



# Social ski driver conditional autoregressive-based deep learning classifier for flight delay prediction

Desmond Bala Bisandu<sup>1,2</sup> · Irene Moulitsas<sup>1,2</sup> · Salvatore Filippone<sup>1,3</sup>

Received: 25 May 2021 / Accepted: 4 January 2022  
© The Author(s) 2022

## Abstract

The importance of robust flight delay prediction has recently increased in the air transportation industry. This industry seeks alternative methods and technologies for more robust flight delay prediction because of its significance for all stakeholders. The most affected are airlines that suffer from monetary and passenger loyalty losses. Several studies have attempted to analyse and solve flight delay prediction problems using machine learning methods. This research proposes a novel alternative method, namely social ski driver conditional autoregressive-based (SSDCA-based) deep learning. Our proposed method combines the Social Ski Driver algorithm with Conditional Autoregressive Value at Risk by Regression Quantiles. We consider the most relevant instances from the training dataset, which are the delayed flights. We applied data transformation to stabilise the data variance using Yeo-Johnson. We then perform the training and testing of our data using deep recurrent neural network (DRNN) and SSDCA-based algorithms. The SSDCA-based optimisation algorithm helped us choose the right network architecture with better accuracy and less error than the existing literature. The results of our proposed SSDCA-based method and existing benchmark methods were compared. The efficiency and computational time of our proposed method are compared against the existing benchmark methods. The SSDCA-based DRNN provides a more accurate flight delay prediction with 0.9361 and 0.9252 accuracy rates on both dataset-1 and dataset-2, respectively. To show the reliability of our method, we compared it with other meta-heuristic approaches. The result is that the SSDCA-based DRNN outperformed all existing benchmark methods tested in our experiment.

**Keywords** Deep long short-term memory · Deep recurrent neural network · Flight delay prediction · Social ski driver · Yeo-Johnson Transformation

## 1 Introduction

The civil aviation sector is a distributed network of large interconnected elements designed to meet the common aim of on-time air transportation and passengers expectations [1, 2]. Because of flight connectivity, flight delays at airports, especially for commercial hub airports, usually propagate to other individual airports or even to the entire air transportation network. Without proper monitoring and control, such delays can expand over time, resulting in poor airport performance and causing unnecessary dissatisfaction for passengers [1, 3]. Over the last few years, this sector has rapidly grown in areas such as customers, infrastructure and territorial coverage [4, 5]. A statistical report from the international air transport association (IATA) reveals that in 2012, airlines' worldwide base incurred over \$160 billion in total energy cost [6–8].

---

✉ Desmond Bala Bisandu  
desmond.bisandu@cranfield.ac.uk

✉ Irene Moulitsas  
i.moulitsas@cranfield.ac.uk

<sup>1</sup> Department of Computational Engineering Sciences, Cranfield University, Bedford Mk43 0AL, UK

<sup>2</sup> Machine Learning and Data Analytics Laboratory, Digital Aviation Research and Technology Centre (DARTEC), Cranfield, UK

<sup>3</sup> Department of Civil and Computer Engineering, Università Di Roma "Tor Vergata", Via del Politecnico 1, 00133 Roma, Italy

With the rapid growth of Communication, Navigation and Surveillance (CNS), increased air traffic and airspace capacities, flight schedule planners try to reduce buffer time among flight departures and arrivals for maximising the utilisation of aircraft.

The air transportation network architecture is tight and complex, making it vital to develop accurate prediction models critical for intelligent aviation systems [9]. It is necessary to predict flight departure or arrival delays with high accuracy. Anxiety by the passengers can be avoided by efficiently arranging their schedules to have access to a prediction model for a specific airport taxi time, with an explanatory variable being computed before prediction [9–11]. Additionally, airport management aims to provide better service with improved airport gate availability [4, 12, 13]. Nowadays, an increase in air traffic and flight delays has become a severe and prominent issue globally [1]. Based on the United States (US) Bureau of Transportation Statistics (BTS) report, most airline flights arrived 15 min late at their destination [14–22], thus incurring a loss of \$30 billion, which is a challenge to the air transportation system [14, 23]. Delay analyses have become an important research topic, and delay prediction has been the subject of earlier studies [14, 24, 25]. As state earlier, delays have a significant financial impact. Therefore, it is important to introduce intelligent systems to automate airports, passengers, and commercial airlines' monitoring and decision processes [26]. Highly accurate predictions and real-time monitoring systems are indispensable tools to that effect [1, 9–11, 26–32]. Proposed strategies include collaborative decision making (CDM) [4, 33], ground delay programs (GDP) [4, 34], and air traffic flow management (ATFM) [4, 35] to improve the information flow among participating airports. A few years ago, the research methodologies utilised for predicting delay propagation were from statistical, network theory, machine learning and agent-enabled methods [4, 36].

Comparing statistical approaches with machine learning methods have become popular in recent years. The field of transportation systems and aerospace research has experienced a significant number of machine learning models such as K-nearest neighbour (k-NN), support vector machine (SVM) and artificial neural network (ANN) models [9]. Some studies seek to improve the model prediction performance by introducing variant neural network models [37–39]. Using only machine learning methods on historical data without an optimisation algorithm has proven ineffective. In contrast, our proposed method uses historical data from two datasets with a novel optimisation algorithm. The authors [14, 40] employed the traditional statistical approach to characterise and distribute flight delays. In [14, 41], the authors combine terminal airports weather forecasts, convective weather forecasts, and the

scheduled flights for predicting daily airport delay time in terms of the weather impacted traffic index (WITI) metric. As reported in the 2017 BTS report, only 0.72% of flight delays were attributed to extreme weather [14]. The most broadly used traffic prediction techniques are deep learning classifiers [42–45]. The deep learning techniques come under supervised and unsupervised machine learning algorithms [37–39, 42, 46–51]. None of the previous studies has taken multiple routes full account in the prediction. In contrast, we introduce a feature fusion method that utilises the complete flight information on different routes and combines them to improve the performance. In our studies, we will focus on non-weather-induced delays.

Our paper aims to propose a novel flight delay prediction strategy that utilises social ski driver conditional autoregressive-based (SSDCA-based) deep long short-term memory (LSTM). We conduct data pre-processing initially to improve the data quality; then, we perform the data transformation based on Yeo-Johnson transformation for further data processing. Yeo-Johnson transformation transfers the data with no loss of its original quality. It works like Box-cox transformation, but data values must not be strictly positive, and it is advantageous over other transformation techniques. Also, it makes the data distribution more symmetric, thereby handling any skewness from the datasets. We then perform feature fusion using the deep recurrent neural network (Deep RNN) for fusing the imperative features. Here, we train the Deep RNN by the developed SSDCA, which improves the model learning process. Finally, we perform flight delay prediction using the Deep LSTM. Furthermore, we compared the newly developed SSDCA with other optimisation algorithms such as social ski driver (SSD), particle swarm optimisation (PSO), ant colony optimisation (ACO), honey-bee optimisation (HBO) and earthworm optimisation algorithm (EWA) [52, 53]. The accuracy (AC) of the SSDCA outperforms the other existing methods. It is also worth mentioning that the proposed method's computational time is less than that of the other methods. In terms of error rate, such as root mean square error (RMSE), mean square error (MSE) and mean absolute error (MAE), the proposed SSDCA outperformed other methods.

The contribution of the paper is:

### 1.1 Proposed SSDCA enabled Deep LSTM for flight delay prediction

We introduced the classifier; SSDCA algorithm drove Deep LSTM by modifying the training process of the Deep LSTM with SSDCA algorithm newly proposed by incorporating SSD with CAViaR for biases and weights optimal tuning. We utilise the Deep RNN for feature fusion, which

is trained by the proposed SSDCA. Also, we adapted the SSDCA enabled Deep LSTM for predicting flight delays.

The rest of this paper is organised as follows: Sect. 2 describes the conventional flight delay prediction strategies employed in the literature and the challenges that inspire the development of the novel technique. Section 3 describes the proposed model for flight delay prediction based on the SSDCA-based deep learning classifier. Section 4 presents our model results and compares them with results from existing methods. Finally, Sect. 5 contains the results and discussion of the findings to conclude with possible future directions.

## 2 Related literature

The vast volume of collected data from the commercial aviation system makes developing machine learning and artificial intelligence algorithms a popular candidate approach in predicting flight delays. Traditional methods such as support vector machine, neural network, fuzzy logic, tree-based methods and K-nearest neighbour are the most common data-driven methods [2]. Güvercin et al. [48] proposed a clustered airport modelling approach for forecasting flight delays using airport networks. The method provided accurate forecasts for flight delays. However, during the training, the method uses only a few samples, which has an adverse effect on the model's prediction performance. Lambelho et al. [49] assessed airports generic strategic schedules using flight cancellation and delay predictions. The method's performance was good in cancellations and delayed flight departure but did not consider other features such as origin and destination to improve the predictions. Tu et al. [26] studied the factors causing major departure delays at Denver International Airport and the departure delay distribution using a probabilistic approach for United Airlines. The study attempts to separate contributing factors but focuses mainly on a single airport and does not consider the network effect. Pathomsiri et al. [50] assessed US joint production on-time and delay performance using a nonparametric function approach.

Some researchers apply operational research, simulation, queueing theory and optimisation to simulate flight delays for an optimised policymakers' system. Pyrgiotis et al. [4] studied an extensive network of delay propagation in major airports in the USA through network decomposition and an analytical queueing model. Ankan et al. [52] analyse delay propagation through air traffic networks with empirical data by developing a stochastic model. A Bayesian network method for estimating delay propagation considers the element-oriented and complex network distribution properties in three commercial aviation systems in the USA. These methods are valuable in

understanding interactions and root causes amongst delay occurrence elements. However, for the individual flights, these models did not yield sufficiently accurate predictions [53]. Rebollo et al. [54] predict departure delays by adopting random forest algorithms using air traffic characteristics as input features. When predicting departure delay for a two-hour forecast window, the model had an error of 21 min. Choi et al. [55] employ several machine learning algorithms and combine weather forecasts with flight schedules to predict scheduled times. Perez-Rodriguez et al. [56] proposed a model to predict daily aircraft delay probabilities in arrivals using asymmetric logic probability.

Recently, deep learning algorithms have been employed to improve the accuracy of flight delay prediction. Yu et al. [1] study flights at Beijing International Airport using a novel Deep Believe Network with support vector regression method (SVR) to analyse high-dimensional data. The model achieves a mean absolute error (MAE) of 8.41 min with high accuracy, but this study was limited to a single airport, and the propagation effects were not evaluated. Kim et al. [57] predict departure and arrival flight delays using a recurrent neural network (RNN) of an individual airport with a day-to-day sequence. Their study shows that a more in-depth architecture improved the accuracy of the RNN. However, the model can only perform a binary prediction of delay and does not quantify its magnitude. Chen and Li [15] developed a machine learning method for chained predictions of flight delays. The method provides an averagely reasonable, accurate and practical result for delay prediction but did not include an adequately big dataset which could have improved the accuracy. Ai et al. [33] developed Convolutional LSTM for temporal and spatial distribution flight delay prediction in the network. The method achieved good classification accuracy but did not include human factors.

Guleria et al. [5] presented a Multi-Agent Approach for reactionary flight delay prediction. The method helps in the flight scheduling system by identifying itineraries. However, the technique did not test delay propagation trees for better performance. Chen et al. [7] presented the Information Gain-Support Vector Machine method for determining how improvements in flight delays of the studied airlines based in China can reduce CO<sub>2</sub> emissions. The proposed approach reduces the limitations of the traditional data envelopment analysis (DEA) model. However, the authors focus on a limited number of airlines and recommend further experiments to confirm the method's validity. Maryam et al. [58] proposed a model for predicting flight delays based on the Levenberg–Marquart algorithm and deep learning. The accuracy of the proposed model in forecasting flight delays was good. However, the results show that the imbalanced form's standard deviation is

higher than all balanced evaluation parameters. Balanced data has more tendency to lead to a lower standard deviation (SD). Ehsan and Seyedmirsajad [59] proposed an approach to predict and analyse departure flight delays using the MATLAB R2018b [60] SVM implementations. The author recommended expanding the research to include National Aviation System (NAS)-wide airports in the analysis for more complete results. Daniel et al. [61] proposed predicting air traffic delays with multilevel input layers using a supervised neural network. The primary aim is to present a prediction model for delays in the air route by applying artificial neural networks (ANN) during the model training; authors concluded that parameters such as the day of the week, the block hour, or the airline had a higher influence than meteorology on the delay.

Regardless of the accuracy of the models, our observation on the methods used in the above works is that their training phase is slow, a characteristic that can become a limiting factor when the size of the data set grows. Another aspect of the training phase is the existence of outliers in the data. Such outliers could exist, for example, because of extreme delays that are not often encountered. Yu et al. [1] eliminate extreme value delays from the bottom and top 1% as outliers. Tu et al. [26] reduce the smoothing spline approach's influence by excluding extreme data preparation observations. The authors in [15] and [62] adopted a random forest method because of its low sensitivity to outliers. This average optimisation technique was unsuitable for our study because it may eliminate essential features through the manual selection process.

Accurate prediction of flight delay propagation at the national network level is essential, especially for flights with reoccurring delays, to reduce unwanted expenses. Based on the recent state-of-the-art architectural model of RNN units, LSTM has been promising in addressing this limitation. Researchers have applied it in several fields, including predicting traffic because the ability to learn time series features temporal correlation. Ma et al. [63] performed traffic speed prediction using LSTM on historical microwave detectors speed data and compared their approach with other approaches in terms of stability and accuracy. Liu et al. [64] proposed a generative deep learning model comprising LSTM encoder and decoder layers to predict aircraft trajectories. The LSTM shows that it can perform good feature extraction and learn useful temporal correlation features [65]. However, despite all these advancements, there is still a need to improve prediction performance and reliability, potentially using an optimisation algorithm to generate more efficient and accurate models.

## 2.1 Challenges

We elaborated on some issues faced by the existing predictive approaches for flight delay as follows:

- The deep learning model and convolutional LSTM [33] were developed for predicting flight delays. However, the method failed to consider the airport's delays to optimise take-off and landing intervals.
- In [49], the authors developed a generic strategic schedule assessment for predicting flight delays. However, the method does not utilise features such as unique air carriers, tail number of aircraft, and origin/destination airports with trigonometric transform function in the model training to enhance predictions' accuracy.
- In [1], the authors introduced a deep belief network method to determine flight delays; the method did not use an optimisation algorithm even without access to the required data on air traffic control because of confidentiality considerations for improving performance.

In our proposed method, we include the departure and arrival details in the selected features. Thus, our method overcomes the challenge of take-off and landing by considering the dataset's actual and scheduled time for both arrival and departure features. Our proposed method also uses features that include categorical and numerical data to enhance prediction performance.

## 3 Problem statement

Flight delays may occur because of several unforeseen events, which can affect airlines, airports and passengers. Developing more accurate models for predicting flight delays has become essential because of the rapid increase in flight complex data overflow, the limited number of prediction methods, and the air transportation system network's complexity. In this context, the proposed method builds an accurate flight delay prediction model. We assume the input data, represented by  $D$ , to be a collection of  $D_{i,j}$ 's, as defined in Eq. 1.

$$K = \{D_{i,j}\}; (1 \leq i \leq N'); (1 \leq j \leq T') \quad (1)$$

where  $N'$  is the total number of data points and  $T'$  is the total number of attributes. Hence,  $D_{i,j}$  represents data in database  $K$  depicting  $j$ th mixed attribute of  $i$ th data. Thus, the expression  $[N' \times T']$  denotes the size of the input data, indicated by  $(D)_{N' \times T'}$ . We will be extracting 6 features  $f_i$  from the input data  $D$  and fuse them into  $f$  as shown in Eq. 2 before feeding them as input to the deep learning classifier for predicting the flight delay as a final output  $O_f$ .

