mathematics

Article

Broad Embedded Logistic Regression Classifier for Prediction of Air Pressure Systems Failure

Adegoke A. Muideen [1,2], Carman Ka Man Lee [1,3,*], Jeffery Chan [1], Brandon Pang [1] and Hafiz Alaka [2]

1 Centre For Advances in Reliability and Safety (CAiRS), Hong Kong, China
2 Big Data Technologies and Innovation Laboratory, University of Hertfordshire, Hatfield AL10 9AB, UK
3 Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China
* Correspondence: ckm.lee@polyu.edu.hk

Abstract: In recent years, the latest maintenance modelling techniques that adopt the data-based method, such as machine learning (ML), have brought about a broad range of useful applications. One of the major challenges in the automotive industry is the early detection of component failure for quick response, proper action, and minimizing maintenance costs. A vital component of an automobile system is an air pressure system (APS). Failure of APS without adequate and quick responses may lead to high maintenance costs, loss of lives, and component damages. This paper addresses classification problem where we detect whether a fault does or does not belong to APS. If a failure occurs in APS, it is classified as positive class; otherwise, it is classified as negative class. Hence, in this paper, we propose broad embedded logistic regression (BELR). The proposed BELR is applied to predict APS failure. It combines a broad learning system (BLS) and logistic regression (LogR) classifier as a fusion model. The proposed approach capitalizes on the strength of BLS and LogR for a better APS failure prediction. Additionally, we employ the BLS's feature-mapped nodes for extracting features from the input data. Additionally, we use the enhancement nodes of the BLS to enhance the features from feature-mapped nodes. Hence, we have features that can assist LogR for better classification performances, even when the data is skewed to the positive class or negative class. Furthermore, to prevent the curse of dimensionality, a common problem with high-dimensional data sets, we utilize principal component analysis (PCA) to reduce the data dimension. We validate the proposed BELR using the APS data set and compare the results with the other robust machine learning classifiers. The commonly used evaluation metrics, namely Recall, Precision, an F1-score, to evaluate the model performance. From the results, we validate that performance of the proposed BELR.

Keywords: artificial intelligence; automotive; condition monitoring; machine learning; predictive maintenance

MSC: 68T07

check for updates

Citation: Muideen, A.A.; Lee, C.K.M.; Chan, J.; Pang, B.; Alaka, H. Broad Embedded Logistic Regression Classifier for Prediction of Air Pressure Systems Failure. *Mathematics* **2023**, *11*, 1014. https://doi.org/10.3390/math11041014

Academic Editor: Tao Zhou

Received: 23 December 2022
Revised: 30 January 2023
Accepted: 9 February 2023
Published: 16 February 2023

1. Introduction

Air Processing System (APS) is a critical component in any brake system of heavy-duty vehicles. It plays an essential role in gauging brakes, controlling suspensions, shifting gears, etc. The effects of faulty APS are numerous. For instance, a faulty APS can cause improper functioning of gears, brakes, and suspension. These may lead to unpleasant/undesired situations such as total breakdown of the vehicles, which may lead to high maintenance costs and sometimes, in a critical situation, loss of life. To mitigate this side effect, the proper functioning of APS should be ensured; hence, its monitoring is vital. APS is said to be working properly when the APS supplies compressed air to its major components in an efficient, adequate, and timely manner.

Typically, in an automobile, the main components of APS are control units, circuit protection valves, and air dryers. A circuit-protection valve controls various circuits. Some of the circuits are a service brake circuit, a parking brake circuit, an auxiliary circuit, etc. It simply does this by activating them using different pre-set pressures. Furthermore, the air dryer removes excess moisture from the inlet air generated at the compressor. The control units dictate when to activate the compressor based on the pressure level in the APS. The control units consist of pressure sensors and temperature sensors.

APS failure detection is a key area of research [1,2]. It is a problem to detect whether an APS failure is the cause of a complete system breakdown or not. APS failures can result in huge maintenance costs and sometimes life-threatening situations. In recent years, machine learning-based techniques for APS failure detection are becoming increasingly popular. This is partly due to the large historical data set and the emergence of Industry 4.0 and the Industrial Internet of Things (IIoT). The core problems for APS failure detection/prediction are associated with:

- High volume of missing values in the data.
- Strongly imbalanced distribution of classes.

Figure 1 presents the missing values and imbalanced class distribution for APS failure data sets. From the figures, we notice many missing values in the data set. Hence, to tackle the problem, we cannot use the commonly used deletion method, as the missing value are many and deletion methods will cause large chunks of the data to be deleted. This will leave no or fewer data to be explored or used for the machine learning task. It is obvious from Figure 1 that the number of negative class cases is far higher than the number of positive class cases. Hence, the data set is highly imbalanced.

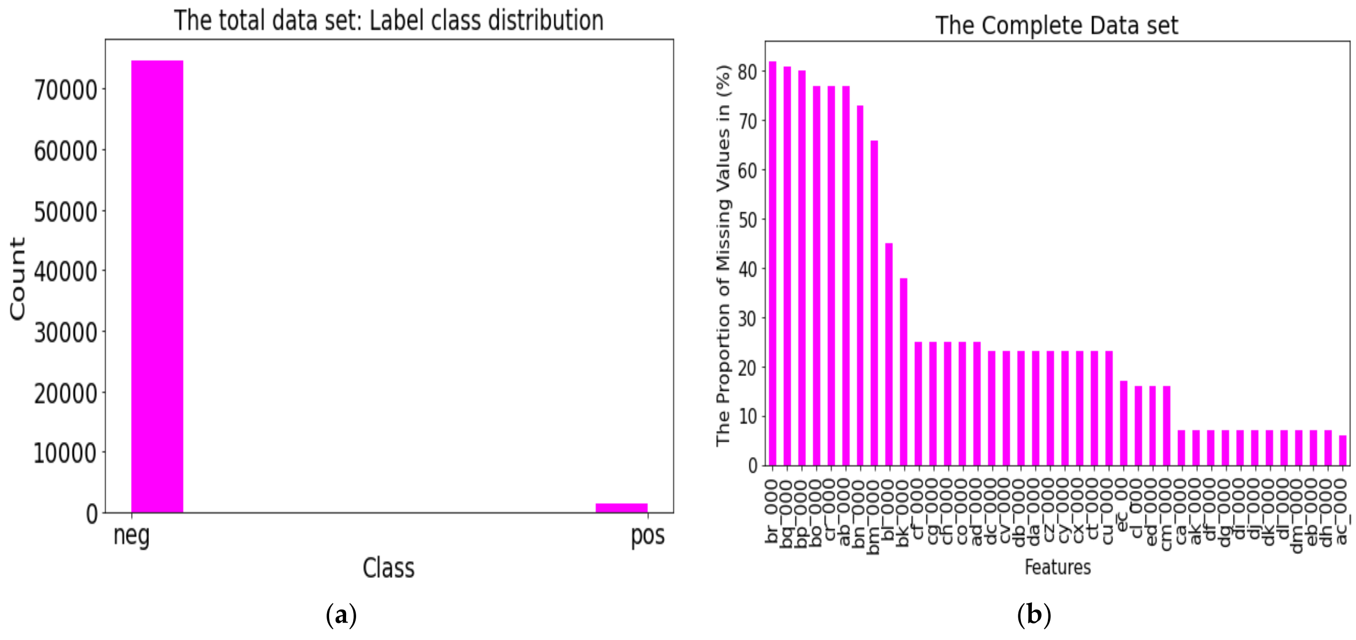


Figure 1. The distribution of the data set: (**a**) shows the class distribution and the imbalance of class distribution for the complete data set; (**b**) shows the percentage of missing values in each feature of the data set.

The two problems, namely class imbalance and missing values, can affect the performance of any machine learning algorithm if proper care is not taken [3,4]. Many researchers have worked on various techniques for handling missing data in APS data sets [1,5]. The existing techniques combined the classical data imputation methods with the traditional machine learning methods. However, the impact of modern data imputation methods is not explored. This could improve the performance of the machine learning method significantly. Additionally, a using modern machine learning method such as flat structure neural networks can enhance performance when the data is skewed to the positive class or negative class.

