

**Contents**

[1 Chapter :01 INTRODUCTION 1](#_Toc73145671)

[1.1 Introduction 1](#_Toc73145672)

[1.2 OBJECTIVES 4](#_Toc73145673)

[1.3 MOTIVATION 4](#_Toc73145674)

[2 Chapter :02 LITERATURE REVIEW 5](#_Toc73145675)

[2.1 Introduction 5](#_Toc73145676)

[2.2 Predicting fire temperature inside compartment 6](#_Toc73145677)

[2.3 Carrying out experimental tests 9](#_Toc73145678)

[2.4 Evacuation models for fire in structures 11](#_Toc73145679)

[2.5 Fire risk models 12](#_Toc73145680)

[2.6 Structural fire resistance 13](#_Toc73145681)

[2.7 Spalling in concrete 15](#_Toc73145682)

[2.8 Structural reliability assessment 16](#_Toc73145683)

[2.9 Active Fire Protection 17](#_Toc73145684)

[3 Chapter: 03 Development of Surrogate Model 21](#_Toc73145685)

[3.1 Introduction 21](#_Toc73145686)

[3.2 Probabilistic Fire Analysis 21](#_Toc73145687)

[3.3 Development of Regression based basic-Surrogate Model. 22](#_Toc73145688)

[3.4 Advanced Non-linear Model: 26](#_Toc73145689)

[3.4.1 Dataset 26](#_Toc73145690)

[3.5 Feature Analysis 29](#_Toc73145691)

[3.5.1 Pearson-based correlation analysis 30](#_Toc73145692)

[3.5.2 Spearman-based correlation analysis 32](#_Toc73145693)

[3.5.3 Kendall-based correlation analysis 33](#_Toc73145694)

[3.6 Best Regression Model Selection 34](#_Toc73145695)

[3.6.1 Data Preprocessing 34](#_Toc73145696)

[3.6.2 Regression Model Selection 35](#_Toc73145697)

[3.6.3 Light Gradient Boosting Regression Algorithm training and hyperparameter tunning 43](#_Toc73145698)

[3.6.4 Feature Importance analysis 49](#_Toc73145699)

[4 Chapter :04 Discussion and Conclusions 54](#_Toc73145700)

[4.1 Introduction 54](#_Toc73145701)

[4.2 Discussion 54](#_Toc73145702)

[4.3 Conclusion 55](#_Toc73145703)

[APPENDIX 57](#_Toc73145704)

[REFERENCE 94](#_Toc73145705)

**List of Figures**

[Figure 1‑1:ML Types And Some Commonly Use ML Algorithm [1] 2](#_Toc73147353)

[Figure 1‑2:Engineering Paradigm [10]. 3](#_Toc73147354)

[Figure 2‑1:Body of Engineering Knowledge [5] 6](#_Toc73147355)

[Figure 2‑2:Schematic of TCNN architecture. 8](#_Toc73147356)

[Figure 2‑3:Histogram of RMSD 9](#_Toc73147357)

[Figure 2‑4 TCNN Prediction of velocity and temperature for x-axis to CFD Simulation 9](#_Toc73147358)

[Figure 2‑6:Decision Boundary which separate flashover (Y=1) From non-flashover(Y=0). 11](#_Toc73147359)

[Figure 2‑7: Analytical Procedure 12](#_Toc73147360)

[Figure 2‑8 16](#_Toc73147361)

[Figure 2‑9:Framework for probabilistic studies of fire exposed structures based on surrogate modelling methodology. 17](#_Toc73147362)

[Figure 2‑10:PDF and CDF for the concrete column based on actual and surrogate model. 17](#_Toc73147363)

[Figure 2‑11:The main application of a proposed fire monitoring system. 20](#_Toc73147364)

[Figure 2‑12:Overall fire detection procedures of the proposed algorithm. 20](#_Toc73147365)

[Figure 3‑1: X-axis: order of polynomial and Y-axis: Difference of mean square error 24](#_Toc73147366)

[Figure 3‑2:Effect of Regularization Parameter 25](#_Toc73147367)

[Figure 3‑3:X-axis: number of sample points. Y-axis: error of cost function 26](#_Toc73147368)

[Figure 3‑4:Left side: Publish study result, Right Side: This study result. 26](#_Toc73147369)

[Figure 3‑5 (a) Null Value analysis for training data; (b) Null Value analysis for testing data 28](#_Toc73147370)

[Figure 3‑6 Distribution Analysis for Training Data 28](#_Toc73147371)

[Figure 3‑7 Distribution analysis for testing data 29](#_Toc73147372)

[Figure 3‑8 Outlier Analysis 30](#_Toc73147373)

[Figure 3‑9 Pearson Correlation Analysis 31](#_Toc73147374)

[Figure 3‑10 Pearson Correlation between features and prediction variable 31](#_Toc73147375)

[Figure 3‑11 Spearman Correlation 32](#_Toc73147376)

[Figure 3‑12 Spearman Correlation between features and prediction variable 32](#_Toc73147377)

[Figure 3‑13 Kendall Correlation 33](#_Toc73147378)

[Figure 3‑14 Kendall Correlation between features and prediction variable 34](#_Toc73147379)

[Figure 3‑15 Distribution for standardized data 35](#_Toc73147380)

[Figure 3‑16 Multi-Linear Regression prediction vs true value plot 37](#_Toc73147381)

[Figure 3‑17 XGBOOST Regression prediction vs true value plot 37](#_Toc73147382)

[Figure 3‑18 Random Forest Regression prediction vs true value plot 38](#_Toc73147383)

[Figure 3‑19 MLP Regression prediction vs true value plot 38](#_Toc73147384)

[Figure 3‑20 Linear Support Vector Regression prediction vs true value plot 39](#_Toc73147385)

[Figure 3‑21 Polynomial Regression (degree=3) prediction vs true value plot 39](#_Toc73147386)

[Figure 3‑22 RBF Support vector regression prediction vs true value plot 40](#_Toc73147387)

[Figure 3‑23 Sigmoid Support vector regression prediction vs true value plot 40](#_Toc73147388)

[Figure 3‑24 K-Nearest Neighbors regression prediction vs true value plot 41](#_Toc73147389)

[Figure 3‑25 Polynomial regression (degree=2) prediction vs true value plot 41](#_Toc73147390)

[Figure 3‑26 Polynomial regression (degree=3) prediction vs true value plot 42](#_Toc73147391)

[Figure 3‑27 Polynomial regression (degree=4) prediction vs true value plot 42](#_Toc73147392)

[Figure 3‑28 Light gradient boosting prediction vs true value plot 43](#_Toc73147393)

[Figure 3‑29 LGBM learning rate vs loss plot 48](#_Toc73147394)

[Figure 3‑30 Number of estimators vs Loss (RMSE) plot 48](#_Toc73147395)

[Figure 3‑31 Feature importance vs feature plot for GAIN 49](#_Toc73147396)

[Figure 3‑32 Feature importance vs feature plot for Split 50](#_Toc73147397)

[Figure 3‑33 Feature Impact on model output analysis using shap value 50](#_Toc73147398)

[Figure 3‑34 nodline\_Y feature dependency plot 51](#_Toc73147399)

[Figure 3‑35 oos feature variable dependency plot 51](#_Toc73147400)

[Figure 3‑36 epskfy feature variable dependency plot 52](#_Toc73147401)

[Figure 3‑37 epskfc feature variable dependency plot 52](#_Toc73147402)

[Figure 3‑38 cover feature dependency plot 52](#_Toc73147403)

[Figure 3‑39 time feature dependency plot 53](#_Toc73147404)

**ABSTRACT**

Machine learning (ML) has been evolved significantly over recent year. ML method has been applied in a variety of disciplines of engineering and science.ML improves the role of data science by learning high level abstractions from data.ML method offers advantages to handle large number of variables of a complex problem and provide computational efficiency. The success of integrating machine learning to tackle complex problem enables ML method as an advance state-of-the - art tool to solve complex scientific problem. This study reviews the application of machine learning in fire safety engineering and analysis the efficacy and appropriateness of the applied method. Second, a surrogate model has redeveloped from a published case study to analysis the potential of surrogate modeling method. Finally feature analysis of a given problem through machine learning method has evaluated.

# Chapter :01 INTRODUCTION

## Introduction

Fire is a second most complex phenomena after life science to understand. The progress of machine learning (ML) shows to solve such complex problem in other parallel field. Recent years ML method evolved rapidly to enhance the role of data science to handle complex problem, provide computational efficiency and facilitate decision making. In 1959, Samuel defines ML is a field of study that gives computer the ability to learn without being explicitly programmed. The classical ML consists of Supervised and unsupervised learning method. The supervised learning method use prior information as a label to map the input of a data set to approximate the relationship between input and output data set. In contrast, Unsupervised ML method infer the hidden pattern in a data set without being using prior information. The ML method primarily depends upon data characteristics and goal of the problem. Supervised ML method can be further subdivided into classification and regression task, while unsupervised learning comprises clustering and dimensionality reduction. Figure 1 summarizes the two types of ML and some commonly used ML algorithms.

The advance method of ML encompasses new sets of computational techniques that heavily rely on machines to mimic human -like realization such as generalization, discovery, association, and abstraction. Such advance ability of ML methods able to solve problem in which there is a large amount of variable and there is unclear relationship between variables and expected outcome. This is a common problem in various branches of engineering. To find such relationship ML method use revolutionary algorithm to predict concealed patterns in a data set. Once a pattern is discovered, this pattern is further exploited to identify its governing parameters [2].

Diagram

Description automatically generated

Figure 1‑1:ML Types And Some Commonly Use ML Algorithm [1]

Industry 4.0 is the common name used to describe the current trend towards a fully connected and automated system. As a result, large volume of data has produced and stored to further analyzed and model to predict the answer of a complex problem. Big data analysis has enabled to examine huge data sets to make associations and spot patterns and trends. We stand on the brink of a technology revolution that will fundamentally alter the way we live, work, and relate to one another. The fourth industrial revolution will shift scientific method from classical paradigm to an advance fourth paradigm: fusion of data driven method [3]. The figure 2 illustrate how an engineering problem is solved from classical method to a current state of the art method. However, research in fire safety engineering (FSE) continues to primarily favor classical experimental, analytical, and numerical methods, As a result a little room for innovation or flexibility [2].

To perform a fire engineering experiment at elevated temperature, require a specialized and sophisticated equipment that most researchers and engineers do not have access to [4]. Integrating ML method in fire safety engineering open up a new oppournity to recycle and maximize finding of past fire test [4].

A picture containing application

Description automatically generated

Figure 1‑2:Engineering Paradigm [10].

As fire experiment is expensive and virtually impossible to design a comprehensive testing program that explore all possible parameter that governing a phenomenon. Similarly developing a numerical model require high level mathematics and computational resources. Furthermore, a true input parameter requires for numerical model which is hard to obtain, as a result fire engineers relay on a holistic approach to assume such input parameters. While a doubt rises on the true adequacy of developed numerical model. It shows a dire need to develop a modern fire assessment tool. Hence The potential of leveraging ML method as a modern fire assessment tool is available. It is expected that ML approaches in fire safety engineering envisioned to supplement fire research to accurately trace fire behavior, fire response of material, fire performance of the structure and in developing new theories and generating new knowledge [5].

## OBJECTIVES

Machine learning (surrogate modelling) approaches are increasingly referenced in fire safety literature. The state-of-the-art is however unclear, both with respect to the range of applications, as well as with respect to the efficacy and appropriateness of the applied methods. This study will investigate the available literature, the performance of published machine learning approaches in fire safety engineering, recalculate the published method and a critical analysis is made for the state of the art machine learning method.

## MOTIVATION

The main motivation of this study is the industrial 4. revolution. The industry 4 revolution will improve industry manufacturing capability and bring more automaton in the field of Engineering. As a result, data is produced. If produce data process effectively will provide valuable insight from the produced data set. Furthermore, the field of artificial intelligence rapidly developed due to availability of hardware, open-source software, and dataset. This development introduces new innovative technology tool such as deep learning and generative modeling. As a result, more sophisticated algorithm has developed to solve complex problem which is difficult to solve with current computation modelling method. Furthermore, Recent years shows number of publications related to application of machine learning in the field of fire safety engineering which motivate to investigate the current stat of the art of machine learning application in the field of FSE.

# Chapter :02 LITERATURE REVIEW

## Introduction

This section describes the application of machine learning approaches in fire safety engineering (FSE). The ML approaches in Fire safety engineering (FSE) have applied in various domains of FSE. The aim of this section is to demystify how a machine learning method can help to solved complex and difficult problem of FSE. In recent years, The ML algorithm used theory base model to solve any physics-based problem.

This year M.Z. Naser [4], published an article mechanistically informed machine learning and artificial intelligence in fire safety journal. This paper shows the current state of art machine learning methods used to solved FSE problems. This is the only paper available during this research which specify state of the art of machine learning approaches in fire safety engineering. However, there are number of articles published for a specific application of ML method to a specific problem of FSE such as detection of fire through image and video and fire resistance of structure.

Fire safety engineering is an interdisciplinary engineering branch described as in the body of engineering knowledge diagram [5] figure 3. Within each subfield of fire safety engineering, ML algorithms have been implemented to solve numerous FSE problems. The primary search journal for this study is fire safety journal which consist of 25 relevant publications since 1990. However Different literatures database such as google scholar, direct science etc was used to search the publication that use machine learning method to solve FSE Problems. The search result indicates that nearly 50 relevant publications are now available. This shows a clear growth of application of ML method in FSE field. The previous publication can be categorized according to the application in a subfield of FSE and ML method. The ML methods and their areas of applications in FSE are investigated to understand the current state of the field. The following subsections discuss briefly how a specific problem of fire safety engineering has been solved by machine learning approaches.

Diagram, venn diagram

Description automatically generated

Figure 2‑1:Body of Engineering Knowledge [5]

## Predicting fire temperature inside compartment

The current-state-of-the-art technique such as fire dynamic simulator (FDS) is used in fire protection engineering to predict complex flow fields inside a compartment. The prediction of flow field has been used for designing a smoke and heat control system or to predict sprinkler activation time. FDS solves the conservation equation of mass, momentum, and energy. FDS model highly depended on true input properties of model variable which is hard to obtained. fire dynamics are highly nonlinear phenomena and the CFD technique is cost-prohibitive and often required a large time to predict flow filed in case of a large structure or parametric study. To remove such limitations of the CFD technique another advanced method has been introduced such as the machine learning technique which enabled scientists to develop a data-driven model to predict complex flow field [6] [7] [8][9].

One such application of machine learning to predict the complex flow fields inside a compartment can be found in [10], where data-driven generative modeling model is used. [11]. This study used a generative machine learning modeling method to predict spatially resolved temperature and velocity in a compartment. Generative modeling generates RGB images and consists of two learning tasks. In the first learning task, a model is used to calculate the optimal vector to describe input data. The second learning task is known as a generator to produce a detailed description from the input data. The proposed method used a transpose convolutional neural network (TCNN) as an architecture to generate RGB images depicted in figure 4. The proposed solution schematic is shown in figure 5. The initial prediction of the fire scene is made by the zone fire model. This prediction and dimension of geometry are used as an input for TCNN architecture. The TCNN used this prior training (input) to predict the thermal flow field. Finally, this prediction is post-processed for the final output prediction of compartment temperature and flow velocity. The accuracy of the result is measured in the form of root mean square difference (RMSD). The RMSD range from a perfect match at 0 and 100 for an opposite match. The result of all tests sampled is shown in the form of a histogram in figure 2.the mean and standard deviation of RMSD is 9.8 and 3.0, respectively. The prediction from the generative neural network for temperature and velocity of a compartment fire is shown in figure 7 with the comparison from the CFD result. The proposed model predicted 95 % accurate test velocity and temperature.

Same researcher has published another study [11] for rapid high fidelity prediction of fire to support fire hazard assessment. This study used another type of machine learning algorithm such as dimensionality reduction (reduce order model) and deep learning with neural network. The result of this study shows that machine learning can provide full field prediction 2 – 3 order of magnitude faster than CFD simulation .likewise another study has published [12] for prediction of velocity and temperature profile in a single compartment fire with improved neural network model. The main feature of the study is to handle noisy data and multi-dimensional prediction problem. This study shows the first application of ANN model for fire studies.

In a parallel field such as fluid mechanics [6] machine learning has been used with large amount of data and image analysis for physics-based application. The preliminary result produce promising result and shows faster results than conventional CFD simulation time. Furthermore, analysis of different machine learning algorithm for prediction of flow field generated by fire revealed that deep learning algorithm which is massively parallel equation [13] produce promising result with less mathematics while other technique used more mathematics such as dimensionality reduction technique .The recent advancement for the solution of partial differential equation [14][15][16][17][18][19] by machine learning algorithm is a new frontier for researcher . The future study of the author will focus for the solution of low Mach number partial differential equation of fire dynamics which is identified as a current research gape.

Diagram

Description automatically generated

Figure 2‑2:Schematic of TCNN architecture.

Chart, histogram

Description automatically generated

Figure 2‑3:Histogram of RMSD

Diagram

Description automatically generated

Figure 2‑4 TCNN Prediction of velocity and temperature for x-axis to CFD Simulation

## Carrying out experimental tests

Fire classification of material is one of the important aspects to ensure safe building design. The classical method to classified building material depends upon fire testing methods such as full-scale fire testing or single burning item (SBI) test to analyze fire reaction to a specimen. while Fire testing is an expensive and labor-intensive task. An alternative method is proposed by the researcher such as reduced small-scale testing. But to justify scale methodology, a detailed understanding of fire behavior is required. Therefore, a European intermediate fire testing method has been introduced for modern construction material such as sandwich panels etc. The derived correlation from reduced scale testing has proved to be less accurate so the dependency on full-scale testing remains to accurately classified modern construction material. The previous researcher [20] highlights that there is a need for a tool to identify experimental configuration and discern relevant parameters for experimental testing to reduce the time and cost of research. one possibility to develop a tool that can aid in ongoing experimentation is a machine learning algorithm. Machine learning algorithm which uses data and experience instead of the rigidly prescribed equation to predict a continuous outcome or classification of data. An interesting research [21] used logistic regression to predict flashover or non-flashover of a specimen (sandwich panel). A detailed description of that research could be found in [1]. Here a brief procedure is mentioned: This research used a 1/5 scale enclosure constructed with a sandwich panel (ISO-13784-1). The first-order linear regression model divides the data point into flashover and non-flashover space by an estimated decision boundary as depicted in figure 2. The value of Y equal to one represents flashover and zero is used to represent the non-flashover result of the specimen. A cost function is used to calculate the error loss between the predicted regression coefficient and training set data. The loss function will update the regression equation coefficient. The proposed machine learning model performance is evaluated through an R2 metric. The cross-validation of the proposed model performance produces R2 = 0.91 which is an improved result after removing bias and variance through a regularized logistic regression. Hence such method could be used by anyone as a screening tool to find out the output of a large or intermediate fire testing experiment. The accuracy of using such method highly depend upon the accuracy of experimental fire testing work and quality of the observed data point.

