Machine Learning

COMP 3010

Assignment Report

Name: Kalana Tharusha Withanage
Curtin ID: 20783462
Kaggle Name: Kalana Tharusha

Table of Contents

Data Cleaning

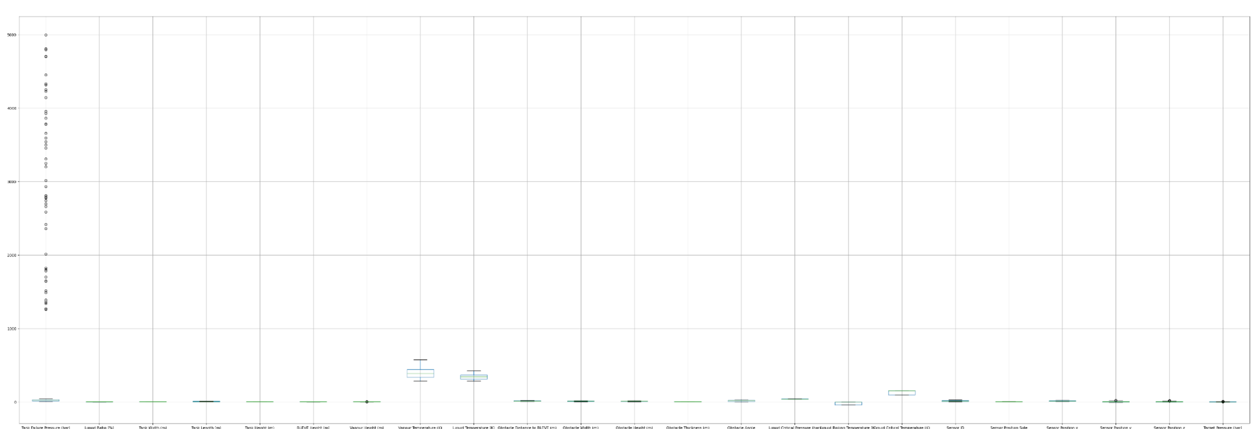
Data Issues Identified

There were a total of 10050 data instances in the training data set.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10050 entries, 0 to 10049
Data columns (total 25 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID                              10045 non-null  float64
 1   Tank Failure Pressure (bar)     10043 non-null  float64
 2   Liquid Ratio (%)                10041 non-null  float64
 3   Tank Width (m)                  10041 non-null  float64
 4   Tank Length (m)                 10041 non-null  float64
 5   Tank Height (m)                 10042 non-null  float64
 6   BLEVE Height (m)                10040 non-null  float64
 7   Vapour Height (m)               10041 non-null  float64
 8   Vapour Temperature (K)          10022 non-null  float64
 9   Liquid Temperature (K)          10023 non-null  float64
 10  Obstacle Distance to BLEVE (m)  10042 non-null  float64
 11  Obstacle Width (m)              10044 non-null  float64
 12  Obstacle Height (m)             10044 non-null  float64
 13  Obstacle Thickness (m)          10043 non-null  float64
 14  Obstacle Angle                  10042 non-null  float64
 15  Status                          10043 non-null  object
 16  Liquid Critical Pressure (bar)  10020 non-null  float64
 17  Liquid Boiling Temperature (K)  10021 non-null  float64
 18  Liquid Critical Temperature (K) 10020 non-null  float64
 19  Sensor ID                       10042 non-null  float64
 20  Sensor Position Side            10042 non-null  float64
 21  Sensor Position x               10043 non-null  float64
 22  Sensor Position y               10042 non-null  float64
 23  Sensor Position z               10041 non-null  float64
 24  Target Pressure (bar)           10040 non-null  float64
dtypes: float64(24), object(1)
memory usage: 1.9+ MB
```

Missing values: All the features had missing values. First, I tried a simple imputer with the strategy of mean value to fill in missing values. However, there was no significant effect on the model's score even if I dropped them instead of using the imputer. Since the total number of missing values was small, I decided to drop all the missing data instances.

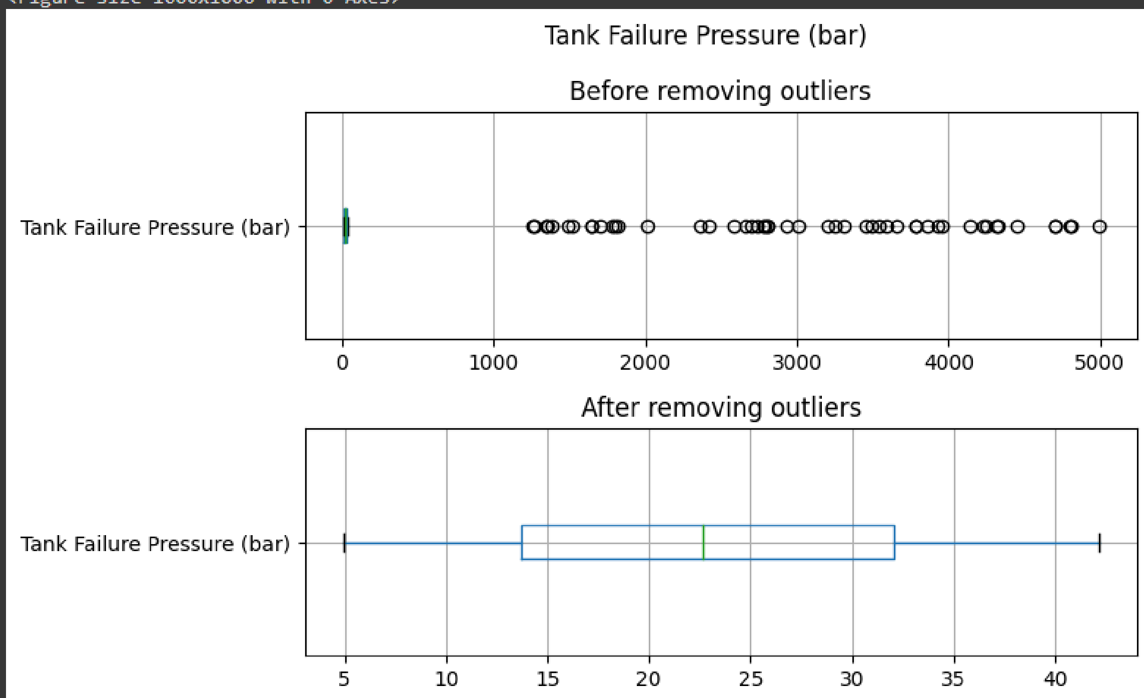
Outliers: Outliers were identified by visualising the boxplot for each feature and using the Z-Score method.



The Z-Score method detected outliers for `Tank Failure Pressure (bar)`, `Vapour Height (m)`, `Sensor Position y`, `Sensor Position z` *and* `Target Pressure (bar)`. Those outliers were removed to prevent distortion of the analysis.

```
column:  Tank Failure Pressure (bar)
upper limit:  718.5009485898352
lower limit:  -642.3946099936331
outlier count:  50

