



OPEN

An intelligent monitoring system for forecasting and anomaly detection in precision beekeeping

Jean-Charles Huet[✉], Lamine Bougueroua & Sid Ahmed Metidji

Bees play a vital ecological role as pollinators, contributing to biodiversity, forest regeneration, and agricultural productivity. In recent years, precision beekeeping has emerged as a promising approach to support hive management through sensor-based monitoring. However, existing systems often lack predictive capabilities, limiting their usefulness in anticipating disruptive events that threaten colony health. To address this gap, we present BeeViz, an intelligent monitoring system that combines time series forecasting and anomaly detection to enhance decision-making in apiculture. The system integrates sensor networks, cloud infrastructure, and AI-based data processing modules to continuously track key hive parameters (temperature, humidity, and weight) and generate short-term forecasts and real-time alerts. Preliminary results show that the system can effectively detect anomalies and generate short-term forecasts for key hive parameters, with promising accuracy across different metrics. By enabling proactive interventions, BeeViz supports more resilient and sustainable beekeeping practices, paving the way for collaborative learning and data-driven hive management.

Keywords Agroecology, Precision beekeeping, Hive management, Intelligent forecasting, Anomaly detection

Bees play a fundamental role in maintaining ecosystems through pollination, contributing to biodiversity, forest regeneration, and agricultural sustainability. Approximately 75% of the world's crops that produce fruits and seeds for human consumption depend, at least partially, on pollinators for sustained production and quality. Beekeeping, or apiculture, encompasses the management of bee colonies to optimize pollination and honey production¹.

During the past two decades, the beekeeping industry has experienced a significant decline in honey yields per colony^{2,3}, severely affecting both productivity and sustainability. For example, in France, honey production in 2024 is estimated at 20,000 tonnes, a sharp drop from 33,900 tonnes in 2023, according to ADA France (Federation of the Regional Beekeeping Development Associations)⁴. According to ADA France, honey production in 2024 is projected to decrease by 27% compared to the average output recorded between 2021 and 2023. This downturn raises pressing concerns about colony health, hive productivity, and the long-term viability of apiculture. It also underscores the growing need for data-centric tools to help beekeepers effectively monitor and manage their operations.

Recent advances in information technology, such as the Internet of Things (IoT) and artificial intelligence (AI), have opened up new possibilities to improve decision making in agriculture, including beekeeping applications^{5,6}. The Food and Agriculture Organization (FAO) has highlighted the strategic role of big data and machine learning in enabling predictive diagnostics and optimizing hive management practices¹.

At the European level, Verbeke et al.⁷ examined the adoption of Digital Beehive Monitoring Technology (DBMT) through a survey of 844 beekeepers in 18 European countries. Their study provides insight into beekeeper characteristics, current usage patterns, perceived benefits, and key factors influencing technology adoption. The findings reveal that 79.1% of the beekeepers had not yet implemented any form of digital monitoring, while 20.9% used it to a limited extent, mainly to track hive weight. Beekeepers primarily expect DBMT to help with hive management and decision making, followed by improving colony health, reducing winter losses, and saving time, all contributing to more efficient and effective beekeeping practices. However, only about a quarter of respondents foresee a reduction in operational costs, suggesting that they take into account the additional investment required for adopting this technology.

In France, the French public bee institute (ITSAP) surveyed over 400 beekeepers to assess their interest in various functionalities that could be integrated into a digital beekeeping platform⁸. Figure 1 presents the results of this survey.

EFREI Research Lab, Université Paris-Panthéon-Assas, 30-32 Avenue de la République, 94800 Villejuif, France.
✉email: jean-charles.huet@efrei.fr

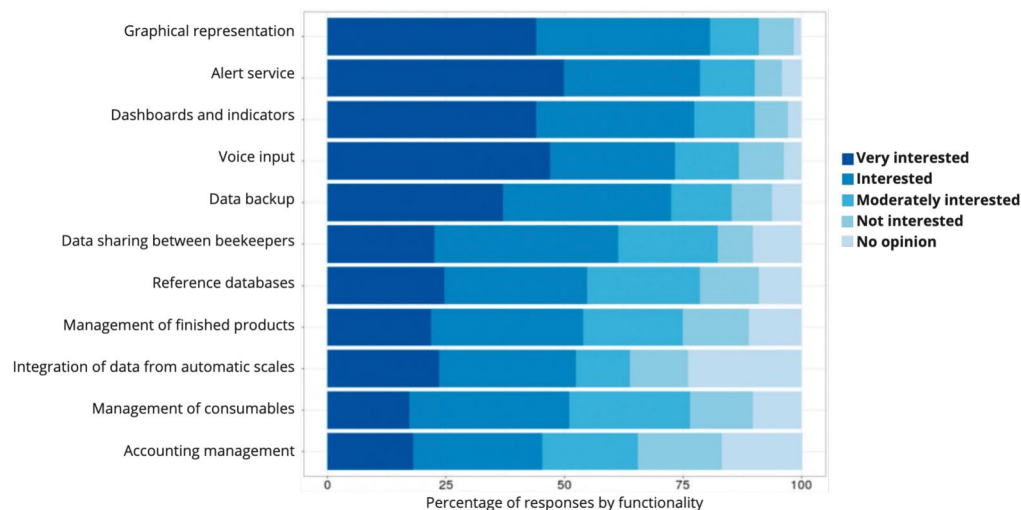


Fig. 1. Beekeepers' expectations regarding the functionalities to be integrated into a farm management tool (extracted and translated from⁸).