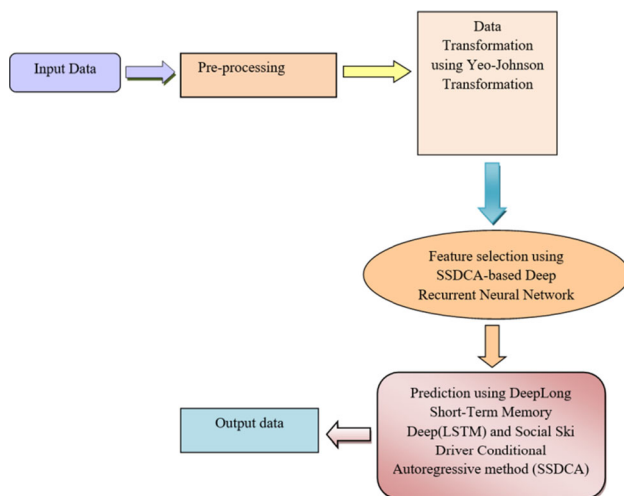
$$f = f_i; \quad i = 1 \text{ to } 6 \quad (2)$$

## 4 Flight delay prediction using the proposed social ski driver conditional autoregressive-based (SSDCA) deep learning classifier

This section explains the proposed SSDCA-based deep learning classifier for predicting flight delay. The steps followed in the developed model are pre-processing, data transformation, feature fusion, and prediction. We initially feed the pre-processing module with the input data and then perform data transmission using Yeo-Johnson transformation [66]. Afterwards, we forward the transformed data to the feature fusion module where Deep RNN performs the feature fusion [67]. We select an optimal number of layers in the Deep RNN method using the proposed SSDCA-based algorithm. The newly developed SSDCA algorithm combines the SSD [68] and CAViaR [69] algorithms. Finally, we perform the prediction using the Deep LSTM method [70] that we trained using the developed SSDCA method. Figure 1 shows the block diagram of the flight delay detection using the newly proposed SSDCA-based Deep learning.

### 4.1 Dataset description

The dataset used in this paper is from the US flight data downloaded from the US Government BTS for January and February of the year 2019 and 2020, respectively [71, 72]. The extracted data feature contains flight information with 21 features for time series analysis and flight delay



**Fig. 1** Block diagram of the flight delay prediction using the proposed SSDCA-based Deep learning

predictions, as shown in Table 1 [17]. There are over 1,000,000 row instances of commercial flights. The datasets records are inconsistent and incomplete, with many missing, duplicates and null values. We initially need to clean to make the data complete and suitable for further pre-processing by converting the attributes to the most appropriate forms for the application of deep learning and machine learning methods.

#### 4.1.1 Features used for the model training and testing

Several features of the dataset are not relevant to our experiment, and we only kept the relevant features that have a high contribution to flight delay. We use the following features in training and testing our model:

- (i) *Flight Date*: The date on which the flight was performed.
- (ii) *Origin*: Departure airport.
- (iii) *Destination*: Arrival airport.
- (iv) *Departure delay*: The difference between actual departure time and scheduled departure time.
- (v) *Arrival delay*: The difference between actual arrival time and scheduled arrival time.
- (vi) *Distance*: The miles covered by the flight.

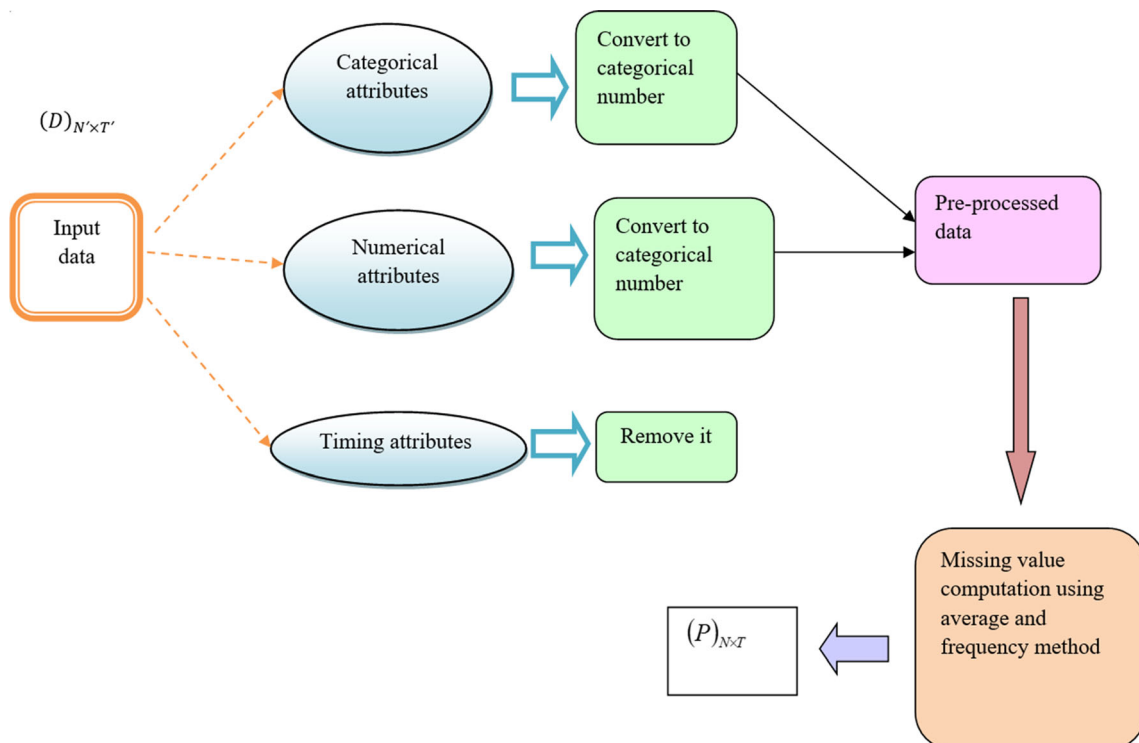
The departure and arrival delays are logically highly correlated, and any experience of delay on the departure flight will certainly affect the flight arrival. The authors in [24] have proved that the origin of congestion at the destination airport is, to a great extent, caused by the departure airport. For this reason, we have the selected features.

### 4.2 Pre-processing

The input data is a collection of categorical, numerical and time attributes. Categorical attributes indicate the airport station, for example, Heathrow, Delhi, Chennai etc. Numerical attributes represent the path values, humidity, etc. Finally, time attributes signify the date and time stamp for departure and arrival. Initially, we convert the categorical attributes to categorical numbers, meaning we assign the airport stations with a unique identifier number 1, 2, 3, etc. After that, we change both the numerical and time attributes from text to number data. Once we get the pre-processed data, we compute the missing value using the average and frequency method for numerical and categorical attributes. Thus, the pre-processed data output is indicated as in  $(P)_{N \times T}$  where the dimension of pre-processed data is  $[N \times T]$ . Figure 2 shows the determination of pre-processed data output.

**Table 1** Datasets Information

| S/No | Feature Name             | Data Type | Attribute Type | Brief format explanation                        |
|------|--------------------------|-----------|----------------|---|
| 1    | Year                     | int64     | Categorical    | In four digits format, e.g. 2000                |
| 2    | Month                    | int64     | Categorical    | In two digits format, e.g. 12                   |
| 3    | Day of month             | int64     | Categorical    | In two digits format, e.g. 01–31                |
| 4    | Day of week              | int64     | Categorical    | In one-digit format, e.g. 1 (Monday)—7 (Sunday) |
| 5    | Departure time           | float64   | Continuous     | In four digits format, e.g. 1456                |
| 6    | Scheduled departure time | float64   | Continuous     | In four digits format, e.g. 1456                |
| 7    | Arrival time             | float64   | Continuous     | In four digits format, e.g. 1456                |
| 8    | Scheduled arrival time   | float64   | Continuous     | In four digits format, e.g. 1456                |
| 9    | Unique carrier code      | object    | Categorical    | 2 letters code, e.g. PS                         |
| 10   | Flight number            | object    | Categorical    | Number of flights, e.g. 1454                    |
| 11   | Actual elapsed time      | float64   | Continuous     | In two digits format, e.g. 94                   |
| 12   | Scheduled elapsed Time   | float64   | Continuous     | In two digits format, e.g. 94                   |
| 13   | Flight time              | float64   | Continuous     | In two digits format, e.g. 94                   |
| 14   | Arrival time             | float64   | Continuous     | In two digits format, e.g. 94                   |
| 15   | Departure time           | float64   | Continuous     | In two digits format, e.g. 94                   |
| 16   | Origin                   | object    | Categorical    | 3 letters code eg. LHR                          |
| 17   | Destination              | object    | Categorical    | 3 letters code, e.g. MAN                        |
| 18   | Distance                 | int64     | Continuous     | The miles covered, e.g. 448                     |
| 19   | Diverted                 | int64     | Continuous     | In two binary format, e.g. 0 or 1               |
| 20   | Carrier delay            | int64     | Continuous     | In two digits format, e.g. 94                   |
| 21   | Weather delay            | int64     | Continuous     | In two digits format, e.g. 94                   |



**Fig. 2** Schematic diagram for the determining-processing output

### 4.3 Data transformation using Yeo-Johnson transformation

We then pass the pre-processed data  $(P)_{N \times T}$  to the data transformation phase, where the Yeo-Johnson transformation is used [66]. We find that the Yeo-Johnson transformation is better than other transformation methods because it produces well-organised data that is easier to use. Correctly formatted and transformed data improves the data quality and protects applications from potential landmines, such as null values, unexpected duplicates, incorrect indexing, and incompatible formats. Furthermore, the Yeo-Johnson transformation can make the data distribution more symmetric, and it does not require that the value be strictly positive. We estimate the parameters by applying this transformation to the response variable using the maximum penalised likelihood model. The Yeo-Johnson transformation expression is shown in Eq. 3.

$$J = \beta(\delta, M) = \begin{cases} \left( \frac{(M + 1)^\delta - 1}{\delta} \right); & M \geq 0, \delta \neq 0 \\ \log(M + 1); & M \geq 0, \delta = 0 \\ -\frac{((-M + 1)^{2-\delta} - 1)}{(2 - \delta)}; & M < 0, \delta \neq 2 \\ -\log(-M + 1); & M < 0, \delta = 2 \end{cases} \quad (3)$$

where  $J = \beta(\delta, M)$  is the output of the data transformation of  $R_j$ ,  $\delta$  is any real number of the power parameter in a piecewise function form that makes it continuous at the point of singularity ( $\delta = 0$ ), where  $\delta = 1$  is an identity transformation and  $M$  is the data vector with zero, negative and positive values or observations without restricting the type of observation needed stabilising the variance of the input datasets, which increase the features distribution symmetry and improving the validity of association measures (such as the correlation between features).

### 4.4 Feature fusion using proposed SSDCA-based deep RNN

Once the data transformation is performed, we then do the feature fusion based on the trained Deep RNN model of our SSDCA algorithm. In feature fusion, SSDs have greater throughput, continuous access times for quicker boot-ups, faster file transfers, and overall excellent performance. The CAViaR model specifies the evolution of the quantile over time using a special type of autoregressive process described in detail later. It applies to real data and can adapt to new risk environments. Thus, the SSDCA has the advantages of both SSD and CAViaR in feature fusion. After data transformation, the size of the data remains  $N \times T$ , hence for reducing the features, the feature fusion

step is fundamental in determining flight delays effectively. For example, the  $T$  attributes having six columns, meaning six features, as shown in the set  $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ .

#### 4.4.1 Correlation-based feature sorting

In this step, we perform the correlation for the six features. For example, we compute the correlation based on  $f_1$  target values; thus, we get six correlation values. Then, we couple the features and change the columns based on high correlation values, and it precedes the feature fusion process for reducing the features. We applied spearman’s rho correlation because our data has an ordinal level of measurement and follows a categorical distribution to measure the pairwise monotonicity of relationships by ranking from highest to lowest and choosing the high correlation values. Also, each of the variable change in one direction of category without necessarily at the same rate. It can be calculated as shown in Eq. 4.

$$C_s = 1 - \frac{6 \sum d_i^2}{(n^2 - n)} \quad (4)$$

where  $C_s$  is the rank between two features,  $d_i$  is the difference between two variable ranks for each data pair while  $\sum d_i^2$  Represents the square sum difference between two features rank, and  $n$  is the number of instances.

#### 4.4.2 Feature fusion and determination of $\beta$ based on deep recurrent neural network

After feature fusion, we determine the  $\beta$  based on the Deep RNN classifier. Here, we take the features as the Deep RNN classifier’s input for performing feature fusion using weights and bias-related with hidden layers.

**4.4.2.1 Feature fusion** Once we change the columns based on the correlation values, we perform the feature fusion based on Eq. 5.

$$f_k^{\text{fused}} = \sum_{i=1}^S \frac{\beta}{j} f_i \quad (5)$$

where  $i = 1 + \frac{T}{l} j = 1 \dots l$ ,  $T$  represents the total features and  $l$  refers to the number of selected features as shown in Eq. 6.

$$l = \frac{T}{S}; 1 \leq k \leq S \quad (6)$$

where the term  $k$  is the index of fused features and  $S$  is the last index of the fused features.

**4.4.2.2 Structure of deep RNN** This network architecture contains several recurrent hidden layers in the network hierarchy. However, the recurrent connection exists only among hidden layers. It takes the previous state output as the input to the next state and the iteration process with the hidden state information begins. The advantage of using a Deep RNN [67] classifier over state-of-art classifiers is that it operates effectively under changes of the input feature length. A recurrent feature associated with Deep RNN yields high performance in feature fusion accuracy and the required number of iterations. We used Deep RNN to find the best parameter values because it can process inputs of any length and approximate any function. We illustrate the structure of the Deep RNN classifier in Fig. 3.

The Deep RNN classifier defined the input vector of the layer  $f$  at time  $h$  as  $U^{(f,h)} = \{U_1^{(f,h)}, U_2^{(f,h)}, \dots, U_r^{(f,h)}, \dots, U_n^{(f,h)}\}$ , and the output vector of

the layer  $f$  at time  $h$  as  $O^{(f,h)} = \{O_1^{(f,h)}, O_2^{(f,h)}, \dots, O_r^{(f,h)}, \dots, O_s^{(f,h)}\}$ . The set of units for each element of output and input vectors  $n$  refers to the arbitrary unit number of layer  $f$  and  $v$  signifies total units. However, the total units of  $(f - 1)^{th}$  layer and the arbitrary unit number  $\gamma$  and  $\phi$ . Here, the input propagation weight from  $(f - 1)^{th}$  layer to  $f^{th}$  layer is shown by  $W^{(f)} \in I^{v \times \gamma}$ , and  $f^{th}$  we denote recurrent layer weight as  $w^{(f)} \in I^{v \times v}$ , the term  $I$  signifies weight set. Therefore, the input vector components are as in Eq. 7.

$$U_n^{(f,h)} = \sum_{u=1}^{\gamma} z_{nu}^{(f)} O_z^{(f-1,h)} + \sum_{n'}^v \varepsilon_{nn'}^{(f)} O_{n'}^{(f,h-1)} \tag{7}$$

where,  $z_{nu}^{(f)}$  and  $\varepsilon_{nn'}^{(f)}$  refer to elements of  $W^{(f)}$  and  $w^{(f)}$ , the term  $n'$  signifies the arbitrary unit number of  $f^{th}$  layer. Thus, the output vector of  $f^{th}$  the layer elements is as in Eq. 8.

