

COMP6013

Report

BLEVE Peak Pressure Prediction Project

Gabriel Sim

12894732

Introduction 3

 Data Preprocessing 3

 Initial 3

 Final..... 3

 Feature Engineering..... 4

 Initial 4

 Final..... 4

 Data Split..... 4

 Model Selection 4

 Neural Network Regression 5

 Hyperparameter Tuning 7

 Model Comparison 7

Model Interpretation 8

 Global Interpretation 8

 Local Interpretation 9

Self-Reflection 10

 Overall Understanding..... 10

 Lessons Learned and Difficulties Encountered..... 10

 Restart with new approach..... 11

Introduction

This is an accompanying report to the 'BLEVE Peak Pressure Prediction' Machine Learning Project submitted May 12th, 2023. This report mainly covers data preprocessing methods, model interpretation and self-reflection. In my project, I was unable to fully complete the Model Interpretation section but have revisited it for the sake of delivering a satisfactory report. Local Interpretation was missing from my project but I have done this for the report.

Data Preprocessing

The first step was to examine the data, both test and train datasets. The generic properties to examine are the number of rows(observations) and columns(features), the data types (integer, string, float), the feature headings (title), percentage of missing values and a histogram of the data points. In the histogram of raw data points, the key details to note are the range, distribution and frequency of values observed. A correlation matrix is plotted to identify features that have a high correlation with one another. The identified features with high correlation are:

- Sensor Position X – Obstacle Distance to BLEVE(m): 97%
- Sensor Position Side – Sensor ID: 93%

The next step looks at dropping features that have no significance to model training. The 'ID' column is dropped as it is only an index for reference therefore it has no contribution to the output. Data Type Conversion is applied to categorical features to convert them to a format that is easier understood by Machine Learning Models. Standard Scaling (sklearn.preprocessing) and Normalisation is applied in an attempt to create a normal distribution. Multiple approaches were undertaken to discover a suitable model that could attain a high R2 score. The final decision on feature dropping and engineering was based on the set of combinations that could achieve a MAPE score of less than 20 on the Kaggle public scoreboard.

Initial

Drop: ID, Sensor Position Side, Tank Length (m), 'Tank Width (m), Tank Height (m), Sensor ID, Obstacle Height (m), Obstacle Width (m), Obstacle Thickness (m), Sensor Position x, Sensor Position y, Sensor Position z

Multiple combinations of these features were dropped due to feature engineering, high correlation or the presumption of low input significance (such as Sensor Position Side due to Sensor ID already indicating which side it would be on)

Encoding: Sensor Location (engineered), Status

Sensor Location is an engineered categorical feature highlighting where the sensor is located on the obstacle wall.

Final

Drop: ID

When running the Neural Network model with only the ID feature dropped, it provided a high R2 Score and reasonable MAPE score as opposed to the initial combination of features dropped.

Encoding: Sensor ID, Sensor Position Side, Status

The model performed significantly better with only the default categorical features encoded.

Feature Engineering

Initial

Two new features were initially created based on the understanding of the dataset. The first being Tank Volume (cubic m) was created as a result of multiplying the length, width and height of the BLEVE tank. The purpose was to reduce the number of features and present a more meaningful value. As the BLEVE tank is a 3D object, an explosion from the tank seemed to be more meaningful if the tank size was measured in volume rather than its separate dimensions.

The second new feature created was 'Sensor Location' derived from 'Sensor ID'. The sensors are located on the front, back and sides of the obstacle. Sensor ID 1-9 is the back sensors, 10-18 the front and 19-27 the sides. It was logical to presume that the back of the obstacle would receive much lesser pressure from the explosion as opposed to the front and side sensors. The idea that there are 9 different sensors for each location did not seem significant thus the decision of grouping them all into a single categorical value.

Final

After numerous model training with the engineered features, it proved to have higher MAPE and lower R2. Without any feature engineering the Neural Network model was able to achieve significantly better performance simply using the original features.

Data Split

Data was split into 80% training and 20% validation sets. It was done with the Sci-Kit Learn library using the 'train_test_split' function.