Chart, scatter chart

Description automatically generated

Figure 2‑5:Decision Boundary which separate flashover (Y=1) From non-flashover(Y=0).

## Evacuation models for fire in structures

Evacuation processes depend upon the measurement of the dynamic movement of evacuees. The stepwise movement is the fundamental recognized process to measure evacuation movement patterns. However, influencing factors for stepwise movements are still not well understood. Research has been published to explore the potential of adopting machine learning methods to study evacuee’s stepwise movement dynamics [22]. The proposed method categorized movement patterns through the two-step cluster method and principal influencing factors were identified through the principal component analysis technique. The proposed method is described in figure 9. Firstly, The movement-related variable is classified through the cluster machine learning method. Second, the Principle component technique is used to identify independent movement variables and discarding other irrelevant variables, and merging interdependent variables. Two quasi videos for emergency evacuation has been used to investigate the relationship between movement pattern and influencing factors through a traditional method such as multinomial logit model and a couple of machine learning model (Decision tree, SVM, K-Nearest neighbor, ANN). The result of two experiment video reveals that distance to the target exit has been a most profound factor for stepwise movement and surrounding action also have a significant effect on stepwise movement procedure. The traditional method well predicts simple scenarios and the machine learning method well predicts in case of a complex scenario. More details could be found in [22]. Furthermore, a study has been published [23] to analysis pre evacuation behavior of people and quantitively investigate pre evacuation behavior through support vector machine. Likewise another study [24] investigate through understanding of pre human behavior with Fuzzy logic and adaptive network.

Diagram

Description automatically generated

Figure 2‑6: Analytical Procedure

## Fire risk models

The fire risk assessment is a procedure to identify potential fire hazards. There are two major fire risk assessment methods such as qualitative and quantitative method. The qualitative method is based on the checklist method and the quantitative method used statistics of fire accidents to find the probability of a potential fire hazard. In other words, the simplest way to assess the fire safety level of an existing building is to check the fire safety features against standards prescribed in the fire codes. However traditional method for fire risk assessment is cumbersome and difficult to produce because of the complicated fire risk assessment process. Therefore, scientists used a new method based on machine learning algorithms such as fuzzy model and support vector techniques for fire risk assessment. The proposed method [25] considers comprehensive risk factors and establishes an index system. Another study [26] combined genetic and neural networks to analyze dwelling fire occurrence in the united kingdom (UK). Similarly, another study [24] for fire risk assessment of the tall building in Hongkong has been done through the machine learning method. An interesting study has been published [27] last year for identifying community fire hazard of carpark ,shopping center and public spaces and based on text input provided by community members. Likewise another interesting study [28] has been published to identify vulnerability of bridges to fire hazards

## Structural fire resistance

The main goal of structural Fire engineering is to calculate fire resistance. The conventional method is based upon a fire testing of element. The fire test is terminated once the fire weakened test sample exceeds a failure limit state. This point in time when a structural element fails is referred to as fire resistance. Results from fire tests were then combined into tables and then used to derive correlation equations that can estimate the fire resistance of samples. Another approach to calculating fire resistance of the structure is finite element (FE) analysis. The FE method has surged due to the growing complexity of fire tests and the lack of testing facilities. But the lack of proper validation and standardization of the solution process continues to hinder the application and acceptance of FE solution. An advanced method has been proposed by the researcher based on a machine learning algorithm. There are series of paper has been published to demonstrate the adopting AI technology to solve complex fire resistance problem. For example : In[29], an AI-based cognition framework has been developed to accurately tracking the response of the concrete structure under elevated temperature. This framework is utilized to derive a simple expression that allows evaluation of thermal and structural response of reinforced concrete (RC) beams and columns at a specific point in time or tracing time-temperature deformation history. The developed AI-based framework successfully comprehends the naturally complex behavior of fire exposed RC structure members. The AI method exploits relationships between key response parameters which were not fully incorporated into structural fire engineering problems. The published framework is developed using a hybrid combination of artificial neural networks (ANN) together with symbolic regression and genetic algorithms. The proposed method identifies critical parameters from engineering judgment and recommendations from published research articles. The developed database of the critical parameter has been used as an input into a neural network and then processed through a genetic algorithm. The candidate solution is derived using symbolic regressions. The obtained expression shows a unique relation between input parameters and output. The result shows that the maximum prediction error between the developed proposed method and experiment result was less than 15 %.Same researcher published another study [30] to calculate fire resistance of timber structure through artificial intelligence. The same procedure has been adopted such as hybrid combination of genetic algorithm and symbolic regression to derive simple expression for temperature dependent material equation for timber wood. The coefficient of determination of such derive expression is 0.99 which indicate a good accuracy of proposed method. Likewise same researcher has published another study [31] for calculating properties of modern construction material such as high strength concrete/steel and fiber reinforced polymer composites with the same proposed method express in earlier studies. The artificial neural network (ANN) and evolutionary genetic algorithm (GAs) is an advance machine learning algorithm if properly used and train then produce very good accuracy as mentioned in reference studies.

Furthermore there are series of paper [32] [33][34][31][35][36][37] published by different researcher to explore the potential of machine learning to solve structural fire engineering problem. It is inferring that artificial neural network (ANN) is an advance algorithm used in the publish work and produce accurate result.

## Spalling in concrete

Concrete is a naturally occurring inert material commonly used in the built environment. The concrete has low thermal conductivity, high specific heat and slow degradation of concrete strength make it superior to use under elevated temperature conditions (Fire Condition). However concrete shows physio-chemical changes at elevated temperature. As a result, an increase in high pore pressure and thermal gradient often leads to concrete spalling in a piece of chunk driven by fire. This exception behavior of concrete shows a complex and randomness of fire-induced concrete spalling. Currently no analytical method exist that can predict accurate prediction of fire-induced concrete spalling. Recent advancements in computing power and improve machine learning methods helped to develop a method for understanding fire induced spalling phenomena. The research [38] used machine cognition (MC), a branch of machine intelligence to derive an expression to calculate intensity of fire induced concrete spalling. The derived expression considers geometric, material, and specific properties related to reinforcement concrete (RC). The proposed method attempts to mimic human like reasoning process to solve such complex problem that may not be solved by traditional methods. The proposed method utilized evolutionary algorithms such as genetic programming to learn patterns hidden in random points by carrying out systematic analysis. Once a pattern is discovered, this pattern is used to solve the problem through training and adaptive learning. Figure 2 shows a flow chart of proposed method. The result of study shows derive expression for major, minor and no spalling phenomena can be calculate and further detail can be found in [38].Same researcher has published another study [39] for fire induced spalling through ensemble machine learning and surrogate modelling to investigate valuable insights of fire induced spalling.

Diagram

Description automatically generated

Figure 2‑7

## Structural reliability assessment

An explicit fire risk profile of the structure should be calculated to demonstrate adequate fire safety. The current approach follows a probabilistic method to calculate the fire risk profile of the structure. However, the current method is laborious and computationally expensive. An alternative method to calculate the explicit fire risk profile of structure is through surrogate modeling as demonstrated in [40]. The study used a machine-learning algorithm to mimic high fidelity simulation to construct a fragility curve of fire exposed RC column. The research used the polynomial regression-based surrogate model to predict the response of fire exposed structure. The surrogate model used input from the result of a SAFIR model. A gradient descent-based optimization technique is used to minimize the cost function of the surrogate model. The proposed framework for probabilistic studies of fire exposed structures based on surrogate modeling methodology is shown in figure 11. The result of the proof of concept of this study is depicted in figure 12. The result shows a comparison of PDF and CDF obtained from surrogate modeling and high fidelity SAFIR model. The results show that the predicted mean load capacity is quite close with Monte Carlo simulation and the proposed surrogate model. The predicted error is observed less than 5%.

Diagram

Description automatically generated

Figure 2‑8:Framework for probabilistic studies of fire exposed structures based on surrogate modelling methodology.

Chart, line chart

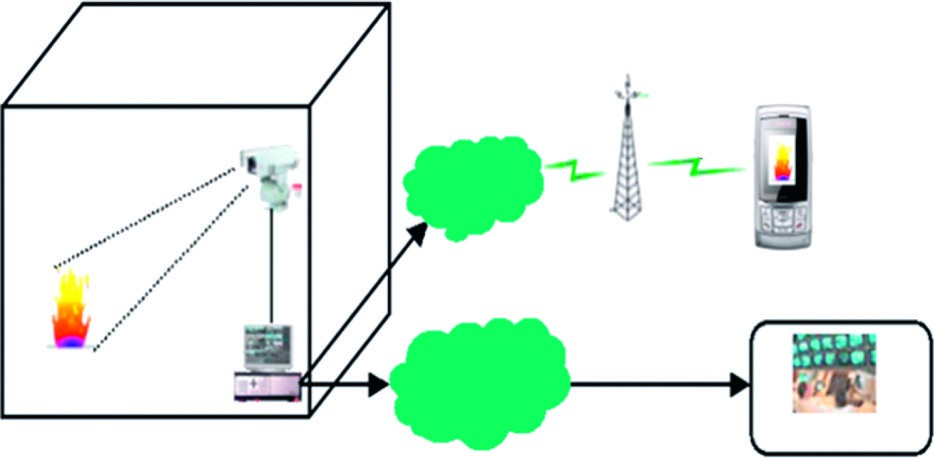
Description automatically generated

Figure 2‑9:PDF and CDF for the concrete column based on actual and surrogate model.

## Active Fire Protection

The active fire protection detection domain consists of detectors to detect fire. Currently, there are multiple types of detectors available such as smoke, heat, and infrared fire detector. These detector works on the principle of detecting physical substances produce from the fire such as smoke, heat, and light using ionization or photometry mechanism. But there are some limitations with respect to the environment to use these detectors. The most common problem with the detector is the false alarm signal and delay fire alarm. As a result, fire spread and cause severe damage with respect to property loss or human loss injury. There are number of research has been conducted to solve false alarm prediction and improve early fire detection based on different algorithm. Here we will discuss briefly most advance model for fire detection which used state of art technology such as computer vision algorithms. A vision-based sensor [41] has been proposed to remove the false alarm and improve early fire detection. The proposed vision-based system is described in figure 13. It is installed in a home-based server system to monitor a room or indoor space by analyzing a sequence of videos from a CCD camera. If a fire is detected, then the system sounds an alarm and sends an image sequence of fire to both user's cellular phone and fire station to check whether it is an actual fire or not. The overall procedure to detect fire is shown in figure 14. The proposed method first detects the fire region through a moving region or fire-colored pixels. Then, two additional methods are applied to fire pixels such as a luminous map and support vector machine. The luminous map is used to remove non-fire images and SVM is used to verify fire pixels. The effectiveness of the proposed method has been checked and validated with the previously published work. The result shows an average detection rate of 86.5% compared with the previously performed study rate of 71.3% more details can be found in [41].

Moreover The preliminary study which used machine learning algorithm has been publish in 1991 by Yoshiaki Okayama [42]. This study uses artificial neural network (ANN) to improve the classical sensor system. In [42], analogue data produced from sensor is used as an input data for ANN to predict early fire warning. In 2004, Fouzia Derbel [43], published a study which used ANN machine algorithm and shows a performance improvement of fire detection system by removing threshold value of a sensor. In 2009, Jaya Vardhan Gubbi [44], carried out a research to find the solution of an existing fire detection system which cannot be used in a critical scenario. He proposes a smoke detection system based on a video surveillance system. He used a ML algorithm such as support vector machine (SVM) and KNN classifier. The results of that study are impressive with limited false alarms. In 2010, Byoung chul ko [41],used a computer vision method to distinguish fire from color moving object, This method gives faster response time to detect fire and could be used in a large area, It also gives additional information about location of fire, size of fire and degree of burning, In [41],in case of false alarm, the system manager can confirm the existence of a fire through the surveillance monitor without visiting the location, This method is used as a commercial product by axon X company. In 2011, Truong Xuan Tung [45], publish a novel method for early fire detection which outperform all previous smoke detection method by using fuzzy ML method. In the same year, Feiniu Yuan [46],proposed a video -bases smoke detection based on histogram sequence which solved the rotation and illumination problem of previous smoke detection method. In 2014, Byoung chulko [47], published a result by using ML probabilistic fuzzy logic to calculate exact location and volume of fire and used as an input for automatic fire suppression system to suppress the fire. In 2015, Jong myam Kim [45], used a thermal images to classify fire in a fire fighting robot to detect and suppress a fire. In 2017, Tafazal Chaudhry [48], used ML method such as random forest to identifying burning material from its smoke with 99.6% accuracy. In the same year Florian [49] and Gao [50],used a semantic technology to find out fire ground location and smoke detection with deep domain method respectively .In 2018,Nyma [51] and Jie [52] published their research to detect smoke and real time for casting of fire by using SVM and ensemble Kalman filter respectively .In 2020,Chenghua li [53].published a novel data processing pipeline based on convolutional neural network (CNN) to detect video base smoke with high accuracy and efficiency. Similarly in 2021, Puli , used CNN method to detect fire and Luyao kou [54] used another variant of ANN such as gated recurrent unit to identify fire source location and predict intensity of fire. Hence, fire detection algorithm improved with the passage of time and computer vision outperform all previous model and improve real time detection.



CCD Camera

Mobile

Network

WAN/ADSL/Cable

Internet

Video

sequences

User Check

File House

Human Check

BTS

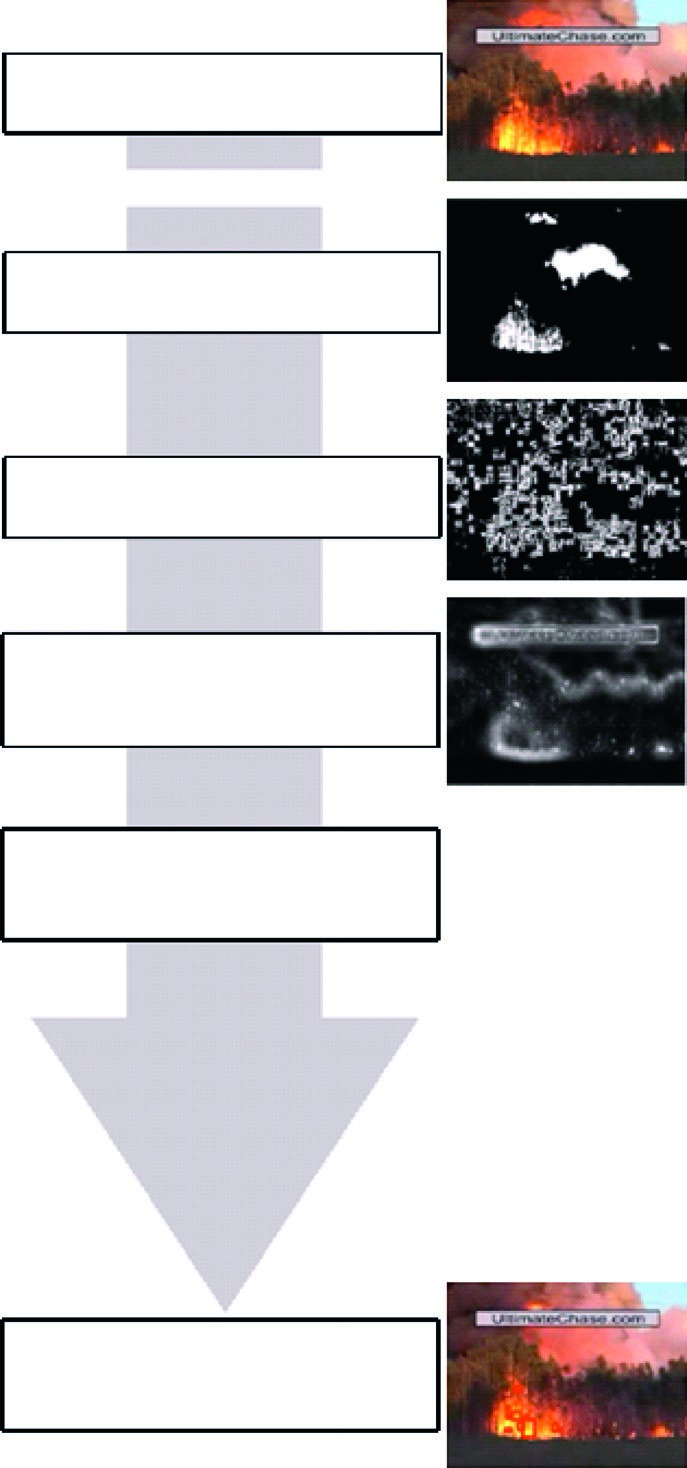
Jpeg

images

MMS Notify

Home Network

Figure 2‑10:The main application of a proposed fire monitoring system.



Input Video

Fire colored-pixel

detection

Moving pixel

detection

Non-fire pixcel removal using

temporal luminance variations

Fire-pixel verification using SVM

Morphological closing &

region merging

Figure 2‑11:Overall fire detection procedures of the proposed algorithm.

# Chapter: 03 Development of Surrogate Model

## Introduction

Theliterature review of machine learning application in fire safety engineering has been discussed in the previous chapter which highlights the potential and merit of adopting machine learning algorithms to solve the FSE problem. This chapter discusses the development of a regression-based surrogate modeling to solve structural fire engineering problems through probabilistic fire analysis of structure. Furthermore, a recently published study [55] has been recalculated and analyzed the result of the developed surrogate model.

## Probabilistic Fire Analysis

The general approach in fire safety engineering for fire safety design of the structure is dependent upon prescriptive guidance and code base recommendation. However, the obtained nominal fire resistance does not ensure an adequate safety level in the case of a complex and innovative design structure. To overcome such an issue performance-based approach could be used to calculate the explicit risk profile of structure under the range of nominal fire exposure and combination of different fire load. In other words, the full range of possible fire scenario and their associated probabilities is incorporated to evaluate the fire performance of the exceptional design to meet adequate fire safety levels which is not easy to find through code base recommendation or required safety level is based upon precedent [55].