Additionally, a deep neural network with traditional classifiers can be used for many classification tasks. For instance, in [6], a deep neural network is trained together with a classifier where highly efficient features are extracted from raw data by a deep neural network, and a classifier, logistic regression [7], is used for classification. However, using a deep neural network requires high computation resources, and it takes a long time to train. Instead of using deep structure, some researchers have investigated random vector functional-link neural networks (RVFLNN). Chen and Liu [8] proposed BLS based on the concept of RVFLNN and obtained a promising result in classification accuracy and learning speed. The BLS network [8,9] is different from the deep neural network in many aspects, and its structure can be constructed widely. Additionally, the BLS network adopted incremental learning, which has quick remodelling without needing to re-train the network from scratch, provided that the performance of the initial network is not acceptable. In essence, the BLS network can be trained quickly and has good generalization performance. Additionally, BLS is regarded as a universal approximator with sufficient nodes in the network.

This paper explores the strength of a broad learning system and logistic regression for APS failure prediction. The BLS network is a flat structure neural network. It has a feature-mapped layer and a feature-enhancement layer. In addition, another common issue in a typical real-life big data set is high dimension. The high dimensional data set can affect the machine learning algorithm's performance. Additionally, it will increase the computational cost. To handle this, we explore principal component analysis (PCA) [10]. PCA is the widely used dimensionality reduction algorithm. Hence, to prevent the issue of the curse of dimensionally [11], we use PCA to reduce the dimension of the data set.

Furthermore, to the best of our knowledge, previous studies on APS failure detection have not investigated the performance of a flat structure neural network. A typical flat structure neural network example is a broad learning system. It has a broad layer structure where nodes are connected widely. More details of the network will be given in Section 3.

Generally, many machine learning models have difficulty handling imbalanced class distribution problems. However, the proposed approach can perform well under imbalanced class distribution problems. The proposed BELR does not require an additional or external method for imbalanced class distribution, as the feature-mapped nodes can extract features discriminative enough to enhance a classifier to separate classes from each other. The feature can be further enhanced by feature-enhancement nodes to uniquely classify each of the classes.

In summary, our method performance is reliable under challenging data sets such as the APS data set, which has an imbalance distribution problem. The idea of a missing mechanism is formalized in [12,13] where the study of missing data such as Missing at random (MAR), Missing not at random (MNAR), and Missing completely at random (MCAR) are presented in details. The data we employ in this paper have missing values across the feature set. A total of 169 features have missing values out of 170 features. Figure 2 shows the pattern of missing values in the data set across the features. From the Figure, the purple color represents the missing values, while the white or clear space represents where values exist in the data.

Additionally, the distribution of the data set and missing data is presented in [14]. In addition, Figure 2 shows the pattern of the missing values. From the idea in [12,13] and from the literature, the pattern of the missing values in the APS data set may be missing completely at random (MCAR). In our paper, the focus is not to categorize the pattern of the missing value in the APS data set. However, an appropriate method can be selected based on missing data mechanism in the data set. Thus, to prevent the destruction of the data set that could come from some missing value imputation method, we employ KNN imputer. In other words, we impute the missing value using the KNN technique [15]. Other methods such as imputation based on generative adversarial network (GAN) [14] could be explored. In [16], median imputation is explored for the problem of missing value. In this paper, we explore KNN imputation concept to tackle the problem of missing value. Figure 3 shows the pipeline of the proposed approach.

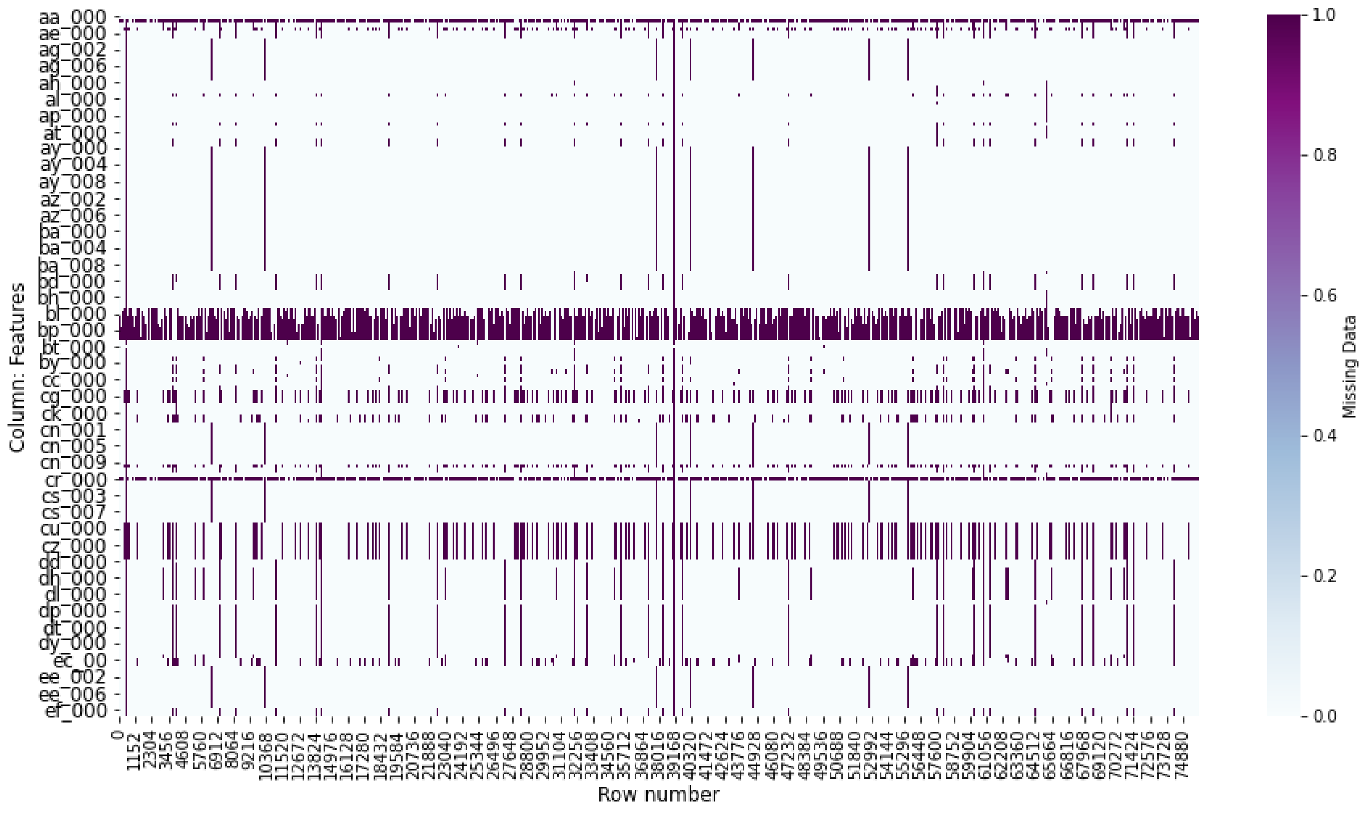


Figure 2. Pattern of the missing values in the data set across the features.

In summary, the contributions of this paper are summarized as follows:

- We propose broad embedded logistic regression (BELR). It is the fusion of broad learning system and logistic regression. We apply the proposed BELR to predict APS's failure.
- We propose a hybrid objective function based on the classical logistic regression objective function.
- We impute the missing value using the KNN algorithm.
- We propose and explore feature-mapped nodes of the BLS to extract discriminative features from the input data and enhancement nodes for further separation of the two classes such that the skewed distribution data set cannot affect the performance of the proposed broad embedded logistic regression (BELR).
- We explore principal component analysis (PCA) for dimensionality reduction and combine BLS and logistic regression classifier for the prediction of air pressure failure detection.

The rest of the paper is presented as follows. Section 2 presents related work on APS failure prediction from the literature. In Section 3, we describe the broad learning system (BLS) and give the mathematical model of the BLS. Additionally, Logistic regression (LogR) is discussed. Section 4 presents the experiment where the APS data set is described, and the numerical results are presented. In addition, in Section 4, the performances of the comparison algorithms and results are discussed. Section 5 gives the conclusion.