The results suggest that beekeepers prioritize features involving straightforward data representation, such as visual dashboards and summary tables. This underscores the central role of usability in the adoption and effectiveness of digital tools within apicultural contexts. Furthermore, there was notable enthusiasm for functionalities enabling real-time alerts and data sharing, reflecting a clear demand for collaborative platforms that support anticipatory and collective hive management. Beekeepers appear willing to modernize their practices and embrace digital tools, including data entry on dedicated platforms and the use of IoT devices⁹. The survey reveals that many already record their data on a computer or other IT devices, and that half of them express little reluctance toward adopting a suitable digital application.

From their perspective, such an application should prioritize user-friendliness and provide straightforward functionalities, allowing easy access to historical data accumulated during hive management. In addition, the application should incorporate alert features, such as weather updates and health event notifications, to facilitate timely information sharing. Ultimately, a participatory approach in both the platform's development and the definition of its terms of use appears to be the most effective way to create a reliable and well-adapted digital tool for beekeepers.

These expectations highlight the need for a comprehensive and user-friendly digital solution that supports both real-time monitoring and collaborative hive management. To meet these needs, this article introduces BeeViz, a monitoring platform that integrates traditional apicultural knowledge with modern data-driven decision support systems. Unlike existing solutions, which often provide fragmented or isolated data visualization tools, BeeViz integrates continuous data collection, predictive analytics, and anomaly detection to help beekeepers manage real-time hives and long-term strategic planning. The system not only allows beekeepers to track hive dynamics monitoring parameters such as internal temperature, humidity, and weight but also detects early warning signs of anomalies through machine learning-based time-series analysis. This approach enables the anticipation of colony stress, food shortages, or environmental hazards, helping beekeepers take proactive measures rather than reacting to adverse conditions.

In addition to monitoring individual hives, the system paves the way for collaborative learning approaches by taking advantage of a network of connected hives to constantly refine predictive models. Unlike traditional threshold-based alerts, which rely on fixed rules, our system can dynamically adapt by incorporating live data from multiple sources, improving accuracy over time. Furthermore, expanding the monitored parameters to include CO₂ levels and foraging activity could provide deeper insights into colony metabolism and environmental interactions.

This paper is structured as follows: “[Related work](#)” provides a review of existing monitoring and anomaly detection techniques, “[Our approach to prediction and anomaly detection](#)” details the proposed system architecture, “[Implementation, testing, and visualization](#)” presents implementation and experimental results, “[Discussion and future research directions](#)” addresses the limitations of our study and presents avenues for future research, finally, “[Conclusion](#)” presents conclusions.

Related work

Hive monitoring approaches

Hive monitoring in precision apiculture has evolved from manual inspection to data-driven techniques. Traditional methods rely on qualitative observations recorded by beekeepers, often on paper or hive covers¹⁰. These practices, while valuable, limit longitudinal tracking and advanced analysis.

Modern approaches fall into two main categories: *computer vision-based systems* and *sensor-based time series monitoring*. Figure 2 summarizes this classification.

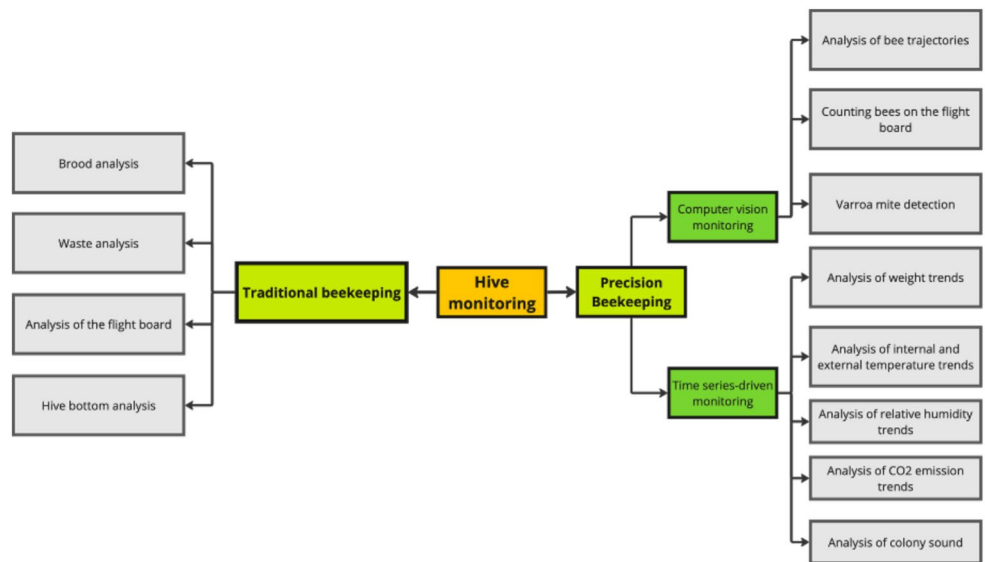


Fig. 2. Taxonomy of hive monitoring approaches.