The current approach for the fire design of the structure is a probabilistic analysis of the structure element exposed to fire. The probabilistic method depends upon the stochastic realization of the variable through Monte Carlo simulation. While the major drawback of Monte Carlo simulation is the computational expense and number of simulations. As a result, an efficient method for probabilistic analysis of structure is developed by the researcher to reduce the number of simulations and cost of advance modeling. However, it is realized that an improved probabilistic method depends upon advanced numerical model calculation and not suitable in the case of design iteration during fire safety design and does not provide fast design iteration. Therefore, a simple method is required which allow fast design iteration for quasi instantaneously updating probabilistic analysis in case of large and complex structure model. The current methodology for probabilistic fire analysis of structure used to develop a fragility curve which demonstrates the exceedance of probability of the predefined state of damage in terms of intensity measure or engineering demand parameter used in a state limit function. The previous research develops such fragility curves for steel frame structure and reinforcement concrete building through Monte Carlo simulation and expert elicitation decision. For the application of such a fragility curve in SFE, an intermediate or simplified model is required to incorporate the infinite number of design alternatives without using an advance numerical model. such limitation could be removed by the application of the machine learning method which is instrumental to approximate advanced numerical models by developing a surrogate model. A recently published study [55] effectively highlights the potential of the surrogate modeling technique for probabilistic fire analysis of the structure to generate a fragility curve. This study will redevelop the surrogate model as described in [55]. The following subsection briefly explains the development of such surrogate models. A published works has been recalculated to analyze such potential of surrogate modeling to develop a fragility curve used for fire safety design of the structure.

## Development of Regression based basic-Surrogate Model.

A general procedure for the development of a surrogate model consists of the selection of model variables, generation of input data, and model evaluation. This general procedure could be found elsewhere in more detail [55]. Here a part of a recently published study [55] for the development of surrogate modeling has been discussed in details and recalculation of the developed surrogate model is performed. For this case, a simple analytical model has been used for the calculation of the bending moment capacity of the RC slab for the known temperature of the rebar. The moment resistance of RC slab exposed to fire could be calculated through a non-linear model described in equation 1.Where, As is the area of tensile reinforcing bars in the slab, ᵩ is the diameter of the tensile reinforcing bars in the slab, h and b refer to the depth and width of the slab, c is the concrete cover to the reinforcement, fc,20\_C refers to the 20\_C compressive strength of the concrete, and fy,20\_C) and kfy(T) are respectively the reinforcement yield strength at 20\_C and the yield strength retention factor at T degrees Celsius [55].

The development of the surrogate model starts with the selection of model-independent variables. The model variable selected for this study are identified as fc,20\_C, fy,T = (kfy(T) Æfy,20\_C), c, h and As. The slab capacity is evaluated for a fixed unit width (b = 1 m) and the reinforcing bars are 12 mm in diameter. Next, the range of interest for the surrogate model is specified for each of the independent variables by an upper and a lower limit, as listed in table 1 [55]. It is intended that the develop surrogate model is accurately approximate the physical model within these upper and lower bounds. The surrogate model has been constructed in a Jupiter notebook by using python programming.The input data for the surrogate model has been generated through the physical model described in equation 1 by the combination of selected model variables. A sampling technique such as Latin hypercube sampling (LHS) is used for the generation of the data point. The generated data point which consists of 2000 data point have been used to train the surrogate model as well as for cross-validation. The performance of the surrogate model is highly dependent upon the optimum value for the hyperparameters such as the number of samples points (N), regularization parameter, and degree of the model hypothesis used during training of the model. The optimum value of the hyperparameter has been calculated by the grid search method.

The published work shows a polynomial order of degree 2 has higher accuracy while this study shows the polynomial degree of 3 has higher accuracy. The difference of polynomial degree is occurred due to different testing sample has been used. However, there is no significant improvement for the surrogate model with higher-order m > 3 is observed during this study. Therefore, a 3rd-degree polynomial is adopted as the surrogate model hypothesis for the fire-exposed RC slab depicted in figure 2.

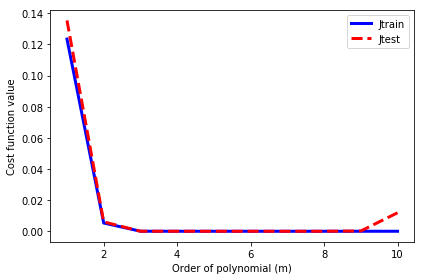


Figure 3‑1: X-axis: order of polynomial and Y-axis: Difference of mean square error

The remaining hyperparameter such as the number of a training sample points, and regularization factor has been shown in Figures 16 and 15. This study shows that the optimum value of the number of required samples is 1600 which produces a converged value of cost function. Similarly, the optimum value of the regularization parameter is 0 where the minimum value of the cost function has been observed. further increase of the regularization parameter increases the value of the cost function. The analysis of the obtained result revealed that the regularization parameter does not influence the developed surrogate model and could be neglected, and furthermore linear regression model seem to be accurate to describe given data set.

The performance of the developed surrogate model is depicted in Figures 17 for the published and current study. The performance of the surrogate model is determined by the actual and predicted moment capacity of the RC slab for the test data set. The initial input data has been split into 75% for training and 25% for testing purposes. This 25 % testing data has been used in the performance evaluation of the surrogate model and consider to be unseen data for the surrogate model. The coefficient of determination (R2) is found to be 99.53% for this study and 99.75 % for the case of published study. The result shows good accuracy of the developed surrogate model to predict moment resistance of RC slab exposed to fire. The analysis of R2 revealed that cross validation test data has influence on the accuracy of the result. As this study used the given 25 % of data set while published study generate another data set for cross validation purpose through Monte Carlo simulation as mentioned in [55].

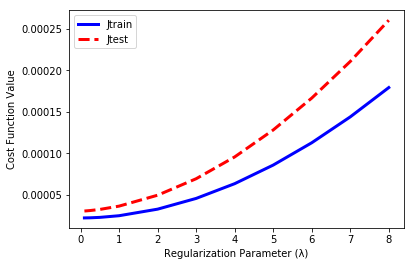


Figure 3‑2:Effect of Regularization Parameter

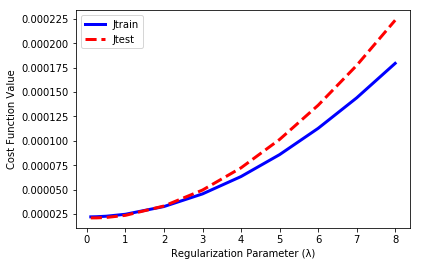


Figure 3‑3:X-axis: number of sample points. Y-axis: error of cost function

|  |  |
| --- | --- |
| Chart, scatter chart  Description automatically generated |  |

Figure 3‑4:Left side: Publish study result, Right Side: This study result.

It can be noted that the proposed models are optimally matching the performance of the reported results.

## Advanced Non-linear Model:

### Dataset

an advanced dataset providing the data samples for the ISO834 model is used for the analysis and final surrogate model designing. The dataset consists of 9369 training and 2459 testing or cross-validation data samples or observations. Seven feature vectors are available in the datasets along with the load as the prediction variable. The dataset has been generated by the use of Finite Element software (SAFIR). The detailed analysis of the dataset has been performed and is reported in the Exploratory data analysis section. The complete analysis of the dataset has been performed by using python.

#### Exploratory Data analysis

The complete statistical details of the training dataset and testing datasets is reported in Table 3.1 and 3.2, respectively. Here, the total number of training and testing example pairs are 9368 and 2458, respectively.

Table 3‑1 Description of Training Data

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | nodeline\_Y | Oos | Oop | epskfy | epskfc | cover | time | Load |
| mean | 0.000037 | 0.000003 | 0.000019 | 0.022815 | 0.069157 | 56.000994 | 173.878813 | 5669.650262 |
| std | 0.017295 | 0.017277 | 0.005774 | 2.308672 | 2.280203 | 23.098603 | 133.33923 | 2717.815084 |
| min | -0.029997 | -0.029997 | -0.009999 | -3.9996 | -3.9996 | 16.004 | 0.2 | 1000.475 |
| 25% | -0.014963 | -0.014915 | -0.004978 | -1.9766 | -1.8886 | 36.032 | 47.8125 | 3314.4375 |
| 50% | 0.000096 | 0.000018 | 0.000024 | 0.0348 | 0.08 | 55.96 | 154.891667 | 5650.25 |
| 75% | 0.015 | 0.01492 | 0.005026 | 2.0222 | 2.0382 | 76.086 | 315.241667 | 7994.6125 |
| max | 0.029997 | 0.029991 | 0.009999 | 3.9996 | 3.9996 | 95.996 | 360 | 10499.525 |

Table 3‑2 Description of testing data

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | nodeline\_Y | Oos | Oop | epskfy | epskfc | cover | time | Load |
| mean | 0.000081 | -0.000098 | 0.000037 | -0.051516 | 0.065318 | 54.686656 | 151.105024 | 6071.667565 |
| std | 0.017472 | 0.01747 | 0.005702 | 2.305567 | 2.156996 | 23.043199 | 100.010397 | 2325.624123 |
| min | -0.029978 | -0.029992 | -0.009997 | -3.997 | -3.999 | 16.01 | 2.3 | 1024.9375 |
| 25% | -0.014899 | -0.015379 | -0.004831 | -2.024 | -1.698 | 34.855 | 65.4 | 4248.40625 |
| 50% | 0.000165 | -0.00003 | 0.00003 | -0.086 | 0.094 | 53.86 | 135.1 | 6016 |
| 75% | 0.015184 | 0.015094 | 0.004965 | 1.956 | 1.8345 | 74.565 | 230.529167 | 7982.5 |
| max | 0.029992 | 0.029978 | 0.009997 | 3.999 | 3.997 | 95.99 | 359.85 | 10498.8125 |

From Tables it can be noted the max and minimum values for the data are not indicating the existence of null values, however the info analysis has been performed for the checking of the null value in the data the same has been reported in Figure 3.5 (a) and (b) for the training and testing datasets, respectively. It can be noted there are no-null value present in any columns in the both splits of the total dataset. In addition to that all the columns have numerical values.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
| Figure 3‑5 (a) Null Value analysis for training data; (b) Null Value analysis for testing data | |

For the understanding of the distribution of the each column in the dataset, the distribution analysis has been performed using histogram method for both the dataset. The same has been reported in Figure 3-6 and 3-7 for training and testing data. Here, it can be noted that all the vectors except time have uniform or close to uniform distribution. However, all the feature vectors have different mean and standard deviation and therefore the data must be standardized before applying the machine learning techniques.

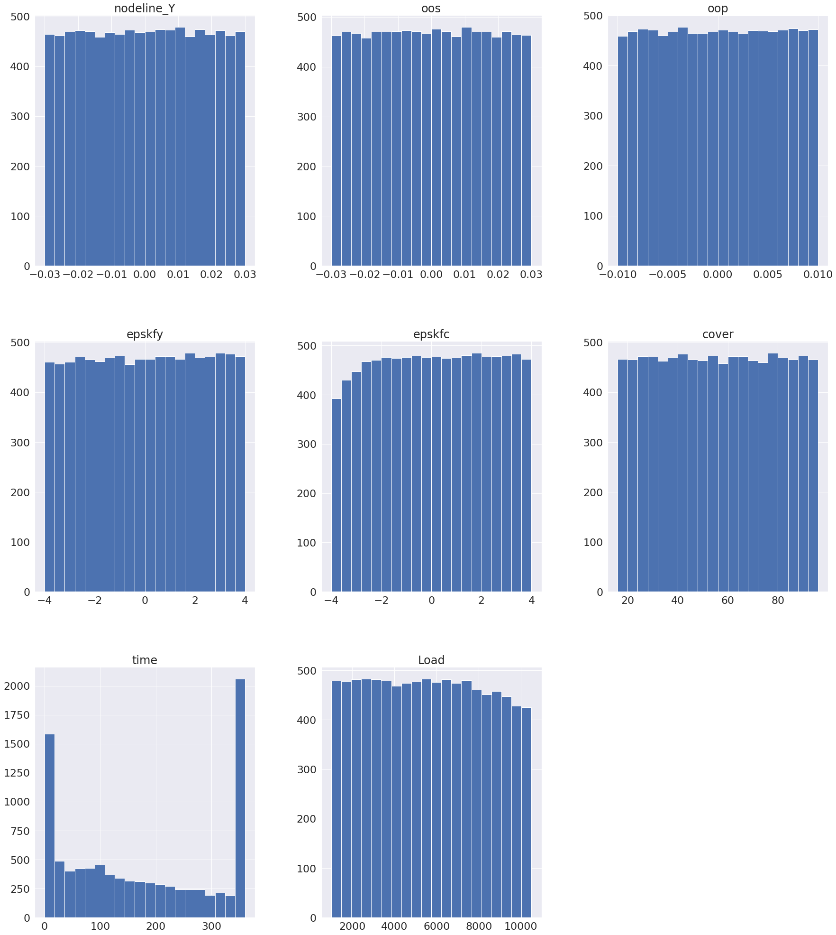


Figure 3‑6 Distribution Analysis for Training Data

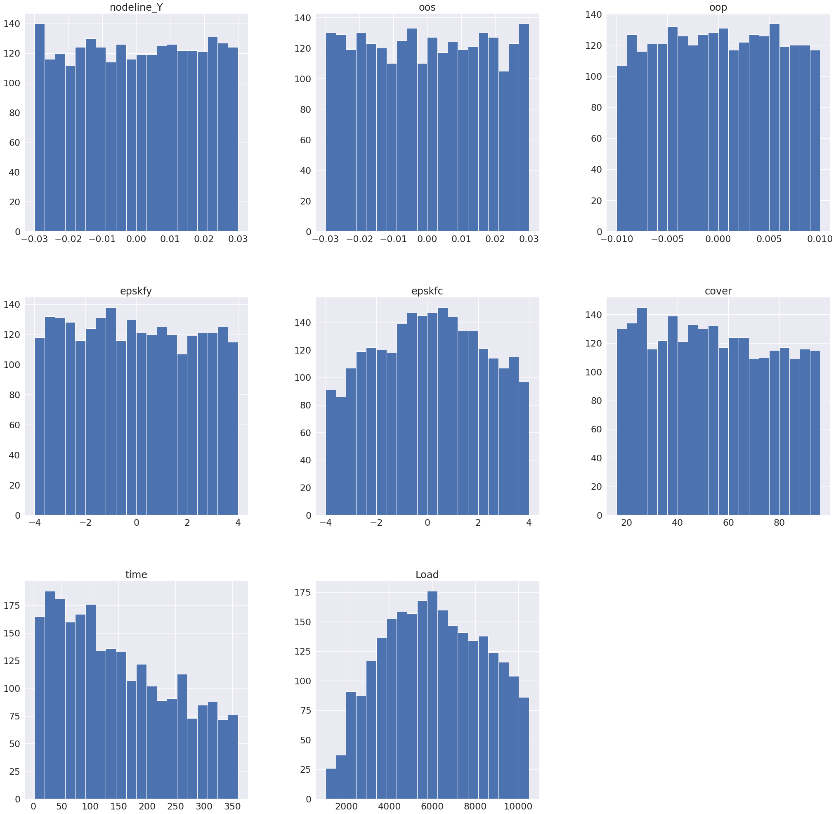


Figure 3‑7 Distribution analysis for testing data

After the understanding of the distribution of the data analysis has been performed for the outlier detection in the data. The analysis is referred to as outlier detection analysis. The analysis has been performed at the training data as some of the regression algorithms are sensitive to the outliers.

It can be noted from the boxplot reported in Figure 3-8, that neither of the variable has any outlier, as all of the values are present in the inter quartile range. Inter quartile range method is one of the most common method used in the literature for the detection and resolution of the outlier issue.

## Feature Analysis

Before using any regression algorithm, a simple feature analysis has been performed on the available dataset for the understanding of the behavior of features and correlation with the prediction variable the Load. There are three type of correlation analysis used in the literature for the understanding referred to as Pearson, Spearman, and Kindall.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |
|  |  |
| (e) | (f) |
| Figure 3‑8 Outlier Analysis | |

### Pearson-based correlation analysis

Pearson correlation analysis method is referred to as parametric test or analysis in the literature. The method assumes that the variables that are compared have the Gaussian distribution. The test is used to detect the strength and direction of correlation. The correlation matrix for the training data is reported in Figure 3-9, whereas the correlation with prediction variable Y=Load is reported in Figure 3-10. Here, it can be noted in Figure 3-9 that neither of the variable despite the time variable has correlation above 0.7 with the load variable.

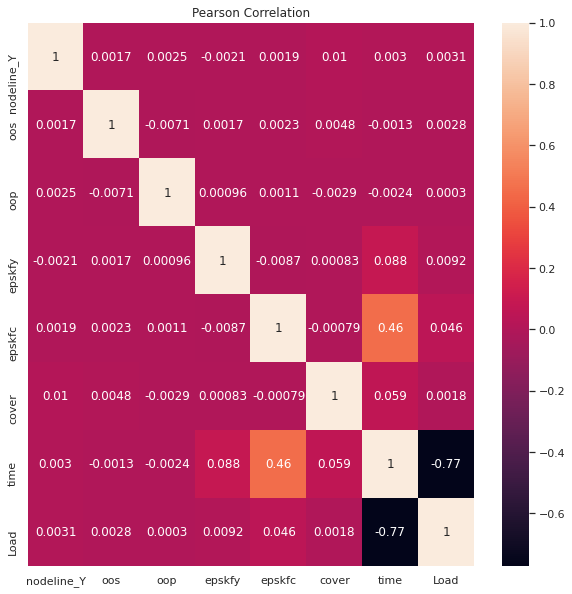


Figure 3‑9 Pearson Correlation Analysis

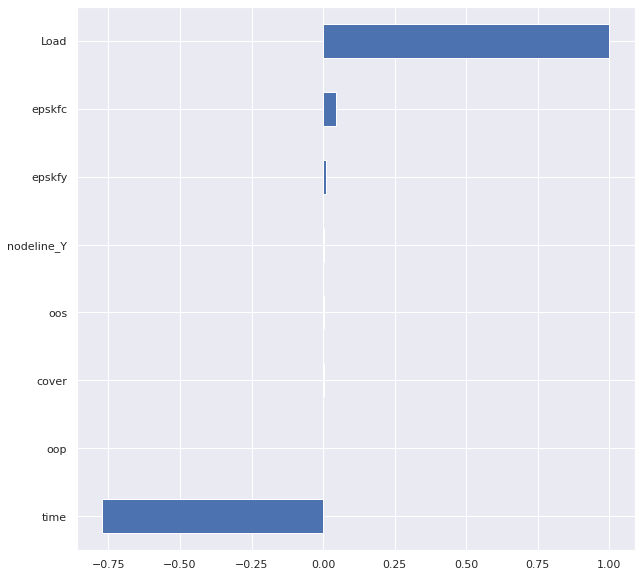


Figure 3‑10 Pearson Correlation between features and prediction variable

Here, the relationship between the two is in inverse, means load will increase with reduction in time. Time variable also have correlation with epskfy and epskfc, however as the value is not significantly high therefore the issue of multi-collinearity does not exist for the dataset.