Split Train Data to Model-building Data and Validation Data

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_percentage_error

# Initialise X and y
X = dfTrain.drop('Target Pressure (bar)',axis=1)
y = dfTrain['Target Pressure (bar)']

# Split 20% data as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

A parameter of 0.2 indicates a 20% split for the validation set and the remaining data point (80%) represents the train set.

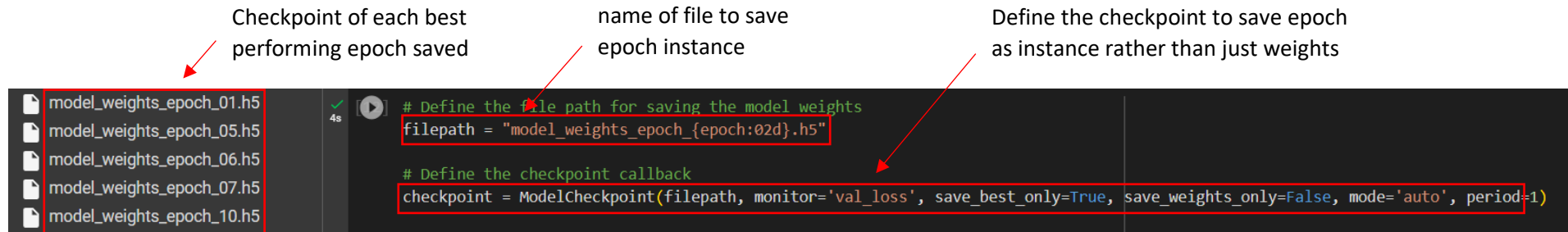
Model Selection

Four regression models were used to attempt the training and prediction of the BLEVE dataset. Linear regression was used with a combination of feature selection and engineering which provided a very low R2 score. Random Forest Regressor was also used with the same combination and had also provided a low R2 score.

XGBoost and MLP Regressor provided decent metrics with hyperparameter tuning, however the uploaded prediction on Kaggle proved that otherwise.

Neural Network Regression

This was the best performing model. After hyperparameter tuning with GCSV, the initial model is trained with the parameters: {'activation': 'relu', 'dropout_rate': 0.0, 'l2_regularizer': 0.001, 'optimizer': 'Adam'}. Although the initial metric score is much lower than other models, the method here is for the model to undergo retraining with specifically selected weights during its 50 epoch run.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer lists five files: `model_weights_epoch_01.h5`, `model_weights_epoch_05.h5`, `model_weights_epoch_06.h5`, `model_weights_epoch_07.h5`, and `model_weights_epoch_10.h5`. The code editor contains the following Python code:

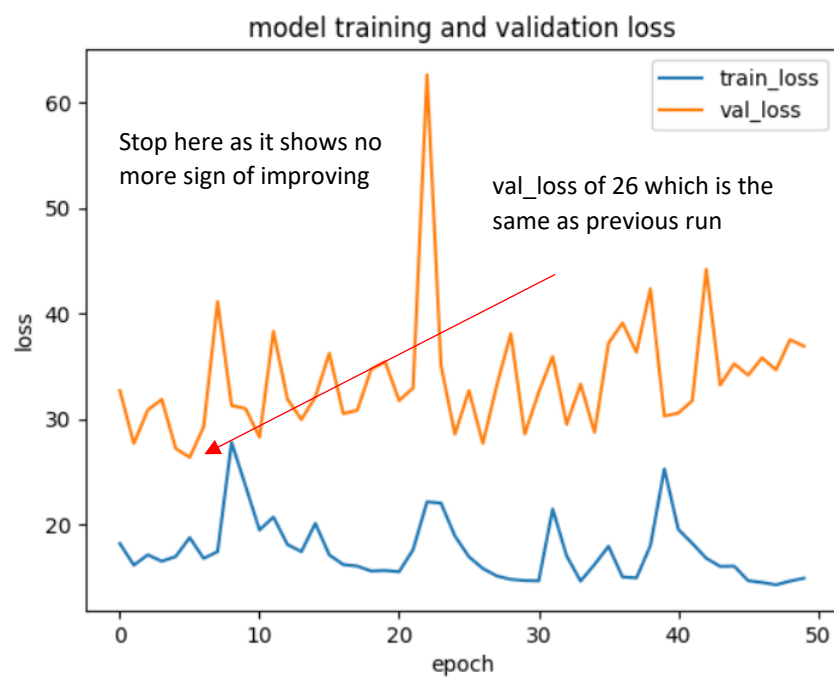
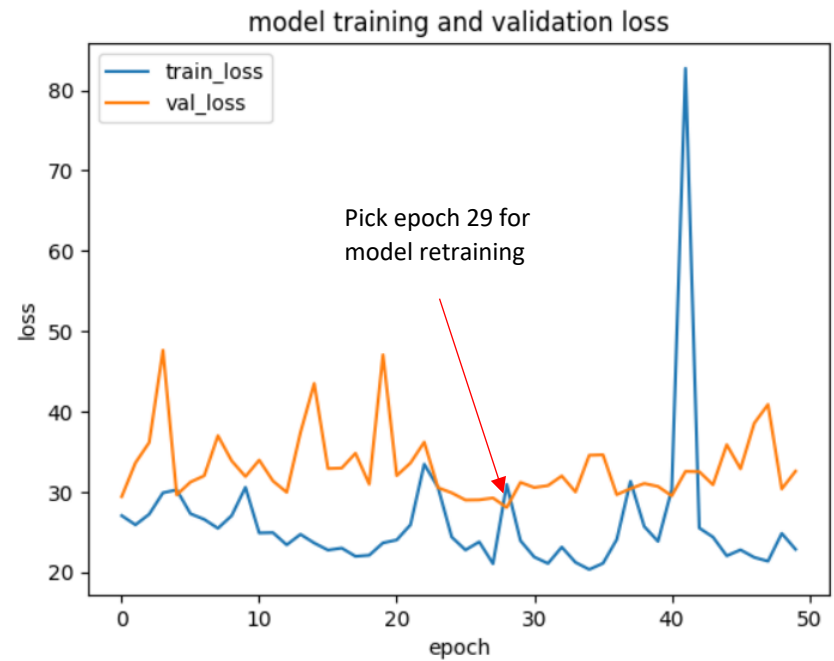
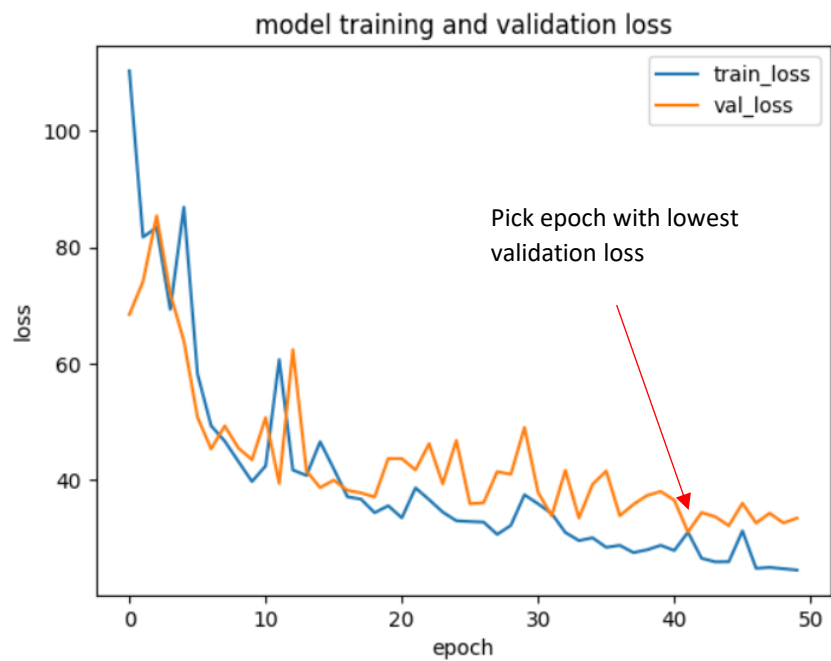
```
# Define the file path for saving the model weights
filepath = "model_weights_epoch_{epoch:02d}.h5"

# Define the checkpoint callback
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', save_best_only=True, save_weights_only=False, mode='auto', period=1)
```