<Figure size 1000x1000 with 0 Axes>
```



Data Integrity: Some duplicate values and incorrect data entries were found in the `Status` feature. Duplicates were removed, and incorrect entries were corrected to ensure data integrity in the training dataset.

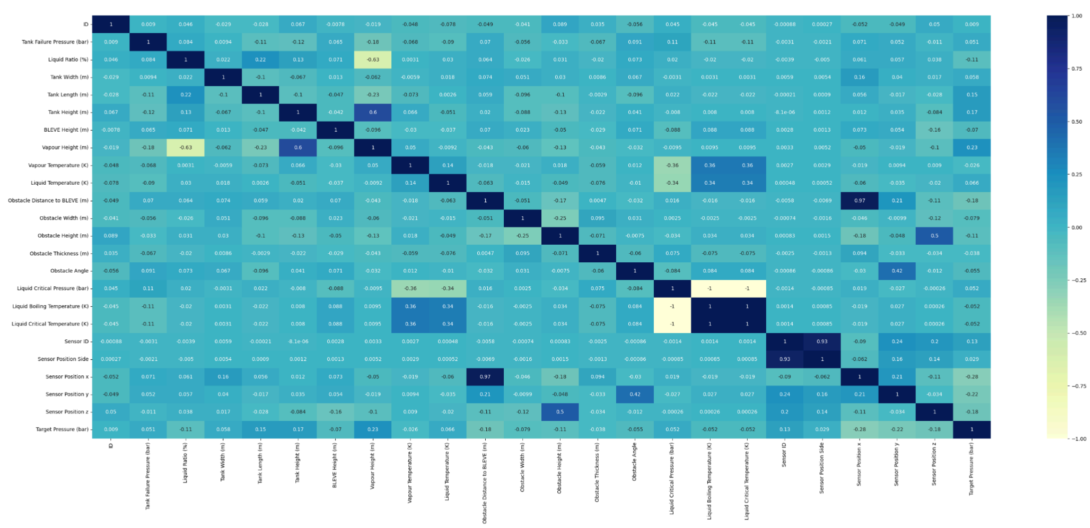
```
Status
Subcooled       6035
Superheated     3513
Subcool           22
subcooled         20
Subcoled          14
superheated        6
Superheat          6
Saperheated        3
Name: count, dtype: int64
```

Data Processing

Exploratory Data Analysis (EDA) was conducted to understand the characteristics of the dataset and the relationships between each feature.

I have visualised distributions of all features to understand their spread and identify potential patterns.

I also created a heatmap to examine pairwise correlations between features and the target variable (peak pressure) to identify strong relationships and dependencies.

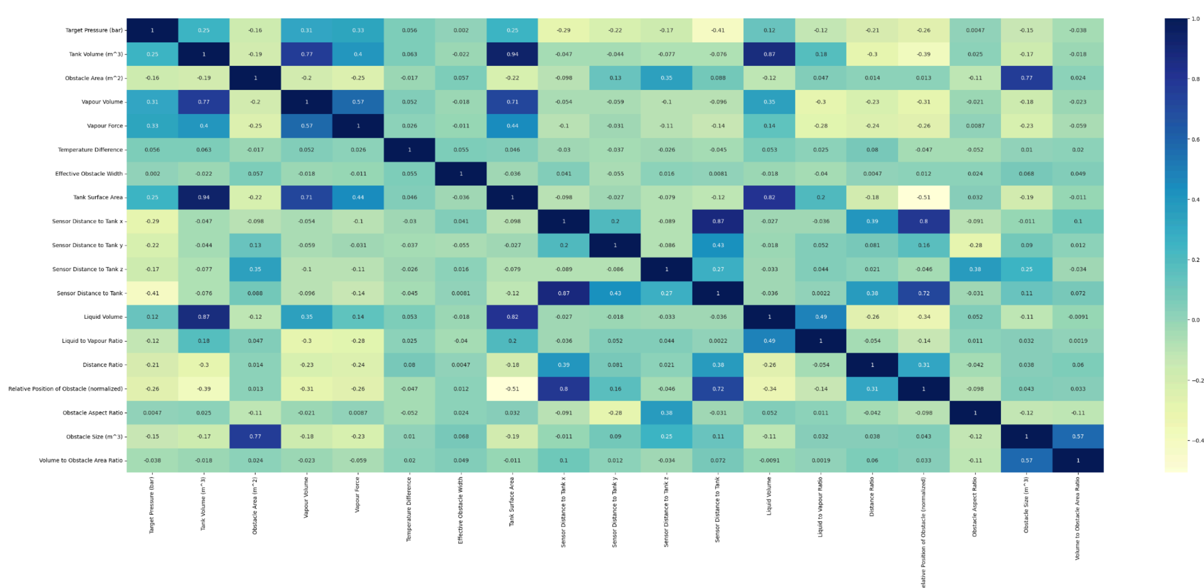


Feature Engineering

According to the above correlation heatmap, most of the features do not have a strong correlation with the target variable. So, I have to do some feature engineering steps and create new features that have a strong correlation with the target variable. Following are the new features that I created.

- $Tank\ Volume\ =\ Tank\ Width\ \times\ Tank\ Length\ \times\ Tank\ Height$
- $Obstacle\ Area\ =\ Obstacle\ Width\ \times\ Obstacle\ Height$
- $Vapour\ Volume\ =\ (1\ -\ Liquid\ Ratio)\ \times\ Tank\ Volume$