### Spearman-based correlation analysis

Spearman correlation analysis is a non-parametric test or analysis that is used to study the monotonic relationship between variables. It is a non-parametric variant of Pearson. It is most commonly used for the ordinal data. However, the current packages perform order-based sorting prior to applying the test. The spearman correlation matrix for the training data is reported in Figure 3-11, while values of correlation with the prediction variables are reported in Figure 3-12.

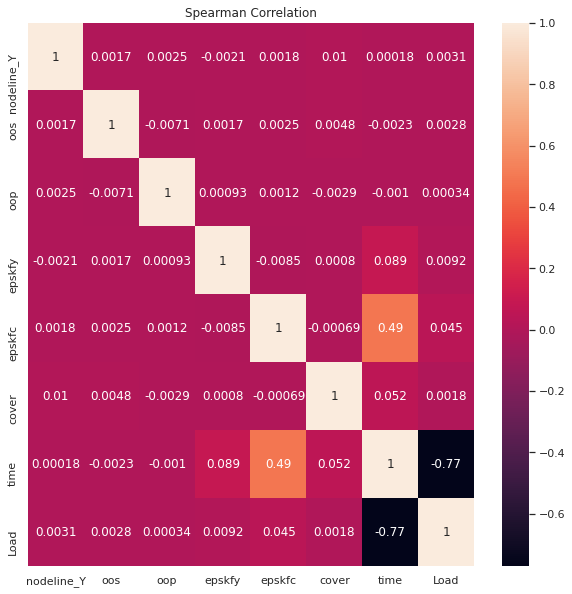


Figure 3‑11 Spearman Correlation

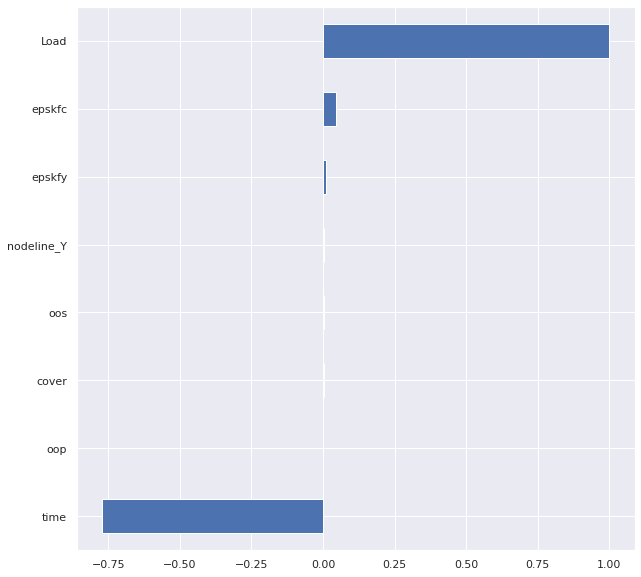


Figure 3‑12 Spearman Correlation between features and prediction variable

Here, it can be noted that the monotonic relationship exists between time and Y=load. Time variable also have monotonic-relationship with epskfy and epskfc, however as the value is not significantly high.

### **Kendall-based correlation analysis**

Kendall just like spearman is a non-parametric test used to find the correlation between variables. The method like spearman and Pearson is an important measure used in literature for studding the correlation. It is more robust than spearman. The correlation matrix is reported in Figure 3-13 while correlation with Y=Load is reported in Figure 3-14.

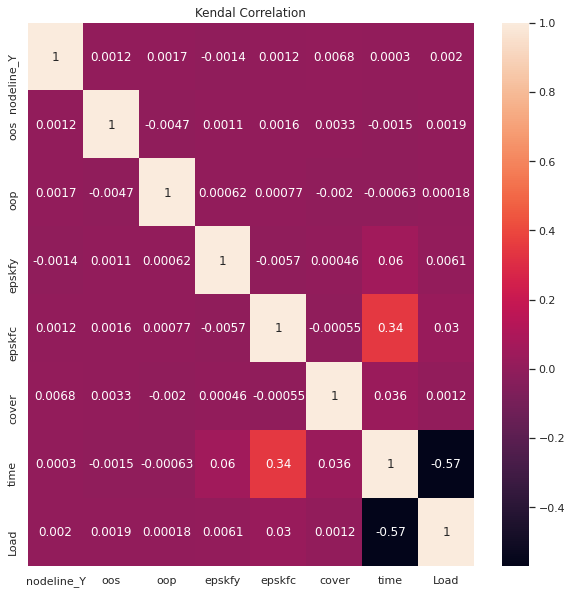


Figure 3‑13 Kendall Correlation

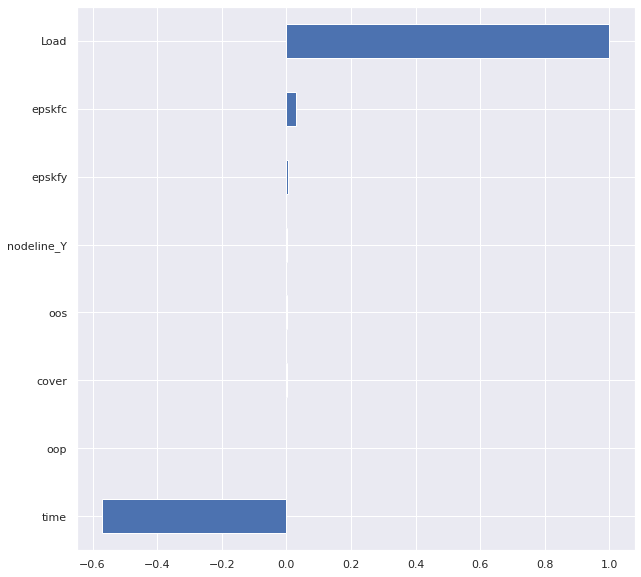


Figure 3‑14 Kendall Correlation between features and prediction variable

## Best Regression Model Selection

From the feature analysis it has been found that in spite to time there is no-linear relationship between any other feature and the prediction variable Y=Load. Therefore, a wide-range of regression must be explored to get the optimal robust regression-model for the design and then the same can be optimized for the dataset.

### Data Preprocessing

Before passing the variables to the group of regression algorithm the dataset needs to be scaled. The feature set is standardized in such a way that all the features have mean zero and standard deviation equal to 1. In addition to that the load value is divided by 1000 to have better and robust convergence. Original output variable is divided by a factor to get a two-digit variant for the same.

So, the equations used for the normalization and standardization and scaling are as below:

The library used for this purpose is sklearn. The new histogram or distribution is reported in Figure 3-15. Here, the mean of all the data is reduced to zero and standard deviation is 1.

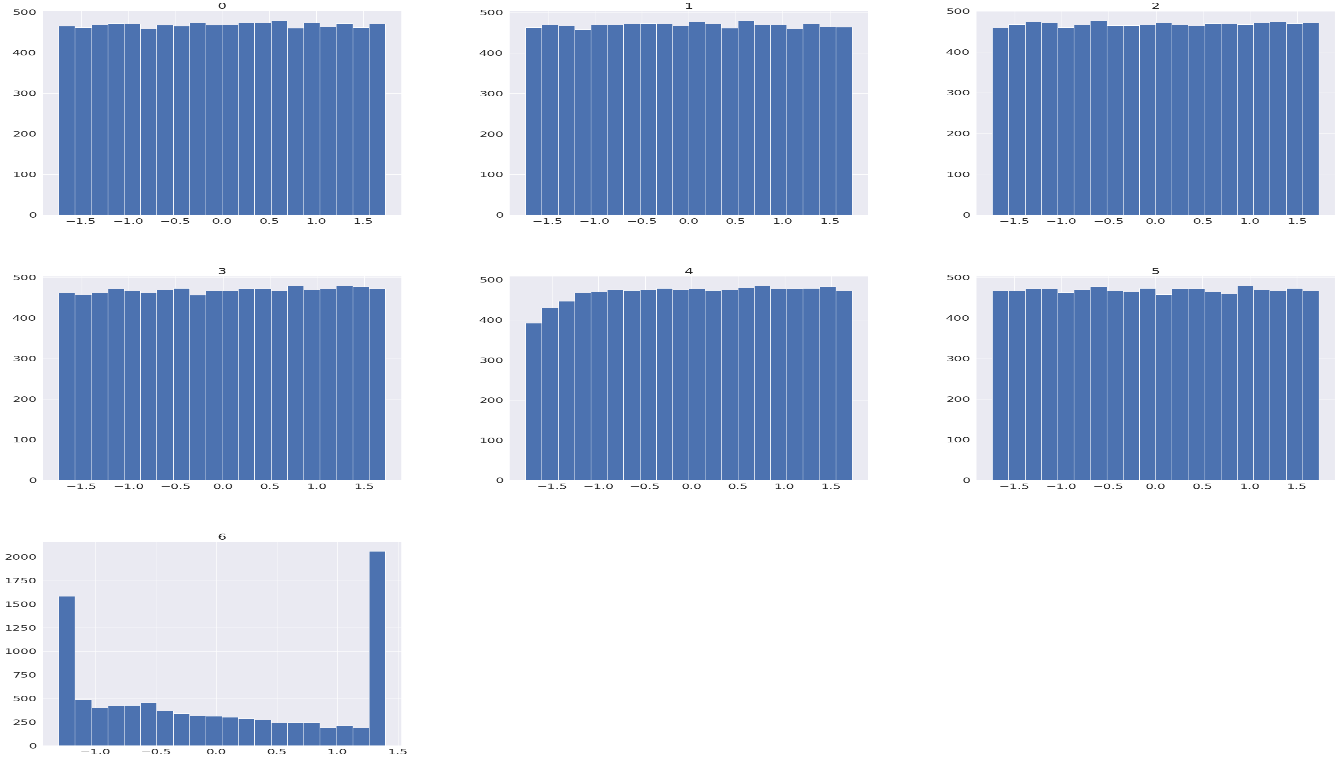


Figure 3‑15 Distribution for standardized data

### Regression Model Selection

For the best model selection, the dataset has been used for the training of different classes of regression algorithm and the same is reported in the section. The performance metrics used for the selection of the algorithm are Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Pearson correlation coefficient or R2, Spearman correlation coefficient, and Kendall correlation coefficient. The values of the performance matrices evaluated at testing data for all the algorithms and the prediction plot are reported below in Table 3-3 and Figures 3-16 to 3-28

Table 3‑3 Comparative analysis of different Regression Algorithms

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Regression  Algorithm | RMSE | MAE |  | Spearman | Kindall |
| Multi-Linear | 1.036758 | 0.847117 | 0.801184 | 0.904038 | 0.904038 |
| XGBOOST Gradient-Boosting | 0.584425 | 0.450628 | 0.936824 | 0.971922 | 0.855835 |
| Random  Forest | 0.692961 | 0.548095 | 0.911179 | 0.955167 | 0.812871 |
| MLP | **0.38071** | **0.249973** | **0.973191** | 0.973191 | **0.9169** |
| Linear-SVR | 1.011218 | 0.827174 | 0.810858 | 0.909594 | 0.735195 |
| Polynomial-SVR Degree=3 | 1.197227 | 0.96423 | 0.734875 | 0.882777 | 0.696033 |
| RBF SVR | 0.442847 | 0.293327 | 0.963725 | **0.981847** | 0.898459 |
| Sigmoid SVR | 0.442847 | 0.293327 | 0.963725 | 0.963725 | 0.898459 |
| K-Nearest  Neighbour | 1.086605 | 0.852119 | 0.781606 | 0.905994 | 0.736503 |
| Polynomial  Regression Degree=2 | 0.730081 | 0.565238 | 0.901409 | 0.950893 | 0.817934 |
| Polynomial  Regression Degree=3 | 0.56723 | 0.427517 | 0.940486 | 0.972855 | 0.862703 |
| Polynomial  Regression Degree=4 | 0.862703 | 0.370187 | 0.951654 | 0.978934 | 0.881167 |
| Light Gradient Boosting | **0.364145** | **0.27134** | **0.975473** | **0.989395** | **0.912025** |

