PROJECT REPORT

ON

PREDICTION OF CO CONVERSION (%) IN WATER GAS SHIFT REACTION USING REGRESSION MODELS

by

MADASU NAGA SRAVYA 21BCE9179

Vellore Institute of Technology, AP

Under the Guidance of

Dr. K. Yamuna Rani

Chief Scientist

Process and Dynamic Control Group

Chemical Engineering Sciences

Indian Institute of Technology

Hyderabad – 500007

India

(2023)

CERTIFICATE

This is to certify that **Ms. Madasu Naga Sravya 21BCE9179,** a student at **Vellore Institute of Technology, AP** has worked on the project titled **"Prediction of CO Conversion (%) in Water Gas Shift Reaction using Regression Models"** from **3-10-2023** to **16-10-2023** under my supervision. This project is carried out at the Process Dynamics and Control Simulation Lab, Indian Institute of Chemical Technology, Hyderabad.

Date:

Signature of the Guide

(Dr. K. Yamuna Rani)

Chief Scientist

IICT, Hyderabad.

TABLE OF CONTENTS

ABSTRACT

Regression models play a crucial role in predicting future outcomes, unraveling intricate relationships between variables, and enhancing decision-making precision in various domains. The aim of this project is to predict the CO conversion rate (%) in the Water Gas Shift Reaction by analyzing and quantifying the impact of independent variables on a dependent variable by developing different regression Models. Different regression models and its related terminology are introduced and the developed regression models i.e, Random Forest, XGBoost, LightGBM, CatBoost, and Artificial Neural Networks are explained in detail. The performance of these models was analyzed using two regression metrics, R – squared score and mean squared error. Ideally, the R^2 score of a model must be close to 1 and the mean squared error must be close to 0. The model that met this criterion was taken as the best performing model. Light Gradient Boosting Machine with optimal parameters was determined to be the best model with a R^2 score of 82.842 % and a mean squared error of 177.0934.

INTRODUCTION

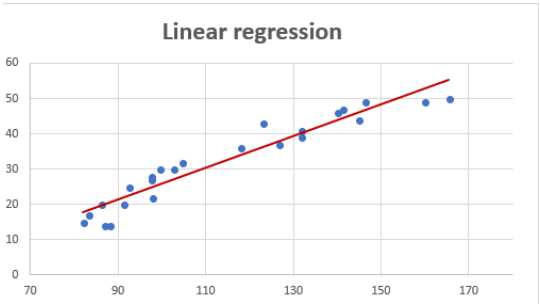
Regression is a statistical method used to find the relationship between a set of independent variables/features and one or more dependent variables. A regression model is generally used to predict a continuous variable/outcome. It does this by drawing a line or curve across each data point on the target – predictor graph in a way that minimizes the vertical distance between the data points and the regression line.

In the real world, there are many situations when we need to forecast future events, such as weather patterns, sales figures, marketing trends, etc. In these situations, we need technology that can forecast events more precisely. Here is where Regression Analysis comes into play.

The simplest form of a Regression Model is a **Linear Regression Model**. It has the following equation –

$$y = \alpha + \beta x$$

Here, $\boldsymbol{\alpha}$ = y - intercept and $\boldsymbol{\beta}$ = slope, **x** = independent variable and **y** = dependent variable.



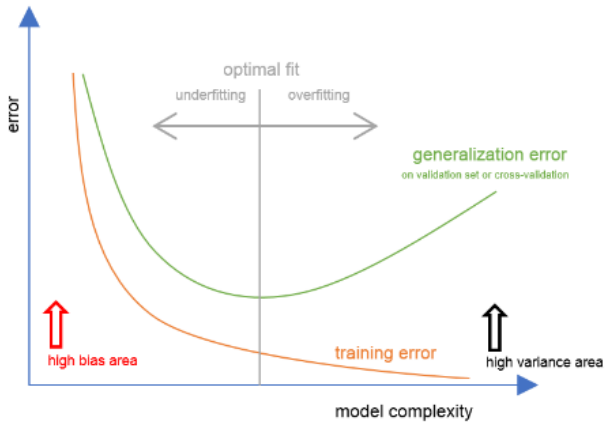
(Figure 1 - Linear Regression Line)

The slope is found using the slope formula $\mathbf{m = Y_2 – Y_1 / X_2 – X_1}$ and y – intercept is found by keeping **X = 0** in the formula.

