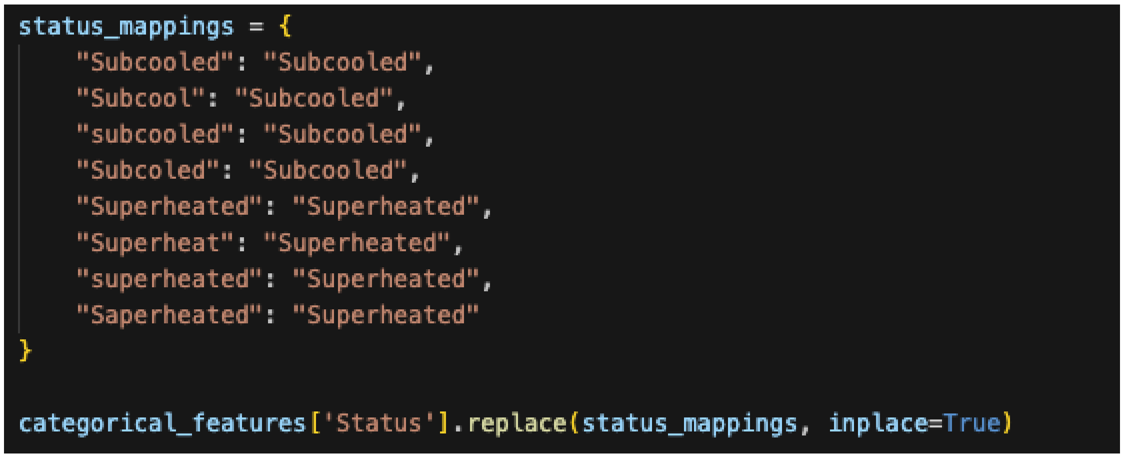
COMP-3010 Assignment Report

Info

Name : Kalindu Amantha Abeysinghe
Curtin ID : 20785947
Kaggle username : kalinduabeysinghe (Kalindu Abeysinghe)

Data Cleaning

1. With regards to null values and duplicates, 379 null values and 50 duplicates were found in train.csv and those values were dropped from the dataframe.
2. Certain incorrect data were found in, mainly in the Status column where the values were in different cases and had spelling errors. For example, “Superheated” is entered as “saperheated” and “superheated” in some records. Those values were replaced using the two selected possible values: “Superheated” and “Subcooled” as below:



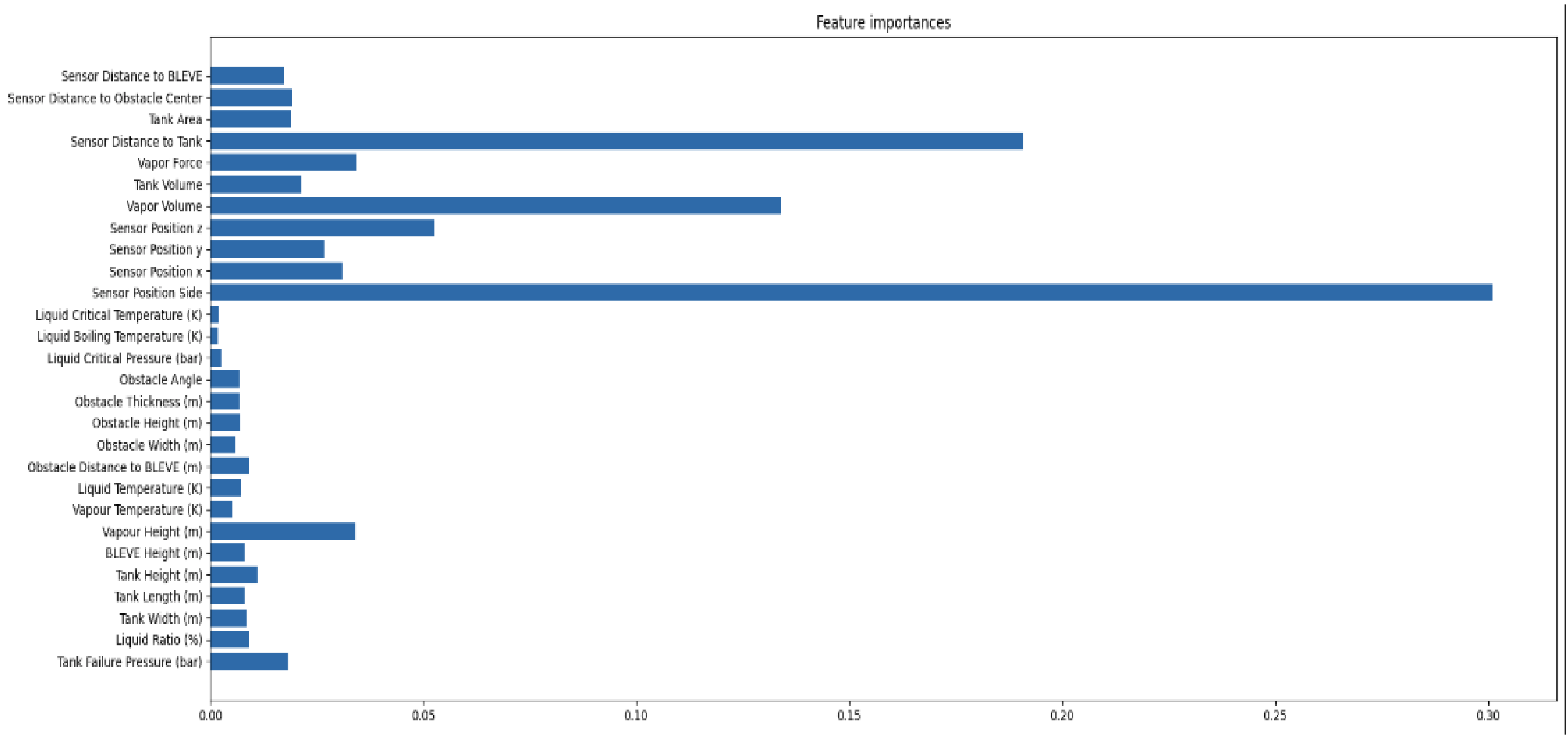
```
status_mappings = {
    "Subcooled": "Subcooled",
    "Subcool": "Subcooled",
    "subcooled": "Subcooled",
    "Subcoled": "Subcooled",
    "Superheated": "Superheated",
    "Superheat": "Superheated",
    "superheated": "Superheated",
    "Saperheated": "Superheated"
}

categorical_features['Status'].replace(status_mappings, inplace=True)
```

3. The outliers were treated by calculating the z-score for the features and only filtering values with a z-score below the threshold of 3.

Data Processing

1. ## Feature selection

   Only features that had contributed to the prediction of the Target Pressure were included to train the models. For example, the \`ID\` and the `Sensor ID` columns were dropped from the dataset. The feature contribution was selection calculating the feature importances using an \`ExtraTreesRegressor\` and by viewing the heatmap of the correlation coefficients of the features. Some snapshots are attached below :



2. ## Feature engineering

   A number of new features were calculated from the provided features in the dataset, which improved the accuracy of the model. The new features calculated are :
   - Vapor volume
   - Tank volume
   - Sensor Distance to the tank
   - Tank area
   - Sensor Distance to Obstacle Center
   - Sensor Distance to BLEVE.

3. ## Normalization / Scaling

   The dataset was scaled using a MinMaxScaler from \`sklearn preprocessing\` as the trained models showed to be sensitive to the magnitude of the feature's value.

4. Data type conversion

   There was one Categorical feature in the dataset, which is the Status Column and the feature was encoded into a numeric value using a LabelEncoder from sklearn.