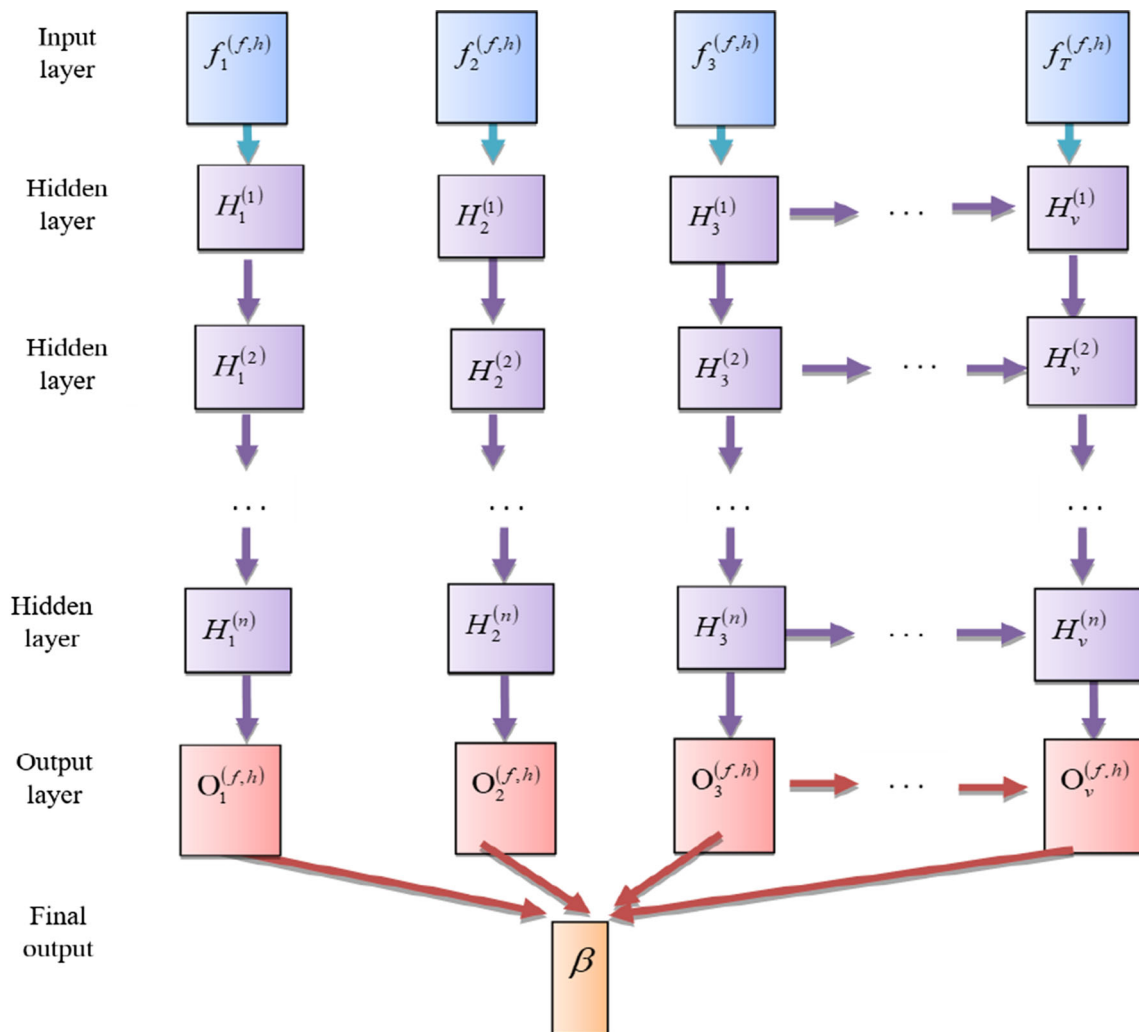


Fig. 3 Structure of Deep RNN



$$O_n^{(f,h)} = \mu^{(f)} \left( U_n^{(f,h)} \right) \tag{8}$$

where  $\mu^{(f)}$  refers to the activation function. Thus, activation functions, like sigmoid function, are indicated by  $\mu(U) = \tanh(U)$  rectified linear unit function (ReLU) is  $\mu(U) = \max(U, \hbar)$ , and we represent the logistic sigmoid function as  $\mu(U) = \frac{1}{(1+e^{-U})}$  are the employed activation function. To simplify the prediction process, consider  $\ell^{th}$  weight as  $z_{n\ell}^{(f)}$  and  $\ell^{th}$  unit as  $O_\ell^{(f-1,h)}$  and hence, we specified the bias as shown in Eq. 9.

$$O^{(f,h)} \text{ or } \beta = \mu^{(f)} \cdot \left( W^{(f)} O^{(f-1,h)} + w^{(f)} \cdot O^{(f,h-1)} \right) \tag{9}$$

Thus, the output of the classifier is indicated as  $O^{(f,h)}$  or  $\beta$ . And  $W^{(f)}$  is the input propagation weight, recurrent layer weight as  $w^{(f)}$ , unit as  $O_\ell^{(f-1,h)}$  and  $\mu^{(f)}$  refers to the activation function. The classifier is tuned by a proposed optimisation algorithm for updating the classifier’s weights, enabling effective prediction performance.

**4.4.2.3 Training data** After the determination  $\beta$ , we computed the training data value by correlating the mean vector of data belonging to the class. The training data step is significant for the selected six features instance in model building to generate an output, as shown in Table 2.

We compute the training of the output  $\beta$  using Eq. 10.

$$\beta_{training}^{(i)} = Correlation(d_i, M_{d_i}) \tag{10}$$

where  $M_{d_i}$  refers to the mean vector of data  $d_i$  belonging to the class.

**4.4.3 Training of deep recurrent neural network using social ski driver conditional autoregressive**

We carried out the training process of Deep RNN [67] using the developed SSDCA technique for finding optimal weights to tune the Deep RNN [30] for feature fusion and flight delay prediction. The feature fusion based on the developed SSDCA categorises data by obtaining optimal weights and dealing with new data characteristics from distributed resources. The naturally

**Table 2** Determination of training data

| $d_i/f_T$ | $f_1$         | $f_2$         | $f_3$         | $f_T$         | Target             |
|-----------|---------------|---------------|---------------|---------------|--------------------|
| $d_1$     | $C(d_1, f_1)$ | $C(d_1, f_2)$ | $C(d_1, f_3)$ | $C(d_1, f_T)$ | $\beta_{training}$ |
| $d_2$     | $C(d_2, f_1)$ | $C(d_2, f_2)$ | $C(d_2, f_3)$ | $C(d_2, f_T)$ | $\beta_{training}$ |
| $d_3$     | $C(d_3, f_1)$ | $C(d_3, f_2)$ | $C(d_3, f_3)$ | $C(d_3, f_T)$ | $\beta_{training}$ |
| $d_4$     | $C(d_4, f_1)$ | $C(d_4, f_2)$ | $C(d_4, f_3)$ | $C(d_4, f_T)$ | $\beta_{training}$ |

inspired SSD algorithm [68] has several evolutionary optimisation algorithms for reducing SVMs parameters and improving system performance. The goal of SSD is to search in space for near-optimal or optimal solutions. Thus, this approach is efficient in generating improved features for solving multi-aim optimisation problems. The method solves highly nonlinear problems with complicated constraints, can deal with heterogeneous data.

On the other hand, the CAViaR model [73] has received much attention to distributing direct returns to the quantile behaviour. We employ regression quantile to estimate and update the parameters. Tests of the model adequacy use a criterion independent of each probability period of all past information processes. The method also increased the convergence process and the diversity of solutions and improved the balance between exploitation and exploration. CAViaR models can adapt to new risk environments. Thus, integrating CAViaR with SSD is to enhance the overall algorithmic performance. We give the algorithmic steps of the proposed SSDCA as follows.

**4.4.3.1 Initialisation** The initial step of the SSDCA algorithm is the search agent’s location initialisation, where the user determines the total number of agents. The agent’s position is, as shown in Eq. 11.

$$X_v^t; \quad (1 \leq v \leq z) \tag{11}$$

where the term  $X_v^t$  refers to the agent’s location at a time  $t$ ,  $v$  is the velocity as and  $z$  is the total number of samples.

**4.4.3.2 Objective function evaluation** We select the optimal agent location using the minimal learning error as an optimal solution. We estimate the objective function error using Eqs. 12, 13, 14 and 15.

$$MSE = \frac{1}{Z} \sum_{h=1}^z (I_h - \beta_h)^2 \tag{12}$$

$$RMSE = \sqrt{\frac{1}{Z} \sum_{h=1}^z (I_h - \beta_h)^2} \tag{13}$$

$$MAE = \frac{1}{Z} \sum_{h=1}^z |I_h - \beta_h| \tag{14}$$

$$AC = \frac{\text{TRUE POSITIVE} + \text{TRUE NEGATIVE}}{\text{TRUE POSITIVE} + \text{FALSE POSITIVE} + \text{TRUE NEGATIVE} + \text{FALSE NEGATIVE}} \tag{15}$$

where the classifiers estimated target, the output is  $I_h$  and  $\beta_h$ . The term  $z$  denotes the total number of samples.

**4.4.3.3 Solution update using the SSDCA algorithm** Once we compute the objective functions, the solution undergoes the location update based on SSDCA as shown in Eq. 16. The standard equation of the SSD velocity  $K_{lm}^{t+1}$  is given by Eq. 17.

$$X_{lm}^{t+1} = X_{lm}^t + K_{lm}^{t+1} \tag{16}$$

$$K_{lm}^{t+1} = \begin{cases} e \sin(m_1)(B_{lm}^t - X_{lm}^t) + \sin(m_1)(A_{lm}^t - X_{lm}^t); & s_2 \leq 0.5 \\ e \cos(m_1)(B_{lm}^t - X_{lm}^t) + \cos(m_1)(A_{lm}^t - X_{lm}^t); & s_2 > 0.5 \end{cases} \tag{17}$$

where  $K_{lm}$  signifies velocity of  $X_{lm}$ , and uniformly distributed random numbers represent  $m_1$  and  $m_2$  range between 0 to 1.  $B_{lm}$  refers to the optimal agent solution and  $A_{lm}$  denotes means a global solution for the entire population. Hence, Eq. (17) depicts the standard SSD equation that incorporates the CAViaR updated equation. Therefore, the CAViaR standard equation is given as in Eq. 18.

$$X_{lm}^t = \alpha_p + \sum_{h=1}^k \alpha_h X_{lm}(t-h) + \sum_{l=1}^r \alpha_l f(X_{lm}(t-h)) \tag{18}$$

Substituting  $k = r = 2$  in Eq. 18 becomes Eq. 19.

$$X_{lm}^t = \alpha_0 + \alpha_1 X_{lm}(t-1) + \alpha_2 X_{lm}(t-2) + \alpha_1 f(X_{lm}(t-1)) + \alpha_2 f(X_{lm}(t-2)) \tag{19}$$

$$X_{lm}^{t+1} = X_{lm}^t + e \sin(m_1)(B_{lm}^t - X_{lm}^t) + \sin(m_1)(A_{lm}^t - X_{lm}^t) \tag{20}$$

Rearranging, Eq. 20, the solution becomes,

$$X_{lm}^{t+1} = X_{lm}^t [1 - e \sin(m_1) - \sin(m_1)] + e \sin(m_1) B_{lm}^t + \sin(m_1) A_{lm}^t \tag{21}$$

Substituting Eqs. 19 in 21 yields:

$$X_{lm}^{t+1} = \alpha_0 + \alpha_1 X_{lm}(t-1) + \alpha_2 X_{lm}(t-2) + \alpha_1 f(X_{lm}(t-1)) + \alpha_2 f(X_{lm}(t-2)) [1 - e \sin(m_1) - \sin(m_1)] + e \sin(m_1) B_{lm}^t + \sin(m_1) A_{lm}^t \tag{22}$$

Following the same procedure when considering case-2 in Eq. 17, which we refer to as Eq. 17b, we get:

$$K_{lm}^{t+1} = X_{lm}^t [e \cos(m_1) + \cos(m_1)] + e \cos(m_1) B_{lm}^t + \cos(m_1) * A_{lm}^t \tag{23}$$

Similarly, substituting Eqs. 19 in 23 results to:

$$X_{lm}^{t+1} = \alpha_0 + \alpha_1 X_{lm}(t-1) + \alpha_2 X_{lm}(t-2) + \alpha_1 f(X_{lm}(t-1)) + \alpha_2 f(X_{lm}(t-2)) [e \cos(m_1) + \cos(m_1)] + e \cos(m_1) B_{lm}^t + \cos(m_1) * A_{lm}^t \tag{24}$$

Thus, the final expression for the updated equation of the proposed SSDCA applied in performing the flight delay prediction is shown below in Eq. 25:

$$K_{lm}^{t+1} = \begin{cases} \alpha_0 + \alpha_1 X_{lm}(t-1) + \alpha_2 X_{lm}(t-2) + \alpha_1 f(X_{lm}(t-1)) + \alpha_2 f(X_{lm}(t-2)) \\ [1 - e \sin(m_1) - \sin(m_1)] + e \sin(m_1) B_{lm}^t + \sin(m_1) A_{lm}^t & ; s_2 \leq 0.5 \\ \alpha_0 + \alpha_1 X_{lm}(t-1) + \alpha_2 X_{lm}(t-2) + \alpha_1 f(X_{lm}(t-1)) + \alpha_2 f(X_{lm}(t-2)) \\ [e \cos(m_1) + \cos(m_1)] + e \cos(m_1) B_{lm}^t + \cos(m_1) * A_{lm}^t; & s_2 > 0.5 \end{cases} \tag{25}$$

where  $f(\cdot)$  denotes the fitness function considering the case-1 in Eq. 16, which we refer to as Eq. 16a.

Now substituting Eq. 17a in Eq. 16 results to:

**4.4.3.4 Recheck the feasibility** Once we test the updated position and each solution of the objective functions, we consider the optimal solution to be the one with maximal fitness.

---

### 4.3.3 Algorithm 1: Pseudocode of the SSDCA-based Deep RNN algorithm

---

```

1:   Input: Search agents location  $X'_v; (1 \leq v \leq z)$ 
2:   Output: Optimal location
3:   Procedure:
4:   begin
5:       Agent's velocities and position are initialised (11)
6:       Consider fitness value should be minimal
7:       while stopping criteria are not met, do
8:           for whole agents do
9:               Compute fitness value based on equation (12)
10:              Sort agents based on fitness value
11:              Determine mean global solution and previous optimal solution
12:              Update the SSD equation using (17)
13:              Update the CAViaR equation using (18)
14:              Update the SSDCA equation based on (25)
15:           end for
16:       Check solution feasibility
17:       Return optimal solution
18:        $t = t + 1$ 
19:       end while
20:       The best solution is achieved
21:   End

```

---

**4.4.3.5 Termination** The steps  $i$  to  $iv$  continue until we meet a specified maximum iteration number or achieve an optimal solution. The pseudocode of the developed SSDCA approach is shown in Algorithm 1.

We call  $F$  the output obtained from line 20 of algorithm 1.

#### 4.4.4 Flight delay prediction based on proposed SSDCA-based deep LSTM

The feature fusion output  $F$  is plug into the Deep LSTM classifier [70] for predicting the flight delays training by the novel SSDCA, incorporating the SSD algorithm [68] and CAViaR [69]. We employ the Deep LSTM classifier because it takes less computational time to process the input time series data. Another significant benefit of Deep LSTM is that it requires less training data and can generate optimal results at a specific instance. This section presents an elaborate discussion of the Deep LSTM structure and framework utilised for optimising training weights.

### 4.5 Deep LSTM Architecture

We feed the Deep LSTM classifier with the fusion features  $F$  obtained from the feature fusion to accomplish the flight delay prediction. The Deep LSTM effectively achieves the

flight delay prediction by applying the classifier's memory cell, advantageous for the other classifiers. Here, the memory cell utilises the stored state information and acts as an accumulator. The Deep LSTM uses the input and their neighbours' past states to predict future states based on the convolutional operator and Hadamard product. We achieve an effective flight delay prediction using the high transitional kernel where the encoding and forecasting layers form the Deep LSTM structure. The forecasting network receives the initial input and outputs the cell encoding network. Also, the Deep LSTM uses a self-parametrised gate to clear cell access. The memory cell receives the state information if the gate gets activated when subject to the input. Whenever the forget gate is ON, the classifier forgets the past cell information. Because of insufficient decaying error backflow, information storage takes a long time over extended time intervals with a recurrent backpropagation. A significant advantage of the classifier is how it manages the information flow.