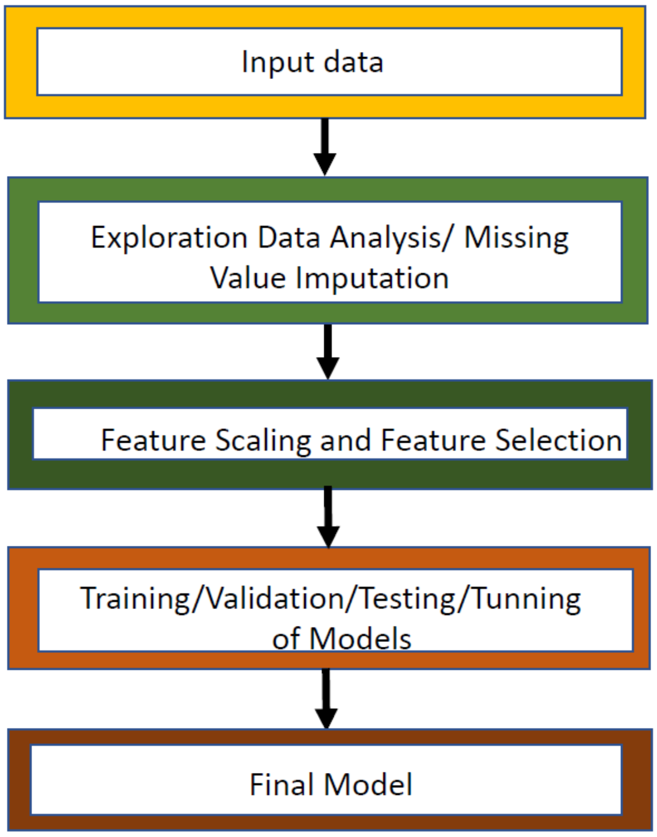

Input data
Exploration Data Analysis/ Missing Value Imputation
Feature Scaling and Feature Selection
Training/Validation/Testing/Tunning of Models
Final Model

Figure 3. The Flow Chart of the Experimental Process.

2. Related Work

Diagnosis of transportation systems is a common task in the automotive industry. The problem is commonly handled using data analysis and machine learning methods. In this section, we focus on related works to APS failure prediction. Additionally, we present related work on the imbalanced classification problem from the literature.

2.1. APS Failure Prediction

First, standard machine learning approaches for APS failure prediction have been applied to the mentioned task. There are works on APS failure prediction. For instance, in [16,17], the failure of APS of heavy-duty vehicles is studied. In the paper, the weighted loss function is employed to improve the performance of the network architecture used. In addition, in [18], a fuzzy-based machine learning algorithm is utilized for air pressure failure prediction. The fuzzy-based algorithm was combined with a relaxed prediction horizon for better air-pressure failure-prediction performance. Furthermore, APS failure prediction was analyzed by [16,19] using various machine learning algorithms, namely Support vector machines (SVM), Multi-Layer Perception (MLP), and Naive Bayes. The author extracts a feature from the raw data set using the feature engineering method namely histogram. In addition, feature ranking was implemented in their feature-selection approach. In the work, the preprocessing method used for the missing value replacement is KNN imputation, where the nearest neighbour in each feature column replaces the missing value. The metric used in terms of cost is based on total misclassification by the algorithm. In the metric, fp is set as a false positive, which predicts the failure wrongly, and fn is set as a false negative, which is missing a failure. In their proposed approach, missing a failure has a cost of 500, while the cost of falsely predicting a failure is given by 10. In their approach, a mean cost of 0.6 was achieved. The mean cost is given by $\frac{1}{Number\ of\ test\ sample}(10 * fp + 500 * fn)$.

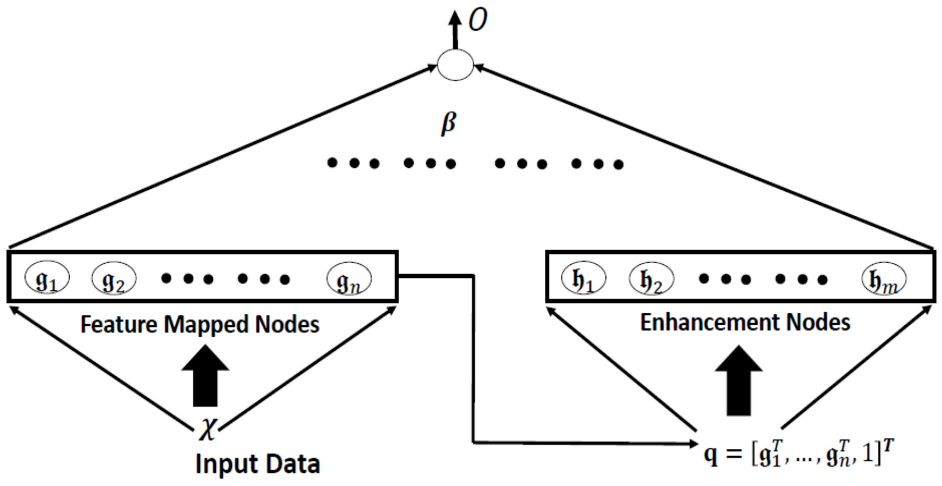
Additionally, to consider imbalance class issues, in some works, weighted data classifiers are used for APS failure prediction [14], logistic regression (LogR), and SVM classifiers. In their method, class-specific weights were integrated into the classifier. The value of the weight for each class is chosen such that it is inversely proportional to the number of samples in a class. Other classical machine learning methods have been applied to the APS data set. For instance, the performance of many machine learning techniques on the APS data set was investigated in Refs. [20–22]. The problem is a binary classification task. In their approach, they resolve the class imbalance problem with the aid of SMOTE (Synthetic Minority Oversampling Technique) algorithm to balance the positive class and negative class examples. Additionally, the author performs feature engineering before applying the machine learning algorithm. Besides, a new method for predicting APS failure was proposed in Ref. [23]. The method maximizes Area Under the Curves (max AUC) by utilizing a linear decision boundary. It is specifically designed to handle imbalance class distribution in the data set.

From all the previous studies on APS failure prediction, most authors focus on exploring the classical machine learning algorithms such as SVM, KNN, NB, LogR, etc. In summary, to the best of our knowledge, no work on APS failure prediction has used the neural network or flat structure-based machine learning methods, namely extreme learning machine (ELM) [24–26] and broad learning system (BLS) [8,26–29]. However, ELM and BLS algorithms are popular among researchers, and they are widely used in many applications. This is partly because they are universal approximators; with sufficient hide nodes, they can estimate any functions. Additionally, they are fast and easy to implement. Hence, in this work, we proposed to combine a broad learning system (BLS) and logistic regression (LogR) to predict APS failure. The background details of BLS and LogR are given in the next sections.

2.2. *Broad Learning System (BLS) and Logistic Regression (LogR)*

2.2.1. Broad Learning System (BLS)

The concept of BLS [8] is a new technique. BLS and other variants [8,30,31] connect hidden nodes of a neural network broadly. As shown in Figure 4, the nodes are put together in a broad flat structure. A BLS network contains two hidden layers, namely the enhancement layer and feature-mapped layer.



O
β
$\mathfrak{g}_1$ $\mathfrak{g}_2$ $\mathfrak{g}_n$
Feature Mapped Nodes
$\mathfrak{h}_1$ $\mathfrak{h}_2$ $\mathfrak{h}_m$
Enhancement Nodes
χ
Input Data
$\mathfrak{q} = [\mathfrak{g}_1^T, \ldots, \mathfrak{g}_n^T, 1]^T$

Figure 4. A typical structure of a BLS network.

The concept introduced in the BLS framework is promising. It is an efficient, simple learning algorithm. Due to the efficient feature-extraction capacity of the nodes in the feature-mapped layer and enhancement layer of the BLS, the original BLS and hybrid methods, where feature-mapped layer of BLS and other techniques are combined, have been used in many applications. However, much work has not been completed using neural network-based algorithms to predict APS failure in the case of APS failure. In view of the points and that BLS's feature-mapped nodes and enhancement nodes can extract effective features from the input data, which can enhance the performance of a classifier, we combine BLS and logistic regression (LogR) to study APS failure prediction. Thus, we propose broad embedded logistic regression (BELR) for APS failure prediction.