The former includes visual and acoustic analysis using cameras and microphones. For example, Tashakkori et al.¹¹ tracked bee movements using threshold-based algorithms, while Bjerger et al.¹² used ultraviolet (UV) light and convolutional neural networks (CNNs) to detect varroa mites. Magnier et al.¹³ modeled bee flight trajectories to infer environmental stress, and Kriouile et al.⁶ applied Mask R-CNN (Region-based Convolutional Neural Network) for object detection inside hives.

Sensor-based systems, which underpin our work, use embedded devices to monitor internal hive parameters such as temperature, humidity, and weight. These systems enable real-time data acquisition and remote monitoring^{14,15}. Studies have shown that these metrics are closely linked to colony health and behavior. For instance, internal temperature regulation is essential for brood development¹⁶, while excessive humidity can degrade honey quality and promote mold^{17,18}. Hive weight variations can signal nectar flow, swarming, or seasonal transitions¹⁹.

Time series forecasting in beekeeping

Forecasting hive conditions using time series data has gained traction in recent years. Approaches vary by modeling technique and prediction target.

Robustillo et al.²⁰ compared four statistical models: vector autoregressive (VAR), time-varying VAR (tvVAR), dynamic linear model (DLM), and generalized additive model (GAM) to forecast temperature, humidity, and weight. The tvVAR model showed the best performance across hives. Erdal and Tolga²¹ used Autoregressive Integrated Moving Average (ARIMA) to predict national honey production in Turkey based on historical data.

Deep learning models offer greater flexibility in capturing nonlinear patterns. Anwar et al.²² developed a bidirectional long short-term memory (Bi-LSTM) with attention to estimate cumulative hive weight using internal and external sensor data. Meireles Filho et al.²³ compared long short-term memory (LSTM), gated recurrent unit (GRU), and CNN architectures to forecast annual honey production, finding LSTM to be the most accurate.

These studies demonstrate the potential of time series forecasting to support proactive hive management, though most focus on isolated metrics or lack integration with anomaly detection.

Anomaly detection in beekeeping

Anomaly detection is critical for identifying disruptive events such as swarming, predator intrusion, or hive damage. Approaches fall into two categories: *rule-based systems* and *AI-driven models*.

Rule-based systems define thresholds or patterns manually, often validated statistically^{24,25}. While effective in specific scenarios, they lack adaptability and scalability. Recent work by Chen et al.²⁶ introduced an edge-computing system using acoustic signals for multiclass anomaly classification, highlighting the growing interest in multimodal detection.

AI-based methods include both non-regressive and regressive models. Davidson et al.²⁷ evaluated deep recurrent AutoEncoders (AEs), isolation forests, and one-class Support Vector Machines (SVMs) for detecting swarming events using temperature data. Their results showed that AutoEncoders performed best in multivariate settings, while rule-based methods excelled in univariate contexts. In the context of precision apiculture, they identified three main categories of anomalies detectable through sensor data:

- **Swarming events:** this phenomenon is an important behavioral anomaly for beekeepers, as it can lead to complete exhaustion of a colony, described by a sudden drop in hive weight. It leads to a reduction in honey production and requires immediate action to gather new colonies.

- Sensor-related anomalies which may arise from hardware failure or calibration issues, leading to erroneous or misleading measurements.
- Opening the hive: where manual intervention by the beekeeper can temporarily distort sensor readings, especially those related to internal temperature and weight.

Senger et al.²⁸ used ARIMA and Seasonal ARIMA (SARIMA) models combined with Generalized Extreme Studentized Deviate (GESD) testing to detect anomalies based on residuals, achieving up to 75% detection accuracy. Further expanding on these classifications, they reported two additional types of anomalies observed in monitored hives:

- Predator intrusions, such as mice entering hives during cold seasons, disrupting the bees' natural clustering behavior used for thermoregulation.
- Physical damage to the beehive, often caused by environmental factors like strong winds or the invasion of ants or wasps. These events can compromise hive integrity and may be detectable via spikes in humidity or irregular weight data.

Discussion

While the literature offers valuable insights into hive monitoring, forecasting, and anomaly detection, most approaches treat these components separately. Few systems integrate predictive analytics and anomaly detection into a unified, user-oriented platform. Moreover, generalization across hives remains a challenge due to environmental variability and data heterogeneity.

Our work addresses these gaps by combining time series forecasting and anomaly detection in a modular system designed for real-time hive management. By integrating sensor data, intelligent processing, and user-friendly visualization, BeeViz aims to support both individual and collaborative decision-making in precision apiculture.

Our approach to prediction and anomaly detection

In this section, we first introduce the main functionalities of the system, followed by its conceptual architecture. We then present the intelligent time series processing techniques and the approaches employed for anomaly detection.