Three red arrows point from text annotations to the code:

- An arrow points from "Checkpoint of each best performing epoch saved" to the `filepath` variable.
- An arrow points from "name of file to save epoch instance" to the `filepath` variable.
- An arrow points from "Define the checkpoint to save epoch as instance rather than just weights" to the `ModelCheckpoint` function call.

Below is a training and validation loss plot showing loss over training run decreases. The epoch with the lowest validation loss is selected each training run until it shows no more sign of improvement.



R2 score from initial training: 0.91 --> 0.94 --> 0.97 --> 0.96

By observing our metrics, it was clear that at the third training session (2nd retraining) we hit our highest R2 score and it was satisfactory enough to predict on the test set. After submitting our prediction to Kaggle we wound up with a MAPE of 19.69 on the public scoreboard.

Hyperparameter Tuning

Grid Search Cross Validation (GSCV) was used on the MLP Regressor Model and Neural Network Regression Model. A range of parameters are set with a scoring metric to assess which set of parameters will produce the best results. The parameter grid for the two models are:

```
from sklearn.neural_network import MLPRegressor
#Initialise MLP Neural Network
estimator_mlp = MLPRegressor(max_iter = 500, random_state=42)

params = {'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,1)],
          'activation': ['relu','tanh','logistic'],
          'alpha': [0.0001, 0.05],
          'learning_rate': ['constant','adaptive'],
          'solver': ['adam', 'sgd']}

gsc = GridSearchCV(estimator = estimator_mlp, param_grid = params,
                  cv=5, scoring='neg_mean_squared_error', verbose=0, n_jobs=-1)

gsc.fit(X_train, y_train)
gsc.best_params_

{'activation': 'tanh',
 'alpha': 0.0001,
 'hidden_layer_sizes': (50, 100, 50),
 'learning_rate': 'constant',
 'solver': 'adam'}
```

```
# create a KerasRegressor object
model = KerasRegressor(build_fn=create_model, epochs=100, batch_size=32, verbose=0)

# define the parameter grid
param_grid = {
    'activation': ['relu', 'sigmoid', 'tanh'],
    'optimizer': ['Adam', 'SGD'],
    'dropout_rate': [0.0, 0.1, 0.2],
    'l2_regularizer': [0.0, 0.001, 0.01]
}

Best parameters: {'activation': 'relu', 'dropout_rate': 0.0, 'l2_regularizer': 0.001,
                  'optimizer': 'Adam'}
Best R2 score: 0.8642656558047279
Best MAPE score: 0.5928030364285949
```

MLP NN on Test Set | r2: 0.9807678829325414 | mse: 0.025709450749297733 | mape: 0.27017165072129495

Model Comparison

Below is the table of performance metrics with the initial and final preprocessed features.

	Linear Regression (sklearn)		Random Forest Regressor (sklearn)		XGBoost Regressor (xgb)		MLP Regressor (sklearn)		Neural Network (keras)	
R2	0.50	0.90	0.66	0.85	0.68	0.97	0.65	0.98	0.13	0.97
MAPE	23.09	0.69	10.37	0.75	10.98	0.28	5.22	0.27	0.9	0.31