- $Vapour\ Force\ =\ Tank\ Failure\ Pressure\ (bar)\ \times\ Tank\ Length\ \times\ Vapour\ Height$
- $Temperature\ Difference\ =\ Liquid\ Temperature\ -\ Vapour\ Temperature$
- $Effective\ Obstacle\ Width\ =\ Obstacle\ Width\ \times\ Cos(Obstacle\ Angle)$
- $Tank\ Surface\ Area\ =$
  $2((Tank\ Length\ \times\ Tank\ Width)\ +\ (Tank\ Length\ \times\ Tank\ Height)\ +\ (Tank\ Width\ \times\ Tank\ Height))$
- $Sensor\ Distance\ to\ Tank\ =$
  $\sqrt{(Sensor\ Position\ x\ -\ Tank\ Length/2)^2\ +\ (Sensor\ Position\ y\ -\ Tank\ Width/2)^2\ +\ (Sensor\ Position\ z\ -\ Tank\ Height/2)^2}$
- $Liquid\ Volume\ =\ Tank\ Volume\ -\ Vapour\ Volume$
- $Liquid\ to\ Vapour\ Ratio\ =\ Liquid\ Volume/Vapour\ Volume$
- $Distance\ Ratio\ =\ Obstacle\ Distance\ to\ BLEVE\ /\ Tank\ Height$
- $Relative\ Position\ of\ Obstacle\ (normalized)\ =$
  $Obstacle\ Distance\ to\ BLEVE\ /\ (Tank\ Width\ +\ Tank\ Height\ +\ Tank\ Length)$
- $Obstacle\ Aspect\ Ratio\ =\ Obstacle\ Height\ +\ Obstacle\ Width$
- $Obstacle\ Size\ =\ Obstacle\ Width\ \times\ Obstacle\ Height\ \times\ Obstacle\ Thickness$
- $Volume\ to\ Obstacle\ Area\ Ratio\ =\ Obstacle\ Size\ /\ Obstacle\ Area$

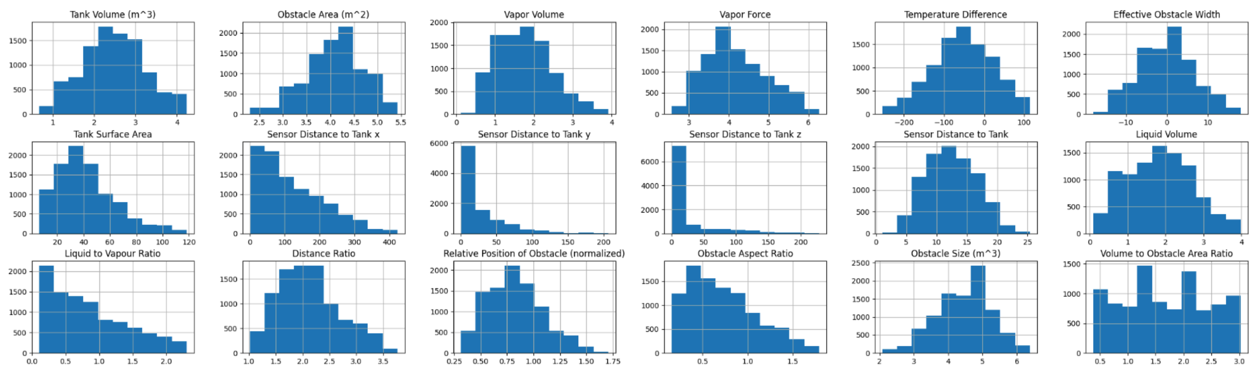
Those newly created features had a stronger correlation with the target variable.



Data Preprocessing

In the above histogram, you can see some of the features do not have bell curve distribution. To achieve this bell shape, I use their log values instead of using raw values.