2.2.2. Operation of BLS Networks

This subsection gives background knowledge on the operation of a BLS network. This paper proposed BLS for solving the air pressure system failure classification problem. In this paper, the classification problem is formulated as nonlinear logistic regression, where the input of a logistic regression algorithm is the feature from the BLS network. The final output is the sign of the predicted value or the probability of the predicted output. In other words, if the sign is positive, then the predicted value belongs to the positive class. Otherwise, the predicted class is negative. Let $\mathbf{x} \in \mathbb{R}^D$ be the input data to the BLS network, where D is the dimension of the input data, and $o \in \mathbb{R}$ be the output of a BLS network. In this section, for smooth and clear presentation, we present the input $\mathbf{x}$ augmented with 1 as $\boldsymbol{\chi} = [\mathbf{x}^{\mathrm{T}}, 1]^{\mathrm{T}}$.

(a) Feature-mapped nodes

The BLS network has two main layers, namely feature-mapped layer and enhancement layer. The feature-mapped layer is to extract features from input data. For the feature-mapped layer, there are n groups of the feature-mapped nodes. These n groups are concatenated together to form one main feature extraction. The output of the main feature-extraction group is passed to the output layer, and another layer is called the enhancement layer. Each group from the n groups is used to extract distinctive features. Each group has it on a specific number of nodes. For instance, in this paper, f_i represents the i-th group of the feature-mapped nodes. Hence, for n groups of features-mapped nodes, the total number of features-mapped nodes is the following:

$$f = \sum_{i=1}^{n} f_i \tag{1}$$

It should be noted that f_i for $i = 1, \ldots n$ may not be equal. For each group of the feature-mapped node, which is the i-th group of features-mapped nodes, there is an associated learned projection matrix, and the i-th learned projection matrix is given by:

$$\boldsymbol{\Psi}_i = \begin{pmatrix} \psi_{i,1,1} & \cdots & \psi_{i,1,(D+1)} \\ \vdots & \ddots & \vdots \\ \psi_{i,f_i,1} & \cdots & \psi_{i,f_i,(D+1)} \end{pmatrix} \tag{2}$$

where $\boldsymbol{\Psi}_i \in \mathbb{R}^{f_i \times (D+1)}$. It is designed to generate features from the input data. The i-th group of mapped features g_i are obtained by projecting the input data with the matrix $\boldsymbol{\Psi}_i$. They are given by the following:

$$\begin{aligned} \mathfrak{g}_i &= && [g_{i,1}, \cdots, g_{i,f_i}]^{\mathrm{T}} \\ &= \boldsymbol{\Psi}_i \boldsymbol{\chi} && \forall\, i = 1, \cdots, n, \end{aligned} \tag{3}$$

where $g_{i,u}$ is the u-th feature of the u-th group, where $i = 1, \cdots, n$, and $u = 1, \cdots, f_i$.

In the classical BLS, $\boldsymbol{\Psi}_i{}'$s is constructed based on sparse optimization steps. There are many ways to achieve these steps. One way is to solve sparse optimization problems based on the alternating direction method of multipliers ADMM [32] algorithm. In Section 2.2.3-(a), we present the construction procedure of $\boldsymbol{\Psi}_i{}'$s. In the classical BLS scheme, a linear operation is applied on $\mathfrak{g}_i$s. It should be noted that $\mathfrak{g}_i$s is not $\boldsymbol{\chi}$ but features-extracted from $\boldsymbol{\chi}$. Similarly, a nonlinear operation can be applied on $\mathfrak{g}_i$'s as well.

In this paper, we apply a linear operation on $\mathfrak{g}_i$'s, that is, this paper follows the classical BLS framework. The outputs from the n group of the feature-mapped nodes are gathered as

$$\mathbf{g} = [\mathfrak{g}_1^{\mathrm{T}}, \cdots, \mathfrak{g}_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^f \tag{4}$$

Additionally, we let

$$\mathbf{q} = [\mathbf{g}^{\mathrm{T}}, 1]^{\mathrm{T}} \in \mathbb{R}^{f+1} \tag{5}$$

for a smooth mathematical model presentation and as the augmented vector of **g**.

(b) Enhancement nodes

Like the feature-mapped nodes, the enhancement nodes of the BLS network have m groups of enhancement nodes. In the enhancement layer, the j-th group of enhancement nodes has e_j nodes. The total number of enhancement nodes in the BLS network is given by

$$e = \sum_{j=1}^{m} e_j \tag{6}$$

In addition, the output of j-th group of enhancement nodes is given by

$$\mathfrak{H}_j = \left[\mathfrak{H}_{j,1}, \ldots \mathfrak{H}_{j,e_j}\right]^T = \xi(\mathbf{W}_j\, \mathbf{q}) \tag{7}$$

where $j = 1, \cdots, m$ and $\mathbf{W}_j$ is the weight that connects the output of feature-mapped nodes to the input of the enhancement nodes together. It should be known that, in the original BLS framework, $\mathbf{W}_j$ is a randomly generated. The elements of $\mathbf{W}_j$ are denoted as

$$\mathbf{W}_j = \begin{pmatrix} w_{j,1,1} & \cdots & w_{j,1,f+1} \\ \vdots & \ddots & \vdots \\ w_{j,e_j,1} & \cdots & w_{j,e_j,f+1} \end{pmatrix}. \tag{8}$$

It should be known that $\xi(\cdot)$ is the activation function for enhancement nodes. Each group of enhancement nodes can have its activation function. In the original BLS algorithm, the hyperbolic tangent is employed as the activation function for all the enhancement nodes. This paper uses hyperbolic tangent as the activation function for all enhancement nodes. We gather all the enhancement node outputs together as

$$\boldsymbol{\eta} = [\mathfrak{h}_1^{\mathrm{T}}, \cdots, \mathfrak{h}_m^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^e \tag{9}$$

(c) Network Output

For a given input vector $\mathbf{x}$, the output of the network is

$$o = \left[\mathbf{g}^{\mathrm{T}} \middle| \boldsymbol{\eta}^{\mathrm{T}}\right] \boldsymbol{\beta} \tag{10}$$

where $\boldsymbol{\beta}$ is the output weight vector. The number of elements in $\boldsymbol{\beta}$ is equal to $f + e$. Hence, its components are given by

$$\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{f+e}]^{\mathrm{T}} \tag{11}$$

2.2.3. Construction of Weight Matrices and Vectors

Given N training pairs $\mathbb{D}_{train} = \{(\mathbf{x}_k, y_k) : \quad k = 1, \ldots, N\}$, where $\mathbf{x}_k = [x_{k,1}, \cdots, x_{k,D}]^{\mathrm{T}}$ is a D dimensional training input and y_k is the corresponding target output as the training set. Consider that the training data matrix is formed by packing all the input $\mathbf{x}_k$'s together. The augmented data matrix denoted as $\mathbf{X}$ is given by

$$\mathbf{X} = \left(\begin{array}{c|c} \mathbf{x}_1^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \mathbf{x}_N^{\mathrm{T}} & 1 \end{array} \right) \tag{12}$$

(a) Construction of the Projection Matrix $\mathbf{\Psi}_i$

For each group of features-mapped nodes, one important thing in the framework of BLS is building the projection matrix $\mathbf{\Psi}_i$. An important question then arises as follows: how to build the projection matrix? The approach presented here follows the procedures of [8,33]. In the BLS, a random matrix $\mathcal{P}_i \in \mathbb{R}^{(D+1)\times f_i}$ is generated first for each group of features-mapped nodes. Afterwards, we can obtain a random-projection data matrix $\mathbf{Q}_i$, given by

$$\mathbf{Q}_i = \mathbf{X}\mathcal{P}_i \tag{13}$$

The projection matrix $\mathbf{\Psi}_i$ is the result of the sparse approximation problem given by

$$\min_{\mathbf{\Psi}_i} \left\{ \| \mathbf{Q}_i \mathbf{\Psi}_i - \mathbf{X} \|_F^2 + \rho \| \mathbf{\Psi}_i \|_1 \right\} \tag{14}$$

where in (14), the term $\rho \| \mathbf{\Psi}_i \|_1$ is to enforce the solution of (14) to be a sparse matrix. Additionally, ρ is a regularization parameter for sparse regularization. In addition, F is the popular Frobenius norm and $\| \,.\, \|_1$ is norm 1.