Bias of an estimator is the difference between the estimator's expected value and the true value of the parameter being estimated. **Variance** is the difference in the accuracy of a machine learning model's predictions between the training data and the test data.

A model is said to be **Underfitted** if it is unable to perform good on both training and testing data. This might be because the model is unable to learn even the basic patterns in the data. They have **high bias** and **low variance.**

A model is said to be **Overfitted** if it performs very well on training data but fails to perform on new data. This might be due to the model learning from the noise present within the data. They have **low bias** and **high variance.**



(Figure 2 – Model complexity vs error graph)

R – squared (R^2) score or Coefficient of determination is a statistical measure that represents the goodness of a fit. It is the value that is 1 minus the proportion of sum of squared difference between the true value and the predicted value and sum of squared difference between the mean of the sample and the individual values in that sample. The value of R^2 lies between 0 and 1. The closer to 1, the better the regression fit.

$$R^2 = 1 - \frac{\sum_{1=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{1=1}^{N} (y_i - \bar{y})^2}$$

Mean Squared Error (MSE) is the average of the sum of squared difference between the true value and the predicted value of a dataset. The closer to zero, the better the model is.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Hyperparameters is an important terminology to be noted**. Hyperparameters** are those parameters that control the learning process of a model. They can vary depending on the model and can be tuned to optimize the model.

A model can be **optimized** by **hyperparameter tuning** using various method including – GridSearchCV Method, RandomSearchCV method, HalvingGridSearch Method, Bayesian Optimization and more.

For this project, we used **GridSearchCV** method which takes an **estimator, parameter grid and CV** as input. It tries all possible combinations of values passed in parameter grid and provides the optimal values by using **Cross – Validation method.** The optimal parameters that provide the best R^2 score and minimum MSE were finalized parameters for the respective regression models.

TYPES OF REGRESSION MODELS

There are many types of Regression models namely –

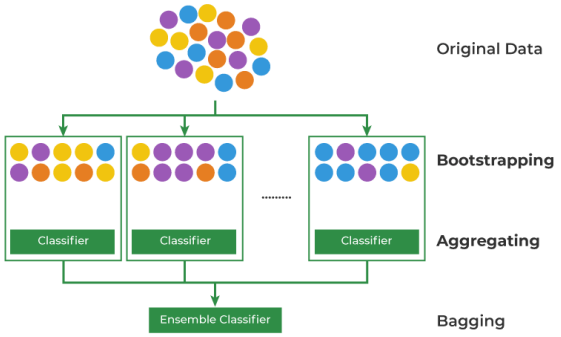
a) Linear Regression Models (Multiple, Ridge, Lasso, etc)
b) Non - Linear Regression Models (Support Vector Machine, K – Nearest Neighbours, Decision Trees, etc)
c) Ensemble Methods (Random Forest, XGBoost, CatBoost, Light GBM, etc)
d) Artificial Neural Networks (ANN).

Here, we will be covering in detail about **Random Forest**, **XGBoost**, **LightGBM, CatBoost** and the **ANN** Regression Models.

Random Forest Regression Model is an ensemble learning model that works by building multiple decision trees during training time. For classification, the majority class of the trees is chosen as the output and for regression, the mean or average of the prediction made by individual decision tree is returned as output.

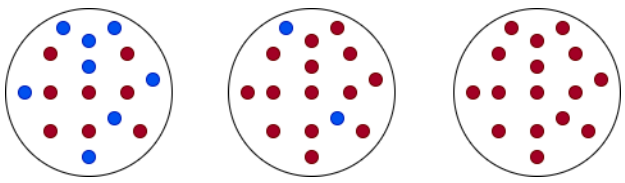
It uses a technique called **Bagging (also known as Bootstrap aggregating),** which is used for lowering variance in a noisy dataset.

In bagging, a random sample of data is picked with replacement from the training set, allowing each data point to be selected more than once. These weak models are then trained individually after generating several data samples. Depending on the job (regression or classification), the average or majority of those predictions produce a more accurate estimate.



(Figure 3 - Bagging)

The features in each decision tree are split in a specific manner. The feature that reduces the **entropy (randomness of a data)** the most after a split is chosen as the **root node** of the tree. The decrease in entropy after a split is known as **Information Gain.** It speaks for itself that a **larger gain** is preferred. The process is continued iteratively until the **leaf nodes** either reach homogeneity or come close to it.