Main functionalities

Monitoring smart beehives presents a number of technical and operational challenges, particularly in ensuring timely and accurate detection of anomalies that may compromise colony health. To support this goal, the system incorporates a real-time anomaly detection mechanism that alerts beekeepers to unusual conditions as they arise. As illustrated in Fig. 3, the platform provides several key features:

- View real-time sensor data for three critical parameters: internal temperature, relative humidity, and hive weight.
- Selection and management of individual hives, including historical data and triggered alerts.
- Access to short-term predictive insights based on time-series forecasting for the selected hive.
- The option to pause data collection and anomaly detection temporarily, in order to avoid false positives during interventions such as hive inspections or honey extraction.
- Customization of alert sensitivity thresholds to adapt detection granularity to specific operational contexts.

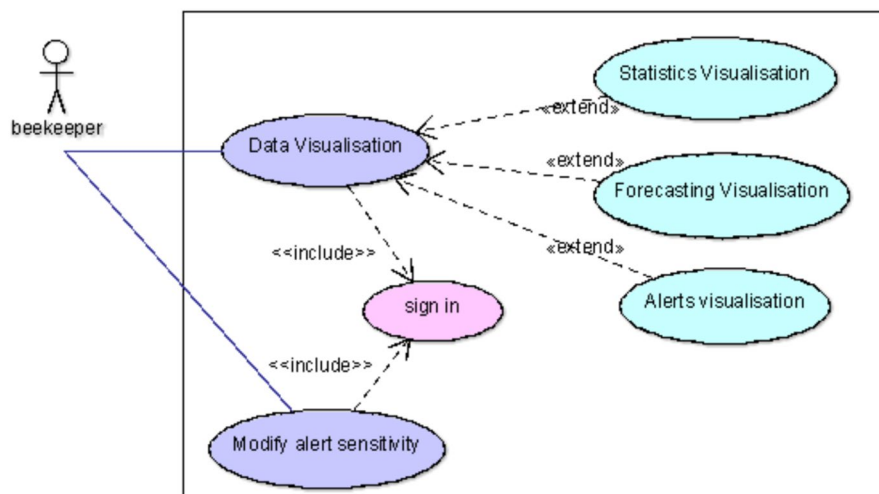


Fig. 3. use case diagram of BeeViz.

Proposed architecture

Effective hive management increasingly requires the deployment of connected devices and advanced data processing frameworks. In this work, we design a modular system architecture composed of six functional units, each responsible for a specific step in the data workflow, from sensor acquisition to predictive analysis. The system enables continuous data capture from smart hives, secure cloud-based storage, and real-time processing using time-series forecasting techniques. This architecture is designed to support informed decision-making by offering real-time monitoring capabilities and early detection of anomalies, thereby helping beekeepers respond proactively to environmental and colony-specific challenges.

The six component groups are (Fig. 4):

1. Connected hives: each user-owned hive is equipped with embedded sensors that continuously capture and transmit internal parameters—temperature, humidity, and weight—in real time.
2. Cloud database: real-time data collected from the hives is securely stored in a cloud-based infrastructure, ensuring both accessibility and scalability for long-term data management.
3. Intelligent Time Series Processing Systems: these modules analyze the incoming data streams to perform anomaly detection and generate short-term forecasts for the three monitored metrics.
4. Learning Database: this repository aggregates historical data, serving as a foundation for training, validating, and testing predictive models and anomaly detection algorithms.
5. Web-based Dashboard: the dashboard offers beekeepers an intuitive interface to visualize live hive data, forecast trends, and real-time alerts generated by the system.
6. Communication APIs: acting as communication gateways, the APIs facilitate data exchange between intelligent modules, the database management system, and the hive sensors, enabling real-time updates on the dashboard.

Designing intelligent time series processing

We implement a supervised learning approach that will take our two-column database as input:

- Timestamp : contains the time points at which observations were recorded; the distance between two timestamps of a metric is determined according to the resampling step determined previously.
- Y: contains the values (temperature, humidity, or weight) recorded at each corresponding timestamp.

We have tested the performance and predictive capacity of two types of architecture in our training database.

1. Recurrent neural networks (RNNs): RNNs are a class of artificial neural networks specifically designed to handle sequential data. Their architecture allows them to capture temporal dependencies by maintaining an internal memory of past input, which is essential for modeling time series where the order of observations impacts future values. This ability to leverage historical context makes RNNs particularly suitable for forecasting applications involving dynamic and evolving data patterns.
2. Facebook Prophet: Facebook Prophet is an open-source forecasting tool developed by Facebook's data science team. It is specially designed for time series forecasting and is particularly useful for commercial and operational forecasting tasks. The Prophet procedure is basically an additive regression model.

