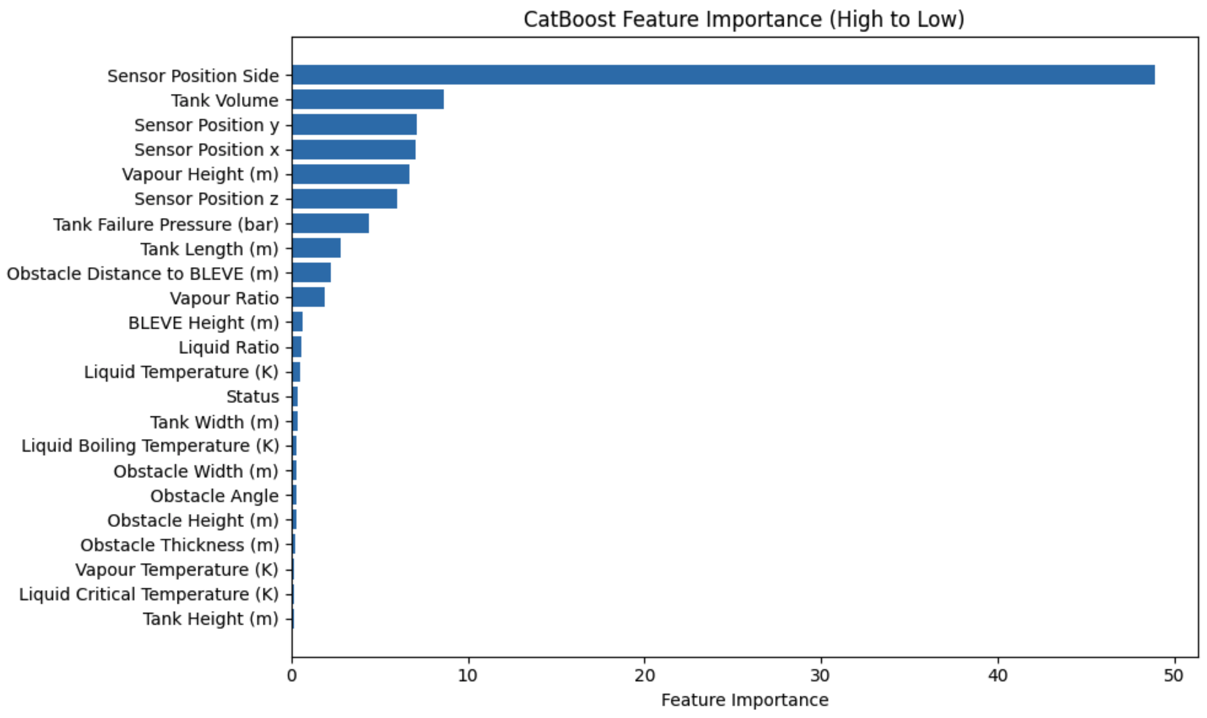


Figure 6: Feature Importance plot

A Partial Dependence Plot (PDP) is used for modelling the marginal effect of each feature on the predicted outcome. Instead of consider all the features, only the top five features are considered which are as follows:

- Sensor Position Side
- Tank Volume

- Sensor Position Y
- Sensor Position X
- Vapour Height (m)

Figure 7 shows the marginal effects of the top five features for CatBoost mode's prediction. As illustrated in the figure, effects of each feature are:

1. **Sensor Position Side**: As the position of the sensor changes, there is massive fluctuation in the target pressure.
2. **Tank Volume**: As the tank volume increases, so does the target pressure.
3. **Sensor Position Y**: When the sensor position value is less than 0, target pressure seems to increase; however, for values greater than 0, the target pressure decreases.
4. **Sensor Position X**: The target pressure decreases as the sensor position X increases.
5. **Vapour Height**: As vapour height increases, so does the target pressure up to 1.5; beyond that, the target pressure remains almost the same.