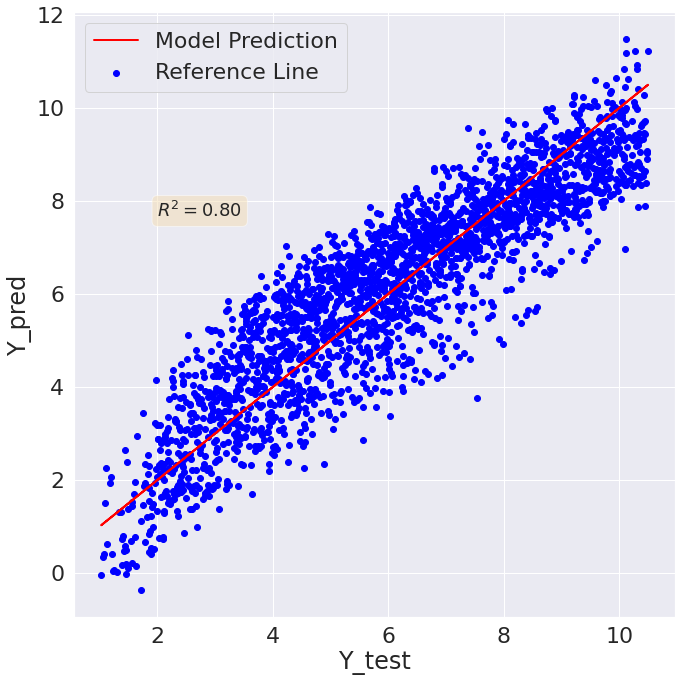


Figure 3‑16 Multi-Linear Regression prediction vs true value plot

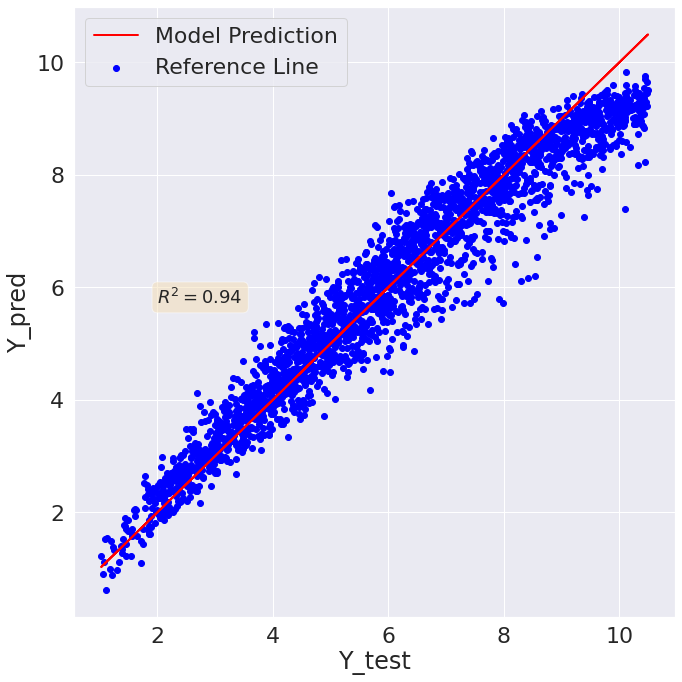


Figure 3‑17 XGBOOST Regression prediction vs true value plot

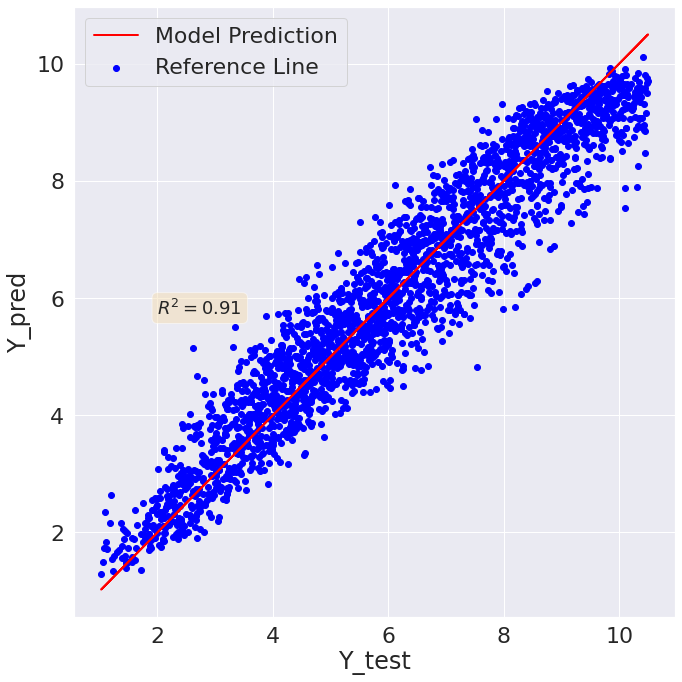


Figure 3‑18 Random Forest Regression prediction vs true value plot

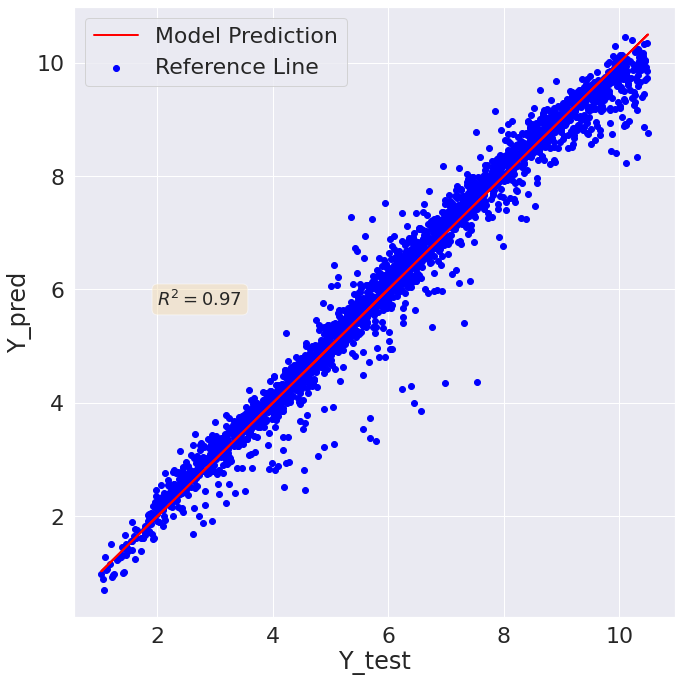


Figure 3‑19 MLP Regression prediction vs true value plot

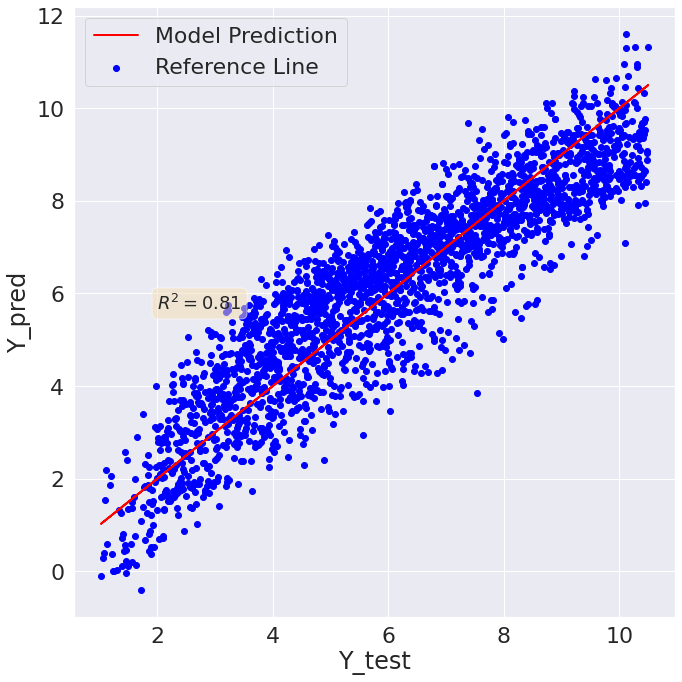


Figure 3‑20 Linear Support Vector Regression prediction vs true value plot

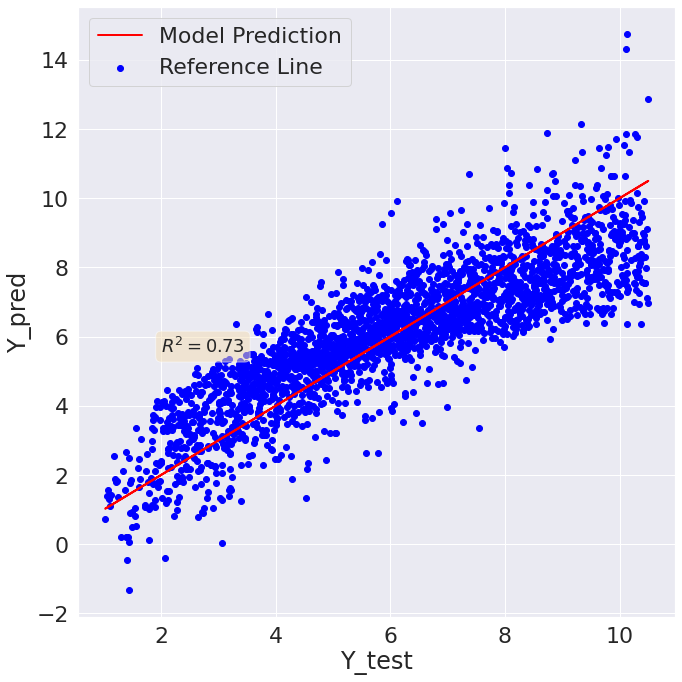


Figure 3‑21 Polynomial Regression (degree=3) prediction vs true value plot

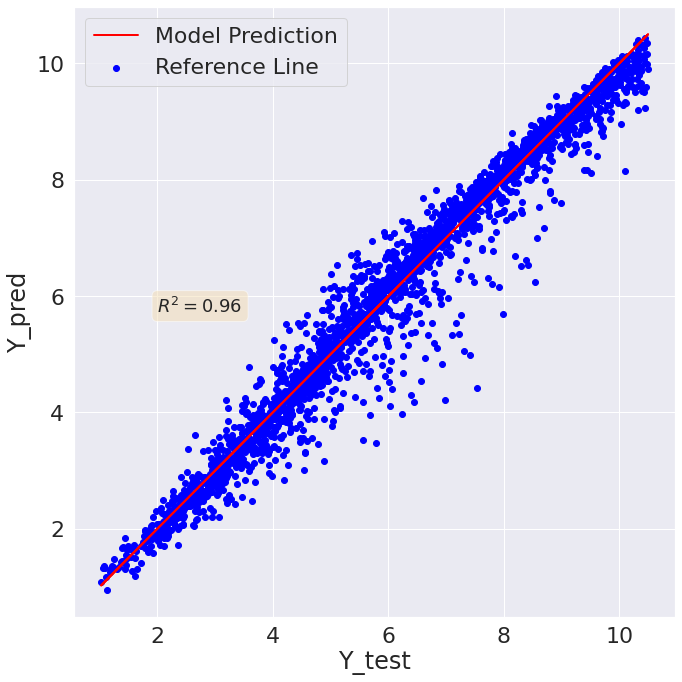


Figure 3‑22 RBF Support vector regression prediction vs true value plot

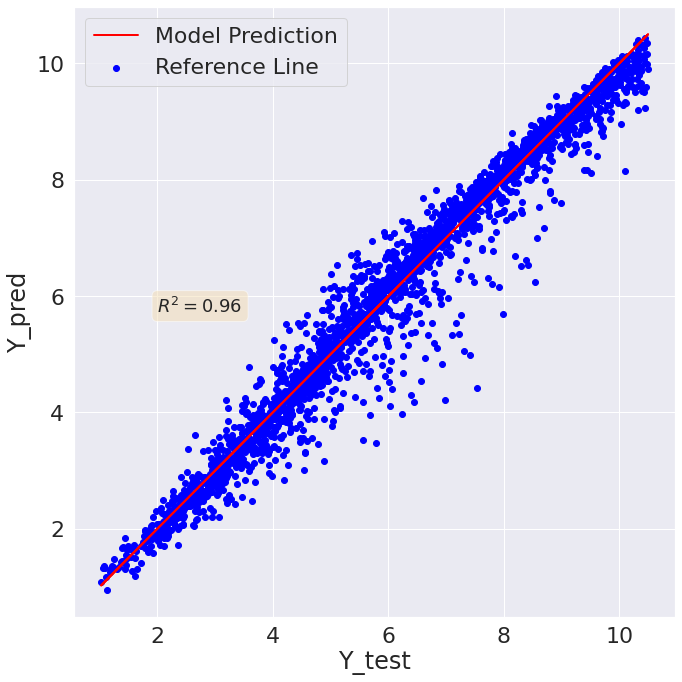


Figure 3‑23 Sigmoid Support vector regression prediction vs true value plot

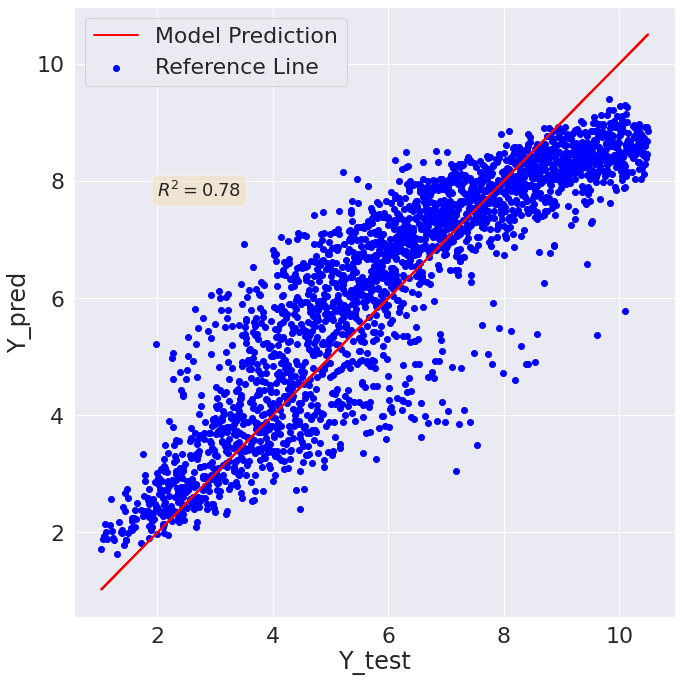


Figure 3‑24 K-Nearest Neighbors regression prediction vs true value plot

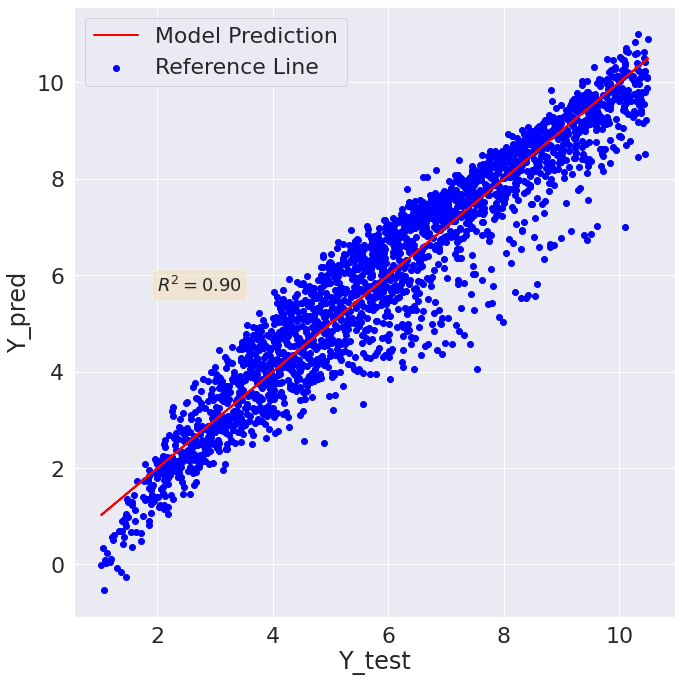


Figure 3‑25 Polynomial regression (degree=2) prediction vs true value plot



Figure 3‑26 Polynomial regression (degree=3) prediction vs true value plot

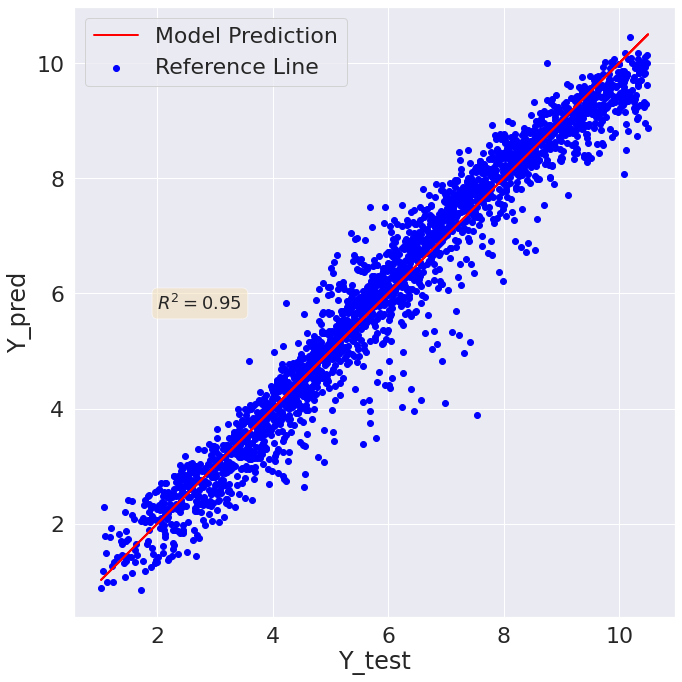


Figure 3‑27 Polynomial regression (degree=4) prediction vs true value plot

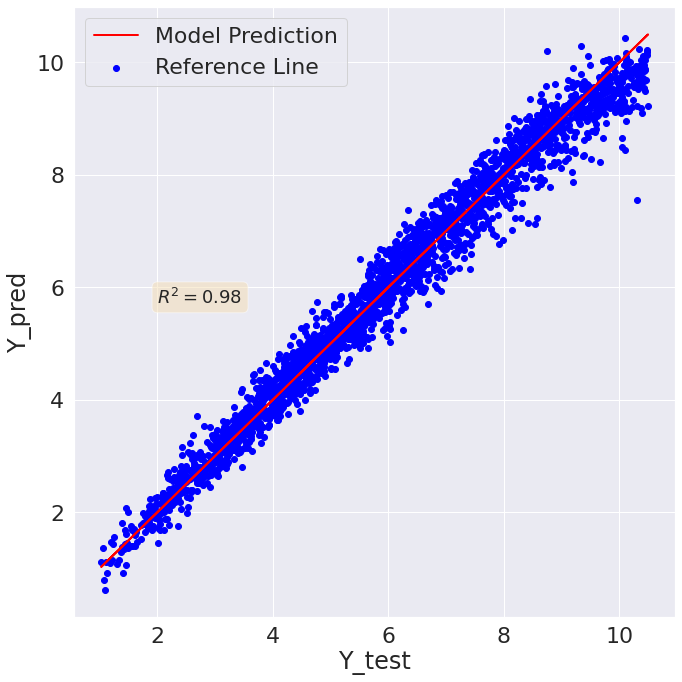


Figure 3‑28 Light gradient boosting prediction vs true value plot

After the detailed analysis it has been found that MLP and LGBM have the competing values of the performance metrics (that are very close to one another). However, as ensemble techniques have better characteristics when it comes to generality and bias variance, therefore the best models selected is LGBM.

### Light Gradient Boosting Regression Algorithm training and hyperparameter tunning.

For the design of final optimized algorithm for the problem the algorithm is trained by using the training and testing data. The training progress is reported below:

[1] valid\_0's l2: 4.71755

Training until validation scores don't improve for 1000 rounds.

[2] valid\_0's l2: 4.02545

[3] valid\_0's l2: 3.45463

[4] valid\_0's l2: 2.98594

[5] valid\_0's l2: 2.60406

[6] valid\_0's l2: 2.2888

[7] valid\_0's l2: 2.03236

[8] valid\_0's l2: 1.81835

[9] valid\_0's l2: 1.63677

[10] valid\_0's l2: 1.48515

[11] valid\_0's l2: 1.35661

[12] valid\_0's l2: 1.24509

[13] valid\_0's l2: 1.15809

[14] valid\_0's l2: 1.07843

[15] valid\_0's l2: 1.01201

[16] valid\_0's l2: 0.949022

[17] valid\_0's l2: 0.898606

[18] valid\_0's l2: 0.85434

[19] valid\_0's l2: 0.808065

[20] valid\_0's l2: 0.759383

[21] valid\_0's l2: 0.732017

[22] valid\_0's l2: 0.700499

[23] valid\_0's l2: 0.661881

[24] valid\_0's l2: 0.635641

[25] valid\_0's l2: 0.610706

[26] valid\_0's l2: 0.589474

[27] valid\_0's l2: 0.566385

[28] valid\_0's l2: 0.527414

[29] valid\_0's l2: 0.503662

[30] valid\_0's l2: 0.482114

[31] valid\_0's l2: 0.460048

[32] valid\_0's l2: 0.441962

[33] valid\_0's l2: 0.430857

[34] valid\_0's l2: 0.419929

[35] valid\_0's l2: 0.39793

[36] valid\_0's l2: 0.388054

[37] valid\_0's l2: 0.38012

[38] valid\_0's l2: 0.356069

[39] valid\_0's l2: 0.344601

[40] valid\_0's l2: 0.336937

[41] valid\_0's l2: 0.320667

[42] valid\_0's l2: 0.315208

[43] valid\_0's l2: 0.300847

[44] valid\_0's l2: 0.292659

[45] valid\_0's l2: 0.283456

[46] valid\_0's l2: 0.279764

[47] valid\_0's l2: 0.268241

[48] valid\_0's l2: 0.259819

[49] valid\_0's l2: 0.25653

[50] valid\_0's l2: 0.24418

[51] valid\_0's l2: 0.237753

[52] valid\_0's l2: 0.231417

[53] valid\_0's l2: 0.228078

[54] valid\_0's l2: 0.222847

[55] valid\_0's l2: 0.21811

[56] valid\_0's l2: 0.211838

[57] valid\_0's l2: 0.208467

[58] valid\_0's l2: 0.204211

[59] valid\_0's l2: 0.199086

[60] valid\_0's l2: 0.197295

[61] valid\_0's l2: 0.190143

[62] valid\_0's l2: 0.18795

[63] valid\_0's l2: 0.186573

[64] valid\_0's l2: 0.184837

[65] valid\_0's l2: 0.183412

[66] valid\_0's l2: 0.18112

[67] valid\_0's l2: 0.176192

[68] valid\_0's l2: 0.175188

[69] valid\_0's l2: 0.170651

[70] valid\_0's l2: 0.168679

[71] valid\_0's l2: 0.165861

[72] valid\_0's l2: 0.164816

[73] valid\_0's l2: 0.162093

[74] valid\_0's l2: 0.160432

[75] valid\_0's l2: 0.15817

[76] valid\_0's l2: 0.155229

[77] valid\_0's l2: 0.153704

[78] valid\_0's l2: 0.152707

[79] valid\_0's l2: 0.151857

[80] valid\_0's l2: 0.151251

[81] valid\_0's l2: 0.147607

[82] valid\_0's l2: 0.147441

[83] valid\_0's l2: 0.146914

[84] valid\_0's l2: 0.145444

[85] valid\_0's l2: 0.145167

[86] valid\_0's l2: 0.141714

[87] valid\_0's l2: 0.14051

[88] valid\_0's l2: 0.139937

[89] valid\_0's l2: 0.139183

[90] valid\_0's l2: 0.138164

[91] valid\_0's l2: 0.136879

[92] valid\_0's l2: 0.136435

[93] valid\_0's l2: 0.136087

[94] valid\_0's l2: 0.135296

[95] valid\_0's l2: 0.134041

[96] valid\_0's l2: 0.133645

[97] valid\_0's l2: 0.133526

[98] valid\_0's l2: 0.133442

[99] valid\_0's l2: 0.132955

[100] valid\_0's l2: 0.132602

Did not meet early stopping. Best iteration is:

[100] valid\_0's l2: 0.132602

**The best Parameters while training**

LGBMRegressor(boosting\_type='gbdt', class\_weight=None, colsample\_bytree=1.0,

importance\_type='split', learning\_rate=0.1, max\_depth=-1,

min\_child\_samples=20, min\_child\_weight=0.001, min\_split\_gain=0.0,

n\_estimators=100, n\_jobs=-1, num\_leaves=31, objective=None,

random\_state=None, reg\_alpha=0.0, reg\_lambda=0.0, silent=True,

subsample=1.0, subsample\_for\_bin=200000, subsample\_freq=0)

After that some of the important hyperparameters are tuned in a more depth for the avoidance of overfitting. Each one of which is described below in detail.