LSTM is an efficient and gradient-based method. The LSTM can bridge the minimal time lags learning to enforce constant error flow-through "constant error carrousel" over 1000 discrete time steps within the particular unit. In terms of space and time, the LSTM is local. The LSTM algorithm can solve long-time lags of complex tasks that

other recurrent neural network algorithms cannot solve. Here, there is a convolutional architecture in the input to the state-to-state transition so that the forecasting problem solve has a structure in the LSTM. The classifier's rows and columns are the 3D tensors in spatial dimensions with inputs, hidden state and gates utilise. We define the input cell and states as vectors in spatial grid form to enhance the flight delay prediction performance. However, the classifier employs the previous state and neighbours' input cell to compute the future cell states. Thus, the Deep LSTM classifier includes an input  $\{T_1, \dots, T_m\}$ , hidden states  $\{I_1, \dots, I_m\}$ , cell output  $\{D_1, \dots, D_m\}$ , and gates  $x_m, y_m, z_m$ . Figure 4 describes the architecture of the Deep LSTM classifier.

Equation 26 defines the input gate output:

$$x_m = \gamma (S_x^T * T_m + S_x^I * I_{m-1} + S_x^D \circ D_{m-1} + \lambda^x) \quad (26)$$

where  $T_m$  is the input vector and  $S_x^T$  is the weight among the input layer and gate. The gate activation function is given by  $\gamma$ , and  $S_x^D$  is the weight vector between the cell input and output layer, and  $S_x^I$  is the weight between memory input and output layer, while  $I_{m-1}$  represents the previous cell output, and  $D_{m-1}$  is the previous memory unit output. The term  $\lambda^x$  represents input layer bias, and the character  $*$  represents the Convolutional operator, and the character  $\circ$  is the element-wise multiplication. Equation 27 represents the forget gate output:

$$y_m = \gamma (S_y^T * T_m + S_y^I * I_{m-1} + S_y^D \circ D_{m-1} + \lambda^y) \quad (27)$$

where  $S_y^T$  denotes weight among the input layer forget gate while the term  $S_y^I$  is the weight among the memory unit of

the previous layer and output gate and  $S_y^D$  is the weight among output gate and cells. The term  $\lambda^y$  represents the bias that relates to forgetting gates. Equation 28 represents the output generated from the output gate:

$$z_m = \gamma (S_z^T * T_m + S_z^I * I_{m-1} + S_z^D \circ D_{m-1} + \lambda^z) \quad (28)$$

where  $S_z^T$  represents the weight between the input layer and the output gate and  $S_z^I$  denotes the weight between the output gate and memory unit, and  $S_z^D$  denotes the weight between the output gate and the cell; the term  $\lambda^z$  is the output gate bias based on the activation function. Equation 29 represents the temporary cell state output.

$$\tilde{D}_m = \tanh (S_v^T * T_m + S_v^I * I_{m-1} + \lambda^v) \quad (29)$$

where  $\lambda^v$  represents the bias and  $S_v^T$  represents the weight between cell and input layer, while the symbol  $S_v^I$  represents the weight between cell and the memory unit. The cell output estimates adding the previous and current layer, temporary cell state and a memory unit. Equation 30 and Eq. 31 represent the cell output estimation.

$$D_\tau = y_m \circ D_{m-1} + x_m \circ \tilde{D}_\tau \quad (30)$$

$$D_m = y_m \circ D_{m-1} + x_m \circ \tanh (S_v^T * T_m + S_v^I * I_{m-1} + \lambda^v) \quad (31)$$

Equation 32 expresses the memory unit generated output:

$$I_m = z_m \circ \tanh (D_m) \quad (32)$$

where  $z_m$  represents the output gate, and the term  $I_m$  represents the block output memory. Thus, Eq. 33 represents the generated output  $O_m$ :

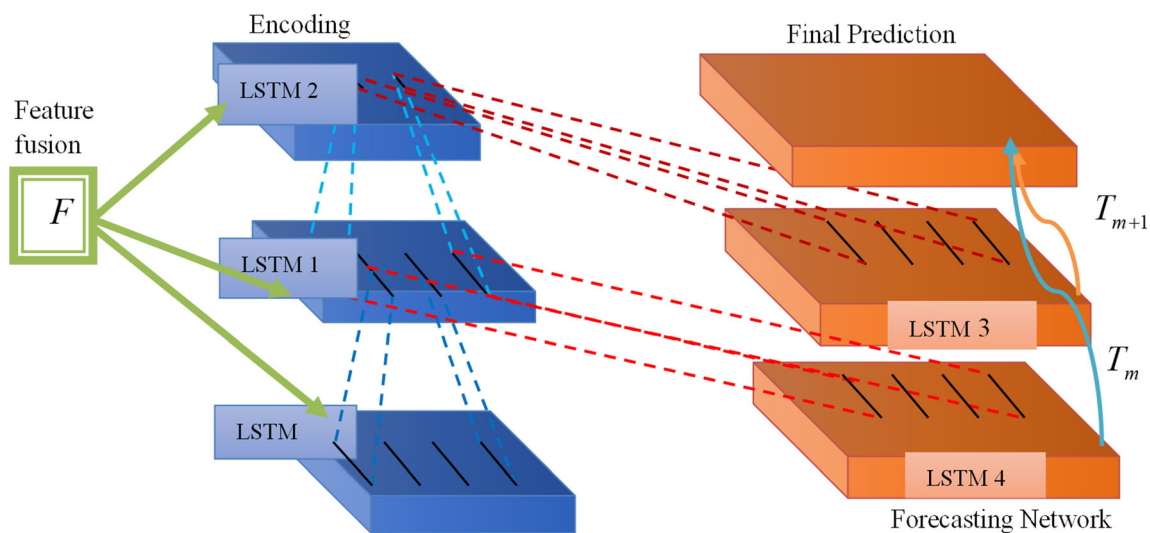


Fig. 4 Architecture of Deep LSTM classifier

$$O_m = \alpha (S_O^t \cdot I_m + \lambda^o) \quad (33)$$

where the output vector of weight among the memory unit is  $S_O^t$  and  $\lambda^o$  representing the output layer bias.

## 5 Results and discussion

This section describes the results of the proposed SSDCA-based Deep LSTM based on some benchmark metrics and compares our method with a set of methods from the literature. Table 3 presents the considered methods and the corresponding special names we give them.

### 5.1 Dataset source

The dataset considered for the experimentation is the flight delay prediction dataset with US flight data downloaded from the US Government Bureau of transportation statistics [71, 72].

#### 5.1.1 January flight delay prediction dataset (dataset-1 [71])

This dataset contains the flights in January 2019 and January 2020. It contains over 400,000 flights, which translate into 400,000 rows. It contains 21 feature columns that specify the features of each flight. It includes destination airport, origin airport, departure time, arrival time, and aircraft information. We used the dataset for predicting flight delays at the destination airport for January.

#### 5.1.2 February flight delay prediction dataset (dataset-2 [72])

This dataset contains the flights in February 2019 and February 2020. It contains over 400,000 flights, which translate into 400,000 rows. It contains 21 feature columns that specify the features of each flight. It includes

**Table 3** Methods considered

| S/No | Methods                      | Special name |
|------|------------------------------|--------------|
| 1    | Deep belief network (DBN)    | Med 1        |
| 2    | Gradient boosting classifier | Med 2        |
| 3    | Information gain-SVM         | Med 3        |
| 4    | Multi-agent approach         | Med 4        |
| 5    | Deep LSTM                    | Med 5        |
| 6    | PSO + Deep LSTM              | Med 6        |
| 7    | ACO + Deep LSTM              | Med 7        |
| 8    | SSD + Deep LSTM              | Med 8        |
| 9    | HBO + Deep LSTM              | Med 9        |
| 10   | EWA + Deep LSTM              | Med 10       |

destination airport, origin airport, departure time, arrival time, and aircraft information. We used the dataset for predicting flight delays at the destination airport for February.

### 5.2 Evaluation metrics

We evaluate the proposed model performance using the MSE, RMSE, MAE and Accuracy metrics [17, 74–76].

#### 5.2.1 MSE

This measures the average square difference between the estimated and target values, as shown in Eq. 10.

#### 5.2.2 RMSE

This measures the square root of the average square difference between the actual and the predicted value, as shown in Eq. 11.

#### 5.2.3 MAE

This measures the Mean of the absolute values of individual prediction errors over all instances in the test set, as shown in Eq. 12.

#### 5.2.4 Accuracy

This measures the estimated value's closeness to a standard or actual value, as shown in Eq. 13.

### 5.3 Method comparisons

We analyse the developed model performance and compare it with the existing methods from the literature [1, 4, 6, 26] and other meta-heuristic approaches implemented with the Deep LSTM in our study.

#### 5.3.1 Model evaluation

In this section, we compare the proposed SSDCA-based Deep LSTM by varying the percentage of the training and testing datasets to investigate the impact on the model's performance.

**5.3.1.1 Analysis using dataset-1** Figure 9 in the appendix illustrates the analysis of the developed approach based on dataset-1 by considering the feature size as '6'. Figure 9a shows the analysis of RMSE regarding training data, with smaller values of RMSE being the favourable ones. When we consider 60% as training data, RMSE measured shows that our proposed method outperforms the existing

**Table 4** Delay prediction using dataset-1

| Method                    | DBN  | Gradient boosting classifier | Information gain-SVM | Multi-agent approach | Deep LSTM | Proposed SSDCA-based deep LSTM |
|---------------------------|------|------------------------------|----------------------|----------------------|-----------|--------------------------------|
| Number of flights delayed | 5300 | 3337                         | 2534                 | 1365                 | 1365      | 1640                           |

methods, as shown in Table 9. For 90% training data, the developed model gained a minimum RMSE of 0.1114, whereas the remaining existing methods did not perform as well, with the actual values shown in Table 9.

Figure 9b shows the analysis of MSE in terms of training and testing data percentage. For 70% of training data, the MSE measured by the existing methods and our proposed method is shown in Table 9. For 90% training data, the developed model achieved a minimum MSE of 0.0134 compared to the other methods, with detailed values shown in Table 9.

Figure 9c shows the analysis of MAE in terms of training and testing data percentage. For 60% of training data, the MAE measured by our proposed method outperforms the other existing methods with a value of 0.1615. Detailed values are shown in Table 9. For 90% training data, the developed model achieved a minimum MAE of 0.0511 compared to the other methods, with detailed values shown in Table 9.

Figure 9d shows the analysis of Accuracy in terms of training and testing data percentage. For 70% of training data, the Accuracy measured by our proposed method outperforms the other existing methods with a value of 0.9156. Table 9 shows the detailed values of the other existing methods. For 90% training data, the developed model achieves an accuracy of 0.9362 compared to the other methods, with detailed values shown in Table 9.

**5.3.1.2 Analysis using dataset-2** Figure 10 in the appendix illustrates the analysis of the developed approach based on dataset-2 by considering the feature size as '6'. Figure 10a shows the analysis of RMSE regarding training data, with smaller values of RMSE being the favourable ones. When we consider 60% as training data, RMSE measured shows that our proposed method outperforms the existing methods, as shown in Table 10. For 90% training data, the developed model gained a minimum RMSE of 0.1157, whereas the remaining existing methods did not perform as well, with the actual values shown in Table 10.

Figure 10b shows the analysis of MSE in terms of training and testing data percentage. For 70% of training data, the MSE measured by the existing methods and our proposed method is shown in Table 9. For 90% training data, the developed model achieved a minimum MSE of 0.0134 compared to the other methods, with detailed values shown in Table 10.

Figure 10c shows the analysis of MAE in terms of training and testing data percentage. For 60% of training data, the MAE measured by our proposed method outperforms the other existing methods with a value of 0.1701. Detailed values are shown in Table 9. For 90% training data, the developed model achieved a minimum MAE of 0.0557 compared to the other methods, with detailed values shown in Table 10.

Figure 10d shows the analysis of Accuracy in terms of training and testing data percentage. For 70% of training data, the Accuracy measured by our proposed method outperforms the other existing methods with a value of 0.9085. Table 10 shows the detailed values of the other existing methods. For 90% training data, the developed model achieves an accuracy of 0.9252 compared to the other methods, with detailed values shown in Table 10.

### 5.3.2 Delay prediction analysis

We now set the training data to be 70% of the dataset. Figure 5 represents the delay prediction gained using dataset-1. On January 15, 2020, the actual number of flights delayed was 1707, shown under the (original label). Our proposed method predicts 1640 and being the most accurate method. Detailed results of all methods appear in Table 4.

We now set the training data to be 70% of the dataset. Figure 6 represents the delay prediction gained using dataset-1. On February 20, 2020, the actual number of flights delayed was 2772, shown under the (original label). Our proposed method predicts 2874 and being the most accurate method. Detailed results of all methods appear in Table 5.

### 5.3.3 Convergence analysis

Figure 7 shows the analysis for convergence using dataset-1. For dataset-1, the MSE of the considered Deep LSTM method coupled with the different optimisers. PSO + Deep LSTM is 0.0132, ACO + Deep LSTM is 0.0130, SSD + Deep LSTM is 0.0130, HBO + Deep LSTM is 0.0128, and EWA + Deep LSTM the convergence is 0.0127. The MSE of the proposed SSDCA-based Deep LSTM is 0.0124. Hence, the proposed algorithm has the best convergence when compared to the other algorithms.