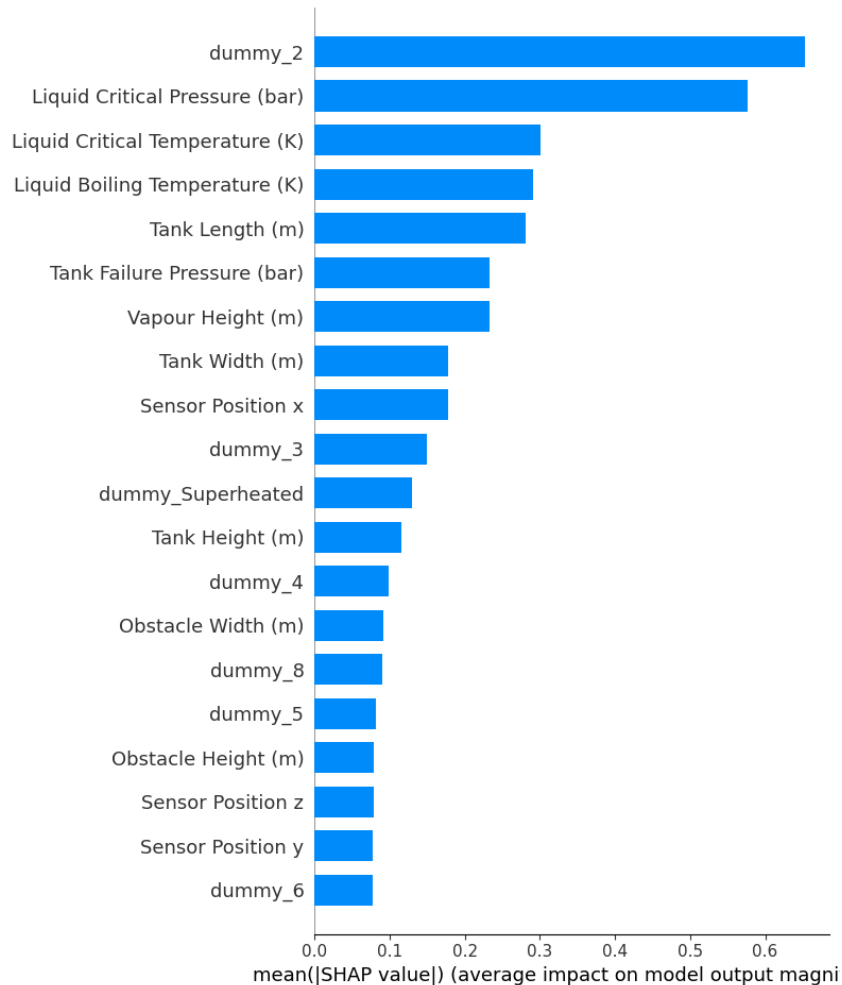
In hindsight, it was clear that with the final preprocessed features which is the default dataset with only ID dropped and categorical features encoded, the performance metrics are much better.

Model Interpretation

Global Interpretation

*Note: In my project I only did the **Feature Importance plot** and did not do the feature effect plot.*

Using the SHAP library, a sample explainer of 200 data points was used to illustrate the feature important and feature effect plots.