(Figure 4 – Change in entropy after split)

Few **advantages** include – i) Reduces risk of overfitting, ii) Makes it easy to assess the significance of a feature, iii) Provides flexibility.

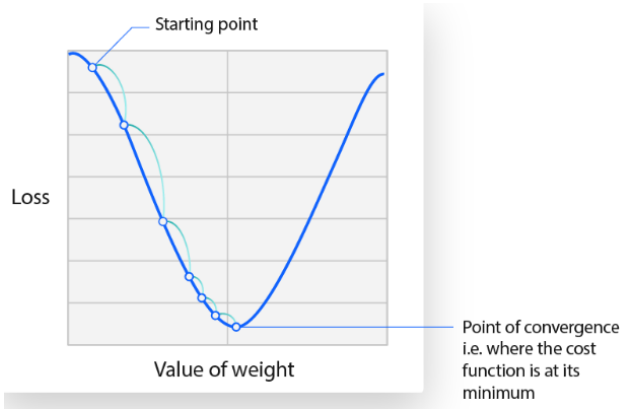
Few **disadvantages** include – i) Time – consuming process, ii) Requires more computer resources for computational power, and iii) Can get complex very easily.

Table 1 – Hyperparameters of Random Forest:

Hyperparameter	Typical Range (Small Dataset)	Typical Range (Medium Dataset)	Typical Range (Large Dataset)
n_estimators	[10, 100, 500]	[100, 500, 1000]	[1000, 5000, 10000]
max_depth	[None, 10, 20]	[None, 20, 30]	[None, 30, 50]
min_samples_split	[2, 5, 10]	[2, 5, 10]	[5, 10, 20]
min_samples_leaf	[1, 2, 4]	[1, 2, 4]	[2, 4, 8]
max_features	['auto', 0.2, 0.5]	['auto', 0.2, 0.5]	['auto', 0.5, 'sqrt']
bootstrap	[True, False]	[True, False]	[True]
criterion	['mse', 'mae']	['mse', 'mae']	['mse']
max_samples (if used)	[0.7, 0.8, 1.0]	[0.8, 1.0]	[1.0]
n_jobs	[-1, 1, 2]	[-1, 2, 4]	[-1, 4, 8]
verbose	[0, 1, 2]	[0, 1, 2]	[0]

XGBoost is a **gradient boosting algorithm** that is used to optimize a decision tree by minimizing the **objective function** that combines a **loss function** (the difference between predicted values and actual values) and a **regularization term** (L1 for Lasso and L2 for Ridge) for model complexity.

Boosting is a sequential process in which every new model aims to fix the mistakes made by the preceding one. The preceding model serves as a foundation for the subsequent versions. While the performance of each model may not be optimal across the full dataset, it is effective for specific regions of the dataset. Thus, each model boosts the performance of the previous model.



(Figure 5 – Gradient Descent Curve)

XGBoost has a wide range of hyperparameters that can be adjusted to optimize the performance of the model.

Table 2 – Hyperparameters of XGBoost

Hyperparameter	Typical Range (Small Dataset)	Typical Range (Medium Dataset)	Typical Range (Large Dataset)
`n_estimators`	[100, 500, 1000]	[500, 1000, 2000]	[1000, 2000, 5000]
`learning_rate`	[0.01, 0.05, 0.1]	[0.05, 0.1, 0.2]	[0.1, 0.2, 0.3]
`max_depth`	[3, 5, 7]	[5, 7, 9]	[7, 9, 11]
`min_child_weight`	[1, 3, 5]	[3, 5, 7]	[5, 7, 9]
`subsample`	[0.7, 0.8, 0.9]	[0.8, 0.9, 1.0]	[0.9, 1.0]
`colsample_bytree`	[0.7, 0.8, 0.9]	[0.8, 0.9, 1.0]	[0.9, 1.0]
`gamma`	[0, 0.1, 0.2]	[0.1, 0.2, 0.3]	[0.2, 0.3, 0.4]
`alpha` (L1 regularization)	[0, 0.1, 0.2]	[0.1, 0.2, 0.3]	[0.2, 0.3, 0.4]
`lambda` (L2 regularization)	[0, 0.1, 0.2]	[0.1, 0.2, 0.3]	[0.2, 0.3, 0.4]
`n_jobs`	[1, 2, 4]	[2, 4, 8]	[4, 8, 16]