1. **Learning Rate:**

The learning rate vs Loss (RMSE), plot for the algorithm for the training and the testing data is reported in Figure 3-29. It can be observed that after 0.3 the algorithm starts overfitting and the best value of the learning rate for the algorithm is 0.20.

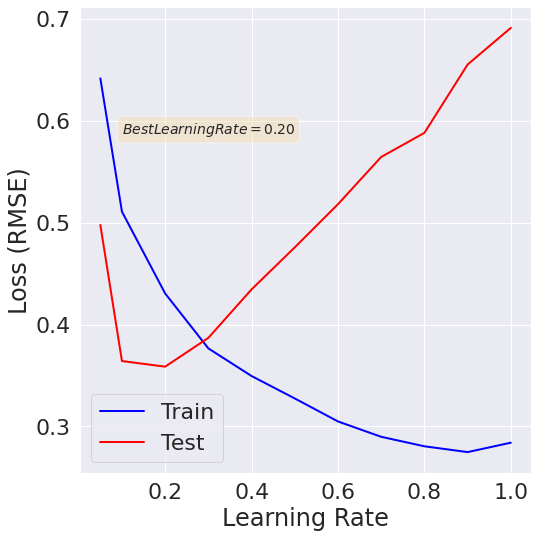


Figure 3‑29 LGBM learning rate vs loss plot

1. **Number of estimators**

The number of estimators vs Loss (RMSE), plot for the algorithm for the training and the testing data is reported in Figure 3-30. It can be observed that the best learning rate value is 190 after which the algorithm starts overfitting.

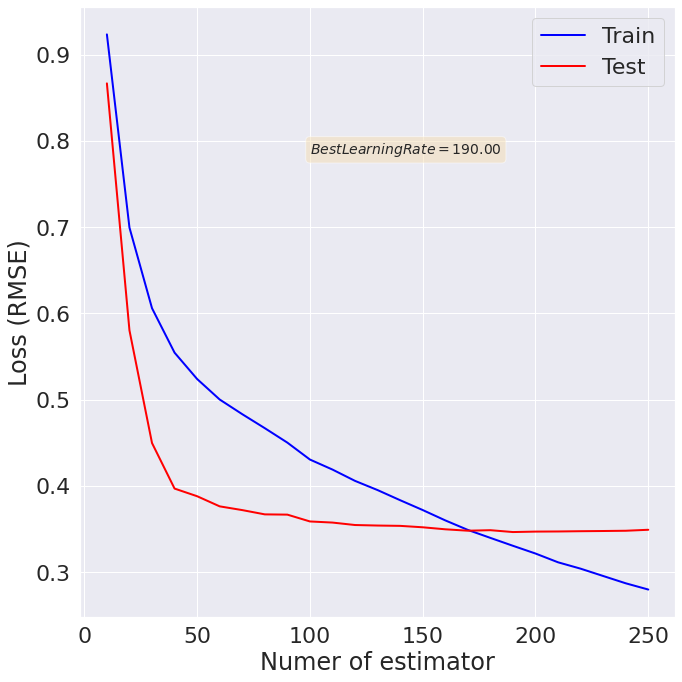


Figure 3‑30 Number of estimators vs Loss (RMSE) plot

The final model performance metrics values are:

The computed root mean squared error is = 0.3586572280632759

The computed mean absolute error is = 0.27162564937484396

The r2 measure for the model is = 0.9762065811348676

The Spearman for the data is= 0.9894022942163794

The Kendall correlation for the data is= 0.9104625597709406

### Feature Importance analysis

#### LGBM Based

The importance of the features for the LGBM model can be determined on the basis of splitting or gain. The most common method used is splitting. Tree based methods make decision on the basis of the features so the feature that is used more in making principal decisions has higher score than the other features. The importance plot for the regression algorithm is reported in Figure 3-31 and 3-32 from gain and split.

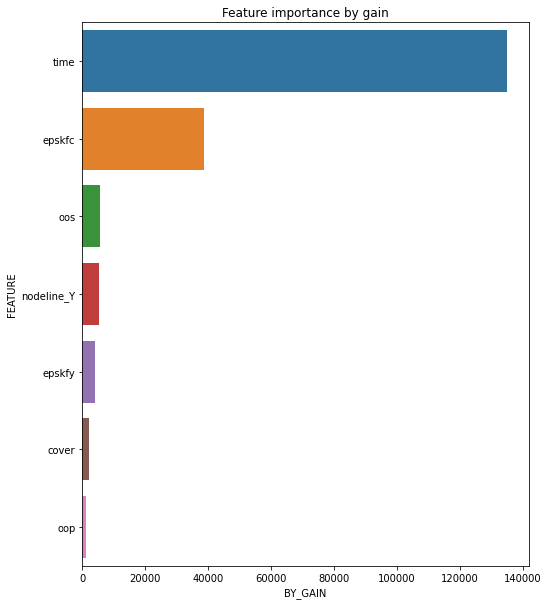


Figure 3‑31 Feature importance vs feature plot for GAIN



Figure 3‑32 Feature importance vs feature plot for Split

So, from the analysis it can be noted that epskfc has the highest value.

#### LGBM and Shap-based feature analysis

Shap is one the most famous library used for the feature importance analysis specially with tree-based algorithm. The same has been used for the further analysis and all the importance plots are reported below:

1. **Model output vs Feature importance**

The impact of features on the model output has been study and it has been found that time has the highest impact on the output, the same is reported in Figure 3-32.

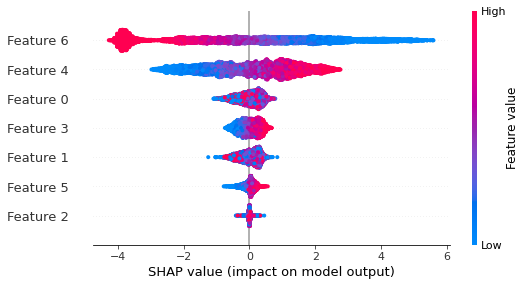


Figure 3‑33 Feature Impact on model output analysis using shap value

1. **Dependency plot**

The dependency plot is used for the determination of dependency of the feature variable and the output, where the color is automatically selected by the library. The plots for each one of the features are reported from Figure 3-34 to

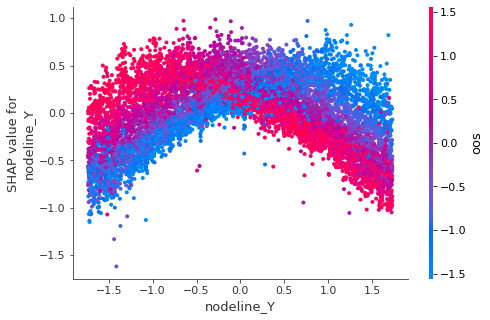


Figure 3‑34 nodline\_Y feature dependency plot

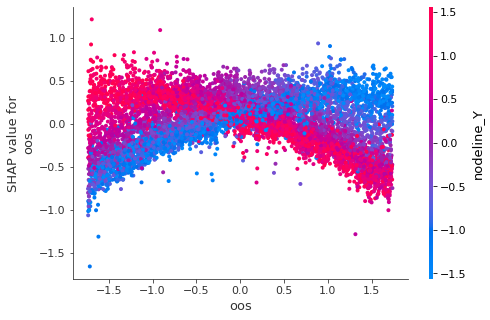


Figure 3‑35 oos feature variable dependency plot

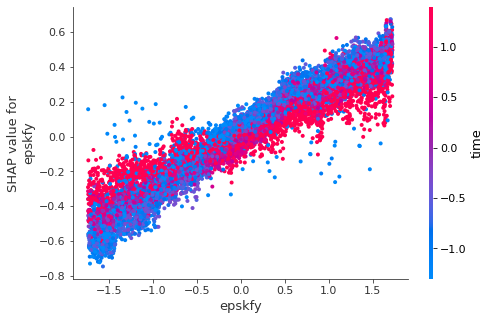


Figure 3‑36 epskfy feature variable dependency plot

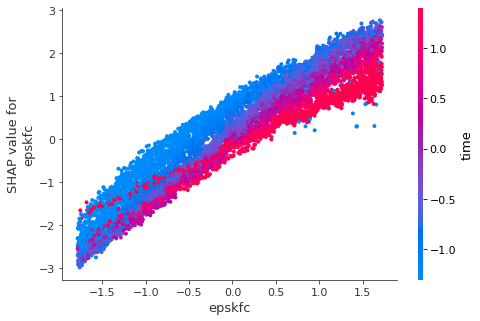


Figure 3‑37 epskfc feature variable dependency plot

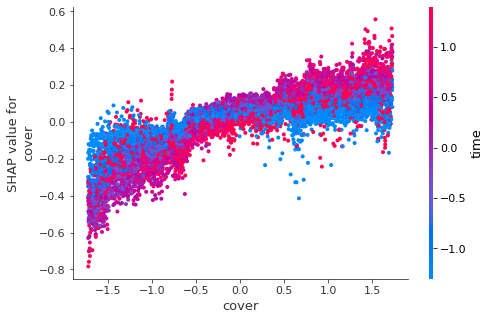


Figure 3‑38 cover feature dependency plot

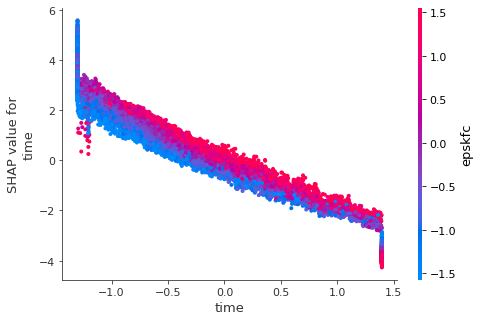


Figure 3‑39 time feature dependency plot

After the analysis it has been found that time, epskfc, and nodline\_Y have the highest effect on the output.

# Chapter :04 Discussion and Conclusions

## Introduction

This chapter discuss glimpse of some of the challenges, future research, and conclusion in order to successful adoption of machine learning as a modern tool to supplement existing method such as experimentation and fire dynamic simulation.

## Discussion

The domain knowledge of fire safety engineering will help to formulate machine learning hypothesis. The selection of the specific technique from the range of available machine learning algorithm only possible if user have a domain knowledge of FSE and machine learning methods. To prove soundness of the machine learning method, one should understand physics of fire problem before formulating machine learning hypothesis. A proper hypothesis is only established if user have domain knowledge related to fire test and feature of governing phenomena.

The facility of fire testing is not available for everyone. Fire testing of a problem is a complex and expensive task. Owing to this condition few problems of fire safety engineering has been solved with machine learning method. Similarly, concept of big data and imbalance data is a current challenge for FSE community. It is recommended to use rational assumptions to generate data and improved algorithm where small data set exist. furthermore, use those machine learning method which specific design for small data set.

It is encouraging that future experimentation and simulation performed in a such way that produce data set could be used through machine learning model to predict outcome result of new data set. In simple words if a surrogate model train properly then design iteration can be achieved fast and reduce the cost of design iteration by without running advance non liner model such as FDS OR SAFIR.

## Conclusion

This study presents a literature overview for machine learning applications in the field of fire safety engineering. The positive potential of machine learning algorithm represents as a modern technique to solve complex problems of fire safety engineering and supplement experimental and simulation study. As a result, advancing research and practice tool for FSE. The literature review starts by covering published research work consist of problems in FSE field solved by machine learning methods. After that, a simple case is considered from a published paper to develop a surrogate model to predict the moment capacity of reinforced concrete slab. The following conclusions could be drawn from this study:

* The machine learning methods have a tremendous potential to solve a range of fire safety engineering problem such as fire detection, fire risk assessment, fire dynamics and fire performance of a structure etc.
* The performance of machine learning method is better and fast to solve ill-defined and multidimensional problem such as fire induced flow field and fire induced spalling.
* The field of machine learning developing with a rapid pace to develop algorithm which solve problem with small data. The available small data in the field of FSE hinder to adopt application of machine learning method such as deep learning.
* The machine learning model has been validated in a parallel field by computer science developer and produce minimum doubt for the true advocacy of advance tool in the field of FSE.
* This study infers that artificial neural network mostly used to solve majority of FSE problem.
* The developed regression based surrogate model accurately predict the moment capacity of an analytical model.
* The development of surrogate model does not require much time and easily use for design iteration phase.
* The python programming skill is required to adopt machine learning model in fire safety engineering field. Hence it is recommended for updating curriculum for fire safety engineering field and incorporate automation as a new subject.

# APPENDIX

This is the python code used for the development of basic surrogate model:

**import** pandas **as** pd

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.model\_selection **import** validation\_curve, learning\_curve

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.metrics **import** r2\_score

**from** sklearn.model\_selection **import** GridSearchCV

**from** sklearn.linear\_model **import** Ridge,SGDRegressor

**import** matplotlib.pyplot **as** plt

**from** sklearn.pipeline **import** Pipeline, FeatureUnion

**from** sklearn.preprocessing **import** PolynomialFeatures

**from** sklearn.preprocessing **import** StandardScaler

**import** numpy **as** np

**from** sklearn.metrics **import** mean\_squared\_error

In [4]:

**%cd** C:\Users\Engineer Asad yousuf\Documents\case\_3

C:\Users\Engineer Asad yousuf\Documents\case\_3

In [6]:

X **=** pd**.**read\_excel('LearningVariables.xlsx')**.**values

Y **=** pd**.**read\_excel('LearningY.xlsx')**.**values

x**=**X

y**=**Y

In [7]:

sc**=**StandardScaler()

x\_data**=**sc**.**fit\_transform(x)

In [8]:

mean\_x\_data**=**x\_data**.**mean(axis**=**0)

print("mean after scaling per column ",mean\_x\_data)

std\_x\_data**=**x\_data**.**std(axis**=**0)

print("std after scaling per column ",std\_x\_data)

mean after scaling per column [-3.55271368e-18 0.00000000e+00 -3.01980663e-16 -3.81916720e-17

2.15827356e-16]

std after scaling per column [1. 1. 1. 1. 1.]

In [9]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x\_data,y,test\_size**=**0.20,random\_state**=**0)

In [61]:

Input**=**[('polynomial',PolynomialFeatures()),('model',Ridge())]

pipe**=**Pipeline(Input)

*#pipe.get\_params().keys()*

In [98]:

alpha**=**[1,0.1,0.01,0.001,0.0001,0]

degree**=**[1,2,3,4,5,6,7,8,9,10]

grid**=**GridSearchCV(estimator**=**pipe, param\_grid**=**{'polynomial\_\_degree':degree,'model\_\_alpha':alpha},cv**=**5, scoring**=**'neg\_mean\_squared\_error', return\_train\_score**=True**)

In [99]:

Out[99]:

GridSearchCV(cv=5, error\_score='raise-deprecating',

estimator=Pipeline(memory=None,

steps=[('polynomial', PolynomialFeatures(degree=2, include\_bias=True, interaction\_only=False)), ('model', Ridge(alpha=0, copy\_X=True, fit\_intercept=True, max\_iter=None,

normalize=False, random\_state=None, solver='auto', tol=0.001))]),

fit\_params=None, iid='warn', n\_jobs=None,

param\_grid={'polynomial\_\_degree': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'model\_\_alpha': [1, 0.1, 0.01, 0.001, 0.0001, 0]},

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=True,

scoring='neg\_mean\_squared\_error', verbose=0)

In [100]:

grid**.**best\_estimator\_

Out[100]:

Pipeline(memory=None,

steps=[('polynomial', PolynomialFeatures(degree=7, include\_bias=True, interaction\_only=False)), ('model', Ridge(alpha=0.001, copy\_X=True, fit\_intercept=True, max\_iter=None,

normalize=False, random\_state=None, solver='auto', tol=0.001))])

In [101]:

**def** plot\_grid\_search(cv\_results, grid\_param\_1, grid\_param\_2, name\_param\_1, name\_param\_2):

*# Get Test Scores Mean and std for each grid search*

scores\_mean **=** cv\_results['mean\_test\_score']

scores\_mean **=** np**.**array(scores\_mean)**.**reshape(len(grid\_param\_2),len(grid\_param\_1))

scores\_sd **=** cv\_results['std\_test\_score']

scores\_sd **=** np**.**array(scores\_sd)**.**reshape(len(grid\_param\_2),len(grid\_param\_1))

*# Plot Grid search scores*

\_, ax **=** plt**.**subplots(1,1)

*# Param1 is the X-axis, Param 2 is represented as a different curve (color line)*

**for** idx, val **in** enumerate(grid\_param\_2):

ax**.**plot(grid\_param\_1, scores\_mean[idx,:], '-o', label**=** name\_param\_2 **+** ': ' **+** str(val))

ax**.**set\_title("Grid Search Scores", fontsize**=**20, fontweight**=**'bold')

ax**.**set\_xlabel(name\_param\_1, fontsize**=**16)

ax**.**set\_ylabel('CV Average Score', fontsize**=**16)

ax**.**legend(loc**=**"best", fontsize**=**15)

ax**.**grid('on')

In [102]:

*# Calling Method*

plot\_grid\_search(grid**.**cv\_results\_,degree,alpha, 'degree', 'alpha')

D:\Anaconda3\lib\site-packages\matplotlib\cbook\\_\_init\_\_.py:424: MatplotlibDeprecationWarning:

Passing one of 'on', 'true', 'off', 'false' as a boolean is deprecated; use an actual boolean (True/False) instead.

warn\_deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false' as a "

So, best alpha=0 and degree=3

In [79]:

Input**=**[('polynomial',PolynomialFeatures()),('model',Ridge(alpha**=**0))]

pipe**=**Pipeline(Input)

In [108]:

mse\_test**=**[]

mse\_train**=**[]