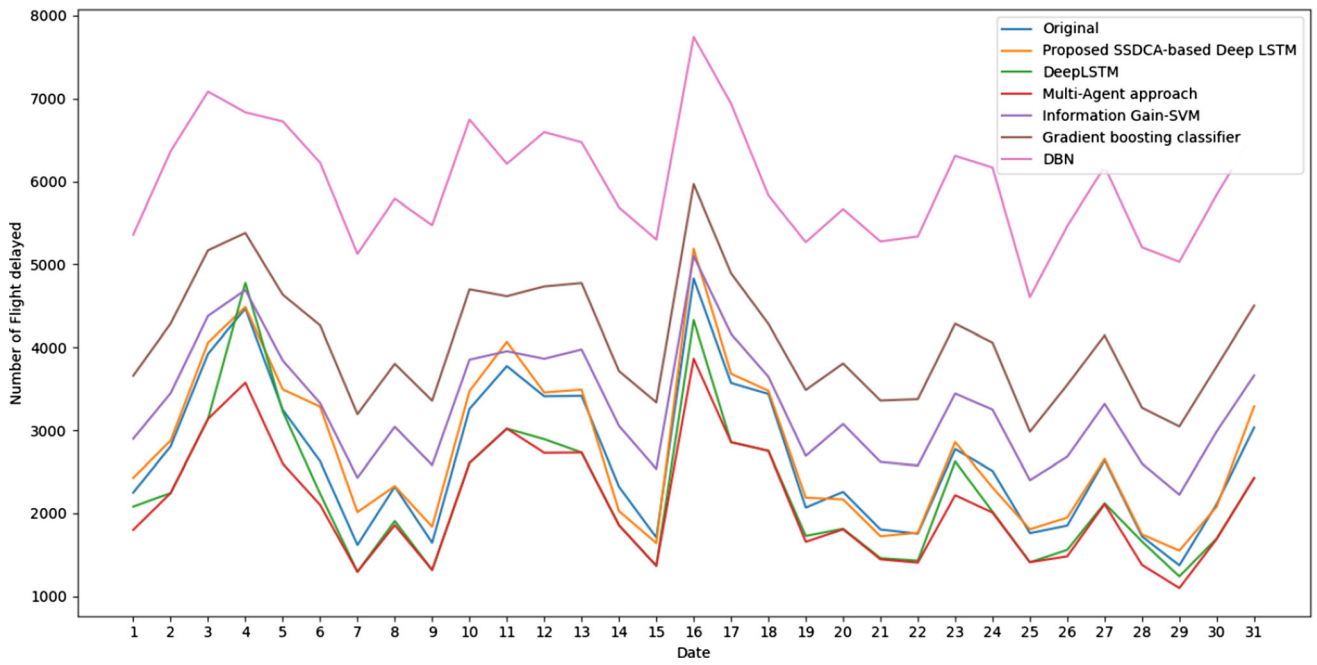


Fig. 5 Delay prediction using dataset-1

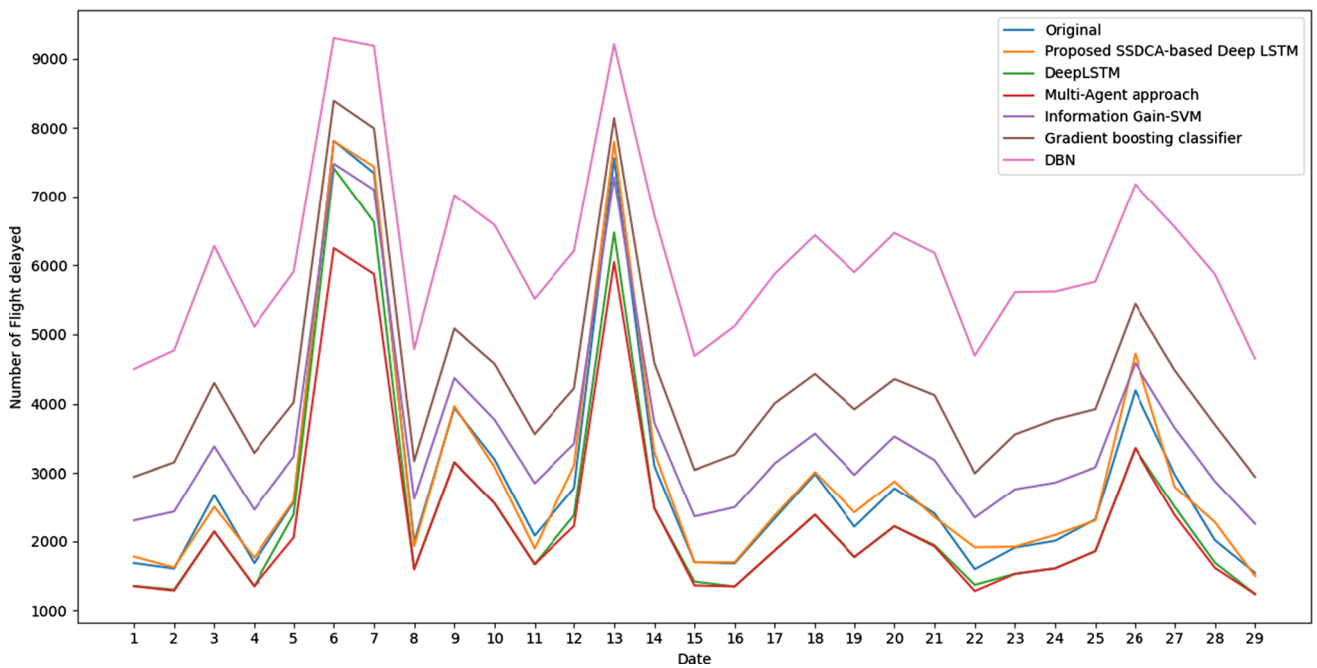
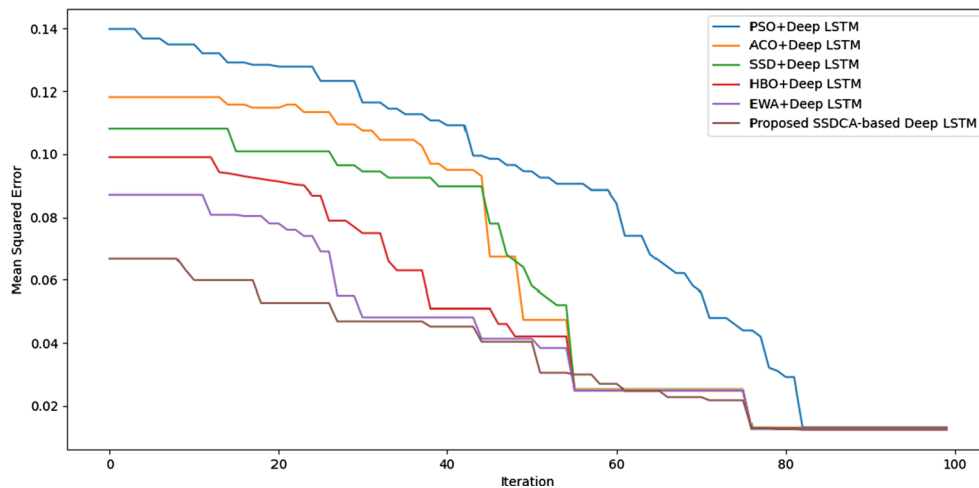


Fig. 6 Delay prediction using dataset-2

Table 5 Delay prediction using dataset-2

| Method                    | DBN  | Gradient boosting classifier | Information gain-SVM | Multi-agent approach | Deep LSTM | Proposed SSDCA-based deep LSTM |
|---------------------------|------|------------------------------|----------------------|----------------------|-----------|--------------------------------|
| Number of flights delayed | 6472 | 4357                         | 3525                 | 2217                 | 2217      | 2874                           |



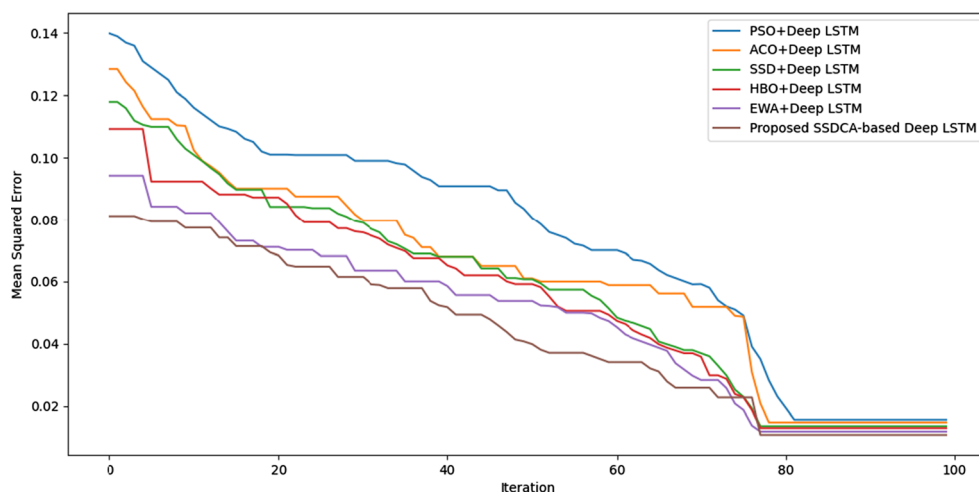
**Fig. 7** Convergence analysis using dataset-1

Figure 8 shows the analysis of convergence using dataset-2. For dataset-2, the MSE of the considered Deep LSTM method coupled with the different optimisers. PSO + Deep LSTM is 0.0157, ACO + Deep LSTM is 0.0148, SSD + Deep LSTM is 0.0136, HBO + Deep LSTM is 0.0130, and the convergence of EWA + Deep LSTM is 0.0119, and for the Proposed SSDCA-based Deep LSTM the convergence is 0.0108. Hence, the proposed algorithm has the best convergence when compared to the other algorithms.

#### 5.4 Comparative discussion

Table 6 illustrates the comparative results of the proposed SSDCA-based Deep LSTM approach. When considering the feature size as '6' for dataset-1 our proposed approach gained minimum RMSE and MSE values. The RMSE measured by the proposed SSDCA-based Deep LSTM is

0.1114. In contrast, the existing DBN, gradient boosting classifier, information gain-SVM, multi-agent approach, and Deep LSTM achieved the RMSE of 0.2953, 0.2229, 0.1878, 0.1165, and 0.1154, respectively. The MSE achieved by the existing DBN, gradient boosting classifier, information gain-SVM, multi-agent approach, Deep LSTM, and proposed SSDCA-based Deep LSTM for feature size '6' is 0.0872, 0.0497, 0.0353, 0.0136, 0.0133, and 0.0124 respectively. Table 3 shows that the proposed approach achieved minimal RMSE and MSE of 0.1065 and 0.0113 with dataset-2. The existing DBN, gradient boosting classifier, information gain-SVM, multi-agent approach, and Deep LSTM achieved the RMSE of 0.2971, 0.2240, 0.1892, 0.121, 0.1143 and 0.1065, respectively. The MSE achieved by the existing DBN, gradient boosting classifier, information gain-SVM, multi-agent approach, Deep LSTM, and proposed SSDCA-based Deep LSTM for



**Fig. 8** Convergence analysis using dataset-2



**Table 6** Comparative discussion

| Dataset   | Metrics | DBN    | Gradient boosting classifier | Information gain-SVM | Multi-agent approach | Deep LSTM | Proposed SSDCA-based deep LSTM |
|-----------|---------|--------|------------------------------|----------------------|----------------------|-----------|--------------------------------|
| Dataset 1 | RMSE    | 0.2953 | 0.2229                       | 0.1878               | 0.1165               | 0.1154    | 0.1114                         |
|           | MSE     | 0.0872 | 0.0497                       | 0.0353               | 0.0136               | 0.0133    | 0.0124                         |
| Dataset 2 | RMSE    | 0.2971 | 0.2240                       | 0.1892               | 0.1215               | 0.1143    | 0.1065                         |
|           | MSE     | 0.0883 | 0.0502                       | 0.0358               | 0.0148               | 0.0131    | 0.0113                         |

feature size '6' is 0.0883, 0.0502, 0.0358, 0.0148, 0.0131 and 0.0113 respectively.

From our analysis, we show that our proposed method offers higher accuracy and needs fewer iterations than the existing methods. Also, it is more versatile in terms of skewness. CAViaR models can adapt to new risk environments, and LSTMs avoid the long-term dependency problem. RNN can process inputs of any length, and RNN with a deep network can approximate any function. Deep RNN can be much more efficient in terms of computation and the number of parameters.

### 5.5 Statistical analysis

Table 7 shows the statistical analysis. In the statistical analysis, the proposed method's RMSE and MSE Mean is 0.1113, 0.0124, and the variance is 0.0002 using dataset-1. Conducting the same statistical analysis using dataset-2, the mean and variance of the proposed method were 0.1063, 0.0111 and 0.0002. The proposed method has a

minimum variance than the other existing methods for both RMSE and MSE.

### 5.6 Computational time analysis

We perform our experiment on a Personal Computer (PC) with Intel(R) Core(TM) i7-9700 CPU with a processor speed of 3.00 GHz and 32 GB RAM. We used libraries such as TensorFlow Core-2.4.1, TensorFlow GPU-2.4.1, NumPy-1.19.1, pandas-0.25.3, sci-kit learn-0.23.2, Scipy-1.5.2, PySimpleGUI-4.29.0 and Matplotlib-3.3.1. Table 8 shows the computational wall time. The computational time analysis shows that our proposed SSDCA-based Deep LSTM needs 300 s and outperforms all other methods. Table 8 shows detail computational time requirements (Table 8).

**Table 7** Statistical analysis

| Dataset   | Metrics  | DBN    | Gradient boosting classifier | Information gain-SVM | Multi-agent approach | Deep LSTM | Proposed SSDCA-based deep LSTM |
|-----------|----------|--------|------------------------------|----------------------|----------------------|-----------|--------------------------------|
| Dataset-1 | RMSE     | 0.2953 | 0.2229                       | 0.1878               | 0.1165               | 0.1154    | 0.1115                         |
|           | Mean     | 0.2947 | 0.2225                       | 0.1872               | 0.1162               | 0.1150    | 0.1113                         |
|           | Variance | 0.0006 | 0.0004                       | 0.0006               | 0.0003               | 0.0004    | 0.0002                         |
|           | MSE      | 0.0872 | 0.0497                       | 0.0353               | 0.0136               | 0.0133    | 0.0124                         |
|           | Mean     | 0.0868 | 0.0490                       | 0.0350               | 0.0131               | 0.0130    | 0.0122                         |
|           | Variance | 0.0004 | 0.0007                       | 0.0003               | 0.0005               | 0.0003    | 0.0002                         |
| Dataset-2 | RMSE     | 0.2971 | 0.2240                       | 0.1892               | 0.1215               | 0.1143    | 0.1065                         |
|           | Mean     | 0.2965 | 0.2236                       | 0.1885               | 0.1210               | 0.1139    | 0.1063                         |
|           | Variance | 0.0006 | 0.0004                       | 0.0007               | 0.0005               | 0.0004    | 0.0002                         |
|           | MSE      | 0.0883 | 0.0502                       | 0.0358               | 0.0148               | 0.0131    | 0.0113                         |
|           | Mean     | 0.0878 | 0.0498                       | 0.0353               | 0.0144               | 0.0126    | 0.0111                         |
|           | Variance | 0.0005 | 0.0004                       | 0.0005               | 0.0004               | 0.0005    | 0.0002                         |

**Table 8** Computational time

| Methods                        | Computational time (sec) |
|--------------------------------|--------------------------|
| DBN                            | 660.00                   |
| Gradient boosting classifier   | 690.00                   |
| Information gain-SVM           | 480.00                   |
| Multi-agent approach           | 630.00                   |
| Deep LSTM                      | 420.00                   |
| Proposed SSDCA-based Deep LSTM | 300.00                   |

## 6 Conclusions and future work

Many factors can cause flight delays, ranging from failure in processes to late departure or aircraft arrival. The reasons for flight delay generate an enormous, complex amount of data used by machine learning methods to make crucial decisions because of the importance of flights arriving or departing on-time for the airport, airlines and passengers. Developing flight delay prediction models with high accuracy is necessary. In this paper, we propose a novel optimised forecasting model with Deep LSTM for flight delay prediction. We utilise the developed SSDCA to train the Deep LSTM and the Deep RNN for fusing the features. The newly proposed SSDCA works by combining SSD and CAViaR algorithms. We employ the novel SSDCA to identify the best weights for effective flight delay prediction using the US Government Bureau of transportation statistics dataset.

Initially, we perform the pre-processing from the input data and then the data transformation based on the Yeo-Johnson transformation. Afterwards, we perform the feature fusion using the Deep RNN to extract the useful features from the original datasets containing complex and nonlinear structures with spatial and temporal correlations. We introduced a two-step training procedure to help integrate the fused features and prediction layer of the model. Finally, we apply the Deep LSTM for flight delay prediction. Here, we use an advanced optimisation method named SSDCA to train the Deep LSTM and Deep RNN. We evaluate our proposed SSDCA-based Deep LSTM and

compare it with four benchmark methods: DBN, gradient boosting classifier, information gain-SVM, multi-agent approach, and Deep LSTM, including other developed meta-heuristic approaches PSO + Deep LSTM, ACO + Deep LSTM, SSD + Deep LSTM, HBO + Deep LSTM and EWA + Deep LSTM. Our approach had a minimal RMSE and MSE of 0.1115 and 0.0124 on dataset-1, with 0.1065 and 0.0113 on dataset-2. The novel SSDCA enabled approach performance has shown superior accuracy with a higher convergence rate than the other five meta-heuristic approaches regarding model accuracy and convergence analysis.

The results we get from our approach are promising. It illustrates how we can improve predicting flight delays using deep learning techniques with optimisation algorithms to inform departing and arriving policies and better airport facilities management. At the same time, the stakeholders can attain efficient and improve passenger satisfaction. The study can be extended further. One direction could be to consider more attributes. While there is no perfect delay prediction, in the future, we plan to investigate further flights with significant delay even though they are rare events. At the same time, we keep the error range on a regular flight with a minimally acceptable level so that departing and arriving policies are developed with less complexity. Secondly, to investigate the performance, we will test our proposed technique on other data sets or other sampling data from different sectors, such as maritime or rail. Finally, we will explore advanced optimisation approaches to improve the proposed method's performance towards a data-driven departure and arrival planning strategy, supporting airports, airlines and passengers to plan travel.

## Appendix

Plots showing the performance results of the methods from Tables 9 and 10.