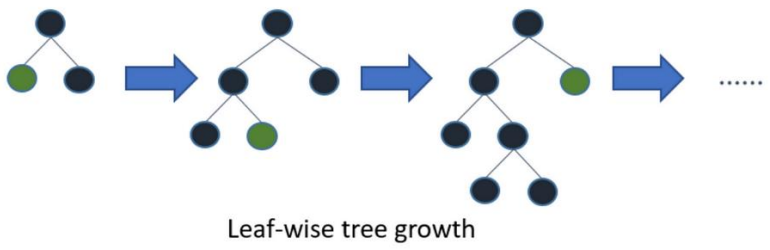
Some of the **advantages** of XGBoost are – i) It has built – in L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents over – fitting, ii) Utilizes parallel processing to reduce time to construct models, iii) It performs effective tree pruning, iv) provides built – in Cross Validation feature.

Light Gradient Boosting Machine (GBM) is a framework based on decision trees to increase the efficiency of the model and reduces memory usage. It uses two methods to do so which are **Gradient – Based One Side Sampling (GOSS)** and **Exclusive Feature Bundling (EFB).**

In **GOSS**, since the data instances (rows) with different gradients play different role in computing information gain, the instances with larger gradients will contribute more to the information gain. Therefore, to retain accuracy of the information, GOSS randomly drops the instances with small gradients and keeps the ones with larger gradients.

In **EFB,** the aim is to reduce the dimensionality of the data. Features like One -hot encoded features are the best examples of exclusive features. EFB improves efficiency while maintaining a high level of accuracy.

LightGBM creates decision trees that grow leaf wise, which means that given a condition, only a single leaf is split, depending on the gain. Leaf-wise trees can sometimes overfit especially with smaller datasets. Limiting the tree depth can help to avoid overfitting.



(Figure 6 - Leaf – wise tree growth)

Light GBM is known for its speed and efficiency. It is designed to handle large datasets and has a distributed computing framework, making it suitable for big data scenarios.

Table 3 – Hyperparameters of Light GBM

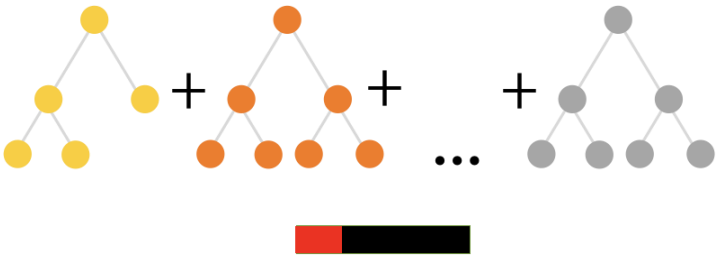
Hyperparameter	Typical Range (Small Dataset)	Typical Range (Medium Dataset)	Typical Range (Large Dataset)
`n_estimators`	[100, 500]	[500, 1000]	[1000, 5000]
`learning_rate`	[0.01, 0.1]	[0.001, 0.1]	[0.001, 0.1]
`max_depth`	[5, 10]	[5, 15]	[10, 20]
`num_leaves`	[10, 31]	[31, 63]	[63, 127]
`min_child_samples`	[10, 20]	[10, 20]	[10, 20]
`subsample`	[0.6, 0.8]	[0.6, 0.8]	[0.6, 0.8]
`colsample_bytree`	[0.6, 0.8]	[0.6, 0.8]	[0.6, 0.8]
`min_split_gain`	[0.0, 0.1]	[0.0, 0.1]	[0.0, 0.1]
`reg_alpha`	[0.0, 0.1]	[0.0, 0.1]	[0.0, 0.1]
`reg_lambda`	[0.0, 0.1]	[0.0, 0.1]	[0.0, 0.1]
`n_jobs`	[1]	[1, 4]	[4, 8]

CatBoost algorithm as the name suggests works with **categorical data** and uses **gradient boosting algorithm.**

Usual we require categorical data to be converted into numerical data to build the model. CatBoost on the other hand does the preprocessing as a part of its algorithm.

CatBoost uses a method called **ordered encoding** to encode categorical features. To determine a value to replace the categorical feature, ordered encoding considers the target statistics from every row before a data point.

Another unique characteristic of CatBoost is that it uses symmetric trees. This means that at every depth level, all the decision nodes use the same split condition.