**for** d **in** degree:

Input**=**[('polynomial',PolynomialFeatures(degree**=**d)),('model',Ridge(alpha**=**0))]

pipe**=**Pipeline(Input)

pipe**.**fit(x\_train,y\_train)

y\_pred\_train**=**pipe**.**predict(x\_train)

mse\_temp\_train**=**mean\_squared\_error(y\_pred\_train,y\_train)

mse\_train**.**append(mse\_temp\_train)

y\_pred\_test**=**pipe**.**predict(x\_test)

mse\_temp\_test**=**mean\_squared\_error(y\_pred\_test,y\_test)

mse\_test**.**append(mse\_temp\_test)

D:\Anaconda3\lib\site-packages\sklearn\linear\_model\ridge.py:165: LinAlgWarning: Ill-conditioned matrix (rcond=1.70317e-19): result may not be accurate.

overwrite\_a=False)

D:\Anaconda3\lib\site-packages\sklearn\linear\_model\ridge.py:165: LinAlgWarning: Ill-conditioned matrix (rcond=6.7983e-20): result may not be accurate.

overwrite\_a=False)

In [127]:

plt**.**plot(degree,mse\_train,label**=**'Jtrain',color**=**'b',linewidth**=**3)

plt**.**plot(degree,mse\_test,label**=**'Jtest',color**=**'r',linestyle**=**'--',linewidth**=**3)

plt**.**legend()

plt**.**show()

In [212]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x\_data,y,test\_size**=**0.25,random\_state**=**0)

In [200]:

data**=**[1600**-**0.75**\***1600,1600**-**0.5**\***1600,1600**-**0.25**\***1600,1599]

In [207]:

mse\_test**=**[]

mse\_train**=**[]

test**=**[0.75,0.5,0.25,0]

**for** t **in** test:

x\_train,x\_t,y\_train,y\_t**=**train\_test\_split(x\_train\_a,y\_train\_a,test\_size**=**t,random\_state**=**0)

Input**=**[('polynomial',PolynomialFeatures(degree**=**3)),('model',Ridge(alpha**=**0))]

pipe**=**Pipeline(Input)

pipe**.**fit(x\_train,y\_train)

y\_pred\_train**=**pipe**.**predict(x\_train)

mse\_temp\_train**=**mean\_squared\_error(y\_pred\_train,y\_train)

mse\_train**.**append(mse\_temp\_train)

y\_pred\_test**=**pipe**.**predict(x\_test)

mse\_temp\_test**=**mean\_squared\_error(y\_pred\_test,y\_test)

mse\_test**.**append(mse\_temp\_test)

In [208]:

mse\_test

Out[208]:

[15127.844294188668,

14139.320350559734,

13807.008596350342,

13471.179071985514]

In [209]:

plt**.**plot(data,mse\_train,label**=**'Jtrain',color**=**'b',linewidth**=**3)

plt**.**plot(data,mse\_test,label**=**'Jtest',color**=**'r',linestyle**=**'--',linewidth**=**3)

plt**.**legend()

plt**.**show()

In [213]:

mse\_test**=**[]

mse\_train**=**[]

**for** a **in** alphas:

Input**=**[('polynomial',PolynomialFeatures(degree**=**3)),('model',Ridge(alpha**=**a))]

pipe**=**Pipeline(Input)

pipe**.**fit(x\_train,y\_train)

y\_pred\_train**=**pipe**.**predict(x\_train)

mse\_temp\_train**=**mean\_squared\_error(y\_pred\_train,y\_train)

mse\_train**.**append(mse\_temp\_train)

y\_pred\_test**=**pipe**.**predict(x\_test)

mse\_temp\_test**=**mean\_squared\_error(y\_pred\_test,y\_test)

mse\_test**.**append(mse\_temp\_test)

In [222]:

plt**.**plot(alphas,mse\_train,label**=**'Jtrain',color**=**'b',linewidth**=**3)

plt**.**plot(alphas,mse\_test,label**=**'Jtest',color**=**'r',linestyle**=**'--',linewidth**=**3)

plt**.**legend()

plt**.**xlabel('Regularization Parameter (\u03BB)')

plt**.**ylabel('Cost Function Value')

plt**.**show()

In [10]:

Input**=**[('polynomial',PolynomialFeatures(degree**=**3)),('model',Ridge(alpha**=**0))]

pipe**=**Pipeline(Input)

pipe**.**fit(x\_train,y\_train)

Out[10]:

Pipeline(steps=[('polynomial', PolynomialFeatures(degree=3)),

('model', Ridge(alpha=0))])

In [11]:

y\_pred**=**pipe**.**predict(x\_test)

In [12]:

fig, ax **=** plt**.**subplots(figsize**=**(6,6))

ax**.**scatter(y\_test, y\_pred, c**=**'red',label**=**'Surrogate model')

ax**.**plot(y\_test,y\_test, color**=**'blue', linewidth**=**2,label**=**'Reference line')

ax**.**legend()

ax**.**plot()

ax**.**set\_xlabel("Actual $M\_{R}$ of Slab")

ax**.**set\_ylabel("Y\_pred")

ax**.**set\_title("Predicted $M\_{R}$ of Slab")

R\_square**=**r2\_score(y\_test,y\_pred)

textstr **=** r'$\mathrm{R\_2}=%.4f$' **%** (R\_square)

props **=** dict(boxstyle**=**'round', facecolor**=**'wheat', alpha**=**0.5)

ax**.**text(0.25, 0.25, textstr, transform**=**ax**.**transAxes, fontsize**=**14,verticalalignment**=**'top',bbox**=**props)

plt**.**show()

Code for the Advanced Non-linear model selection and processing

import numpy as np

import pandas as pd

import sklearn

import seaborn as sn

import matplotlib.pyplot as plt

from numpy import percentile

from scipy.stats import spearmanr

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

from scipy.stats import kendalltau

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

Importing and Analyzing the data

%cd /content/drive/MyDrive/Arman\_Thesis\_Work/

train\_input=pd.read\_excel('LearningVariables (1).xlsx',sheet\_name='X\_tr')

train\_output=pd.read\_excel('LearningY (1).xlsx',sheet\_name='X\_tr')

test\_input=pd.read\_excel('LearningVariables (1).xlsx',sheet\_name='X\_cv')

test\_output=pd.read\_excel('LearningY (1).xlsx',sheet\_name='X\_cv')

# Data Analysis

\*\*Step1: Checking data columns and values \*\*

train\_data=pd.concat([train\_input, train\_output], axis=1)

test\_data=pd.concat([test\_input, test\_output], axis=1)

train\_data.describe()

test\_data.describe()

\*\*Step2: Checking data shape \*\*

print("Shape of features train data",train\_input.shape)

print("Shape of Prediction or Y train data",train\_output.shape)

print("Shape of features test data",test\_input.shape)

print("Shape of Prediction or Y test data",test\_output.shape)

\*\*Step-3 Getting the list of all columns\*\*

feat\_columns=list(train\_input.columns)

feat\_columns

\*\*Step-4 Checking all information of the dataset\*\*

train\_data.info()

test\_data.info()

\*\*Step-5 Checking for the missing values\*\*

train\_input.isna().sum()

train\_output.isna().sum()

test\_input.isna().sum()

test\_output.isna().sum()

# Checking the distribution of data

sn.set(font\_scale=2)

train\_data.hist(figsize=(30,35),bins=20)

plt.show()

sn.set(font\_scale=2)

test\_data.hist(figsize=(35,35),bins=20)

plt.show()

def outliers\_removal(data\_frame, column):

# Step-1 sorting the data

sorted(data\_frame)

#Step-2 finding the quartiles

Q3=np.percentile(data\_frame[column], 75)

Q1=np.percentile(data\_frame[column], 25)

# Calculating Inter Quartile Range

IQR=Q3-Q1

# Calculating the bounds

lower\_bound=Q1 - 1.5 \* IQR

upper\_bound=Q3 + 1.5 \* IQR

# Removing the outliers

data\_frame.drop(data\_frame[ (data\_frame[column] > upper\_bound) | (data\_frame[column] < lower\_bound) ].index , inplace=True)

return data\_frame

sn.boxplot(y=train\_input['nodeline\_Y'])

plt.show()

No, outlier

'oos', 'oop', 'epskfy', 'epskfc', 'cover', 'time'

sn.boxplot(y=train\_input['oop'])

plt.show()

No, outlier

sn.boxplot(y=train\_input['epskfy'])

plt.show()

No, outlier

sn.boxplot(y=train\_input['epskfc'])

plt.show()

No outilers

sn.boxplot(y=train\_input['cover'])

plt.show()

No, outlier

sn.boxplot(y=train\_input['time'])

plt.show()

No, outlier

\*\*Checking Pearson Corrleation\*\*

Pearson check the linearity

Spearman check montonacity

Kindeal check the same as that of spearman in a error known mode

train\_data=pd.concat([train\_input, train\_output], axis=1)

train\_data.columns

sn.set(font\_scale=1)

fig, ax = plt.subplots(figsize=(10,10))

# Pearson Correlation

pearson\_corr\_data = train\_data.corr(method='pearson')

sn.heatmap(pearson\_corr\_data,annot=True)

ax.set\_title('Pearson Correlation')

plt.savefig("Pearson\_Corr.png")

plt.figure(figsize=(10,10))

pearson\_corr\_data['Load'].sort\_values().plot(kind="barh")

pearson\_corr\_data['Load'].sort\_values(ascending=False)

fig, ax = plt.subplots(figsize=(10,10))

# spearman Correlation

spearman\_corr\_data =train\_data.corr(method='spearman')

sn.heatmap(spearman\_corr\_data ,annot=True)

ax.set\_title('Spearman Correlation')

plt.savefig("Spearman\_Corr.png")

plt.figure(figsize=(10,10))

spearman\_corr\_data["Load"].sort\_values().plot(kind="barh")

spearman\_corr\_data['Load'].sort\_values(ascending=False)

fig, ax = plt.subplots(figsize=(10,10))

# spearman Correlation

kendal\_corr\_data = train\_data.corr(method='kendall')

sn.heatmap(kendal\_corr\_data ,annot=True)

ax.set\_title('Kendal Correlation')

plt.savefig("Kindall\_Corr.png")

plt.figure(figsize=(10,10))

kendal\_corr\_data["Load"].sort\_values().plot(kind="barh")

kendal\_corr\_data['Load'].sort\_values(ascending=False)

Correlation plot for the price column with the other features

sc\_X = StandardScaler()

X\_data = sc\_X.fit\_transform(train\_input)

data\_std=pd.DataFrame(X\_data)

sn.set(font\_scale=2)

data\_std.hist(figsize=(40,40),bins=20)

plt.show()

# Final Data Preparation

X\_train=train\_input.values

Y\_train=train\_output.values/1000

X\_test=test\_input.values

Y\_test=test\_output.values/1000

# Train and Test split

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.transform(X\_test)

## Selection of Regression algorithm

For the selection of suitable regression algorithm. We have to analyze the relationship between the attributes and the dependent variable.

As, the correlation between the inputs/attributes and the dependent variables is not above 0.7 of correlation measure. Therefore, it is indicating non-linear relation ship between the attributes and the dependent variable inspite of time which is indicating only relationship

\*\*Using simple Multi\_Linear Regression\*\*

from sklearn.linear\_model import LinearRegression

LM=LinearRegression()

LM.fit(X\_train,Y\_train)

Y\_pred=LM.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 8, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Using Ensemble Learning\*\*

Ensembel models ususally perform better as compared to the normal models in most of the cases as they use some basic individual model as base learners and then design a met-model from all of these in the final stage. Therefore, using gradient boosting as first model by using the XGBOOST library

import xgboost as xgb

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import RepeatedKFold

from xgboost import XGBRegressor

from numpy import absolute

#setting one

model=XGBRegressor()

model.fit(X\_train,Y\_train)

Y\_pred=model.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

mse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed mean squared error is = ",mse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

Tree Based Ensembel Exploring other settings

from xgboost import plot\_importance

from matplotlib import pyplot

plot\_importance(model)

pyplot.show()

---

\*\*Using Random Forest Regression\*\*

It's a type of ensemble

from sklearn.ensemble import RandomForestRegressor

RF\_reg=RandomForestRegressor(n\_estimators = 300, max\_features = 'auto', min\_samples\_leaf=2, random\_state=1)

RF\_reg.fit(X\_train,Y\_train)

Y\_pred=RF\_reg.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Neural Network\*\*

from sklearn.neural\_network import MLPRegressor

MLP\_regressor = MLPRegressor(max\_iter=1000)

MLP\_regressor.fit(X\_train,Y\_train)

Y\_pred=MLP\_regressor.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Support Vector Machine\*\*

from sklearn.svm import SVR

model = SVR(kernel='linear')

model.fit(X\_train, Y\_train)

Y\_pred=model.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*SVM with polynomial kernel degree 3\*\*

SVR\_poly=SVR(kernel='poly',degree=3,gamma= 'auto')

SVR\_poly.fit(X\_train,Y\_train)

Y\_pred=SVR\_poly.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*SVR with RBF\*\*

SVR\_rbf=SVR(kernel='rbf')

SVR\_rbf.fit(X\_train,Y\_train)

Y\_pred=SVR\_rbf.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*SVR with sigmoid\*\*

SVR\_sigmoid=SVR(kernel='sigmoid')

SVR\_sigmoid.fit(X\_train,Y\_train)

Y\_pred=SVR\_rbf.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Using K-Nearst Neighbor Regressor\*\*

from sklearn.neighbors import KNeighborsRegressor

KNNR=KNeighborsRegressor(n\_neighbors=20,algorithm='brute')

KNNR.fit(X\_train,Y\_train)

Y\_pred=KNNR.predict(X\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 8, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Polynomial Regression\*\*

from sklearn.preprocessing import PolynomialFeatures

polynomial\_features= PolynomialFeatures(degree=2)

X\_poly\_train = polynomial\_features.fit\_transform(X\_train)

X\_poly\_test = polynomial\_features.fit\_transform(X\_test)

regressor = LinearRegression()

regressor.fit(X\_poly\_train, Y\_train) #training the algorithm

Y\_pred = regressor.predict(X\_poly\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

polynomial\_features= PolynomialFeatures(degree=3)

X\_poly\_train = polynomial\_features.fit\_transform(X\_train)

X\_poly\_test = polynomial\_features.fit\_transform(X\_test)

regressor = LinearRegression()

regressor.fit(X\_poly\_train, Y\_train) #training the algorithm

Y\_pred = regressor.predict(X\_poly\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

polynomial\_features= PolynomialFeatures(degree=4)

X\_poly\_train = polynomial\_features.fit\_transform(X\_train)

X\_poly\_test = polynomial\_features.fit\_transform(X\_test)

regressor = LinearRegression()

regressor.fit(X\_poly\_train, Y\_train) #training the algorithm

Y\_pred = regressor.predict(X\_poly\_test)

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

---

\*\*Light Gardient Boosting\*\*

import lightgbm as lgb

gbm = lgb.LGBMRegressor()

gbm.fit(X\_train, Y\_train.flatten(), eval\_set=[(X\_test,Y\_test.flatten())],eval\_metric='neg\_mean\_absolute\_error',early\_stopping\_rounds=1000)

train\_pred=gbm.predict(X\_train)

Y\_pred=gbm.predict(X\_test)

print("R-squared on train data: ",r2\_score(Y\_train,train\_pred))

print("R-squared on test data: ",r2\_score(Y\_test,Y\_pred))

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

data=[0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]

rmse\_train={}

rmse\_test={}

for l in data:

gbm = lgb.LGBMRegressor(learning\_rate=l)

gbm.fit(X\_train,Y\_train.flatten())

Y\_pred\_train=gbm.predict(X\_train)

rmse\_train\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_train,Y\_train))

rmse\_train[l]=rmse\_train\_temp

Y\_pred\_test=gbm.predict(X\_test)

rmse\_test\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_test,Y\_test))

rmse\_test[l]=rmse\_test\_temp

plt.figure(figsize=(8,8))

value=min(rmse\_test, key=lambda k: rmse\_test[k])

plt.plot(list(rmse\_train.keys()),list(rmse\_train.values()),c='blue',label='Train',linewidth=2)

plt.plot(list(rmse\_test.keys()),list(rmse\_test.values()),color='red', linewidth=2,label="Test")

plt.legend()

plt.xlabel('Learning Rate')

plt.ylabel('Loss (RMSE)')

plt.tight\_layout()

textstr = '$Best Learning Rate=%.2f$' % value

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(0.1, 0.6, textstr, fontsize=14,

verticalalignment='top', bbox=props)

plt.show()

n\_estimators=[10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250]

rmse\_train={}

rmse\_test={}

for l in n\_estimators:

gbm = lgb.LGBMRegressor(n\_estimators=l,learning\_rate=0.2,random\_state=0)

gbm.fit(X\_train,Y\_train.flatten())

Y\_pred\_train=gbm.predict(X\_train)

rmse\_train\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_train,Y\_train))

rmse\_train[l]=rmse\_train\_temp

Y\_pred\_test=gbm.predict(X\_test)

rmse\_test\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_test,Y\_test))

rmse\_test[l]=rmse\_test\_temp

rmse\_train

rmse\_test

plt.figure(figsize=(10,10))

value=min(rmse\_test, key=lambda k: rmse\_test[k])

plt.plot(list(rmse\_train.keys()),list(rmse\_train.values()),c='blue',label='Train',linewidth=2)

plt.plot(list(rmse\_test.keys()),list(rmse\_test.values()),color='red', linewidth=2,label="Test")

plt.legend()

plt.xlabel('Numer of estimator')

plt.ylabel('Loss (RMSE)')

plt.tight\_layout()

textstr = '$Best Learning Rate=%.2f$' % value

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(100, 0.8, textstr, fontsize=14,

verticalalignment='top', bbox=props)

plt.show()

n\_sample\_train=[100,250,500,1000,2000,5000,9367]

rmse\_train={}

rmse\_test={}

for l in n\_sample\_train:

gbm = lgb.LGBMRegressor(n\_estimators=190,learning\_rate=0.2,random\_state=0)

gbm.fit(X\_train[:l,:],Y\_train[:l].flatten())