See Figs. 9 and 10

**Table 9** Results comparison of our proposed method with other existing methods across different evaluation metrics on dataset-1

| S/ No | Training/ testing (%) | Method/metrics | Med 1  | Med 2  | Med 3  | Med 4  | Med 5  | Med 6  | Med 7  | Med 8  | Med 9  | Med 10 | Proposed SSDCA-based deep LSTM |
|-------|-----------------------|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------------------------|
| 1     | 60/40                 | RMSE           | 0.5891 | 0.4471 | 0.3775 | 0.2473 | 0.2328 | 0.2280 | 0.2261 | 0.2258 | 0.2258 | 0.2243 | 0.2235                         |
|       |                       | MSE            | 0.3471 | 0.1999 | 0.1425 | 0.0611 | 0.0542 | 0.0519 | 0.0511 | 0.0511 | 0.0509 | 0.0503 | 0.0499                         |
|       |                       | MAE            | 0.5291 | 0.3971 | 0.3175 | 0.1989 | 0.1845 | 0.1756 | 0.1674 | 0.1645 | 0.1632 | 0.1625 | 0.1614                         |
|       |                       | Accuracy       | 0.7528 | 0.7656 | 0.7857 | 0.7924 | 0.8067 | 0.8156 | 0.8237 | 0.8398 | 0.8478 | 0.8756 | 0.9013                         |
| 2     | 70/30                 | RMSE           | 0.5213 | 0.3874 | 0.3259 | 0.2078 | 0.2016 | 0.1963 | 0.1955 | 0.1950 | 0.1947 | 0.1946 | 0.1931                         |
|       |                       | MSE            | 0.2615 | 0.1501 | 0.1062 | 0.0432 | 0.0406 | 0.0385 | 0.0382 | 0.0380 | 0.0379 | 0.0378 | 0.0373                         |
|       |                       | MAE            | 0.4567 | 0.3367 | 0.2656 | 0.1573 | 0.1515 | 0.1485 | 0.1454 | 0.1450 | 0.1444 | 0.1436 | 0.1375                         |
|       |                       | Accuracy       | 0.7598 | 0.7720 | 0.7896 | 0.7995 | 0.8123 | 0.8195 | 0.8285 | 0.8425 | 0.8534 | 0.8798 | 0.9156                         |
| 3     | 80/20                 | RMSE           | 0.4174 | 0.3154 | 0.2664 | 0.1682 | 0.1646 | 0.1588 | 0.1584 | 0.1582 | 0.1580 | 0.1577 | 0.1576                         |
|       |                       | MSE            | 0.1742 | 0.0995 | 0.0710 | 0.0283 | 0.0271 | 0.0252 | 0.0251 | 0.250  | 0.0249 | 0.0249 | 0.0248                         |
|       |                       | MAE            | 0.3567 | 0.2667 | 0.2056 | 0.1178 | 0.1134 | 0.1076 | 0.1066 | 0.1056 | 0.1020 | 0.1010 | 0.1010                         |
|       |                       | Accuracy       | 0.7754 | 0.7845 | 0.7945 | 0.8045 | 0.8196 | 0.8295 | 0.8367 | 0.8543 | 0.8634 | 0.8956 | 0.9268                         |
| 4     | 90/10                 | RMSE           | 0.2952 | 0.2228 | 0.1878 | 0.1164 | 0.1154 | 0.1148 | 0.1140 | 0.1138 | 0.1133 | 0.1126 | 0.1114                         |
|       |                       | MSE            | 0.0871 | 0.0496 | 0.0352 | 0.1356 | 0.0133 | 0.0131 | 0.0130 | 0.0129 | 0.0128 | 0.0126 | 0.1024                         |
|       |                       | MAE            | 0.2487 | 0.1674 | 0.1267 | 0.0616 | 0.0594 | 0.0585 | 0.0580 | 0.0572 | 0.0572 | 0.0567 | 0.0511                         |
|       |                       | Accuracy       | 0.8034 | 0.8278 | 0.8584 | 0.8645 | 0.8756 | 0.8834 | 0.8907 | 0.9056 | 0.9156 | 0.9265 | 0.9361                         |

**Table 10** Results comparison of our proposed method with other existing methods across different evaluation metrics on dataset-2

| S/ No | Training/ testing (%) | Method/metrics | Med 1  | Med 2  | Med 3  | Med 4  | Med 5  | Med 6  | Med 7  | Med 8  | Med 9  | Med 10 | Proposed SSDCA-based deep LSTM |
|-------|-----------------------|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------------------------|
| 1     | 60/40                 | RMSE           | 0.5904 | 0.4474 | 0.3786 | 0.2478 | 0.2428 | 0.2408 | 0.2397 | 0.2382 | 0.2350 | 0.2338 | 0.2302                         |
|       |                       | MSE            | 0.386  | 0.2001 | 0.1433 | 0.0614 | 0.0589 | 0.0580 | 0.0574 | 0.0567 | 0.0552 | 0.0546 | 0.0530                         |
|       |                       | MAE            | 0.5315 | 0.3956 | 0.3267 | 0.1967 | 0.1915 | 0.1895 | 0.1825 | 0.1795 | 0.1750 | 0.1732 | 0.1701                         |
|       |                       | Accuracy       | 0.7598 | 0.7756 | 0.7834 | 0.7956 | 0.8052 | 0.8167 | 0.8256 | 0.8367 | 0.8578 | 0.8626 | 0.8942                         |
| 2     | 70/30                 | RMSE           | 0.5126 | 0.3869 | 0.3280 | 0.2103 | 0.2091 | 0.2076 | 0.2073 | 0.2058 | 0.2046 | 0.2016 | 0.1984                         |
|       |                       | MSE            | 0.2628 | 0.1497 | 0.1076 | 0.0442 | 0.0437 | 0.0431 | 0.0429 | 0.0423 | 0.0418 | 0.0406 | 0.0393                         |
|       |                       | MAE            | 0.4626 | 0.3367 | 0.2756 | 0.1576 | 0.1491 | 0.1476 | 0.1467 | 0.1445 | 0.1427 | 0.1406 | 0.1375                         |
|       |                       | Accuracy       | 0.7645 | 0.7834 | 0.7945 | 0.8013 | 0.8146 | 0.8245 | 0.8356 | 0.8467 | 0.8596 | 0.8694 | 0.9085                         |
| 3     | 80/20                 | RMSE           | 0.4183 | 0.3159 | 0.2681 | 0.1717 | 0.1703 | 0.1702 | 0.1701 | 0.1696 | 0.1687 | 0.1680 | 0.1675                         |
|       |                       | MSE            | 0.1750 | 0.0998 | 0.0178 | 0.0294 | 0.0290 | 0.0290 | 0.0289 | 0.0287 | 0.0284 | 0.0282 | 0.0280                         |
|       |                       | MAE            | 0.3567 | 0.2658 | 0.2181 | 0.1217 | 0.1203 | 0.1202 | 0.1201 | 0.1196 | 0.1187 | 0.1180 | 0.1175                         |
|       |                       | Accuracy       | 0.7756 | 0.7956 | 0.8056 | 0.8145 | 0.8256 | 0.8352 | 0.8426 | 0.8567 | 0.8631 | 0.8745 | 0.9156                         |
| 4     | 90/10                 | RMSE           | 0.2968 | 0.2238 | 0.1888 | 0.1214 | 0.1200 | 0.1198 | 0.1183 | 0.1181 | 0.1174 | 0.1165 | 0.1157                         |
|       |                       | MSE            | 0.0881 | 0.0500 | 0.0356 | 0.0147 | 0.0144 | 0.0143 | 0.0140 | 0.0139 | 0.0137 | 0.0135 | 0.0134                         |
|       |                       | MAE            | 0.2368 | 0.2238 | 0.1388 | 0.0614 | 0.0600 | 0.0593 | 0.0583 | 0.0581 | 0.0573 | 0.0565 | 0.0557                         |
|       |                       | Accuracy       | 0.7967 | 0.8042 | 0.8152 | 0.8221 | 0.8372 | 0.8442 | 0.8517 | 0.8615 | 0.8742 | 0.9042 | 0.9252                         |

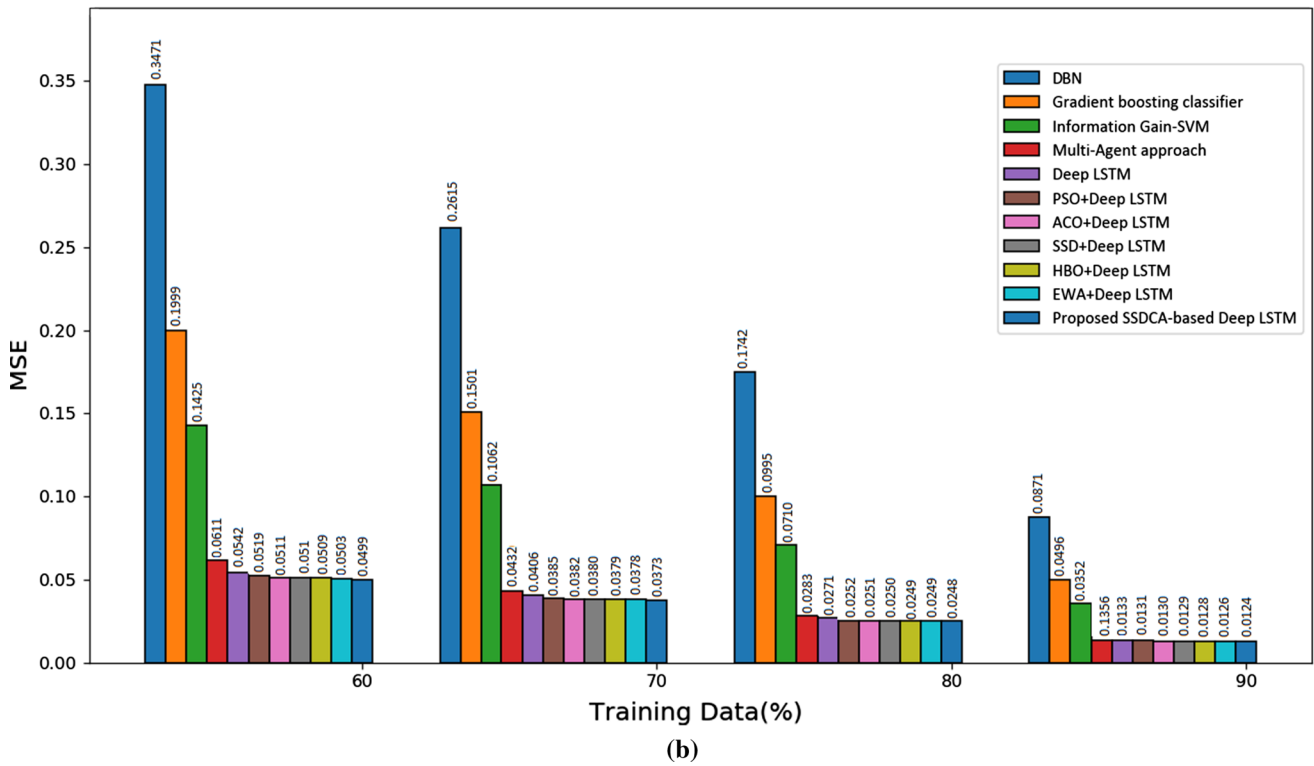
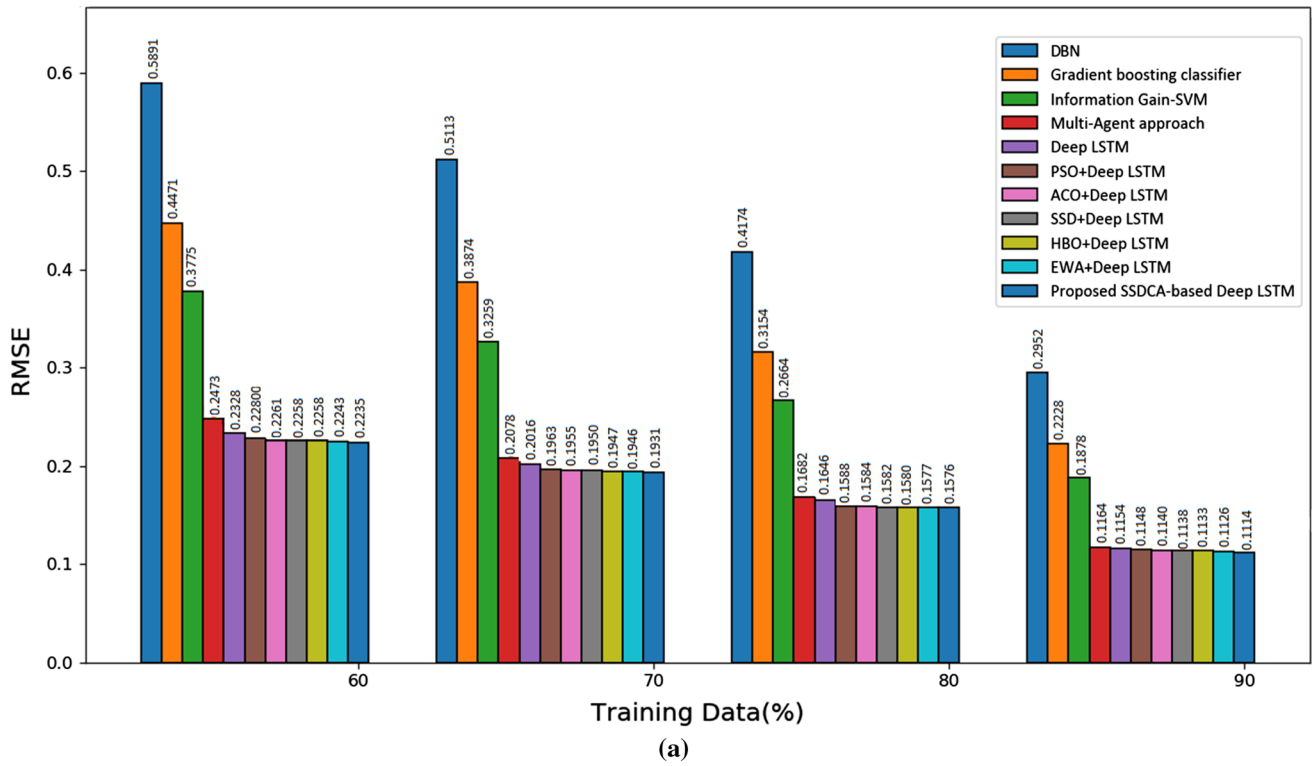


Fig. 9 Analysis using dataset-1 with feature size 6, a RMSE, b MSE, c MAE, d Accuracy