(Figure 7 - Symmetric trees and a loss function)

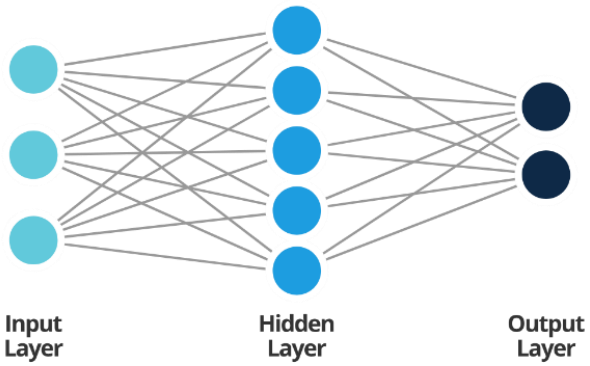
CatBoost provides built – in feature importance, making it easier to understand which features are contributing the most to the model's predictions. This is crucial for feature engineering and model interpretation.

Table 4 – Hyperparameters of CatBoost

Hyperparameter	Typical Range (Small Dataset)	Typical Range (Medium Dataset)	Typical Range (Large Dataset)
`iterations`	[50, 200]	[200, 500]	[500, 1000]
`learning_rate`	[0.01, 0.1]	[0.001, 0.1]	[0.001, 0.1]
`depth`	[4, 6]	[6, 10]	[8, 12]
`l2_leaf_reg`	[1, 3]	[3, 5]	[5, 10]
`border_count`	[32, 64]	[64, 128]	[128, 256]
`bagging_temperature`	[0.6, 0.8]	[0.6, 0.8]	[0.6, 0.8]
`min_data_in_leaf`	[5, 10]	[10, 20]	[20, 30]
`verbose`	[0]	[0, 1]	[1, 2]

Artificial Neural Network (ANN) is a subset of machine and is the base of deep learning, whose structure is inspired by the human brain, mimicking the working of the neurons.

It consists of **three important layers** which are – **input layer,** one or more **hidden layers and** an **output layer.**



(Figure 8 - Layers of a Neural Network)

Weights are assigned after an input layer has been identified. Larger weights contribute more heavily to the output than smaller ones, helping to establish the relative relevance of each variable. After that, each input is **multiplied** by its corresponding weight before being added together. The output is then determined by passing it through an **activation function**.

The **activation function** decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Some of the **commonly used activation functions** are – Binary Step, Rectifiers, Sigmoid, Hyperbolic Tangent, and more.

The node activates and sends data to the following layer of the network if the output surpasses a predetermined threshold. As a result, one node's output becomes the next node's input. This neural network is characterized as a feedforward network by the way that data is passed from one layer to the next.

In each step we find the **cost (loss) function (mean squared error, etc)** and the goal is to minimize it and keep it as low as possible. We use **gradient descent method** to find the minimal cost function value.

Finally, **Backpropagation** is used to move backward in direction, calculate and attribute error associated with each neuron and adjust and fit the parameters of the models appropriately.

Table 5 – Hyperparameters of ANN

Hyperparameter	Typical Range (Small Dataset)	Typical Range (Medium Dataset)	Typical Range (Large Dataset)
`epochs`	[50, 100]	[100, 500]	[500, 1000]
`batch_size`	[8, 32]	[32, 128]	[128, 256]
`learning_rate`	[0.01, 0.1]	[0.001, 0.01]	[0.001, 0.01]
`hidden_layers`	[1, 2]	[1, 2, 3]	[2, 3, 4]
`neurons_per_layer`	[16, 32]	[32, 64]	[64, 128]
`activation`	['relu', 'tanh']	['relu', 'tanh']	['relu', 'tanh']
`dropout_rate`	[0.2, 0.5]	[0.2, 0.5]	[0.2, 0.5]
`optimizer`	['adam', 'sgd']	['adam', 'sgd']	['adam', 'sgd']
`early_stopping`	[Yes/No]	[Yes/No]	[Yes/No]
`regularization`	[L1, L2]	[L1, L2]	[L1, L2]

ANNs can model complex, non-linear relationships in data. This allows them to learn and represent intricate patterns that may be challenging for traditional linear models.

After ANN training, the data may produce output even with incomplete information. The loss of performance here depends on the importance of the missing information.