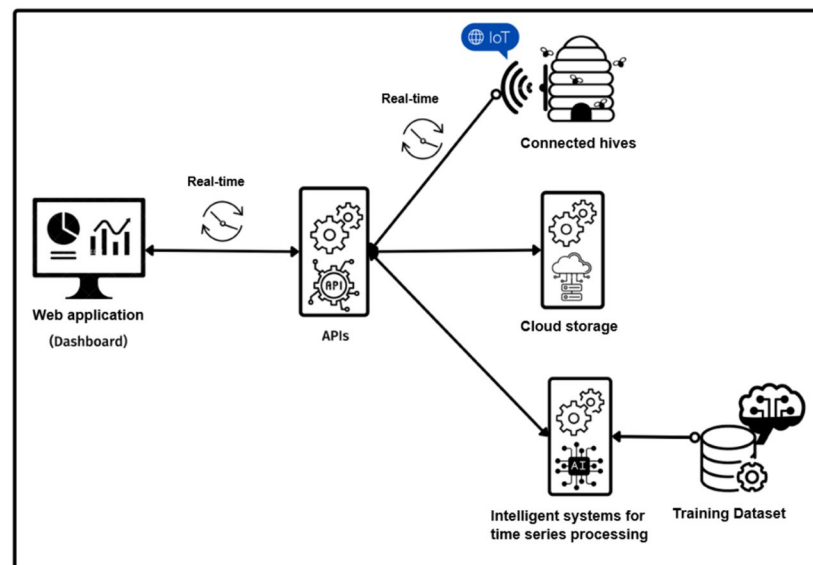


Fig. 4. Conceptual architecture.

Step	Technique used
Missing data imputation	Piecewise cubic hermite interpolating polynomial (PCHIP), selected after comparison with linear and cubic interpolation
Resampling	Hourly for temperature; daily for humidity and weight, based on metric variability and forecast horizon
Normalization	Min–max scaling applied per metric to standardize input ranges
Feature engineering	Rolling statistics (mean, std), lag features for time-series forecasting

Table 1. Summary of preprocessing steps applied to the HOBOS dataset.

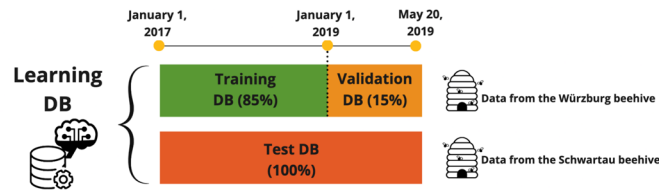


Fig. 5. Learning data distribution for the prediction.

Approaches used to design anomaly detection

Since our training database does not contain information on existing anomalies, we cannot use a classical supervised learning approach to train our models, and we cannot measure the precision of the models developed or select the best model. For this reason, we decided to let the user choose the anomaly model for each metric based on sensitivity. The sensitivity of the models developed is measured in terms of the number of anomalies it was able to detect in the training database. We have chosen three popular approaches to time series anomaly detection:

- **Predictive confidence level approach:** This approach relies on the best-performing time series forecasting model previously trained using the procedure detailed in Algorithm 1. A confidence interval is constructed based on the model's prediction and its Mean Absolute Percentage Error (MAPE), which has already been computed. This interval allows for the dynamic identification of anomalous values by flagging any observation that deviates significantly from the expected range.
- **Statistical profile approach:** This approach has been used effectively in the economic and financial sectors. We will then try to project it onto our learning database.
- **Unsupervised clustering approach:** One of the difficulties common to clustering algorithms for anomaly detection is defining the number of clusters, which is required by most clustering algorithms as a hyperparameter.

Although there are many techniques for estimating the number of clusters in tabular data, it is not possible to dynamically estimate the number of clusters for time series. In fact, we opt for two clustering algorithms that showed significant efficiency in our training database without the need to specify the number of clusters: (1) Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and (2) isolation forest.

Implementation, testing, and visualization

Dataset

Several research projects have used open source data to develop their own version of the monitoring system. Metidji et al.²⁹ compared several datasets, indicating the advantages and disadvantages of each. The dataset proposed by the authors, based on HOBOS, contains data without outliers or missing data. This dataset provides uninterrupted data for several years, covering the three most important parameters (temperature, humidity, and weight). We used this dataset in our study.

To ensure reproducibility and model robustness, we applied a structured data preprocessing pipeline. The main steps are summarized in Table 1. A detailed description of each step is available in our companion paper²⁹. Data normalization was applied at the global level across the entire hive dataset, rather than separately for each hive.

The dataset is made up of two hives located in Germany and contains two years of data. Figures 5 and 6 show the distribution between training, validation, and test data.

For both prediction and anomaly detection tasks, we systematically applied a *leave-one-hive-out* strategy, where one hive was reserved exclusively for testing the trained models. This approach guarantees that:

1. **Prediction:** the model does not overfit to the data of the training hive and can generate forecasts with comparable accuracy on previously unseen hives. To ensure a rigorous evaluation of the model, we verified several key aspects of the experimental setup. The training set (730 days) and validation set (130 days) were strictly separated in time, with no overlap, ensuring temporal independence between the learning and validation phases. The test set, although derived from a different hive, covers a similar time period as the training and

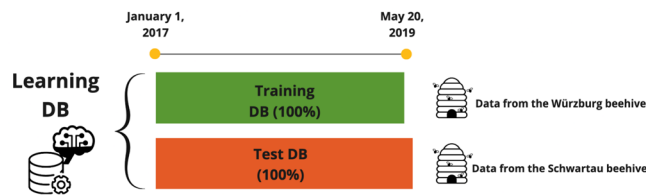


Fig. 6. Learning data distribution for the anomalies detection.

validation data (860 days). This configuration allows us to assess the model’s ability to generalize to a different hive under comparable environmental conditions.