```
train_data['Tank Volume (m^3)'] = np.log(train_data['Tank Volume (m^3)'] + 1)
train_data['Obstacle Area (m^2)'] = np.log(train_data['Obstacle Area (m^2)'] + 1)
train_data['Vapor Volume'] = np.log(train_data['Vapor Volume'] + 1)
train_data['Vapor Force'] = np.log(train_data['Vapor Force'] + 1)
train_data['Liquid Volume'] = np.log(train_data['Liquid Volume'] + 1)
train_data['Liquid to Vapour Ratio'] = np.log(train_data['Liquid to Vapour Ratio'] + 1)
train_data['Relative Position of Obstacle (normalized)'] = np.log(train_data['Relative Position of Obstacle (normalized)'] + 1)
train_data['Obstacle Aspect Ratio'] = np.log(train_data['Obstacle Aspect Ratio'] + 1)
train_data['Obstacle Size (m^3)'] = np.log(train_data['Obstacle Size (m^3)'] + 1)
train_data['Distance Ratio'] = np.log(train_data['Distance Ratio'] + 1)
```



The `Status` is a categorical(Object) data type feature. So, it needs to be converted into numerical or boolean data to fit our machine learning model. I used the pandas get_dummies method to convert the `Status` feature into dummy variables, which means it creates a unique column for each categorical label and marks true or false (1/0) based on the data instance's Status value.

Subcooled	Superheated
False	True
False	True
False	True
False	True
False	True
...	...
False	True
True	False
True	False
False	True
True	False