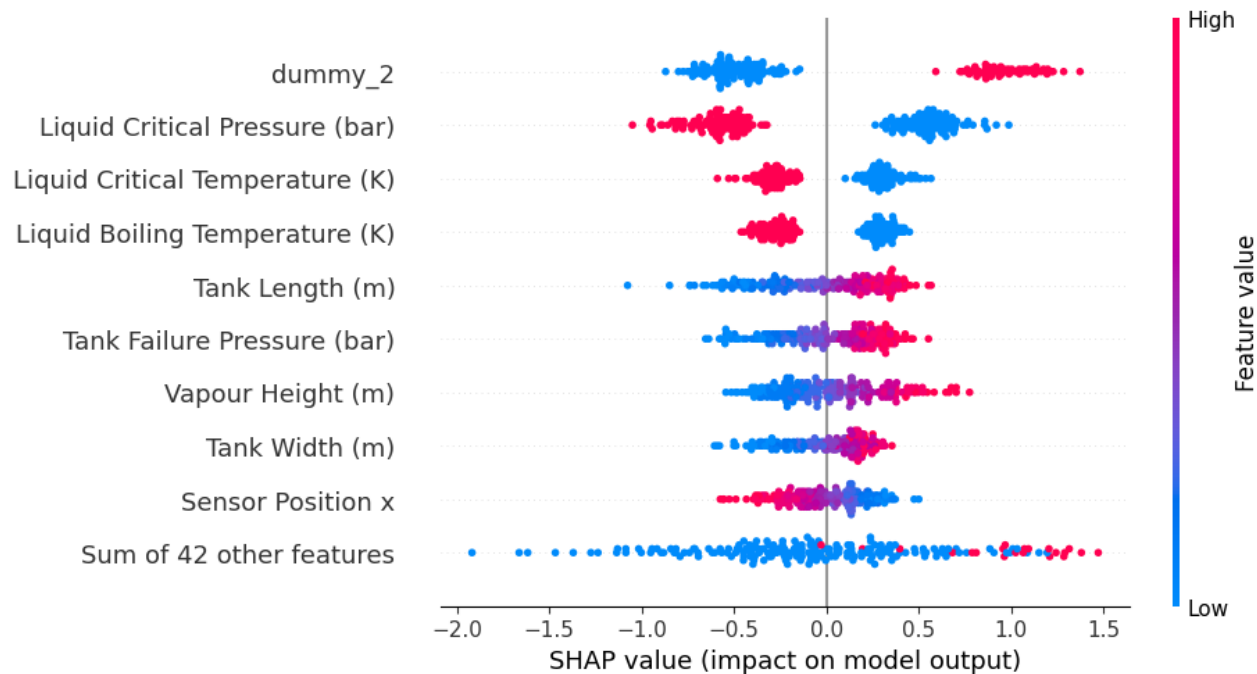
Feature Importance plot

The key features are mostly categorical features. Confusingly, there are two 'dummy_2' features and it isn't clear if it is from the 'Sensor ID' or 'Sensor Position Side' feature. In hindsight, it would have been best to rename the values for Sensor Position Side feature to avoid this confusion.

A normalised value of 0 or 1 for 'dummy_2' and 'Liquid Critical Pressure' have a greater impact on the predicted output by roughly double.

Features that are not listed are of low importance and therefore seen to have little to no impact on the predicted output. They can be removed to reduce the dimensionality of the model to speed up the training and overall performance of the model.

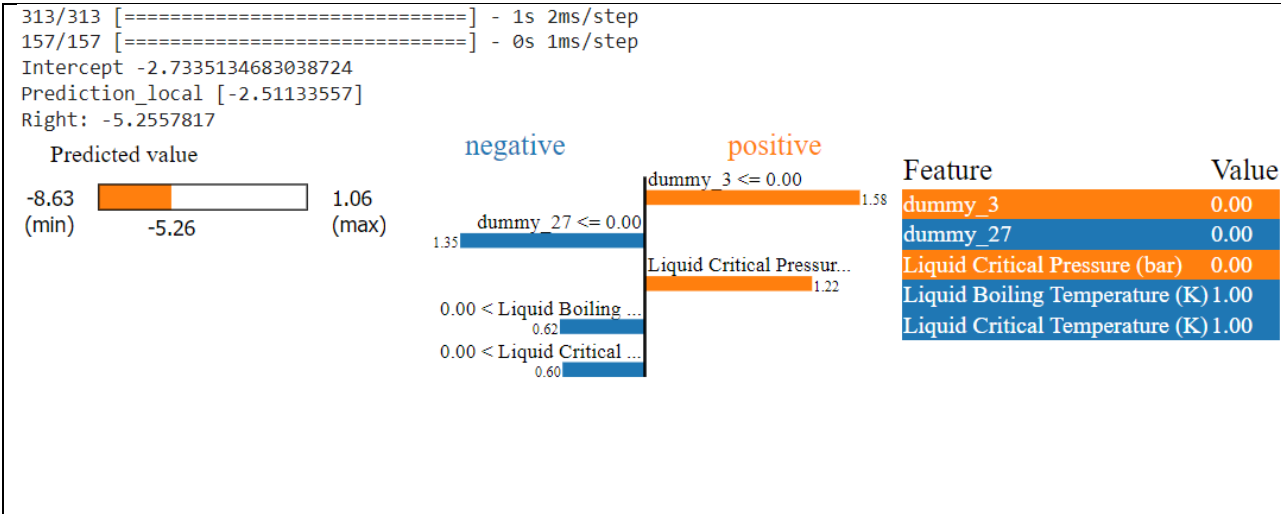
These are the top 9 features with the highest impact. The highest impact dummy_2 indicates that its categorical value of '0'