ABOUT THE DATASET

A moderately exothermic reaction between carbon monoxide and steam, producing hydrogen and carbon dioxide is known as **Water Gas Shift Reaction (WGSR).** In most industrial applications, the WGSR is carried out in two process stages, a high temperature stage involving heating an iron – based catalysts to 320 and 450^{o} C and a low temperature stage involving heating copper – based catalysts to 150 and 250^{o} C.

$$CO + H_2O \rightleftharpoons CO_2 + H_2$$

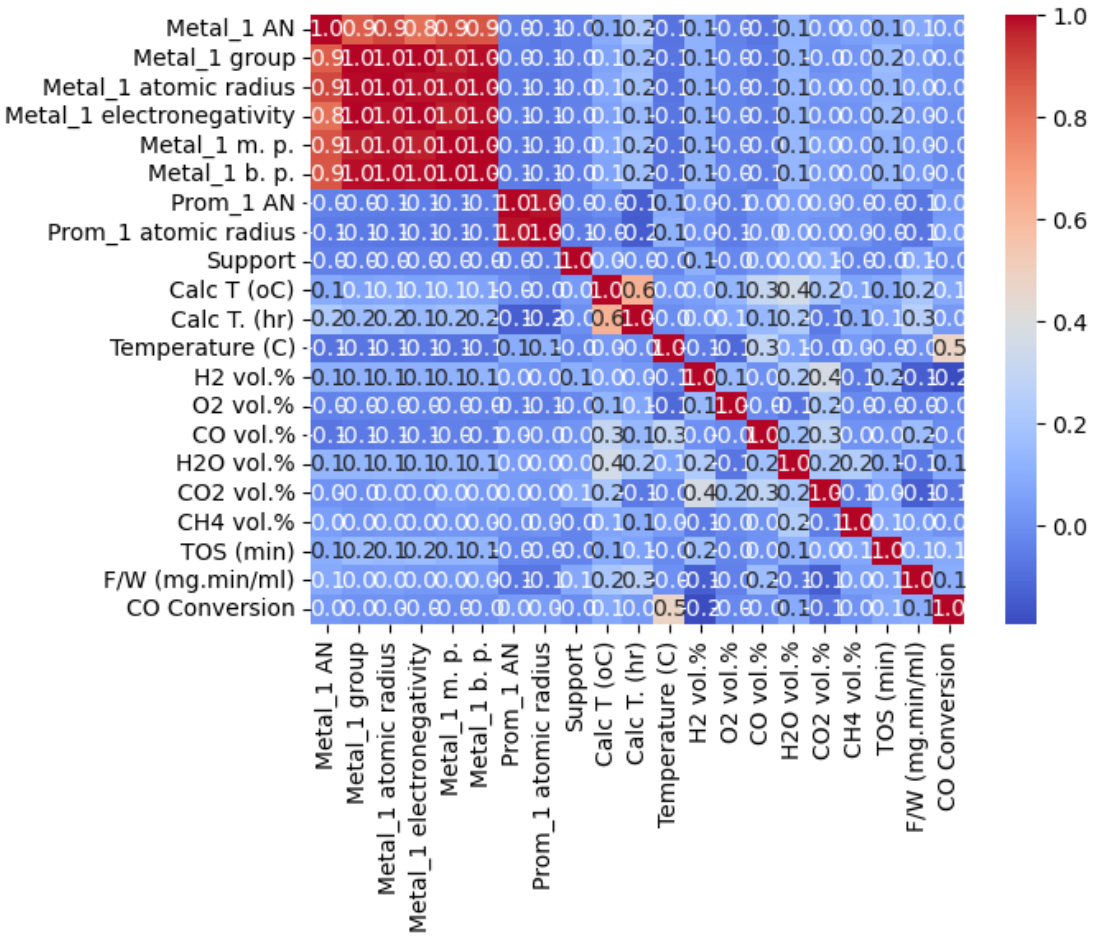
The WGSR is a highly valuable industrial reaction. It is used in manufacturing of ammonia, hydrocarbons, methanol, and hydrogen. Its most important application is in conjunction with the conversion of carbon monoxide from steam reforming of methane or other hydrocarbons in the production of high purity hydrogen.

The dataset initially contained 4185 rows with 8 duplicates which were removed using MS Excel features to get a total of 4177 rows and 21 columns.