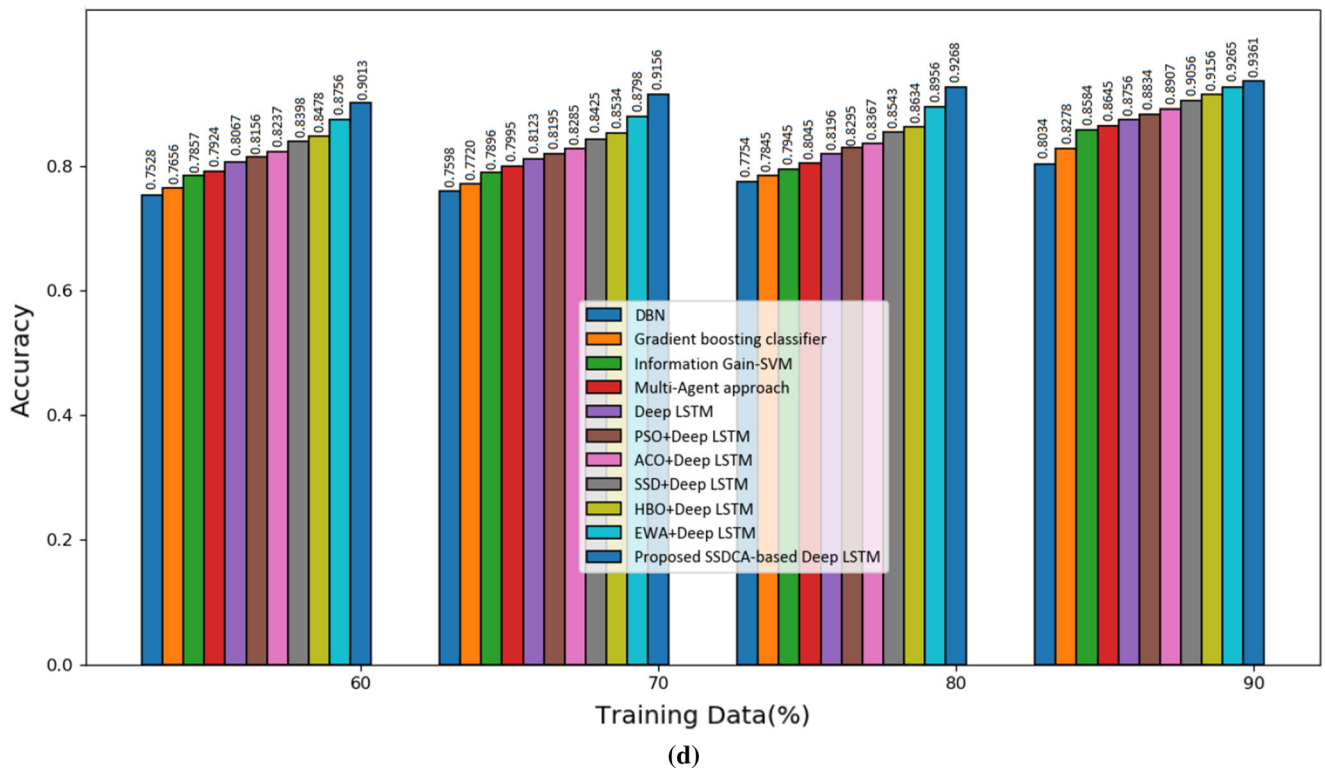
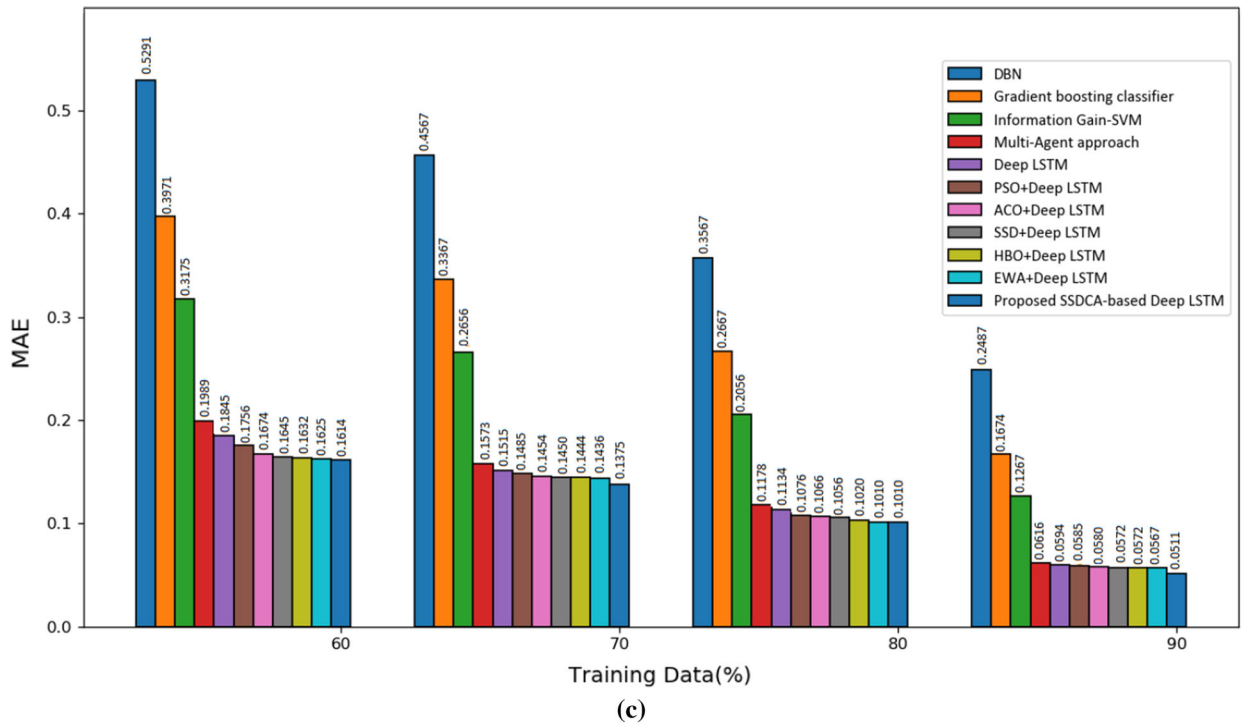


Fig. 9 continued

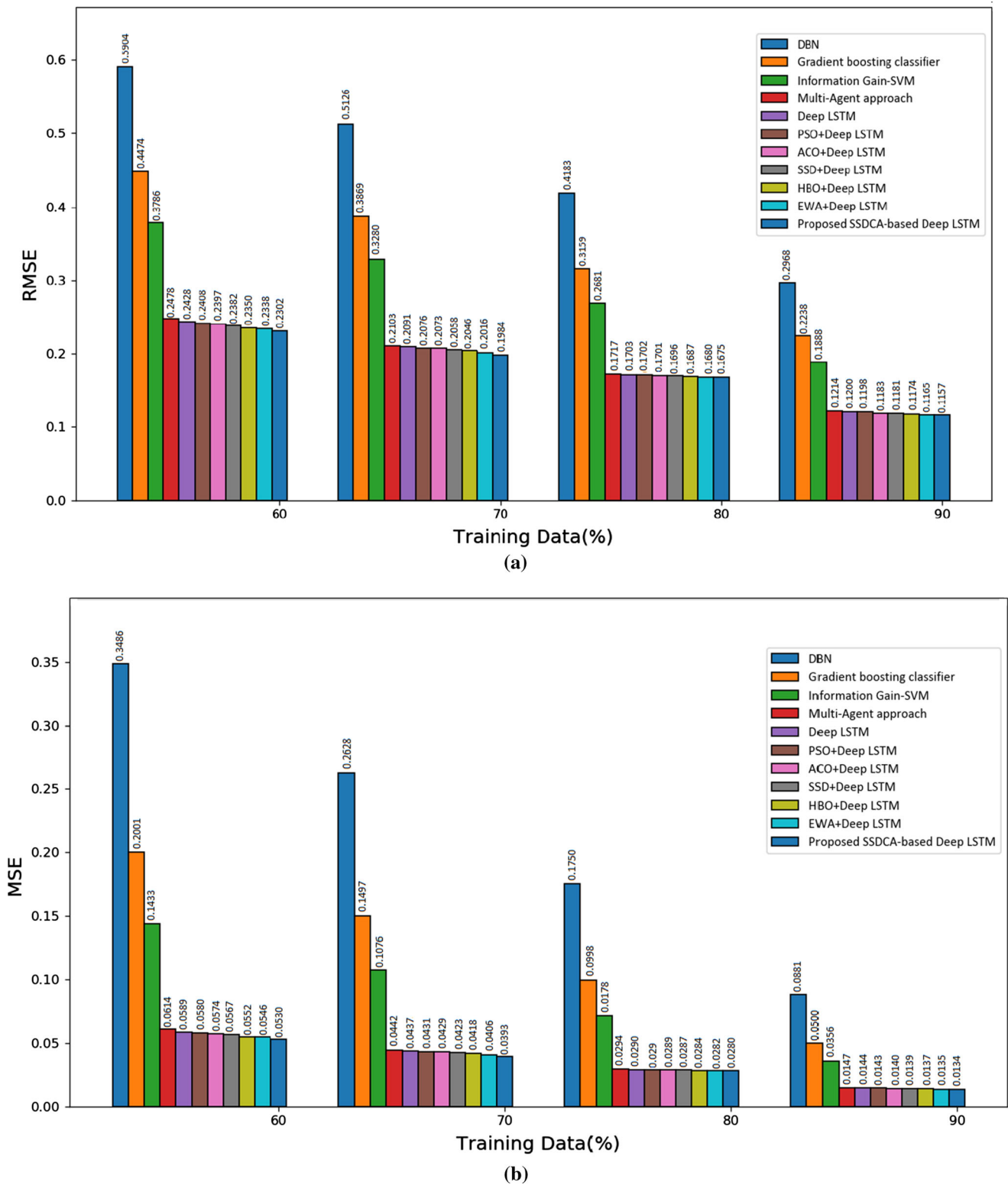


Fig. 10 Analysis using dataset-2 with feature size 6, a RMSE, b MSE, c MAE, d Accuracy

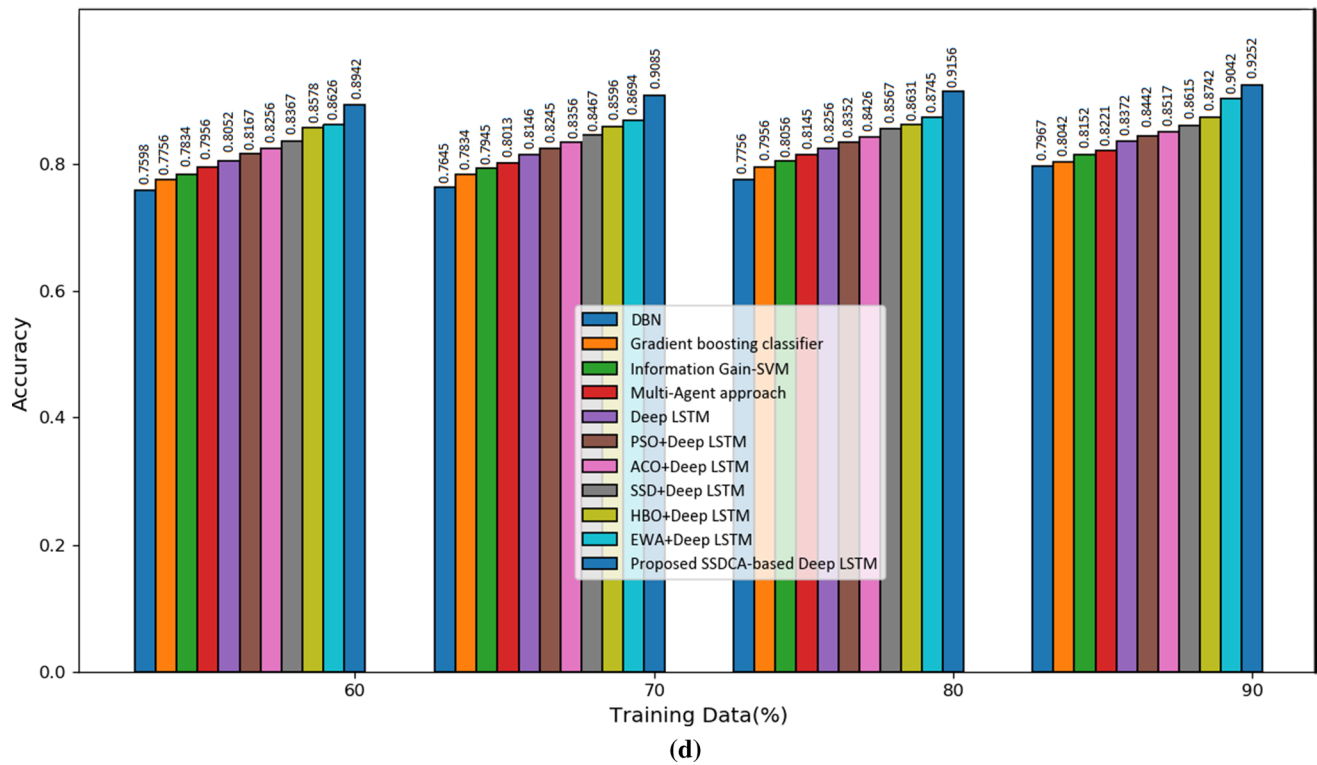
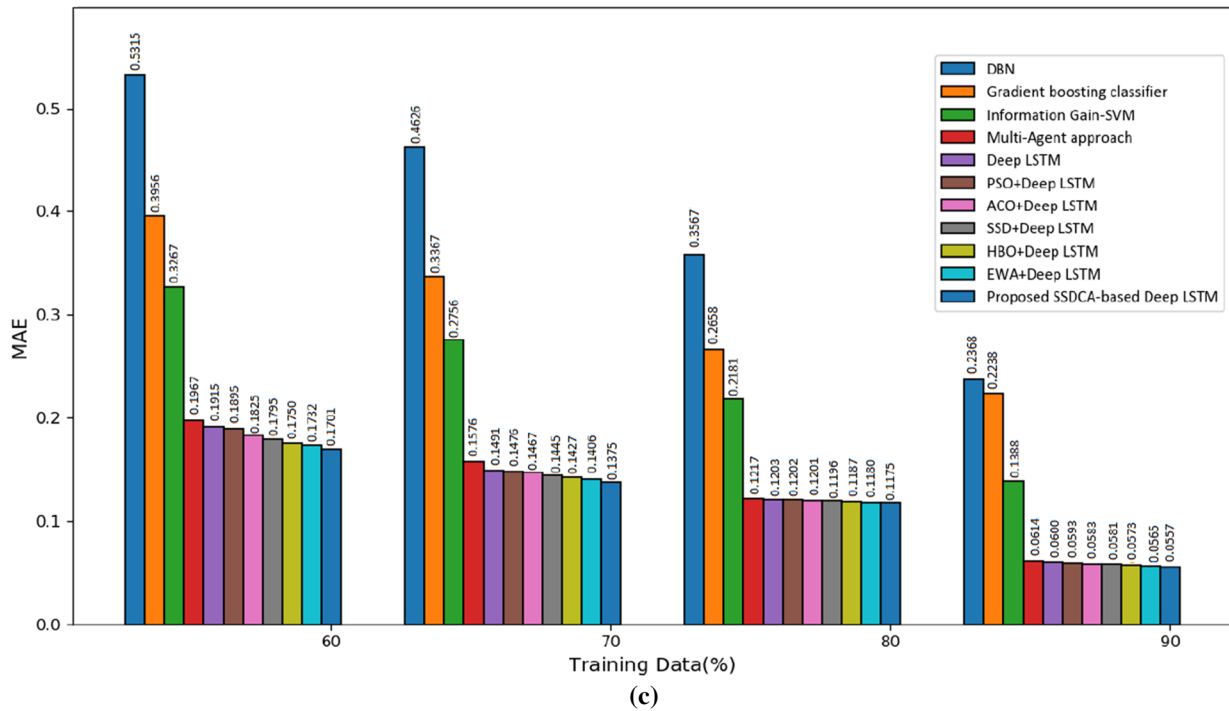


Fig. 10 continued

**Funding** The Petroleum Trust Development Fund (PTDF) Nigeria partially funded this work through grant number PTDF/ED/OSS/PHD/DBB/1558/19 for the first author’s PhD studies.