contributes to a lower predicted output as opposed to a value of '1'. Liquid Critical Pressure, Liquid Critical Temperature and Liquid Boiling Temperature indicate that a '0' input value contributes to a higher predicted output and vice versa. These 4 features clearly show a distinct margin between high and low value features because they are encoded categorical features. The remaining features are of continuous values which do show the impact of its input against a higher or lower predicted output. For example, Tank Length (m) indicates that a higher input value influences the predicted output

Local Interpretation

Note: Local Interpretation is absent from my project. In revisiting it, I have attempted to interpret the 3 required prediction values.



Lowest Predicted Value

There are 3 features that contribute to lowering the predicted output value. 'dummy_27' with value 0 means that it is not located at sensor 27 which is the side of the obstacle. Liquid Boiling Temperature is -1 (K) and Liquid Critical Temperature is 152 (K)

On the contrary, a Liquid Critical Pressure of 37.9 bar (as opposed to 42.5) and not being located at Sensor ID 3 contributes to a higher predicted output value.

<pre> 313/313 [=====] - 1s 3ms/step 157/157 [=====] - 0s 3ms/step Intercept -3.9131011875941346 Prediction_local [-1.15561983] Right: 1.6566195 </pre> <p>Predicted value</p> <p>-7.71 (min) 1.66 (max)</p> <p>negative positive</p> <p>0.00 < Liquid Critical P... 1.23</p> <p>dummy_4 <= 0.00 1.39</p> <p>0.00 < dummy_27 <=... 1.30</p> <p>Liquid Critical Temper... 0.71</p> <p>Liquid Boiling Temper... 0.58</p> <table border="1"> <thead> <tr> <th>Feature</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>dummy_4</td> <td>0.00</td> </tr> <tr> <td>dummy_27</td> <td>1.00</td> </tr> <tr> <td>Liquid Critical Pressure (bar)</td> <td>1.00</td> </tr> <tr> <td>Liquid Critical Temperature (K)</td> <td>0.00</td> </tr> <tr> <td>Liquid Boiling Temperature (K)</td> <td>0.00</td> </tr> </tbody> </table>	Feature	Value	dummy_4	0.00	dummy_27	1.00	Liquid Critical Pressure (bar)	1.00	Liquid Critical Temperature (K)	0.00	Liquid Boiling Temperature (K)	0.00	<p>Highest Predicted Value</p> <p>The only feature that lowered the predicted output value is Liquid Critical Pressure with 42.5 bar.</p> <p>Being located at Sensor ID 27 (side of obstacle), not at Sensor ID 4 (back of obstacle), Liquid Critical Temperature of 96.7 (K) and Liquid Boiling Temperature of −42 (K) contribute to a higher predicted output value.</p>
Feature	Value												
dummy_4	0.00												
dummy_27	1.00												
Liquid Critical Pressure (bar)	1.00												
Liquid Critical Temperature (K)	0.00												
Liquid Boiling Temperature (K)	0.00												
<pre> 313/313 [=====] - 1s 4ms/step 157/157 [=====] - 1s 4ms/step Intercept -4.048145072312133 Prediction_local [-1.1087664] Right: 0.06959428 </pre> <p>Predicted value</p> <p>-9.14 (min) 1.49 (max)</p> <p>negative positive</p> <p>0.00 < Liquid Critical P... 1.18</p> <p>dummy_3 <= 0.00 1.48</p> <p>0.00 < dummy_27 <=... 1.41</p> <p>Liquid Critical Temper... 0.67</p> <p>Liquid Boiling Temper... 0.55</p> <table border="1"> <thead> <tr> <th>Feature</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>dummy_3</td> <td>0.00</td> </tr> <tr> <td>dummy_27</td> <td>1.00</td> </tr> <tr> <td>Liquid Critical Pressure (bar)</td> <td>1.00</td> </tr> <tr> <td>Liquid Critical Temperature (K)</td> <td>0.00</td> </tr> <tr> <td>Liquid Boiling Temperature (K)</td> <td>0.00</td> </tr> </tbody> </table>	Feature	Value	dummy_3	0.00	dummy_27	1.00	Liquid Critical Pressure (bar)	1.00	Liquid Critical Temperature (K)	0.00	Liquid Boiling Temperature (K)	0.00	<p>Largest Error Predicted Value</p> <p>There are also 4 features that contribute to a large error predicted value. Being located at Sensor ID 27 (side) with Liquid Critical Temperature of 96.7 and Liquid Boiling Temperature of −42 (K) contribute to a large error predicted value.</p>
Feature	Value												
dummy_3	0.00												
dummy_27	1.00												
Liquid Critical Pressure (bar)	1.00												
Liquid Critical Temperature (K)	0.00												
Liquid Boiling Temperature (K)	0.00												