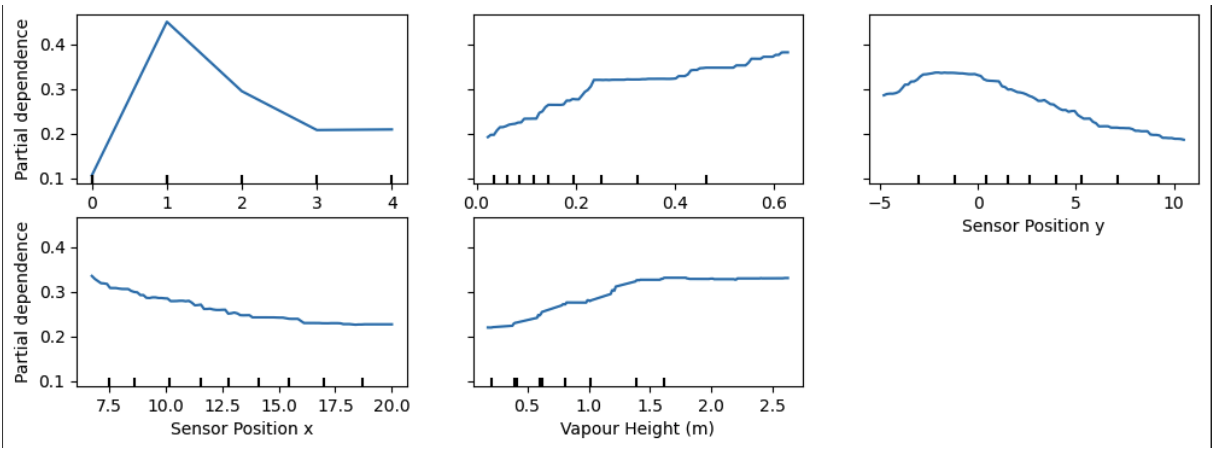


Figure 7: Partial Dependency Plot for top 5 features

1.8.2 Local Interpretation

Three instances of CatBoost model's prediction were selected for local interpretability namely; (1) **lowest predicted target pressure**, (2)**highest predicted target pressure** and (3) **largest error**. The lowest predicted target pressure is **0.01**. Figure 8 shows the features that contribute to the lowest predicted target pressure. As shown in the figure, **Sensor Position Side**, **Vapour Height**, **Sensor Position Y**, and **Tank Volume** are all pushing the target pressure below the baseline where as **Sensor Position X** and **Sensor Position Z** is pushing the values above baseline of $\tilde{0}$**.253**.

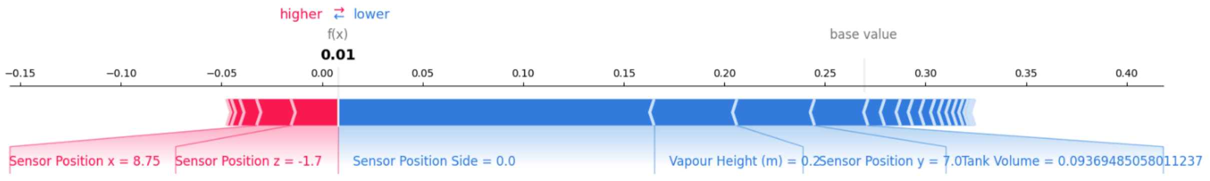


Figure 8: Features Contributing to Lowest Predictions

Figure 9 shows the features responsible for the highest predicted target pressure. There is no noticeable feature that is negatively associated for that instance, while features **Tank Length**, **Tank Volume**, **Vapour Height**, **Sensor Position Y**, **Sensor Position Z**, **Sensor Position X**, and **Sensor Position Side** push the predicted outcome above the baseline

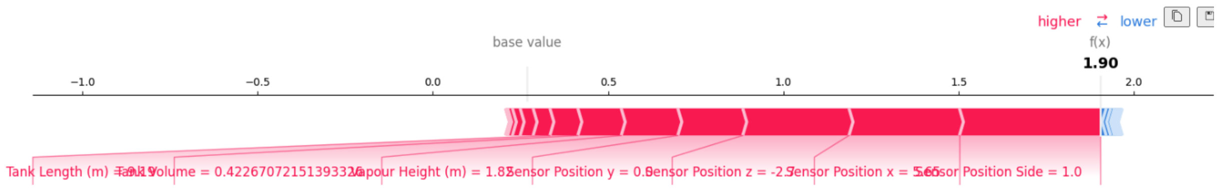


Figure 9: Largest Predicted Target Pressure Instance

Figure 10 shows the largest error predicted by CatBoost Model. **Sensor Position x** and **Sensor Position z** pushes the prediction above the baseline while **Sensor Position Side**,**Vapour Height (m)**, **Sensor Position y** and **Tank Volume** pushes the volume below the baseline.

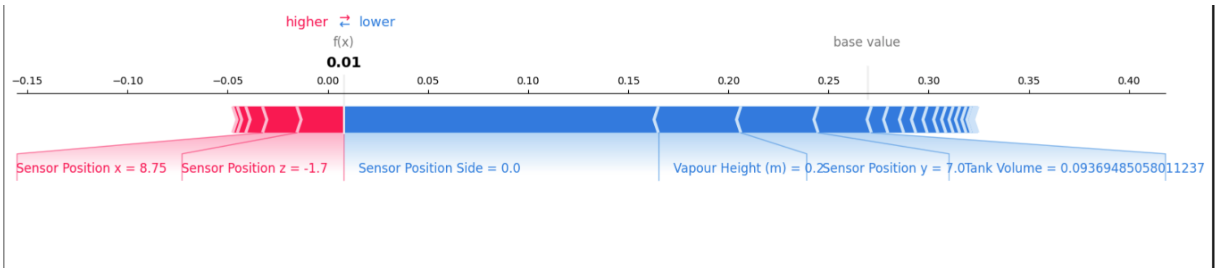


Figure 10: Largest Error Instance

The model's largest error is **0.01**, which indicates the model is overfitting for the training dataset and validation set. This is further evidenced by lower performance on the testing data in Kaggle. Neural network or transformer-based models are likely to perform better than the models discussed in the report.