(b) Construction of the Weight Matrices of the Enhancement Nodes

To some degree, the construction step of $\mathbf{W}_j$'s is a bit straightforward. For instance, as detailed in [8,33,34], the traditional BLS algorithm and other variants randomly generate the weight matrices for each group of the enhancement nodes. Similarly, this paper follows the same procedure to generate $\mathbf{W}_j$'s.

(c) Construction of Output Weight Vector

This section gives the procedure to construct the output weight vector $\boldsymbol{\beta}$. Given the projection matrix $\mathbf{\Psi}_i$'s $\forall\ i = 1, \cdots, n$ of the feature-mapped nodes, and the training data matrix $\mathbf{X}$, the i-th training data feature matrix for all training samples is given by

$$\mathbf{Z}_i = \begin{pmatrix} \mathbf{z}_{i,1}^{\mathrm{T}} \\ \vdots \\ \mathbf{z}_{i,N}^{\mathrm{T}} \end{pmatrix} = \mathbf{X}\mathbf{\Psi}_i^{\mathrm{T}} \tag{15}$$

where $\mathbf{z}_{i,k} = [z_{i,k,1}, \cdots, z_{i,k,f_i}]^{\mathrm{T}}$, and

$$z_{i,k,u} = \sum_{\iota=1}^{D} \psi_{i,u,\iota} x_{k,\iota} + \psi_{i,u,D+1} \tag{16}$$

Let $\mathcal{Z}$ be the collection of all training data feature matrices. Hence, we have

$$\mathcal{Z} = [\mathbf{Z}_1, \cdots, \mathbf{Z}_n] \tag{17}$$

In this way, $\mathcal{Z}$ is an $N \times f$ matrix, denoted as

$$\mathcal{Z} = \begin{pmatrix} \mathcal{Z}_{1,1} & \cdots & \mathcal{Z}_{1,f} \\ \vdots & \ddots & \vdots \\ \mathcal{Z}_{N,1} & \cdots & \mathcal{Z}_{N,f} \end{pmatrix} = \begin{pmatrix} \mathcal{Z}_1^{\mathrm{T}} \\ \vdots \\ \mathcal{Z}_N^{\mathrm{T}} \end{pmatrix} \tag{18}$$

where the k-th row vector $\mathcal{Z}_k^{\mathrm{T}}$ of $\mathcal{Z}$ is the inputs of the enhancement nodes (the outputs of the feature-mapped nodes) for the k-th training input vector $\mathbf{x}_k$. To handle input biases, we augment one vector into $\mathcal{Z}$, given by

$$\mathcal{Z}' = \begin{pmatrix} \mathcal{Z}_1^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \mathcal{Z}_N^{\mathrm{T}} & 1 \end{pmatrix} = \begin{pmatrix} \mathcal{Z}'^{\mathrm{T}}_1 \\ \vdots \\ \mathcal{Z}'^{\mathrm{T}}_N \end{pmatrix} \tag{19}$$

Furthermore, given $\mathcal{Z}$, the enhancement node outputs of the j-th enhancement group for all training data are given by

$$\mathbf{H}_j = \xi\left(\mathcal{Z}'\,\mathbf{W}_j^{\mathrm{T}}\right) = \begin{pmatrix} \mathbf{h}_{j,1}^{\mathrm{T}} \\ \vdots \\ \mathbf{h}_{j,N}^{\mathrm{T}} \end{pmatrix} \tag{20}$$

for $j = 1, \cdots, m$, where

$$\mathbf{h}_{j,k} = \left[h_{j,k,1}, \cdots, h_{j,k,e_j}\right]^{\mathrm{T}} \tag{21}$$

and

$$h_{j,k,v} = \xi\left(\sum_{\tau=1}^{f+1} w_{j,v,\tau}\mathcal{Z}'_{k,\tau}\right) \tag{22}$$

Packing all the enhancement node outputs together, we have

$$\mathcal{H} = [\mathbf{H}_1, \cdots, \mathbf{H}_m] \tag{23}$$

where $\mathcal{H}$ is a $N \times \left(\sum_{j=1}^{m} e_j\right) = N \times e$ matrix.

Define $\mathcal{A} = [\mathcal{Z}|\mathcal{H}]$. The output weight vector $\boldsymbol{\beta}$ can be calculated based on least square techniques:

$$arg\,\min_{\boldsymbol{\beta}} \| \mathcal{A}\boldsymbol{\beta} - \mathbf{y} \|_{\rho}^{\rho} + \varrho \| \boldsymbol{\beta} \|_{\lambda}^{\lambda} \tag{24}$$

where $\mathbf{y} = [y_1, \cdots, y_N]^{\mathrm{T}}$ is the collection of all training outputs. Equation (24) means that we can have different cost functions by setting different values of ρ, ρ, λ. It should be noted the value of ρ and λ are not necessarily the same. In this paper, to explore BLS for air pressure failure prediction, we reformulate the objective function (24) like that of logistic regression. In the next subsection, we give background details of logistic regression (LogR).

2.2.4. Logistic Regression

Logistic regression (LogR) is a widely used and popular probabilistic statistical classification technique. It is designed for binary classification problems. Logistic regression is detailed in [35]. The technique aims to maximize the likelihood function given by

$$J_{LogR} = \prod_{k=1}^{N} t_k^{y_k}\{1 - t_k\}^{1-y_k}; t_k = \sigma\left(\mathbf{w}^T\mathbf{x}_k\right) \tag{25}$$

where $\sigma(z) = \frac{1}{1+\exp^{-z}}$ and $\mathbf{x}_k$ is the k-th input vector. We can further modify (25) as a minimization problem. With manipulations, we turn (25) to a well-known cross-entropy error function (26). By taking the negative logarithm of the likelihood function (25), we arrive at

$$J(w) = -\log\left(J_{LogR}\right) = -\sum_{k=1}^{N}\{y_k \log(t_k) + (1 - y_k)\log(1 - t_k)\} \tag{26}$$

The gradient descent can be used to optimize the error function (25) to obtain an optimal output weight $\mathbf{w}$.

3. The Proposed Technique

In the proposed approach, we capitalize on the strength of a broad learning system (BLS) and logistic regression (LogR). Figure 5 shows the structure of the fused BLS network with logistic regression classifier.

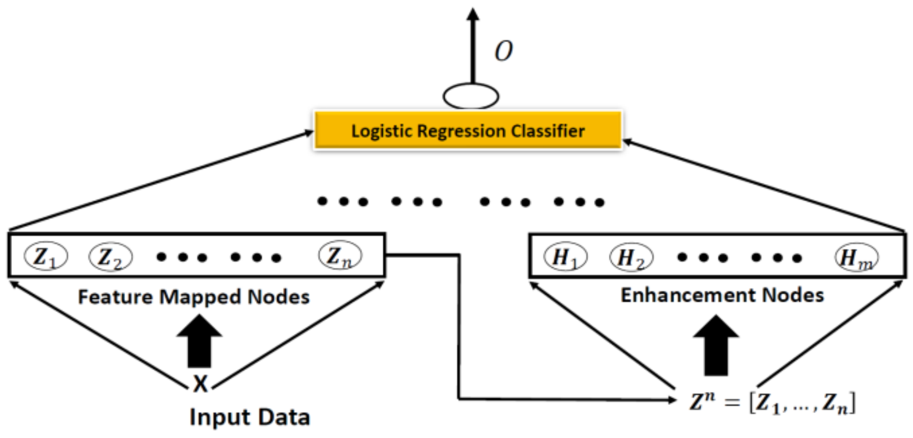


Figure 5. The Flowchart of the proposed network and procedure.