Self-Reflection

Overall Understanding

There is great value in the use of Machine Learning for understanding large complex datasets. With this assignment, several GSCV repetitions were done on several models. Machine Learning was able to automate this process making it very convenient to test a range of parameters without intervening. The final model I decided to use is a deep neural network model which is appropriate with the BLEVE dataset primarily because the total number of features after encoding was 51. The implemented model trained effectively with the large dataset, although very time consuming.

Lessons Learned and Difficulties Encountered

The most important lesson I learned was that I should not have done any feature selection or engineering at the start. I ended up with extremely poor metrics for the basic models and decided that Deep Neural Networks was the best model. And even with Deep NN, I was still encountering poor metrics until I used the default features. When I revisited the basic models with the default features, surprisingly the metrics are on par with the Deep NN model if not better.

Using encoding to create 20+ columns with an overall 54 columns is not necessarily a bad choice. Deep Neural Networks and even some basic Machine Learning Models can handle many categorical features. Encoding is essential for ML models as it removes the amplification of numerical representation in a categorical feature. The number values may only represent a group or category. Without encoding, the training model will treat bigger numbers to have higher values than smaller numbers which will create a bias and result in poor training and prediction. However

With a Deep NN Model, you can retrain it a specific epoch instance once training loss stagnates. Being able to retrain the model with the exact weights at the epoch instance that has the lowest validation loss a couple times gave my best performing R2 score of 97% and an overall MAPE of 19.00361 on the Kaggle private scoreboard.

What I found difficult doing this assignment was that GSCV takes very long to run especially with NN models. I only had time to run it twice on my Deep NN as it took about an hour each time. Another struggle was using the SHAP package to do my global interpretation. I kept running into errors when following the documentation and examples. It took a lot of reading from other sources to figure out how to do it.

Restart with new approach

I would keep the same Data Preprocessing methods as it ensures you have examined the data thoroughly and gained a good understanding of it. I would also still use the same initial models (Linear, Random Forest, XGBoost) to train and test. However, in my new attempt I would not start feature engineering or dropping at the first run.

I would try to get all these models up and running quicker so that I could use GSCV quicker as well. This is due to GSCV requiring time to tune and test the hyperparameters. I would also try to implement Weights and Biases next time to be able to keep track of all the tests as I found it difficult near the end of the project to be able to remember the performance of the older models.

Only after would I transition to Neural Networks as these models take longer to train and tune especially deep neural networks. After testing all the models, I will attempt implementing feature selection and engineering and redo the training process with all models again to gain better metrics.

I would base the feature selection and engineering on the Model Interpretation. Global and Local interpretation can give light into the importance and effect of certain features which in turn informs which features to select and possibly engineer to gain an even better performing model.

I would definitely spend more time on the Model Interpretation section as I failed to get it working in the project. I would get at least 500 samples for a more accurate SHAP plot, get the Partial Dependency Plots working and evaluate more data instances with LIME.

In revisiting the project, the interpretation of features shows very few Sensor ID values having a great impact on output. It leads to investigating whether Sensor ID can be completely dropped, and Sensor Position Side be its replacement.