2 Extended Discussion

2.1 Question 1

Prior to this course, my understanding of interpretability was limited to explaining the outcome of a model. For example, in a linear regression model with the equation $y = 5 + 2x_1 + 5x_2$, I interpreted the model as indicating that `Y` increases by 2 units for every unit increase in X_1 and by 5 units for every unit increase in X_2. However, I had not questioned why the coefficient for X_1 was 2 and for X_2 was 5. Through this course, I have come to understand that interpretability extends beyond reading coefficients and involves exploring how each feature influences the response variable. Tools like SHAP values have helped me appreciate how individual features contribute to a model's predictions, and I have also learned how local interpretability techniques can reveal the reasoning behind a model's decision in a specific instance.

One of the most insightful finding I gained is that interpretability is not a fixed property. Instead, it must be tailored to the needs of different users or audiences. While interpretability is generally defined as the extent to which a human can understand a machine learning model's decision-making process, it must also comply with domain-specific constraints to be truly meaningful [4]. The requirements for interpretability vary depending on the problem context, domain, and target users. For instance, in clinical applications, the model must be interpretable to healthcare professionals such as doctors and clinicians, even if it is not understandable to the general public. Furthermore, I now recognise the critical importance of interpretability in addressing the black-box nature of many machine learning models. It ensures that ML systems are safe, fair, transparent and accountable to use.

2.2 Question 2

Target pressure is defined as the maximum pressure experienced at the centre of an explosion or at a designated reference point, such as an obstacle, as used in the assignment. While target pressure provides valuable insight into the severity of a blast, it does not capture the full scope of the explosion's effects. Specifically, it overlooks the total energy released, the presence of flames in case of combustion, and the thermal radiation emitted. For example, Laboureur et al. [5] examined blast characteristics, focusing on blast wave generation and propagation instead of focusing on target pressure.

BLEVEs are extremely hazardous to human life and can inflict severe damage to infrastructure. Therefore, rather than predicting target pressure at a specific point, estimating the destruction radius or the extent of blast wave propagation may provide more practical and safety-relevant information. By analysing the destruction radius, it becomes possible to determine appropriate safety distances for storing tanks carrying Liquefied Petroleum Gas (LPG), both during transportation and at rest.

For the research, experiments could be conducted at multiple scales. These experiments would record variables such as the mass of the liquid, tank volume, total energy released, and distances at which destructive pressure levels are observed. A potential research question is what are factors that most significantly influence the blast destruction radius, and how can interpretable machine learning models predict it?

2.3 Question 3

The topics I think that would have been helpful and needed further investigation are:

1. **Advanced data processing technique and pipelines**: The course provides excellent coverage of machine learning algorithms and the architecture of various models. However, I would have appreciated a deeper exploration of advanced data preprocessing pipelines and how they are implemented in real-world applications. One of the slides mentioned that, in practice, nearly 70% of the effort in building ML systems is spent on data preprocessing. More practical activities on these these pipelines would have enhanced our understanding of how it works in real world practice and implemented. For instance, I understood more on how important data distribution, data preprocessing, and data processing methods are for building models during assignment.

2. **Application of Transformers in Large Language Models**: LLMs have gained significant attention in recent years and are used across a wide range of applications. While the course touched upon transformers, understanding application of transformers in LLMs would have enhanced my understanding of the models.More importantly, how do we define and evaluate interpretability for such large models, both globally and locally? Understanding this would help users assess the safety, reliability, and correctness of outputs generated by LLMs.

References

[1] Q. Li, Y. Wang, L. Li, H. Hao, R. Wang, and J. Li, "Prediction of bleve loads on structures using machine learning and cfd," *Process Safety and Environmental Protection*, vol. 171, pp. 914–925, 2023.

[2] T. Abbasi and S. Abbasi, "The boiling liquid expanding vapour explosion (bleve): Mechanism, consequence assessment, management," *Journal of Hazardous Materials*, vol. 141, no. 3, pp. 489–519, 2007.

[3] C. S. K. Dash, A. K. Behera, S. Dehuri, and A. Ghosh, "An outliers detection and elimination framework in classification task of data mining," *Decision Analytics Journal*, vol. 6, p. 100164, 2023.

[4] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistic Surveys*, vol. 16, pp. 1–85, 2022.

[5] D. Laboureur, A. Birk, J. Buchlin, P. Rambaud, L. Aprin, F. Heymes, and A. Osmont, "A closer look at bleve overpressure," *Process Safety and Environmental Protection*, vol. 95, pp. 159–171, 2015.

3 Appendix

3.1 Outlier Detection Method: Inter Quartile Range

```
#outlier look
#Use IQR - inter-quartile range to fill in the values for outliers
def replace_outliers(data):
    for col in data.columns:
        if col not in ['Target Pressure (bar)', 'Status']:
            q1 = data[col].quantile(0.25)
            q3 = data[col].quantile(0.75)
            iqr = q3 - q1  # Correct IQR calculation
            # print(f"IQR for {col}: {iqr}")

            #return output
            lower_bound = q1 - 1.5 * iqr
            upper_bound = q3 + 1.5 * iqr
            data[col] = data[col].apply(lambda x: q1 if x < lower_bound els
            return data

replace_outliers(train_data)
# data.describe()
```