In Figure 5, input $\mathbf{X}$ is passed to the feature-mapped layer, where the feature $\mathbf{Z}^n$ is extracted and obtained. This feature is further enhanced to obtain an enhanced feature $\mathbf{H}^m$. Both features are combined as $\mathcal{A} = [\mathbf{Z}^n|\mathbf{H}^m]$. The concatenated features are then passed to the logistic regression classifier for making the decision. The fusion of logistic regression and broad learning system, with the effectiveness of the feature-extraction layer and enhancement layer improves the performance of the network. For instance, when the feature nodes extract features from the input X, the enhancement nodes further enhance the features such that the distance between the positive class and negative class is widened. Hence, it able to separate between class even when the two classes are in balance.

In our approach, we incorporate the objective function of BLS (24) into the objective function of LogR (25). In other words, for the proposed broad embedded logistic regression model, we assume a non-linear relationship between the input of the logistic regression classifier and the output of logistic regression classifier. For easy notation and explanation, we let

$$\mathcal{A} = \left[\mathcal{A}_1, \cdots, \mathcal{A}_{f+e}\right] = \begin{bmatrix} a_{1,1} & \cdots & a_{1,f+e} \\ \vdots & \ddots & \vdots \\ a_{N,1} & \cdots & a_{N,f+e} \end{bmatrix}$$

Additionally, the probability that $y_k = 1$ is given by p_k. In other words, when the model predicts that $y_k = 1$, the prediction probability is given by p_k. Hence, we formulate the relationship between feature $\mathcal{A}$ obtained from the BLS network and the output weight $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{f+e}]^{\mathrm{T}}$. In addition, we add a bias term β_0 and $\mathcal{A}_0 = [1, 1, 1 \dots 1]^T$ into the relationship. Hence, for k-th input, we have

$$l_k = a_{k,0}\beta_0 + \sum_{r=1}^{f+e} \mathrm{a}_{k,r}\beta_r = \log_b \frac{p_k}{1 - p_k} \tag{27}$$

where l_k is the log-odds for the k-th. Furthermore, it should be noted that b is an additional generalization, it is the base of the model.

For a more compact notation and to take the bias term into consideration, we specify the feature variables $\overline{\mathcal{A}}$ and $\overline{\boldsymbol{\beta}}$ as $(f+e+1)$—dimensional vectors. They are given by

$$\begin{aligned} \overline{\mathcal{A}} = & \left[\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_{f+e+1}\right] \\ \overline{\boldsymbol{\beta}} = & \quad [\beta_0, \beta_1, \ldots, \beta_{f+e+1}]^T \end{aligned} \tag{28}$$

where $\mathcal{A}_0 = [a_{1,0}, \ldots a_{N,0}]^T$, $\mathcal{A}_1 = [a_{1,1}, \ldots a_{N,1}]^T, \ldots, \mathcal{A}_{f+e+1} = \left[a_{1,f+e+1}, \ldots a_{N,f+e+1}\right]^T$.

Hence, we rewrite the logit, l_k as

$$l_k = \sum_{r=0}^{f+e} \mathrm{a}_{k,r}\beta_r = \log_b \frac{p_k}{1-p_k} \tag{29}$$

Now, solving for the probability p_k that the model predicts $y_k = 1.$ yields.

$$p_k = \frac{e^{l_k}}{1+e^{l_k}} = \sigma(l_k) \tag{30}$$

where b is substituted by e and it is exponential function and where $\sigma(.)$ is the sigmoid function. With (30), we can easily compute the probability that $y_k = 1$ for a given observation. The optimum $\boldsymbol{\beta}$ can be obtained by minimizing the negative log-likelihood of (30). Hence, the log-likelihood may be written as follows:

$$\begin{aligned} J &= \sum_{k=1}^{N} \{-y_k \log(p_k) - (1-y_k)\log(1-p_k)\} + \rho \parallel \boldsymbol{\beta} \parallel_\lambda^\lambda \\ &= \sum_{j=1}^{N} \{-y_k \log(\sigma_k) - (1-y_k)\log(1-\sigma_k)\} + \rho \parallel \boldsymbol{\beta} \parallel_\lambda^\lambda \end{aligned} \tag{31}$$

where

$$\sigma_k = \frac{1}{1+exp^{(l_k)}} = \frac{1}{1+exp^{(\sum_{r=0}^{f+e} \mathrm{a}_{k,r}\beta_r)}} \tag{32}$$

We employ gradient descent to optimize the proposed objective function (31). We name our proposed technique broad embedded logistic regression (BELR).

From (26) and (31), it should be noted that the traditional logistic regression can only manage the linear relationship between dependent variables and independent variables effectively. In other words, the classical logistic regression does not consider any possible nonlinear relationship between the dependent variable and independent variables. Unlike the classical logistic regression classifier, where the raw data are used as its input directly, in this paper, from (31), the output of the feature-mapped node and enhancement node of BLS is the input of the logistic regression classifier. In other words, enhanced features serve as the input of the logistic regression classifier. This improves the performance of the algorithm.

In addition, the objective function (31) of the proposed approach contains the regularizer $\rho \parallel \boldsymbol{\beta} \parallel_\lambda^\lambda$, where λ can be chosen or set to different values to have different scenarios and to improve the performance of the network. For instance, for $\lambda = 1$, the output weight of the proposed method will have a sparse solution. This setting can allow the network to automatically select a relevant feature from $\overline{\mathcal{A}}$, which may enhance the network performance. Similarly, if λ is set to 2, the output weight will have dense values and the values will be small. This will prevent the network from overfitting. In this paper, our focus is not to have a sparse solution. Hence, in our experiment, we utilize $\lambda = 2$.

4. Experiment and Settings

In this section, we compare the proposed BELR with other linear and non-linear algorithms, namely the original logistic regression (LogR), Random Forest classifier (RF), Gaussian Naive Bayes (GNB), K-nearest neighbour (KNN), and Support Vector Machine

(SVM). We use four evaluation metrics in our comparison. Table 1 presents the evaluation metrics used to evaluate the performance of the comparison algorithms.

Table 1. The METRICS FOR THE MODELS comparison.

Evaluation Metrics	Equivalent Equation
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F1-Score	$\frac{2*(Precision*Recall)}{(Precision+Recall)}$
Accuracy	The fraction of the predictions the model executed correctly $\frac{TP+TN}{(TP+TN+FP+FN)}$

From the Table, False Positive (FP) is the number of examples which are predicted to be positive by the model but belong to the negative class. False Negative (FN) is the number of examples which are predicted to be negative by the model but belong to the positive class. True Positive (TP) is the number of examples which are predicted to be positive by the model and belong to the positive class.

Furthermore, for a fair comparison, in all the comparison algorithms, we use standard settings for all the parameters suggested in the scikit-learn machine learning package [36]. Additionally, we use APS data set [37,38]. This benchmark data set is commonly used to evaluate machine learning algorithms, specifically for APS failure prediction tasks. There are two problems with the data set. First, the data set contains a high number of missing values. Second, the data set has a high imbalance in class distribution.

In some papers, median imputation has been used to fill missing data. For instance, in [16], the median imputation technique was utilized to handle missing values. However, median imputation can cause destruction to the data. Hence, we employ a robust imputer, namely the KNN imputation method. Thus, we replace the missing values in each column using KNN. The data set used in this paper is quite challenging, as it has the issue of imbalanced class distribution. Our proposed BELR has a comparable good performance. This may be attributed to the ability of feature-mapped layer (nodes) to extract features from the input data and enhancement layer (nodes) for further enhancement of the feature such that the classes are separated from each other. Hence, this improves the performance of BELR under skew data set. This is validated when we compare the original logistic regression classifier and the proposed BELR.

After filling the missing data using KNN imputer, we use cross validation method to fit the comparison models. Furthermore, inside cross-validation, we extract features by using BLS on training set, then fit logistic regression on a feature from the training set, then used the test set to estimate quality metrics.

The total data points are split into 10-fold using stratified method of scikit-learn machine learning package and run each algorithm 10 times. For instance, in the first run we combine nine samples of the divided data as the training set and the remaining one sample for test set. We repeat this process 10 times using different set of data points as the training set and test set. Table 2 summarize the details of the data set used in the first run. In the experiment, we present the average performance of each compared algorithm.

Table 2. Details of the data set and further details of the data set.