Model Selection & Hyper Parameter tuning

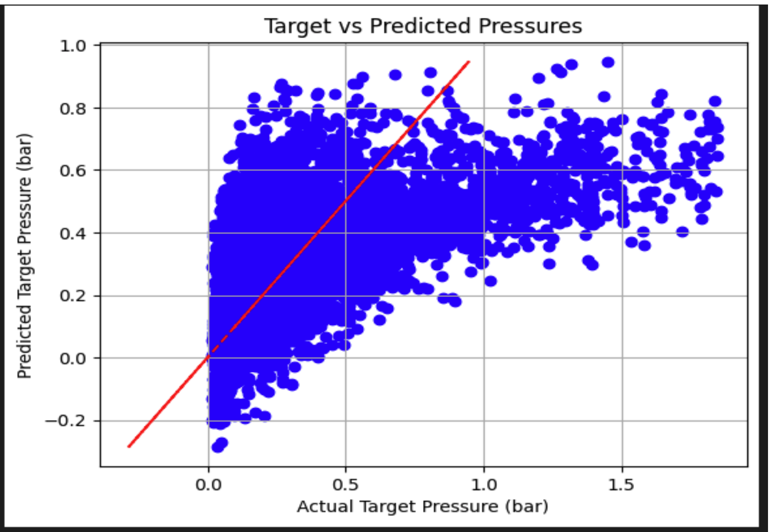
1. Model types used

1. Linear Regression

This model was selected at first since it is the simplest model. A multivalued regression was used against the Target Pressure as target variable. The following features were included as they gave the highest feature importance score :

- Obstacle Distance to BLEVE (m'
- Vapor Force
- Sensor Distance to Tank
- Vapor Volume
- Sensor Position y
- Sensor Position z
- Obstacle Angle
- Liquid Critical Pressure (bar)
- Status
- Sensor Position Side

However the model was rejected since the R^2 score was too low, and the MAPE score was too high. The results are below:



```
R squared : 0.33
Mean Absolute Percentage Error : 1.07
Mean Absolute Error : 0.18
Root Mean Squared Error : 0.25
```

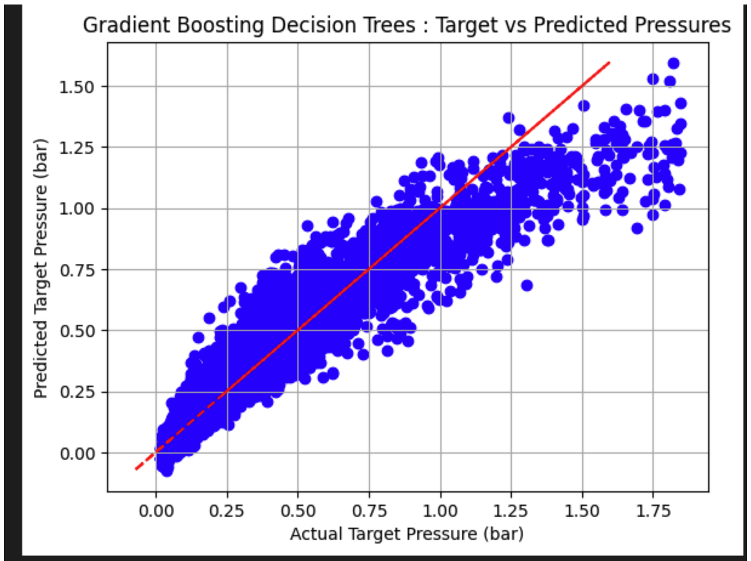
2. Gradient Boosting Decision Trees

This was the next model chosen, since the dataset proved to be non-linear and there were quite a number of features in the dataset, 30 including the newly engineered ones. This was indeed more efficient than Linear Regression. Hyper parameters for are list below :

- n_estimators
- learning_rate
- max_depth
- subsample
- Random_state

Hyper parameter tuning was using GridSearchCV from sklearn. The optimal hyper parameters turned out to be as follows : {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 600, 'subsample': 0.75}. The performance of the model did seem to improve by a significant amount but the model proved to be bit overfitting with high variance as the MAPE when submitted to Kaggle was very much higher than 0.0178 :

Before hyper parameter tuning	After hyper parameter tuning
R^2 score = 0.90	R^2 score = 0.999
MAPE = 0.27	MAPE = 0.0178



```
Gradient boosting regreesion R squared : 0.90
Gradient boosting Mean Absolute Percentage Error : 0.27
Gradient boosting Mean Absolute Error : 0.06
Gradient boosting Root Mean Squared Error : 0.10
```