```
train_data = train_data.join(pd.get_dummies(train_data.Status)).drop(['Status'], axis=1)
train_data
```

Model Selection

Since this is a regression problem, I selected the following three widely used and popular machine learning models.

1. **Linear Regression**
2. **XGBRegressor**
3. **MLPRegressor**
4. **CatBoost Regression**
5. **TensorFlow Neural Network (Sequential Model)**

Liner Regression `(from sklearn.linear_model)`

Linear regression is a simple and interpretable model that can be used to predict values based on linear relationships with input features and target variables. It serves as a baseline model for more advanced and complex algorithms. I thought linear regression would be a good starting point for our Regression problem.

Extreme Gradient Boosting (XGBoost) Regressor `(from xgboost)`

XGBoost is known for its high performance in predictive modelling tasks, specially for tabular datasets. This model also has powerful techniques against overfitting compared to the traditional gradient boosting model. I chose this model for its ability to capture the complex and non-linear relationships in the dataset, which are common in BLEVE scenarios.

Multi-layer Perceptron (MLP) Regressor `(from sklearn.neural_network)`

MLP is one of the best simple neural network models that is capable of finding and learning non-linear relationships between input features and target values. MLP is also good at discovering hidden patterns that are not able to be seen in the original feature space. Its flexibility in adapting different datasets by customising various ranges of hyperparameters (e.g. number of layers, neurons per layer) helps to create a perfect fit for our datasets.

Cat Boost Regressor `(from catboost)`

CatBoost Regression can directly handle categorical values without data preprocessing techniques such as one-hot encoding, which reduces the risk of information losses. This also has robust overfit prevention capabilities on unseen data. CatBoost is a well-optimized, efficient model suitable for large-scale datasets.

TensorFlow NN (Sequential Model) `(from tensorflow)`

This is a deep learning model that can capture non-linear, complex relationships between input features and the target variable. This model is famous for its state-of-the-art performance, which provides advanced architectures and optimisation techniques for performing various tasks.

Hyperparameter Tuning

Hyperparameter tuning was done by using GridSearchCV, which gives the best parameters by comparing all the combinations of the given parameter grid.

For the linear regression model, there is no wide range of hyperparameters to tune. Its basic model gave the Mean Absolute Percentage Error of **0.8094**, and the tuned model had no difference.

The basic model of XGBoost Regression had **0.1566** MAPE value, Which can give more accurate results than the linear regression model. The MAPE of the tuned model was **0.1194.**

MLP Regression basic model had MAPE 0f **1.7713,** and it could be able to reduce to **0.2434** after tuning the hyperparameters.

For ensemble methods, tuned hyperparameters included the number of estimators, max depth, learning rate, and subsample. Cross-validation was also done to maximise the model's accuracy. After hyperparameter tuning, I created ideal models for each selected model with optimal (best) hyperparameters.

Ideal Model	R2 Score	MAPE
Linear Regression	0.6226031178943184	0.8094575769306446
XGBoost Regression	0.941846872399352	0.11943867954388719
MLP Regressor	0.950773950655859	0.2434838391231757
CatBoost Regression	0.9712067599925815	0.11764595342877292
TensorFlow NN	-48.135173293403604	14.347951446512903

Prediction

Even if the CatBoost gave the lowest MAPE and highest R2 score on training and validation data, its unseen test data accuracy was not that good. So, the final predictions were made using the XGBoost model, which gave the next best results of the lowest *Mean Absolute Percentage Error* of **0.1194.**

Self Reflection

I was really interested in investigating predictive analysis methods to understand the dynamics of BLEVEs when I started this project. The project's emphasis on peak pressure prediction in a complex 3D environment pushed me to learn more about feature engineering (physics) and model selection because it posed such an interesting challenge.

One of the main difficulties I faced was creating appropriate new features that could have a strong correlation with the peak pressure. According to my experience (throughout this project), This can greatly affect the accuracy of the model.

By being a part of this project, I learned a lot of important data science and machine learning techniques, including different machine learning algorithms, fine-tuning techniques and evaluating methods.

Overall, this project was so much fun for me, and I am eager to continue applying machine-learning techniques to address real-world problems like this.