Total Number of Data Points	No Rows of Training Data	No Rows of Test Data	
76,000	68,400	7600	
Training Set		**Test Set**	
Negative Case	**Positive Case**	**Negative Case**	**Positive Case**
67,162	123	7462	137

From the table, the ratio of positive case to negative case in the training set is 0.001831, and for the test set, it is 0.018360. It should be noted that we have used stratified method of scikit-learn, a machine learning package, in our cross-validation methods. It takes into consideration the imbalance class of the data to split the data into 10-fold.

The Comparison of the Performance of the Compared Algorithms

In the subsection, we compare the proposed BELR and the original logistic regression (LogR), Random Forest classifier (RF), Gaussian Naive Bayes (GNB), K-nearest neighbour (KNN), and Support Vector Machine (SVM). The average performance in terms of the metrics listed in Table 1 is presented for the comparison algorithms. First, to prevent the effect of the curse of dimensionality, we use principal component analysis (PCA) to reduce the dimension and select an important feature from the input data. A total of 81 principal components are created after applying the PCA technique with a covariance value of 0.95. The initial dimension of input data is 170; however, after applying PCA, the dimension is reduced to 81, which is almost 50% of the feature variables compared to the initial feature variables. After applying PCA, we then apply comparison algorithms on the feature from PCA. We use 10-fold cross-validation concept. In the experiment, the total number of data point is 76,000. For each fold, there are 7600 data points after applying stratified cross-validation, ensuring that each fold has the same proportion of observations with a given categorical value. In the first run, we take one group (7600 data points) as the test set and the remaining nine groups (9×7600 data points) for training of the model. In the second run, we pick another 76,000 data points (a new group) as the test set and the remaining nine groups (9×7600 data points) to train the models. The process continues until we reach the 10th run or trial. The training set contains 67,162 negative cases and 123 positive cases. Similarly, the test set contains 7462 negative cases and 137 positive cases. Table 2 shows the details. From the experiment, the results obtained are presented in Table 3.

Table 3. The performance of comparison algorithms under certain metrics.

Score (%)	**GNB**	**LogR**	**RF**	**SVM**	**KNN**	**BELR**
Precision	32.45	77.81	82.6	92.05	80.23	80.91
Sensitivity (Recall)	79.35	57.89	57.67	27.78	55.78	62.25
F1-Score	46.06	66.39	67.92	42.68	65.81	70.39
Accuracy	96.64	98.94	99.01	98.65	98.95	99.05

From Table 3, we notice that GNB has a recall of 79.35, and the performance looks better than the rest of the algorithms. However, GNB has a very poor performance in precision. It has a precision score of 32.45. In addition, it has a very poor performance in F1-score, with a score of 46.06.

For other algorithms, it is notice that SVM has a very good score in precision but a very poor score in recall. This resulted in a poor value of F1-score. However, LogR, RF, KNN, and the proposed BELR have good precision and recall scores from the Table. Their performances in terms of precision are relatively equal. The proposed BELR has the best average score in terms of Sensitivity (Recall). In addition, when we compared the performance of the compared algorithms in terms of average F1-score, the proposed BELR has the best F1-score, as shown in Table 3. Other scores for other evaluation metrics are presented in the Table. We use boxplot to present the average F1-score of all the compared algorithms. Figure 6 presents the average F1 score of the performance of the compared algorithms. It is noticed that the proposed BELR has a better F1 score from the box plot.

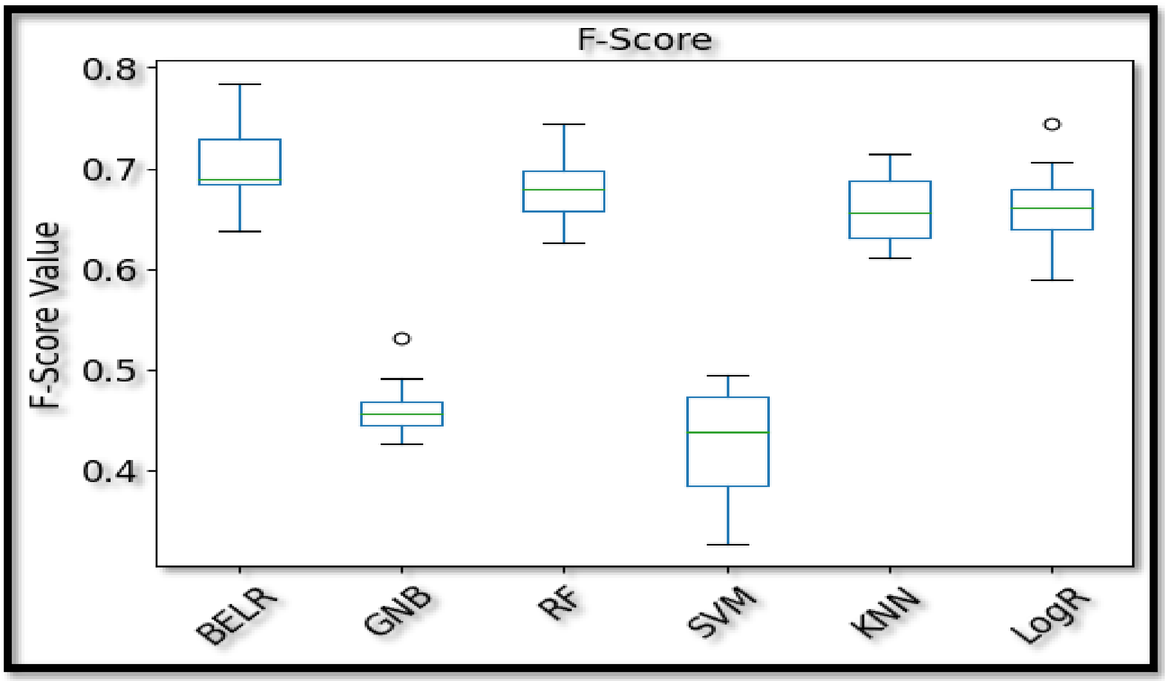


Figure 6. The average F1-score of the compared algorithms.

Overall, we notice that the performance of the proposed BELR is better than the other comparison algorithms under an imbalanced data set.

5. Conclusions

This paper proposes broad embedded logistic regression (BELR) for classification problems, specifically for APS failure prediction. In addition, its performance is studied under an exceedingly difficult data situation and an imbalanced class distribution problem. The feature-mapped nodes and enhancement nodes of the BLS are employed to handle imbalance data set due to the ability of the two types of nodes to generate/extract features that can uniquely separate two classes from each other. Hence, it improves the classification capacity of logistic regression classifier.

Furthermore, the APS data set has a problem of missing data, and in this paper we explore KNN imputation method to solve the problem of missing data using KNN_imputer from Sklearn. Sklearn is a machine learning package commonly used for processing data, building machine learning model. It should be noted that other missing data imputation methods such as generative adversarial network (GAN), etc., could be explored.

The performance of the proposed algorithm is better than other comparison algorithms, namely Gaussian Naive Bayes (GNB), Random Forest, K-nearest neighbor (KNN), Support Vector Machine (SVM), and Logistic Regression (LogR). The performance of the comparison algorithms is evaluated using popular and commonly used metrics in the literature, namely average F1-score, average Recall, average Precision, and average Accuracy. In terms of the F1-score, the performance of the proposed algorithm is the best among the comparison algorithms. The Table and the Figures presented in the experimental section validate that the proposed BELR performances are comparable with other algorithms.

Author Contributions: Validation, B.P.; Writing—original draft, A.A.M.; Writing—review & editing, H.A.; Supervision, C.K.M.L.; Project administration, J.C.; Funding acquisition, C.K.M.L. All authors have read and agreed to the published version of the manuscript.

Funding: ITC-InnoHK Clusters-Innovation and Technology Commission.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Available upon request.