2. XGBoost Regression

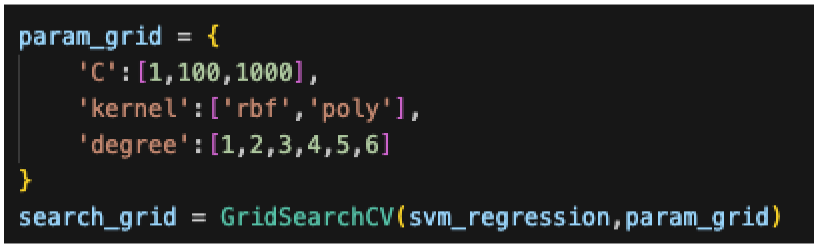
XGBoost was chosen as the next model, since it is a more efficient and regularized version of gradient boosting with advanced L1 and L2 regularization.[1] This was indeed more efficient than GradientBoostingRegression but was too overfitting in my case even after hyper parameter tuning. GridSearchCV was used for hyper parameter tuning for this too. //TODO: include hyper parameters here.
However this gave my second best Kaggle score which was 0.25537

Before hyper parameter tuning	After hyper parameter tuning
R^2 score = 0.992	R^2 score = 0.9998
MAPE = 0.101	MAPE = 0.02019

3. Support Vector Regression

Support Vector Regression was chosen as a model as it supports non-linear functions as

well using different kernel functions. [2] Hyper parameters for SV Regression was mainly considered as the kernel used and the degree, the GridSearchCV tuning used is as below:



```
param_grid = {
    'C':[1,100,1000],
    'kernel':['rbf','poly'],
    'degree':[1,2,3,4,5,6]
}
search_grid = GridSearchCV(svm_regression,param_grid)
```

4. Neural Network

An Artificial neural network was used to incorporate deep learning as a means to predict the Target Pressure in this case. Since the dataset's variables and target variable does not have a linear relationship, a network with hidden layers were used, whose architecture is as described below layer-wise:

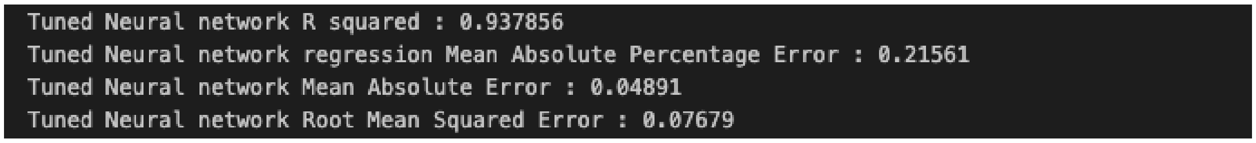
1. Input layer - Dense
2. Dropout layer - To regularize the input data by randomly setting some input units to zero. This helps prevent overfitting.
3. GaussianNoise layer - Another regularization technique which adds Gaussian noise to some inputs in the training set. This too helps prevent overfitting.
4. Dense layer - PReLU activation function
5. Dense layer - PReLU activation function. Fewer neurons than previous layer
6. Dense layer - PReLU activation function. Fewer neurons than previous layer
7. Dense layer - PReLU activation function. Fewer neurons than previous layer
8. Output layer - Another Dense layer, but with `linear` activation function and one neuron.

Hyper parameter tuning for the ANN (Artificial Neural Network) was done using RandomSearch from keras-tuners. The Best hyper parameters are as follows :

```
{'units_1': 156, 'units_2': 52, 'units_3': 29, 'units_4': 23, 'units_5': 12,
'learning_rate': 0.01, 'weight_decay': 0.0004, 'epsilon': 1e-06, 'beta_1': 0.9,
'beta_2': 0.999}
```

Before hyper parameter tuning	After hyper parameter tuning
R^2 score = 0.7635	R^2 score = 0.9999
MAPE = 0.331	MAPE =0.2067

This was my chosen model for the final prediction since it gave me the best score in the Kaggle competition.



```
Tuned Neural network R squared : 0.937856
Tuned Neural network regression Mean Absolute Percentage Error : 0.21561
Tuned Neural network Mean Absolute Error : 0.04891
Tuned Neural network Root Mean Squared Error : 0.07679
```