Y\_pred\_train=gbm.predict(X\_train)

rmse\_train\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_train,Y\_train))

rmse\_train[l]=rmse\_train\_temp

Y\_pred\_test=gbm.predict(X\_test)

rmse\_test\_temp=np.sqrt(mean\_squared\_error(Y\_pred\_test,Y\_test))

rmse\_test[l]=rmse\_test\_temp

plt.figure(figsize=(10,10))

value=min(rmse\_test, key=lambda k: rmse\_test[k])

plt.plot(list(rmse\_train.keys()),list(rmse\_train.values()),c='blue',label='Train',linewidth=2)

plt.plot(list(rmse\_test.keys()),list(rmse\_test.values()),color='red', linewidth=2,label="Test")

plt.legend()

plt.xlabel('Numer of traning samples')

plt.ylabel('Loss (RMSE)')

plt.tight\_layout()

textstr = '$Best traning sample number=%.2f$' % value

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(1000, 0.8, textstr, fontsize=14,

verticalalignment='top', bbox=props)

plt.show()

# Final Model

gbm = lgb.LGBMRegressor(random\_state=0,n\_estimator=190,learning\_rate=0.2)

gbm.fit(X\_train, Y\_train.flatten())

train\_pred=gbm.predict(X\_train)

Y\_pred=gbm.predict(X\_test)

print("R-squared on train data: ",r2\_score(Y\_train,train\_pred))

print("R-squared on test data: ",r2\_score(Y\_test,Y\_pred))

# Metrics

SROCC=spearmanr(Y\_test,Y\_pred)

KROCC=kendalltau(Y\_test,Y\_pred)

R\_square=r2\_score(Y\_test,Y\_pred)

rmse=np.sqrt(mean\_squared\_error(Y\_pred,Y\_test))

mae=mean\_absolute\_error(Y\_pred,Y\_test)

print("The computed root mean squared error is = ",rmse)

print("The computed mean absolute error is = ",mae)

print("The r2 measure for the model is = ",R\_square)

print("The Spearman for the data is = ",SROCC)

print("The Kendall correlation for the data is = ",KROCC)

plt.figure(figsize=(10,10))

plt.scatter(Y\_test, Y\_pred, c='blue',label='Reference Line')

plt.plot(Y\_test,Y\_test, color='red', linewidth=2,label="Model Prediction")

plt.xlabel('Y\_test')

plt.ylabel('Y\_pred')

plt.tight\_layout()

plt.legend()

textstr = '$R^{2}=%.2f$' % R\_square

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

plt.text(2, 6, textstr, fontsize=18,

verticalalignment='top', bbox=props)

plt.show()

plt.figure(figsize=(10,10))

lgb.plot\_importance(gbm)

plt.show()

params = {

'objective': 'regression',

'boosting\_type': 'gbdt',

'learning\_rate': 0.2, # 02,

'n\_estimator': 190,

'seed': 0,

'verbose': -1,

'metric': 'neg\_mean\_absolute\_error'

}

lgb\_train = lgb.Dataset(X\_train, label=Y\_train.flatten(), silent= False, free\_raw\_data=False)

clf = lgb.train(train\_set = lgb\_train, params= params, num\_boost\_round= 8000)

feat\_importance = pd.DataFrame()

feat\_importance['FEATURE'] = feats

feat\_importance['BY\_GAIN'] = clf.feature\_importance(importance\_type='gain').astype('int32')

feat\_importance['BY\_SPLIT'] = clf.feature\_importance(importance\_type='split').astype('int32')

feat\_importance = feat\_importance.sort\_values(by=['BY\_GAIN', 'BY\_SPLIT'], ascending=False)

plt.figure(figsize=(8, 10))

sn.barplot(x= 'BY\_GAIN', y= 'FEATURE', data= feat\_importance)

plt.title('Feature importance by gain')

plt.show()

plt.figure(figsize=(8, 10))

sn.barplot(x= 'BY\_SPLIT', y= 'FEATURE', data= feat\_importance)

plt.title('Feature importance by split')

plt.show()

Shap based analysis

!pip install shap

import shap

feats=list(train\_data.columns)

feats=['nodeline\_Y', 'oos', 'oop', 'epskfy', 'epskfc', 'cover', 'time']

train=pd.DataFrame(X\_train,columns=feats)

train\_df=pd.concat([train, train\_output], axis=1)

total\_columns=list(train\_df.columns)

total\_columns

shap\_values = shap.TreeExplainer(clf).shap\_values(X\_train)

shap\_df = pd.DataFrame(shap\_values, columns = feats)

shap.initjs()

shap.summary\_plot(shap\_df.values, train)

shap.dependence\_plot('nodeline\_Y', shap\_df.values, train)

shap.dependence\_plot('oos', shap\_df.values, train)

shap.dependence\_plot('epskfy', shap\_df.values, train)

shap.dependence\_plot('epskfc', shap\_df.values, train)

shap.dependence\_plot('cover', shap\_df.values, train)

shap.dependence\_plot('time', shap\_df.values, train)

# REFERENCE

[1] Y. Xie, M. Ebad Sichani, J. E. Padgett, and R. DesRoches, “The promise of implementing machine learning in earthquake engineering: A state-of-the-art review,” *Earthq. Spectra*, vol. 36, no. 4, pp. 1769–1801, 2020, doi: 10.1177/8755293020919419.

[2] M. Z. Naser, “AI-based cognitive framework for evaluating response of concrete structures in extreme conditions,” *Eng. Appl. Artif. Intell.*, vol. 81, no. February, pp. 437–449, 2019, doi: 10.1016/j.engappai.2019.03.004.

[3] H. O. W. To, E. Where, M. Learning, W. Contribute, and T. O. Optimal, “Applications for Machine Learning in Engineering and the Physical Sciences How To Evaluate Where Machine Learning.”

[4] M. Z. Naser, *Mechanistically Informed Machine Learning and Artificial Intelligence in Fire Engineering and Sciences*. Springer US, 2021.

[5] S. E. Magnusson *et al.*, “A proposal for a model curriculum in fire safety engineering,” *Fire Saf. J.*, vol. 25, no. 1, pp. 1–88, 1995, doi: 10.1016/S0379-7112(95)00038-0.

[6] J. Lee, S. Lee, and D. You, “Deep learning approach in multi-scale prediction of turbulent mixing-layer,” *arXiv*, pp. 1–21, 2018.

[7] T. P. Miyanawala and R. K. Jaimana, “An efficient deep learning technique for the Navier-Stokes equations: Application to unsteady wake flow dynamics,” *arXiv*, 2017.

[8] S. Lee and D. You, “Data-driven prediction of unsteady flow over a circular cylinder using deep learning,” *J. Fluid Mech.*, vol. 879, no. Schmid 2010, pp. 217–254, 2019, doi: 10.1017/jfm.2019.700.

[9] C. W. Chang and N. T. Dinh, “Classification of machine learning frameworks for data-driven thermal fluid models,” *Int. J. Therm. Sci.*, vol. 135, pp. 559–579, 2019, doi: 10.1016/j.ijthermalsci.2018.09.002.

[10] J. L. Hodges, B. Y. Lattimer, and K. D. Luxbacher, “Compartment fire predictions using transpose convolutional neural networks,” *Fire Saf. J.*, vol. 108, no. November 2018, p. 102854, 2019, doi: 10.1016/j.firesaf.2019.102854.

[11] B. Y. Lattimer, J. L. Hodges, and A. M. Lattimer, “Using machine learning in physics-based simulation of fire,” *Fire Saf. J.*, vol. 114, no. July, 2020, doi: 10.1016/j.firesaf.2020.102991.

[12] R. K. K. Yuen, E. W. M. Lee, S. M. Lo, and G. H. Yeoh, “Prediction of temperature and velocity profiles in a single compartment fire by an improved neural network analysis,” *Fire Saf. J.*, vol. 41, no. 6, pp. 478–485, 2006, doi: 10.1016/j.firesaf.2006.03.003.

[13] J. L. Hodges, “Predicting Large Domain Multi-Physics Fire Behavior Using Artificial Neural Networks,” *PhD Diss. Virginia Tech, Dep. Mech. Eng.*, no. December, p. 205, 2018.

[14] C. Basdevant *et al.*, “Spectral and finite difference solutions of the Burgers equation,” *Comput. Fluids*, vol. 14, no. 1, pp. 23–41, 1986, doi: 10.1016/0045-7930(86)90036-8.

[15] B. Xiong, H. Fu, F. Xu, and Y. Jin, “Data-driven discovery of partial differential equations for multiple-physics electromagnetic problem,” *arXiv*, no. April, pp. 1–7, 2019.

[16] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019, doi: 10.1016/j.jcp.2018.10.045.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Machine learning of linear differential equations using Gaussian processes,” *J. Comput. Phys.*, vol. 348, no. 1, pp. 683–693, 2017, doi: 10.1016/j.jcp.2017.07.050.

[18] M. Raissi and G. E. Karniadakis, “Hidden physics models: Machine learning of nonlinear partial differential equations,” *J. Comput. Phys.*, vol. 357, pp. 125–141, 2018, doi: 10.1016/j.jcp.2017.11.039.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations,” *arXiv*, no. Part I, pp. 1–22, 2017.

[20] R. Ripke and D. Version, “h e i re e rf o rmanc e o f t e e l f ac e d The Fire Performance of ns ul at i o n Panels ane l s w i t h t o ne o o l o r Insulation with Stone Wool or o l y me r Cores o re s Polymer c al i ng and e rans f e r Study t y scaling Heat Transfer s o ul l c,” 2018.

[21] A. Dexters, R. R. Leisted, R. Van Coile, S. Welch, and G. Jomaas, “Testing for knowledge: Application of machine learning techniques for prediction of flashover in a 1/5 scale ISO 13784-1 enclosure,” *Fire Mater.*, no. April, pp. 1–12, 2020, doi: 10.1002/fam.2876.

[22] K. Wang, X. Shi, A. P. X. Goh, and S. Qian, “A machine learning based study on pedestrian movement dynamics under emergency evacuation,” *Fire Saf. J.*, vol. 106, no. April, pp. 163–176, 2019, doi: 10.1016/j.firesaf.2019.04.008.

[23] M. Liu and S. M. Lo, “The quantitative investigation on people’s pre-evacuation behavior under fire,” *Autom. Constr.*, vol. 20, no. 5, pp. 620–628, 2011, doi: 10.1016/j.autcon.2010.12.004.

[24] S. M. Lo, M. Liu, P. H. Zhang, and R. K. K. Yuen, “An artificial neural-network based predictive model for pre-evacuation human response in domestic building fire,” *Fire Technol.*, vol. 45, no. 4, pp. 431–449, 2009, doi: 10.1007/s10694-008-0064-6.

[25] Y. Y. Wei, J. Y. Zhang, and J. Wang, “Research on Building Fire Risk Fast Assessment Method Based on Fuzzy comprehensive evaluation and SVM,” *Procedia Eng.*, vol. 211, pp. 1141–1150, 2018, doi: 10.1016/j.proeng.2017.12.121.

[26] L. Yang, C. W. Dawson, M. R. Brown, and M. Gell, “Neural network and GA approaches for dwelling fire occurrence prediction,” *Knowledge-Based Syst.*, vol. 19, no. 4, pp. 213–219, 2006, doi: 10.1016/j.knosys.2005.11.021.

[27] Z. G. Liu, X. Y. Li, and G. Jomaas, “Identifying Community Fire Hazards from Citizen Communication by Applying Transfer Learning and Machine Learning Techniques,” *Fire Technol.*, 2020, doi: 10.1007/s10694-020-01035-4.

[28] V. K. Kodur and M. Z. Naser, “Classifying bridges for the risk of fire hazard via competitive machine learning,” *Adv. Bridg. Eng.*, vol. 2, no. 1, 2021, doi: 10.1186/s43251-020-00027-2.

[29] M. Z. Naser and A. Seitllari, “Concrete under fire: an assessment through intelligent pattern recognition,” *Eng. Comput.*, vol. 36, no. 4, pp. 1915–1928, 2020, doi: 10.1007/s00366-019-00805-1.

[30] M. Z. Naser, “Fire resistance evaluation through artificial intelligence - A case for timber structures,” *Fire Saf. J.*, vol. 105, no. February, pp. 1–18, 2019, doi: 10.1016/j.firesaf.2019.02.002.

[31] M. Z. Naser, “Properties and material models for modern construction materials at elevated temperatures,” *Comput. Mater. Sci.*, vol. 160, no. December 2018, pp. 16–29, 2019, doi: 10.1016/j.commatsci.2018.12.055.

[32] M. Bilgehan and A. E. Kurtoğlu, “ANFIS-based prediction of moment capacity of reinforced concrete slabs exposed to fire,” *Neural Comput. Appl.*, vol. 27, no. 4, pp. 869–881, 2016, doi: 10.1007/s00521-015-1902-3.

[33] M. Z. Naser, “Autonomous Fire Resistance Evaluation,” *J. Struct. Eng.*, vol. 146, no. 6, p. 04020103, 2020, doi: 10.1061/(asce)st.1943-541x.0002641.

[34] M. Lazarevska, M. Kneževiá, M. Cvetkovska, N. Ivaniševič, T. Samardzioska, and A. Trombeva-Gavriloska, “Fire-resistance prognostic model for reinforced concrete columns,” *Gradjevinar*, vol. 64, no. 7, pp. 565–571, 2012, doi: 10.14256/jce.674.2011.

[35] M. Z. Naser, “Deriving temperature-dependent material models for structural steel through artificial intelligence,” *Constr. Build. Mater.*, vol. 191, pp. 56–68, 2018, doi: 10.1016/j.conbuildmat.2018.09.186.

[36] H. Erdem, “Prediction of the moment capacity of reinforced concrete slabs in fire using artificial neural networks,” *Adv. Eng. Softw.*, vol. 41, no. 2, pp. 270–276, 2010, doi: 10.1016/j.advengsoft.2009.07.006.

[37] F. Fu, “Fire induced progressive collapse potential assessment of steel framed buildings using machine learning,” *J. Constr. Steel Res.*, vol. 166, 2020, doi: 10.1016/j.jcsr.2019.105918.

[38] M. Z. Naser, “Heuristic machine cognition to predict fire-induced spalling and fire resistance of concrete structures,” *Autom. Constr.*, vol. 106, no. August, 2019, doi: 10.1016/j.autcon.2019.102916.

[39] M. Z. Naser, “Observational Analysis of Fire-Induced Spalling of Concrete through Ensemble Machine Learning and Surrogate Modeling,” *J. Mater. Civ. Eng.*, vol. 33, no. 1, p. 04020428, 2021, doi: 10.1061/(asce)mt.1943-5533.0003525.

[40] R. K. Chaudhary, B. Jovanović, T. Gernay, and R. Van Coile, “Generalized fragility curves for concrete columns exposed to fire through surrogate modelling,” 2020, doi: 10.14264/45b645e.

[41] B. Ko, K. H. Cheong, and J. Y. Nam, “Early fire detection algorithm based on irregular patterns of flames and hierarchical Bayesian Networks,” *Fire Saf. J.*, vol. 45, no. 4, pp. 262–270, 2010, doi: 10.1016/j.firesaf.2010.04.001.

[42] Y. Okayama, “A primitive study of a fire detection method controlled by artificial neural net,” *Fire Saf. J.*, vol. 17, no. 6, pp. 535–553, 1991, doi: 10.1016/0379-7112(91)90052-Z.

[43] F. Derbel, “Performance improvement of fire detectors by means of gas sensors and neural networks,” *Fire Saf. J.*, vol. 39, no. 5, pp. 383–398, 2004, doi: 10.1016/j.firesaf.2004.03.001.

[44] J. Gubbi, S. Marusic, and M. Palaniswami, “Smoke detection in video using wavelets and support vector machines,” *Fire Saf. J.*, vol. 44, no. 8, pp. 1110–1115, 2009, doi: 10.1016/j.firesaf.2009.08.003.

[45] T. X. Tung and J. M. Kim, “An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems,” *Fire Saf. J.*, vol. 46, no. 5, pp. 276–282, 2011, doi: 10.1016/j.firesaf.2011.03.003.

[46] F. Yuan, “Video-based smoke detection with histogram sequence of LBP and LBPV pyramids,” *Fire Saf. J.*, vol. 46, no. 3, pp. 132–139, 2011, doi: 10.1016/j.firesaf.2011.01.001.

[47] B. Ko, J. H. Jung, and J. Y. Nam, “Fire detection and 3D surface reconstruction based on stereoscopic pictures and probabilistic fuzzy logic,” *Fire Saf. J.*, vol. 68, pp. 61–70, 2014, doi: 10.1016/j.firesaf.2014.05.015.

[48] T. Chaudhry and K. Moinuddin, “Method of identifying burning material from its smoke using attenuation of light,” *Fire Saf. J.*, vol. 93, no. June 2016, pp. 84–97, 2017, doi: 10.1016/j.firesaf.2017.08.001.

[49] F. Vandecasteele, B. Merci, and S. Verstockt, “Fireground location understanding by semantic linking of visual objects and building information models,” *Fire Saf. J.*, vol. 91, no. February, pp. 1026–1034, 2017, doi: 10.1016/j.firesaf.2017.03.083.

[50] G. Lin, Y. Zhang, G. Xu, and Q. Zhang, “Smoke Detection on Video Sequences Using 3D Convolutional Neural Networks,” *Fire Technol.*, vol. 55, no. 5, pp. 1827–1847, 2019, doi: 10.1007/s10694-019-00832-w.

[51] N. Alamgir, K. Nguyen, V. Chandran, and W. Boles, “Combining multi-channel color space with local binary co-occurrence feature descriptors for accurate smoke detection from surveillance videos,” *Fire Saf. J.*, vol. 102, no. November 2017, pp. 1–10, 2018, doi: 10.1016/j.firesaf.2018.09.003.

[52] J. Ji, C. Liu, Z. Gao, and L. (Leon) Wang, “Real-time forecasting of fire in a two-story building using ensemble Kalman filter method,” *Fire Saf. J.*, vol. 97, no. February, pp. 19–28, 2018, doi: 10.1016/j.firesaf.2018.02.007.

[53] C. Li, B. Yang, H. Ding, H. Shi, X. Jiang, and J. Sun, “Real-time video-based smoke detection with high accuracy and efficiency,” *Fire Saf. J.*, vol. 117, no. June, p. 103184, 2020, doi: 10.1016/j.firesaf.2020.103184.

[54] L. Kou, X. Wang, X. Guo, J. Zhu, and H. Zhang, “Deep learning based inverse model for building fire source location and intensity estimation,” *Fire Saf. J.*, vol. 121, no. March 2020, p. 103310, 2021, doi: 10.1016/j.firesaf.2021.103310.

[55] R. K. Chaudhary, R. Van Coile, and T. Gernay, “Potential of Surrogate Modelling for Probabilistic Fire Analysis of Structures,” *Fire Technol.*, 2021, doi: 10.1007/s10694-021-01126-w.