COMP3010 Machine Learning Assignment Report

Data Cleaning

The data provided for training had many issues with it such as duplicate values, missing values, outliers and incorrect entries. These issues would be problematic for any model to train on and would give it an inaccurate predicting capability. To solve this I first checked for incorrect entries such as in the Status column there were a lot of entries that were not one of the valid values of 'Superheated' and 'Subcooled'. I figured there really was not a fair way to replace these incorrect values as replacing it with one or the other of the correct values would make the data very biased towards whatever was picked so I just omitted the rows with the incorrect entries. Furthermore, for the values that needed to be positive, I looped through the relative columns and made sure they were removed.

For the missing values I filled in all the numeric entries with the median of the column and I filled in all the categorical entries with the mode of the column. For the duplicate values, they were all removed. Finally, for the outliers I calculated the Interquartile Range (IQR) and calculated the lower and upper bound and for the entries with values outside of these bounds they were replacing by the closest bound's value.

Data Processing

For data preprocessing I first did data-type conversions as the models needed basic numeric data for training and so I had to One-Hot Encode the Status and Sensor ID columns into binary values. Feature engineering, the Obstacle Angle was in degrees and the model would not know that 0 and 360 degrees are the same value, so I had to convert the values into radians to make them into their cosine and sine values. Furthermore, I created a ratio column between the tank width and length as it was recommended in the assignment brief.

For scaling I separated the features that require scaling from those that do not and scaled those features. After scaling I put all the features back together in the original order they were in.

Model Selection

I have chosen three models: Linear Regression, XGBoost and Deep Neural Network. For the first model I chose Linear Regression as a baseline for the other models as it is one of the simplest models. Linear Regression works in the case of this assignment as it predicts one continuous variable finding the relationship between input features and the target variable.

I was considering using Gradient Boosted Decision Trees (GBDT) as it has good generalization which provides it more accuracy, it is quite fast and it works for smaller datasets such as the training data provided. However, I found the XGBoost model (stands for eXtreme Gradient Boosting) which uses multiple GBDTs at the same time so its parallel processing means it is still very fast but more accurate. It provides both speed and accuracy so it was a very efficient choice for the model.

The Deep Neural Network (DNN) was lastly chosen as it can find non-linear patterns through its more complex structure, consisting of many layers which is necessary for accuracy because there are so many features for the target variable. The Deep Neural Network follows the sequential model from the Keras library where there is an input layer where the data is inputted, the hidden layers extract patterns and learns the complex relationship between features and the output layer generates the results and in this case provides a continuous output related to Regression.

Hyperparameter Tuning

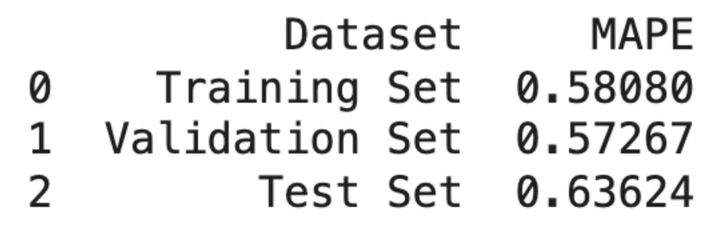
For the Linear Regression model, as it was treated as a baseline model I did not apply any hyperparameter tuning.

For the XGBoost model I used Randomized Search to find the best hyperparameters. For the number of trees I made a range of numbers from 100 – 1200, learning rate from 0.01 – 0.2, max depth from 3 – 7, subsample from 0.7 – 1.0, reg_alpha from 0 – 1. Then the Random Search makes a combination of these hyperparameters and based off the R^2 metric finds the best hyperparameters for the model. The best hyperparameters found were the following: subsample: 0.9, reg_alpha: 0.1, trees: 900, max depth: 7, learning rate: 0.05.

For the Deep Neural Network I used KFold Cross Validation with 5 splits to more accurately evaluate the model and changed the hyperparameters of the model accordingly using the MAPE scores provided by the model. Through this method I found that four hidden layers with neurons descending from 256, 128, 64 and 32 were providing accurate results. Furthermore, I tried adding Batch Normalization and Dropout layers to multiple layers and found out having one Batch Normalization layer on the first hidden layer provided the best results.

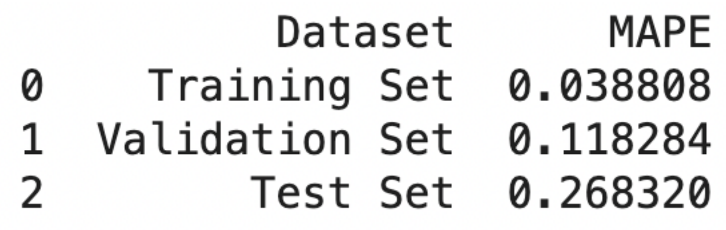
Prediction Performance

Linear Regression:



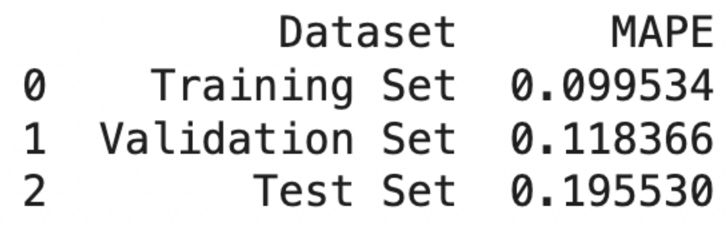
```
           Dataset     MAPE
0     Training Set  0.58080
1   Validation Set  0.57267
2         Test Set  0.63624
```

XGBoost:



```
           Dataset      MAPE
0     Training Set  0.038808
1   Validation Set  0.118284
2         Test Set  0.268320
```

Deep Neural Network:



```
           Dataset      MAPE
0     Training Set  0.099534
1   Validation Set  0.118366
2         Test Set  0.195530
```

Self-Reflection

This assignment was very enjoyable and acted as the gateway to Machine Learning (ML). Learning to create different types of models and applying it to real data felt very meta and thus was more engaging as it felt extremely practical and applicable to reality.

There were many issues such as having to learn a lot of the aspects of ML that were new to me such as all the packages, new language that comes with the packages such as ‘relu’ for activation in the DNN which stands for Rectified Linear Unit, model creation etc.. Also, the training of the models took very long, and it slowed down the whole process as I would have to wait for it to train with every little change. This waiting time showed me the importance of optimizing the model to be fast at training while retaining its accuracy and it also demonstrated how costly ML can be with time and computing costs. Another major issue was the preprocessing as different models would receive differently processed data and it was a struggle to maintain compatibility between different data and different models while trying to minimize code duplication.

For future projects I think I would plan the method more instead of just going straight into it. Having a more methodical plan would provide structure and simplicity rather than just starting with no plan as you would have to jump to different steps back and forth as you try to figure out how to complete the assignment as you go which can just make the whole process messier.