**Data availability and material** All data used in this research was obtained from the United States Bureau of Transportation Statistics website.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Yu B, Guo Z, Asian S, Wang H, Chen G (2019) Flight delay prediction for commercial air transport: a deep learning approach. *Transp Res Part E Logist Transp Rev* 125:203–221. <https://doi.org/10.1016/j.tre.2019.03.013>
2. Sternberg A, Soares J, Carvalho D, Ogasawara E (2017) A review on flight delay prediction, pp. 1–21
3. Pyrgiotis N, Malone KM, Odoni A (2013) Modelling delay propagation within an airport network. *Transp Res Part C Emerg Technol* 27:60–75. <https://doi.org/10.1016/j.trc.2011.05.017>
4. Guleria Y, Cai Q, Alam S, Li L (2019) A multi-agent approach for reactionary delay prediction of flights. *IEEE Access* 7:181565–181579. <https://doi.org/10.1109/ACCESS.2019.2957874>
5. Alharbi B, Prince M (2020) A hybrid artificial intelligence approach to predict flight delay. *Int J Eng Res Technol* 13(4):814–822
6. Chen Z, Wanke P, Antunes JJM, Zhang N (2017) Chinese airline efficiency under CO2 emissions and flight delays: a stochastic network DEA model. *Energy Econ* 68:89–108. <https://doi.org/10.1016/j.eneco.2017.09.015>
7. Şafak Ö, Atamtürk A, Selim Aktürk M (2019) Accommodating new flights into an existing airline flight schedule. *Transp Res Part C Emerg Technol* 104:265–286. <https://doi.org/10.1016/j.trc.2019.05.010>
8. Jiang C, Zheng S (2020) Airline baggage fees and airport congestion. *Transp Res Part C Emerg Technol* 117:102686. <https://doi.org/10.1016/j.trc.2020.102686>
9. Guo Z, Yu B, Hao M, Wang W, Jiang Y, Zong F (2021) A novel hybrid method for flight departure delay prediction using random forest regression and maximal information coefficient. *Aerosp Sci Technol* 116:106822
10. Dalmau R, Ballerini F, Naessens H, Belkoura S, Wangnick S (2021) An explainable machine learning approach to improve take-off time predictions. *J Air Transp Manag* 95:1020
11. Shao W, Prabowo A, Zhao S, Koniusz P, Salim FD (2021) Predicting flight delay with spatio-temporal trajectory convolutional network and airport situational awareness map. *arXiv preprint* <https://arxiv.org/abs/2105.08969>
12. Dixit A, Jakhar SK (2021) Airport capacity management: a review and bibliometric analysis. *J Air Transp Manag* 91(2020):102010. <https://doi.org/10.1016/j.jairtraman.2020.102010>
13. Chiti F, Fantacci R, Rizzo A (2018) An integrated software platform for airport queues prediction with application to resources management. *J Air Transp Manag* 67:11–18. <https://doi.org/10.1016/j.jairtraman.2017.11.003>
14. Chen J, Li M (2019) Chained predictions of flight delay using machine learning. In: *AIAA SciTech 2019 Forum*, pp. 1–25, doi:<https://doi.org/10.2514/6.2019-1661>.
15. Ball M et al. (2010) Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States. In: *Nextor*, pp. 5–14, 25–30
16. Chen J, Sun D (2018) Stochastic ground-delay-program planning in a metroplex. *J Guid Control Dyn* 41(1):231–239. <https://doi.org/10.2514/1.G002964>
17. Bisandu DB, Homaid MS, Moulitsas I, Filippone S (2021) A deep feedforward neural network and shallow architectures effectiveness comparison: flight delays classification perspective. In: The 5th International conference on advances in artificial intelligence (ICAAI 2021) in QAHE at Northumbria University London Campus, UK, 2021, (Accepted)
18. Chen H, Tu S, Xu H (2021) The application of improved grasshopper optimization algorithm to flight delay prediction-based on spark. *Conference on complex, intelligent, and software intensive systems*. Springer, Cham, pp 80–89
19. Giusti L, Carvalho L, Gomes AT, Coutinho R, Soares J, Ogasawara E (2021) Analyzing flight delay prediction under concept drift. *arXiv preprint* <https://arxiv.org/abs/2104.01720>, 2021.
20. Chen W, Li Z, Liu C, Ai Y (2021) A deep learning model with conv-LSTM networks for subway passenger congestion delay prediction. *J Adv Transp*. <https://doi.org/10.1155/2021/6645214>
21. Khan WA, Ma HL, Chung SH, Wen X (2021) Hierarchical integrated machine learning model for predicting flight departure delays and duration in series. *Transp Res Part C: Emerg Technol* 129:103225
22. Alla H, Moumoun L, Balouki Y (2021) A multilayer perceptron neural network with selective-data training for flight arrival delay prediction. *Sci Program*. <https://doi.org/10.1155/2021/5558918>
23. Ferguson J, Kara AQ, Hoffman K, Sherry L (2013) Estimating domestic US airline cost of delay based on European model. *Transp Res Part C Emerg Technol* 33:311–323. <https://doi.org/10.1016/j.trc.2011.10.003>
24. Thiagarajan B, Srinivasan L, Sharma AV, Sreekanthan D, Vijayaraghavan V (2017) A machine learning approach for prediction of on-time performance of flights. In: *IEEE/AIAA 36th digital avionics systems conference*, <https://doi.org/10.1109/DASC.2017.8102138>
25. Buttazzo GC (2011) *Hard real-time computing systems*, vol 24. Springer, Boston
26. Yao S (2019) *Flight arrival delay prediction using gradient boosting classifier*, vol 813. Springer, Singapore
27. Piccialli F, Bessis N, Jung JJ (2020) Guest editorial: data science challenges in industry 4.0. *IEEE Trans Ind Inform* 16(9):5924–5928. <https://doi.org/10.1109/TII.2020.2984061>
28. Makhloof MAA, Waheed ME, Badawi UAER (2014) Real-time aircraft turnaround operations manager. *Prod Plan Control* 25(1):2–25. <https://doi.org/10.1080/09537287.2012.655800>
29. Zhu X, Li L (2020) Improved flight time predictions for fuel loading decisions of scheduled flights with a deep learning approach. no. 11215119, pp. 1–12, 2020.
30. Ball MO, Chen CY, Hoffman R (2001) Collaborative decision making in air traffic management: current and future research directions. In: *New concepts and methods in air traffic management*
31. Vonitsanos G, Panagiotakopoulos T, Kanavos A, Tsakalidis A (2021) Forecasting air flight delays and enabling smart airport services in apache spark. *IFIP International conference on artificial intelligence applications and innovations*. Springer, Cham, pp 407–417



32. Zhu X, Li L (2021) Flight time prediction for fuel loading decisions with a deep learning approach. *Transp Res Part C Emerg Technol* 128:1031
33. Agogino AK, Tumer K (2012) A multi-agent approach to managing air traffic flow. *Auton Agent Multi Agent Syst* 24(1):1–25. <https://doi.org/10.1007/s10458-010-9142-5>
34. Tu Y, Ball MO, Jank WS (2008) Estimating flight departure delay distributions—a statistical approach with long-term trend and short-term pattern. *J Am Stat Assoc* 103(481):112–125. <https://doi.org/10.1198/016214507000000257>
35. Mueller E, Chatterji G (2002) Analysis of aircraft arrival and departure delay characteristics. In: *AIAA's aircraft technology, integration, and operations*, pp. 1–14. <https://doi.org/10.2514/6.2002-5866>
36. Klein A, Craun C, Lee RS (2010) Airport delay prediction using weather-impacted traffic index (WITI) model. In: *AIAA/IEEE Digital avionics systems conference*, pp. 1–13. <https://doi.org/10.1109/DASC.2010.5655493>
37. Somani S, Priyanshu P, Meghna S, Safa M (2021) An approach of applying machine learning model in flight delay prediction—a comparative analysis. *Revista Geintic-Gestao Inovacao E Techlogias* 11(3):1233–1244
38. Aghdam MY, Tabbakh SRK, Chabok SJM, Maryam K (2021) Optimisation of air traffic management efficiency based on deep learning enriched by the long short-term memory (LSTM) and extreme learning machine (ELM). *J Big Data* 8(1):1–26
39. Zoutendijk M, Mihaela M (2021) Probabilistic flight delay predictions using machine learning and applications to the flight-to-gate assignment problem. *Aerospace* 8(6):152
40. Holden AJ et al (2006) Reducing the dimensionality of data with neural networks. *Science* 313:504–507
41. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444. <https://doi.org/10.1038/nature14539>
42. Ai Y, Pan W, Yang C, Wu D, Tang J (2019) A deep learning approach to predict the spatial and temporal distribution of flight delay in network. *J Intell Fuzzy Syst* 37(5):6029–6037. <https://doi.org/10.3233/JIFS-179185>
43. Ahmed I, Ahmad M, Jeon G, Piccialli F (2021) A framework for pandemic prediction using big data analytics. *Big Data Res* 25:100190. <https://doi.org/10.1016/j.bdr.2021.100190>
44. Hesam M, Sultana N, Millward H, Liu L (2021) Ensemble learning activity scheduler for activity based travel demand models. *Transp Res Part C* 123:102972. <https://doi.org/10.1016/j.trc.2021.102972>
45. Farrokhi A, Farahbakhsh R, Rezazadeh J, Minerva R (2021) Application of Internet of Things and artificial intelligence for smart fitness: a survey. *Comput Netw* 189:107859. <https://doi.org/10.1016/j.comnet.2021.107859>
46. Krizhevsky BA, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Commun ACM* 60(6):84–90
47. Truong D (2021) Using causal machine learning for predicting the risk of flight delays in air transportation. *J Air Transp Manag* 91:101993. <https://doi.org/10.1016/j.jairtraman.2020.101993>
48. Sharma N, Sharma R, Jindal N (2021) Machine learning and deep learning applications—a vision. *Glob Trans Proc* 2:24–28. <https://doi.org/10.1016/j.gltp.2021.01.004>
49. Bansal JC, Sharma H, Jadon SS, Clerc M (2014) Spider Monkey Optimization algorithm for numerical optimisation. *Memetic Comput* 6(1):31–47. <https://doi.org/10.1007/s12293-013-0128-0>
50. Chander S, Padmanabha V, Mani J (2020) Jaya Spider Monkey optimization-driven deep convolutional LSTM for the prediction of COVID-19. *Bio-Algor Med-Syst*. <https://doi.org/10.1515/bams-2020-0030>
51. Kim S, Hong S, Joh M, Song SK (2017) DeepRain: ConvLSTM network for precipitation prediction using multichannel radar data. *arXiv*, <https://arxiv.org/abs/1711.02316> pp. 3–6
52. Guvercin M, Ferhatosmanoglu N, Gedik B (2020) Forecasting flight delays using clustered models based on airport networks. *IEEE Trans Intell Transp Syst* 22:1–11. <https://doi.org/10.1109/tits.2020.2990960>
53. Lambelho M, Mitici M, Pickup S, Marsden A (2020) Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions. *J Air Transp Manag* 82:101737. <https://doi.org/10.1016/j.jairtraman.2019.101737>
54. Pathomsiri S, Haghani A, Dresner M, Windle RJ (2008) Impact of undesirable outputs on the productivity of US airports. *Transp Res Part E Logist Transp Rev* 44(2):235–259. <https://doi.org/10.1016/j.tre.2007.07.002>
55. Ankan M, Deshpande V, Sohoni MG (2012) Building reliable air-travel infrastructure using empirical data and stochastic models of airline networks. In: *52nd AGIFORS Annu Proc 2012—Symp Study Gr Meet*, vol. 2, pp. 1107–1142
56. Abraham A, Dutta P, Mandal JK, Bhattacharya A, Dutta S (2018) *Emerging technologies in data mining and information security*, vol 2. Springer, Singapore
57. Rebollo JJ, Balakrishnan H (2014) Characterisation and prediction of air traffic delays. *Transp Res Part C Emerg Technol* 44:231–241. <https://doi.org/10.1016/j.trc.2014.04.007>
58. Choi S, Kim YJ, Briceno S, Mavris D (2016) Prediction of weather-induced airline delays based on machine learning algorithms. In: *2016 AIAA/IEEE Digital avionics systems conference—proceedings*. <https://doi.org/10.1109/DASC.2016.7777956>
59. Pérez-Rodríguez JV, Pérez-Sánchez JM, Gómez-Déniz E (2017) Modelling the asymmetric probabilistic delay of aircraft arrival. *J Air Transp Manag* 62:90–98. <https://doi.org/10.1016/j.jairtraman.2017.03.001>
60. The MathWorks I (2018) *Symbolic math toolbox*. Natick, Massachusetts, United States. Retrieved from <https://www.mathworks.com/help/symbolic/>
61. Kim YJ, Choi S, Briceno S, Mavris D (2016) A deep learning approach to flight delay prediction. In: *AIAA/IEEE digital avionics systems conference—proceedings*, pp. 1–16. <https://doi.org/10.1109/DASC.2016.7778092>
62. Yazdi MF, Kamel SR, Chabok SJM, Kheirabadi M (2020) Flight delay prediction based on deep learning and Levenberg–Marquart algorithm. *J Big Data*. <https://doi.org/10.1186/s40537-020-00380-z>
63. Esmaeilzadeh E, Mokhtarimousavi S (2020) Machine learning approach for flight departure delay prediction and analysis. *Transp Res Rec* 2674(8):145–159. <https://doi.org/10.1177/0361198120930014>
64. Pamplona DA, Weigang L, De Barros AG, Shiguemori EH, Alves CJP (2018) Supervised neural network with multilevel input layers for predicting of air traffic delays. In: *Proceedings of international joint conference on neural networks*, <https://doi.org/10.1109/IJCNN.2018.8489511>
65. Gomes HM et al (2017) Adaptive random forests for evolving data stream classification. *Mach Learn* 106(9–10):1469–1495. <https://doi.org/10.1007/s10994-017-5642-8>
66. Ma X, Tao Z, Wang Y, Yu H, Wang Y (2015) Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transp Res Part C Emerg Technol* 54:187–197
67. Liu Y, Jia G, Tao X, Xu X, Dou W (2014) A stop planning method over big traffic data for airport shuttle bus. In: *2014 IEEE fourth international conference on big data and cloud computing*, pp. 63–70. <https://doi.org/10.1109/BDCLOUD.2014.21>

68. Xu N, Donohue G, Laskey KB, Chen CH. Estimation of delay propagation in the national aviation system using Bayesian networks. In: 6th USA/Europe Air traffic management research and development seminar ATM, pp. 353–363
69. Tsiotas G (2009) On the use of non-linear transformations in stochastic volatility models. *Stat Methods Appl* 18(4):555–583. <https://doi.org/10.1007/s10260-008-0113-9>
70. Inoue M, Inoue S, Nishida T (2018) Deep recurrent neural network for mobile human activity recognition with high throughput. *Artif Life Robot* 23(2):173–185. <https://doi.org/10.1007/s10015-017-0422-x>
71. January Flight Delay Prediction dataset taken from, <https://www.kaggle.com/divyansh22/flight-delay-prediction>, Accessed on July 2020
72. February Flight Delay Prediction dataset taken from, '[https://www.kaggle.com/divyansh22/february-flight-delay-prediction#Feb\\_2020\\_ontime.csv](https://www.kaggle.com/divyansh22/february-flight-delay-prediction#Feb_2020_ontime.csv)', accessed on July 2020
73. Engle RF, Manganelli S (2004) CAViaR: conditional autoregressive value at risk by regression quantiles. *J Bus Econ Stat* 22(4):367–381. <https://doi.org/10.1198/073500104000000370>
74. Thrun MC, Ultsch A (2021) Swarm intelligence for self-organised clustering. *Artif Intell* 290:103237. <https://doi.org/10.1016/j.artint.2020.103237>
75. Zhou J et al (2021) Optimisation of support vector machine through the use of metaheuristic algorithms in forecasting TBM advance rate. *Eng Appl Artif Intell* 97:104015. <https://doi.org/10.1016/j.engappai.2020.104015>
76. Lim TS, Loh WY, Shih YS (2000) Comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach Learn* 40(3):203–228. <https://doi.org/10.1023/A:1007608224229>
77. Al-Tabbakh SM, Mohamed HM, El ZH (2018) Machine learning techniques for analysis of egyptian flight delay. *Int J Data Min Knowl Manag Process* 8(3):01–14. <https://doi.org/10.5121/ijdkp.2018.8301>
78. Qu J, Zhao T, Ye M, Li J, Liu C (2020) Flight delay prediction using deep convolutional neural network based on fusion of meteorological data. *Neural Process Lett* 52(2):1461–1484. <https://doi.org/10.1007/s11063-020-10318-4>
79. Ball MO, Hoffman R, Mukherjee A (2010) Ground delay program planning under uncertainty based on the ration-by-distance principle. *Transp Sci* 44(1):1–14. <https://doi.org/10.1287/trsc.1090.0289>
80. Tharwat A, Gabel T (2020) Parameters optimisation of support vector machines for imbalanced data using social ski driver algorithm. *Neural Comput Appl* 32(11):6925–6938. <https://doi.org/10.1007/s00521-019-04159-z>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.