Acknowledgments: The work was supported by the Centre for Advances in Reliability and Safety (CAiRS) admitted under AIR@InnoHK Research Cluster.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ML	Machine learning
APS	Air Pressure System
BELR	Broad Embedded Logistic Regression
BLS	Broad Learning System
LogR	Logistic Regression
PCA	Principal Component Analysis
RVFLNN	Random Vector Functional-link neural networks
IIoT	Industrial Internet of Things
SVM	Support Vector Machine
MLP	Multi-layer Perceptron
SMOTE	Synthetic Minority Oversampling Technique
ELM	Extreme Learning Machine
KNN	K-Nearest Neighbour
ADMM	Alternating Direction Method of Multipliers
RF	Random Forest
GNB	Gaussian Naïve Bayes
ROC	Receiver Operating Characteristics
max AUC	Maximizes Area Under the Curves
MAR	Missing at random
MNR	Missing not at random
MCAR	Missing completely at random
GAN	Generative Adversarial Network
f	Total number of feature-mapped nodes in the BLS network
e	Total number of enhancement nodes in the BLS network

References

1. Yuantao, F.; Nowaczyk, S.; Antonelo, E.A. Predicting air compressor failures with echo state networks. *PHM Soc. Eur. Conf.* **2016**, *3*, 1.
2. Lokesh, Y.; Nikhil, K.S.S.; Kumar, E.V.; Mohan, G.K. Truck APS Failure Detection using Machine Learning. In Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 13–15 May 2020; IEEE: Piscataway, NJ, USA, 2020.
3. Qu, W.; Balki, I.; Mendez, M.; Valen, J.; Levman, J.; Tyrrell, P.N. Assessing and mitigating the effects of class imbalance in machine learning with application to X-ray imaging. *Int. J. Comput. Assist. Radiol. Surg.* **2020**, *15*, 2041–2048. [CrossRef] [PubMed]
4. Zolanvari, M.; Teixeira, M.A.; Jain, R. Effect of imbalanced datasets on security of industrial IoT using machine learning. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI), Miami, FL, USA, 9–11 November 2018; IEEE: Piscataway, NJ, USA, 2018.
5. Akarte, M.M.; Hemachandra, N. *Predictive Maintenance of Air Pressure System Using Boosting Trees: A Machine Learning Approach*; ORSI: Melle, Belgium, 2018.
6. Wang, G.K.; Sim, C. Context-dependent modelling of deep neural network using logistic regression. In Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 8–12 December 2013; IEEE: Piscataway, NJ, USA, 2013.
7. Wright, R.E. Logistic regression. In *Reading and Understanding Multivariate Statistics*; Grimm, L.G., Yarnold, P.R., Eds.; American Psychological Association: Washington, DC, USA, 1995; pp. 217–244.
8. Chen, C.P.; Liu, Z. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 10–24. [CrossRef] [PubMed]
9. Adegoke, M.; Leung, C.S.; Sum, J. Fault Tolerant Broad Learning System. In *International Conference on Neural Information Processing*; Springer: Cham, Switzerland, 2019; pp. 95–103.
10. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [CrossRef]
11. Köppen, M. The curse of dimensionality. In Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), Online, 4–18 September 2000; Volume 1.
12. Rubin, D.B. Inference and missing data. *Biometrika* **1976**, *63*, 581–592. [CrossRef]

13. Little, R.J.; Rubin, D.B. *Statistical Analysis with Missing Data*; John Wiley & Sons: Hoboken, NJ, USA, 2019; p. 793.
14. Guo, Z.; Wan, Y.; Ye, H. A data imputation method for multivariate time series based on generative adversarial network. *Neurocomputing* **2019**, *360*, 185–197. [CrossRef]
15. Zhang, S. Nearest neighbor selection for iteratively kNN imputation. *J. Syst. Softw.* **2012**, *85*, 2541–2552. [CrossRef]
16. Gondek, C.; Daniel, H.; Oliver, R.S. Prediction of failures in the air pressure system of scania trucks using a random forest and feature engineering. In *International Symposium on Intelligent Data Analysis*; Springer: Cham, Switzerland, 2016.
17. Rengasamy, D.; Jafari, M.; Rothwell, B.; Chen, X.; Figueredo, G.P. Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors* **2020**, *20*, 723. [CrossRef] [PubMed]
18. Nowaczyk, S.; Prytz, R.; Rögnvaldsson, T.; Byttner, S. Towards a machine learning algorithm for predicting truck compressor failures using logged vehicle data. In Proceedings of the 12th Scandinavian Conference on Artificial Intelligence, Aalborg, Denmark, 20–23 November 2013; IOS Press: Washington, DC, USA, 2013; pp. 20–22.
19. Costa, C.F.; Nascimento, M.A. Ida 2016 industrial challenge: Using machine learning for predicting failures. In *International Symposium on Intelligent Data Analysis*; Springer: Cham, Switzerland, 2016.
20. Cerqueira, V.; Pinto, F.; Sá, C.; Soares, C. Combining boosted trees with meta feature engineering for predictive maintenance. In *International Symposium on Intelligent Data Analysis*; Springer: Cham, Switzerland, 2016.
21. Ozan, E.C.; Riabchenko, E.; Kiranyaz, S.; Gabbouj, M. An optimized k-nn approach for classification on imbalanced datasets with missing data. In *International Symposium on Intelligent Data Analysis*; Springer: Cham, Switzerland, 2016.
22. Jose, C.; Gopakumar, G. An Improved Random Forest Algorithm for classification in an imbalanced dataset. In Proceedings of the 2019 URSI Asia-Pacific Radio Science Conference (AP-RASC), New Delhi, India, 9–15 March 2019; IEEE: Piscataway, NJ, USA, 2019.
23. Syed, M.N.; Hassan, R.; Ahmad, I.; Hassan, M.M.; De Albuquerque, V.H.C. A Novel Linear Classifier for Class Imbalance Data Arising in Failure-Prone Air Pressure Systems. *IEEE Access* **2020**, *9*, 4211–4222. [CrossRef]
24. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
25. Huang, G.B.; Ding, X.J.; Zhou, H.M. Optimization method based extreme learning machine for classification. *Neurocomputing* **2010**, *74*, 155–163. [CrossRef]
26. Ding, S.; Zhao, H.; Zhang, Y.; Xu, X.; Nie, R. Extreme learning machine: Algorithm, theory and applications. *Artif. Intell. Rev.* **2015**, *44*, 103–115. [CrossRef]
27. Gong, X.; Zhang, T.; Chen, C.L.P.; Liu, Z. Research review for broad learning system: Algorithms, theory, and applications. *IEEE Trans. Cybern.* **2021**, *52*, 8922–8950. [CrossRef] [PubMed]
28. Xu, M.; Han, M.; Chen, C.L.P.; Qiu, T. Recurrent broad learning systems for time series prediction. *IEEE Trans. Cybern.* **2018**, *50*, 1405–1417. [CrossRef] [PubMed]
29. Zhao, H.; Zheng, J.; Xu, J.; Deng, W. Fault diagnosis method based on principal component analysis and broad learning system. *IEEE Access* **2019**, *7*, 99263–99272. [CrossRef]
30. Liu, Z.; Chen, C.P. Broad learning system: Structural extensions on single-layer and multi-layer neural networks. In Proceedings of the 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), Shenzhen, China, 15–17 December 2017; IEEE: Piscataway, NJ, USA, 2017.
31. Feng, S.; Chen, C.P. Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification. *IEEE Trans. Cybern.* **2018**, *50*, 414–424. [CrossRef] [PubMed]
32. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2011**, *3*, 1–122. [CrossRef]
33. Jin, J.; Liu, Z.; Chen, C.L. Discriminative graph regularized broad learning system for image recognition. *Sci. China Inf. Sci.* **2018**, *61*, 112209. [CrossRef]
34. Muideen, A.; Wong, H.T.; Leung, C.S. A fault aware broad learning system for concurrent network failure situations. *IEEE Access* **2021**, *9*, 46129–46142.
35. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 4, No. 4.
36. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
37. Asuncion, A.; Newman, D. UCI Machine Learning Repository. 2007. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=31.%09Asuncion%2C+A.%3B+Newman%2C+D.+UCI+ma-chine+learning+repository&btnG=#d=gs_cit&t=1676228269803&u=%2Fscholar%3Fq%3Dinfo%3AbqbHDUKR2lMJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den (accessed on 11 November 2022).
38. Available online: https://www.kaggle.com/datasets/uciml/aps-failure-at-scania-trucks-data-set (accessed on 3 October 2022).