2. Anomaly detection: the detection models maintain consistent sensitivity across the reference (training) and test hives. The training phase was conducted using data from a single hive, while the test phase relied on data from a second, distinct hive. This choice was motivated by the absence of labeled anomaly data for validation purposes as detailed in the “[Approaches used to design anomaly detection](#)”. By using a separate hive for testing, we ensured that the evaluation of the model’s ability to detect anomalies was performed on entirely unseen data, both in terms of location and hive-specific dynamics. As explained in the “[Implementation of the anomaly detection system](#)”, for instance, the confidence interval–based method for humidity anomaly detection achieves nearly identical sensitivity values for both hives (38% vs. 41%). This demonstrates that the proposed approach reliably produces a high-sensitivity anomaly detector applicable across different hives.

Implementation of the prediction system

Prediction model training algorithm

To develop an accurate and robust prediction system for hive monitoring, we implemented a model training algorithm that explores different time series forecasting techniques. The process, outlined in Algorithm 1, focuses on two primary forecasting models: Recurrent Neural Networks (RNNs) and Prophet. The system aims to predict crucial parameters of the hive, including internal temperature, humidity, and weight, using historical data.

The training process follows a structured pipeline:

- Technology and variable selection: the system assesses multiple forecasting methods—specifically RNN and Prophet—in key environmental variables, including internal temperature, humidity, and hive weight.
- Data preparation: for each method-variable pair, the input data are preprocessed to ensure quality, involving normalization procedures and the handling of missing or inconsistent values.
- Model initialization and evaluation: an initial predictive model is built for each configuration. If early evaluation metrics are satisfactory, the model is refined further; otherwise, the process shifts to alternative techniques.
- Optimization and validation: promising models undergo hyperparameter tuning and are validated against unseen data. Forecast accuracy is assessed by comparing actual versus predicted trends, often supported by visual inspection.
- Model selection and storage: if optimization yields improved results, the improved model is retained for future deployment; otherwise, the best available baseline is selected. Models that fail to meet performance thresholds are discarded in favor of other approaches.

The RNN models were implemented using the Huber loss function and a fixed random seed of 2 to ensure reproducibility. No early-stopping criterion was applied. All models used Batch Normalization with a batch size of 32. We allow the user to specify their desired prediction horizon directly from the dashboard (see “[The dashboard](#)”).

This approach ensures a systematic evaluation of different forecasting methodologies while adapting to the complexities of hive behavior. The final models selected through this process will be integrated into the monitoring system, enabling accurate short-term predictions that support beekeepers in proactive decision-making.

```

1: Technologies ← ['RNNs', 'Prophet']
2: Variables ← ['Temperature', 'Humidity', 'Weight']
3: for T in Technologies do
4:   for V in Variables do
5:     Perform the necessary data pre-processing of V for T
6:      $M_{0T} \leftarrow \text{build\_initial\_version}(T)$ 
7:     if  $M_{0T}$  gives promising results then
8:        $M_{\text{optim}T} \leftarrow \text{optimize\_model}(M_{0T}, T)$ 
9:       Visualize actual values, predicted values by  $M_{0T}$  and  $M_{\text{optim}T}$  in the
       validation period
10:       $\text{save\_model}(M_{\text{optim}T})$ 
11:     else if  $M_{0T}$  gives poor results then
12:       continue
13:     else
14:        $\text{save\_model}(M_{0T})$ 
15:     end if
16:   end for
17: end for

```

Algorithm 1. Prediction model training algorithm.

Optimizing initial models with grid search

After identifying the most important hyperparameters for each technology (see Table 2), we set the initial values of these hyperparameters to build M_{0T} , the initial model.

After evaluating the initial model, we adopted an optimization technique to select the optimal hyperparameters values defined with the aim of minimizing the value of the chosen error metric. To achieve this, we adopted the Grid Search technique.

It is a technique used in machine learning and model selection to find the optimal hyperparameters for a given model. Hyperparameters are parameters that are not learned during training but are defined prior to training and influence the learning process. The grid search consists of systematically testing different combinations of hyperparameter values to determine the set of values that gives the best model performance.

The term “Grid Search” refers to the process of creating a grid of hyperparameter values and exhaustively searching all possible combinations of these values to find the best combination. Each combination is evaluated using an error metric selected from the validation DB. The combination that produces the best performance is then selected as the optimal set of hyperparameters for the model.

Table 2 summarizes the hyperparameters chosen for each technology and the optimal values possible for each.

Table 3 presents the optimal parameters selected following a series of validation experiments, with final values chosen for each RNN model according to the specific characteristics of each parameter (weight, humidity and temperature).