The input ranges of the dataset are mentioned below:

INPUT FEATURE	MINIMUM	MAXIMUM
Metal_1 AN	0.92	2114.58
Metal_1 group	0.2	437.8
Metal_1 atomic radius	0.0388	57.71
Metal_1 electronegativity	0.027	69.65
Metal_1 m. p.	36.561	55342.354
Metal_1 b. p.	64.72	127360
Prom_1 AN	0	4564.6
Prom_1 atomic radius	0	145.595
Support	1	111
Calc T (oC)	1	700
Calc T. (hr)	0	10
Temperature (C)	4.78	809
H2 vol.%	0	60
O2 vol.%	0	1.358
CO vol.%	0.2	99.5
H2O vol.%	0	93.1
CO2 vol.%	0	24.84
CH4 vol.%	0	50.5
TOS (min)	0	5920
F/W (mg.min/ml)	0	40.1
CO Conversion	0	100

The correlation matrix of the all the variables is shown below:



(Figure 9)

The input variables were found to be **normally distributed** by plotting a Histogram and by running the variables through a function that checks for normality.

A **Standard Scaler function** was used to scale the input variables since they are normally distributed. It is a preprocessing technique that is used to scale the data by ensuring the mean is 0 and the standard deviation is 1. The formula is as follows:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

RESULTS AND DISCUSSIONS

The dataset was then split into training and testing datasets respectively and the models were built using their base hyperparameters. The split taken was 80 – 20 with 80 % for training and 20% for testing.

Table 6 – Comparison of R^2 scores and MSEs for base models built.

MODEL	R2 SCORE (%)	MSE
Random Forest	76.9588	238.9716
XGBoost	**81.8927**	**187.7579**
Light GBM	79.8292	209.5491
CatBoost	81.6321	190.4284
ANN	67.1259	357.0208

By analysing the above table, we can see that Gradient Boosted Regression Trees have performed good compared to Artificial Neural Network (ANN). Among these, the XGBoost gave the highest R^2 score and the least MSE. Conversely, the ANN model gave the least R^2 score and the highest MSE value.

The models were optimized by **hyperparameter tuning** using **GridSearchCV** method. The hyperparameters used were taken from the previously mentioned tables (Table 1, 2, 3, 4 and 5). They were further tuned manually to obtain the optimal hyperparameters for all the models.

Table 7 – Comparison of R^2 scores and MSEs of the models after optimization.

MODEL	R2 SCORE (%)	MSE
Random Forest	80.8892	197.6365
XGBoost	82.7692	177.9538
Light GBM	**82.8420**	**177.0934**
CatBoost	81.8875	187.2652
ANN	73.5458	287.2986

By analysing the above table, the **Light GBM model** gave the highest R^2 score of **82.842 %** after optimization. Light GBM also gave the least mean squared error of **177.0934** which made it the best model to choose to predict CO Conversion (%).

Optimal Parameters of all the models:

Random Forest –

{'criterion': 'poisson', 'max_depth': None, 'max_features': 0.2, 'max_samples': 1.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 326}

XGBoost –

{'colsample_bytree': 0.8, 'gamma': 0.125, 'learning_rate': 0.1, 'max_depth': None, 'min_child_weight': 5, 'n_estimators': 326, 'reg_alpha': 0.1, 'reg_lambda': 0.1, 'subsample': 0.9}

LightGBM –

{'colsample_bytree': 0.6, 'learning_rate': 0.1, 'max_depth': None, 'min_child_samples': 11, 'min_split_gain': 0.0, 'n_estimators': 328, 'num_leaves': 32, 'reg_alpha': 0.1, 'reg_lambda': 0.1, 'subsample': 0.4}