2. Evaluation metrics

The evaluation metrics used to evaluate each model are :

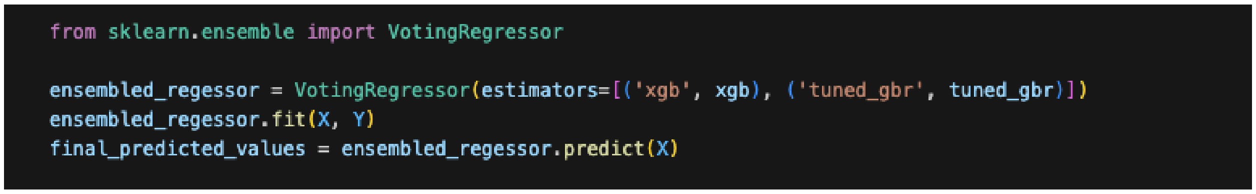
1. R-squared score of the model.
2. MAPE (Mean absolute percentage error)
3. MAE (Mean absolute error)
4. RMSE (Root Mean squared error)

The evaluation results are outlined in the table below :

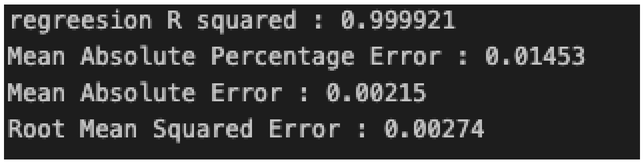
Model	R-squared score	MAPE	MAE	RMSE
Linear Regression	0.33	1.07	0.18	0.25
Gradient Boosting Decision Trees	0.90	0.27	0.06	0.10
Gradient Boosting Decision Trees (TUNED)	0.9988	0.0178	0.00264	0.0032
XGBoost Regression	0.992	0.1016	0.01917	0.02718
XGBoost Regression (TUNED)	0.9998	0.02019	0.00298	0.00384
Support Vector Regression	0.827	0.437	0.082	0.128
Artificial Neural Network	0.7635	0.2922	0.0719	0.1498
Artificial Neural Network (TUNED)	0.9378	0.2067	0.04891	0.07679

3. Model Ensembling

With regards to model ensembling, VotingRegressor from `sklearn.ensemble` was used to combine results from the XGBoostRegression and GradientBoostingRegressor models. In this, both the tuned models were used. A sample code snippet is as below with its results.



```
from sklearn.ensemble import VotingRegressor

ensembled_regessor = VotingRegressor(estimators=[('xgb', xgb), ('tuned_gbr', tuned_gbr)])
ensembled_regessor.fit(X, Y)
final_predicted_values = ensembled_regessor.predict(X)
```



```
regreesion R squared : 0.999921
Mean Absolute Percentage Error : 0.01453
Mean Absolute Error : 0.00215
Root Mean Squared Error : 0.00274
```

Prediction

The final prediction on the test set was arrived at using the Artificial Neural network, as it proved to give me the best score without a huge overfitting.

Self-reflection

I gained a great understanding of machine learning from this assignment as a beginner to this aspect. I gained practical first hand experience about understanding a given dataset, detecting outliers in it, normalizing/standardizing it and preprocessing it.
Below are some activities I would have done differently if I had to start the assignment all over again :

- Would have spent more time researching the domain field (BLEVE explosions in this case) of the problem, so I could engineer more features which would have a better correlation to the Target Pressure.
- Would have looked into how to make the XGBoost regression model less overfitting, and retraining the model for a better performance.
- Would have researched more about Neural networks, mainly about its layers, neurons and activation functions, and most importantly how to design a good neural network model architecture for a given problem and its dataset.

References

1. Khandelwal, Neetika. "A Brief Introduction to XGBoost. Extreme Gradient Boosting with XGBoost! | by Neetika Khandelwal | Towards Data Science." Towards Data Science. https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6#:~:text=XGBoost%20vs%20Gradient%20Boosting,can%20be%20parallelized%20across%20clusters.
2. "Support Vector Regression Tutorial for Machine Learning." Analytics Vindhya. https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/.