Hyper-parameters	Description	Grid search
Layer count	Number of intermediate layers in the neural network	2, 3, 4
Number of hidden units (neurons)	Determines the ability of the RNN to learn complex patterns in the data	32, 64, 128
RNN neurons type	Gated recurrent units (GRUs) / Long short-term memory (LSTM)	LSTM, GRU
Optimization algorithm	Stochastic gradient descent (SGD) or adaptive moment estimation (ADAM)	SGD, ADAM
Learning rate	Controls step size during optimization	5×10^{-6} , 10^{-4} , 10^{-5}
Window size	Number of past time steps the model examines to predict future events	100, 200, 300
Number of epochs	Number of times the training dataset is run through the model during training	50, 100, 200
Prophet model : trend scale	Controls the flexibility of the trend component. Smaller values make the trend more rigid; larger values make it more flexible	0.01, 0.03, 0.05, 0.07
Prophet model : trend interval	Proportion of historical data where potential change points can be placed	0.6, 0.7, 0.8, 0.9
Prophet model : seasonality scale	Controls the flexibility of the seasonal component. Smaller values make seasonal models more rigid; larger values make them more flexible	1, 5, 10, 15
Prophet model : seasonality mode	Defines how seasonal components are incorporated with trend components (additively or multiplicatively)	Additive, multiplicative

Table 2. List of hyper-parameters to be manipulated in grid search.

Parameter	Layers number	Units number	Layers type	Learning rate	Optimizer	Window size	Epochs number
Weight	3	128	GRU	0.00005	SGD	200	100
Humidity	2	128	GRU	0.000005	ADAM	100	200
Temperature	2	32	LSTM	0.00005	SGD	200	100

Table 3. Final chosen values for each parameter-specific RNN model.

Metric	MAE (M_0)	To optimize?	MAE (M_{optim})	Optimization rate
Humidity (RNN)	4.558 %	Yes	4.017 %	13 %
Humidity (PROPHET)	6.290 %	Yes	5.705 %	10 %
Weight (RNN)	0.550 Kg	Yes	0.374 Kg	47 %
Weight (PROPHET)	14.59 Kg	Yes	5.248 Kg	178 %
Temperature (RNN)	0.461 °C	No	–	–

Table 4. RNN & PROPHET optimization.

Humidity		Temperature		Weight	
Validation DB	Test DB	Validation DB	Test DB	Validation DB	Test DB
4.017%	3.80%	0.461 °C	0.61 °C	0.37 Kg	0.94 Kg

Table 5. Summary of MAE error rates for the three forecasting models.

Model results during the validation period

Table 4 shows the evaluation of the initial models M_0 and the results after optimization of the humidity, weight, and temperature models. As mentioned in “[Designing intelligent time series processing](#)”, we use the RNN and Facebook prophet architectures. The database used here is the validation database of the Fig. 5.

We note that the initial temperature model M_{ORNN} gave us satisfactory results that make its optimization unnecessary, especially as this may lead to model overlearning in the training data. In addition, the Grid Search technique produced a considerably improved version of our model for humidity and weight, as shown in Table 4.

For the Prophet-type of architecture, Grid Search gave a very significant optimization in the models initially designed, but none of them was ideally able to follow the local trend variation in values.

We note a significant difference between the prediction quality of the two versions, where the RNNs predictions seem more sophisticated and realistic. We will therefore adopt M_{ORNN} for the temperature and $M_{optimRNN}$ for weight and humidity.

We now present the accuracy of the forecasting systems developed using the test dataset. Table 5 and Fig. 7 summarize and compare the MAE of the three final models for the three metrics. The results show that the errors observed on the test dataset (originating from a different hive than the training data) are comparable to those obtained during validation. This provides a form of cross-hive validation and suggests that the models may exhibit a degree of scalability. However, we acknowledge that broader cross-site validation, involving hives from different geographic locations and environmental conditions, is necessary to fully assess the generalizability and robustness of the proposed system.

Implementation of the anomaly detection system

This section explains the development of the four anomaly detection models using the algorithms introduced in “[Approaches used to design anomaly detection](#)”, and then presents their level of sensitivity as a function of their detection capability.

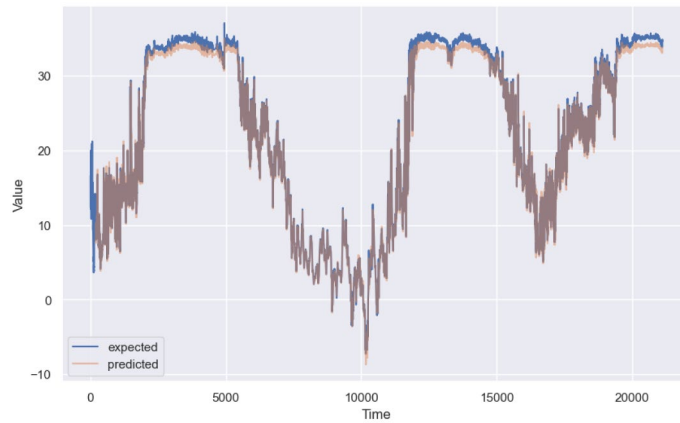
Regarding humidity and the training DB, we note that the predictive confidence level approach is the most sensitive algorithm (38.33%), while DBSCAN is the least sensitive (6.81%), and the others have moderate sensitivity (around 20–25%). For temperature, we note that the Isolation Forest approach is the most sensitive algorithm (20.06%), while DBSCAN is the least sensitive (0.28%), and the others have moderate sensitivity (around 8–16%). For weight, we note that the predictive confidence level approach is the most sensitive algorithm (34.16%), while DBSCAN is the least sensitive (1.83%), and the others have moderate sensitivity (around 21–25%).