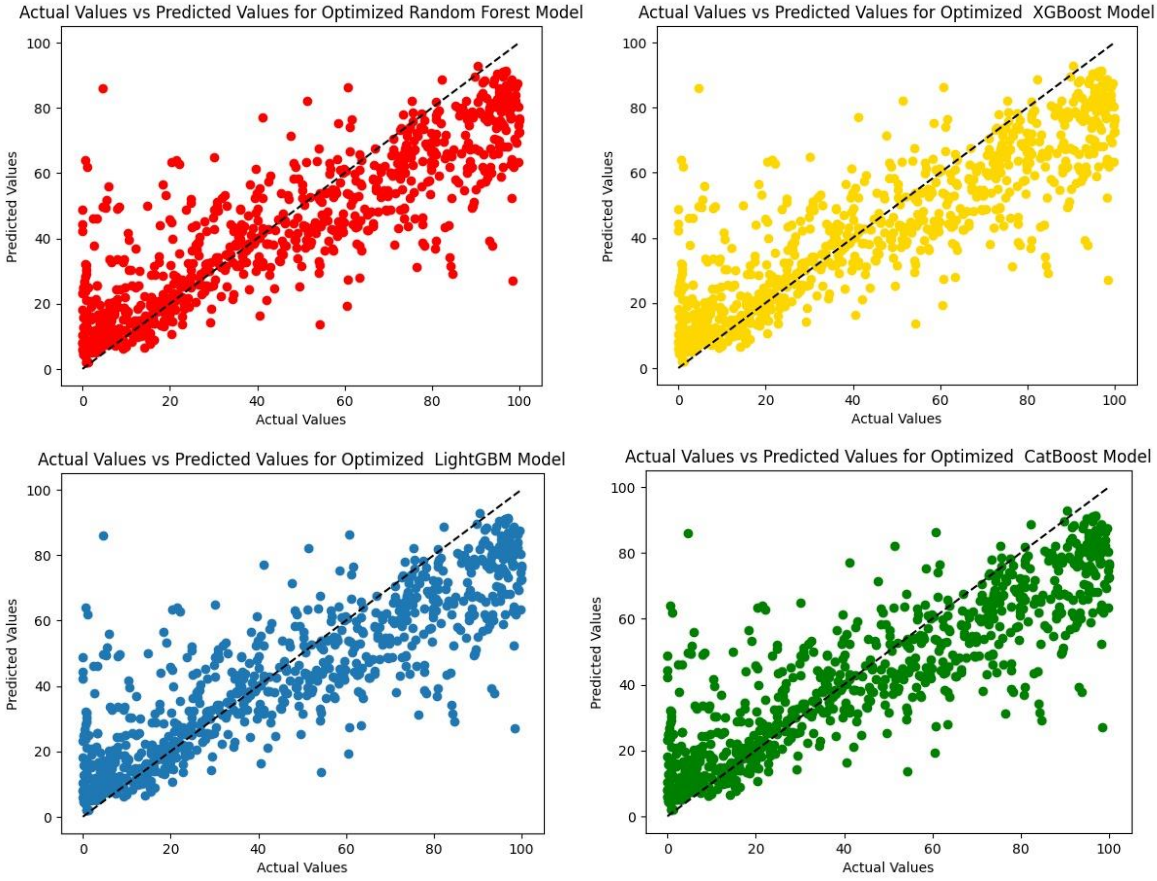
CatBoost –

{'bagging_temperature': 0.6, 'border_count': 64, 'iterations': 350, 'l2_leaf_reg': 1, 'learning_rate': 0.1, 'min_data_in_leaf': 4}

ANN –

{'learning_rate': 0.05, 'epochs': 200, 'batch_size': 32, 'optimizer': adam, 'activation': ['relu', 'tanh']}

The actual vs predicted CO Conversion (%) and a line representing the ideal were plotted on a scatter plot to visualize the models:



CONCLUSION

This project aimed to predict the percentage of CO conversion in the water gas shift reaction using a variety of regression models, including XGBoost, Random Forest, LightGBM, CatBoost, and Artificial Neural Networks (ANN). Through comprehensive experimentation, it was observed that LightGBM outperformed the other models, yielding the highest R2 score of 82.84%.

The success of LightGBM in predicting CO conversion underscores its effectiveness in capturing the complex relationships within the dataset. The high R2 score indicates a strong correlation between the predicted and actual values, showcasing the model's robustness in explaining the variance in CO conversion.

The findings suggest that ensemble methods like LightGBM are particularly well-suited for this prediction task, likely due to their ability to handle non-linear relationships and interactions among various features influencing the water gas shift reaction. The importance of feature engineering and model tuning, especially in the context of boosting algorithms, was highlighted during the model development process.

While LightGBM emerged as the top performer, the exploration of multiple models, including XGBoost, Random Forest, CatBoost, and ANN, provided valuable insights into their respective strengths and weaknesses. The comparative analysis (Tables 6 & 7) contributes to a more comprehensive understanding of the predictive capabilities of different regression techniques in the context of the water gas shift reaction.

As with any scientific endeavor, it is essential to acknowledge the limitations and potential areas for improvement. Future work could involve exploring additional features, refining model hyperparameters, and incorporating domain-specific insights to further enhance prediction accuracy.

In conclusion, the successful application of LightGBM in predicting CO conversion in the water gas shift reaction reaffirms the significance of employing advanced regression models in chemical reaction prediction. The insights gained from this project contribute to the growing body of knowledge in the field of machine learning applications in chemistry and pave the way for more sophisticated predictive models in the optimization of industrial processes.