Table 6 summarizes and compares the sensitivity rates of the four models across the two databases and shows that, in most cases, the results are relatively consistent. This suggests that the models may generalize reasonably well across similar hive contexts, although further validation is needed to confirm their robustness in more diverse settings.

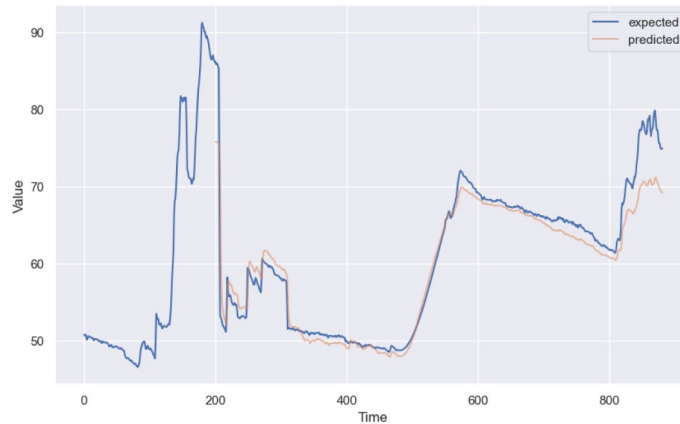
While most anomaly detection models demonstrated consistent sensitivity across training and test datasets, the Isolation Forest algorithm showed a significant disparity (21.83% vs. 66.51%), as highlighted in table 6. This inconsistency suggests a lack of generalization and potential overfitting to the training hive data. As a result, we



(a) Relative Humidity (%), Elapsed time in days



(b) Temperature (°C), Elapsed time in hours



(c) Weight (kg), Elapsed time in days

Fig. 7. Forecast results in the test DB.

excluded Isolation Forest from the final prototype. These findings emphasize the importance of hive-specific calibration when deploying anomaly detection models in real-world settings. In our current implementation, each model is trained on data from a single hive, which improves reliability but limits generalization. To address this, future work will explore more robust evaluation strategies, such as hive-specific cross-validation folds, to better assess model adaptability. We recommend to investigate transfer learning techniques (e.g., fine-tuning on subsets of hive data) to enhance model performance across heterogeneous apiaries. These directions aim to ensure that the system remains reliable and adaptable in diverse operational contexts.

Detection models	Humidity		Temperature		Weight	
	Train DB	Test DB	Train DB	Test DB	Train DB	Test DB
Confidence prediction	38.33%	41.35%	8.76%	8.36%	34.16%	35.38%
Statistical profile	25.86%	24.51%	16.22%	16.22%	21.95%	39.04%
DBSCAN	6.81%	6.32%	0.28%	0.18%	1.83%	5.79%
Isolation forest	21.83%	66.51%	20.06%	56.24%	25.55%	54.71%

Table 6. Sensitivity of anomaly detection models.



Fig. 8. Dashboard: visualization interface.



Fig. 9. Dashboard: prediction configuration interface.

The dashboard

Since connected beehives are not yet fed by bee colonies that can generate data close to reality, we have replaced connected beehives in our prototype with an API that will play the role of connected beehives by periodically and continuously broadcasting values for internal humidity, temperature, and weight that simulate real values.

To generate random numbers with the same distribution as the actual hive data, we used a function to generate Gaussian random numbers, i.e. with a normal distribution. This function will take as input the mean and standard deviation of the metric. The BeeViz interface (Figs. 8, 9 and 10) provides a visual dashboard for hive monitoring, displaying temperature, humidity, and weight data over a selected time range (e.g., May to September). Each graph includes historical values (solid blue lines), predictions (dashed blue lines), statistical profiles (yellow areas), and predictive confidence intervals (green areas). Orange warning icons highlight detected anomalies. Users can configure the view by selecting the hive, adjusting the prediction horizon and sensitivity, and toggling the display of predictions and anomalies (Figs. 9 and 10).

Since our predictive models are based on iterative forecasting rather than direct multi-step prediction, we allow the user to specify their desired prediction horizon directly from the dashboard. Based on this input, the system recursively generates forecasts day by day (predicting $N + 1$, then $N + 2$, etc.) until the user's chosen horizon.

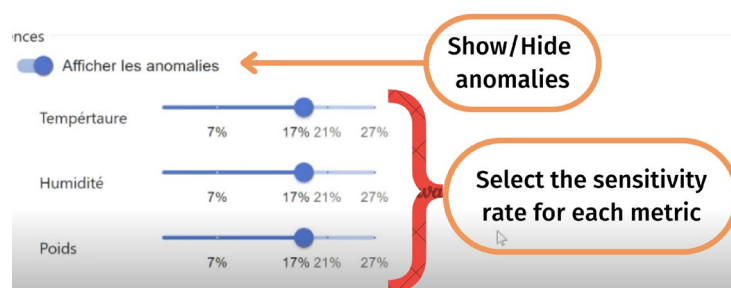


Fig. 10. Dashboard: anomaly configuration interface.

Software (full name)	Version	URL (accessed: 2026-01-15)
Python (programming language)	3.10.10	https://www.python.org/
TensorFlow (machine learning platform)	2.10.1	https://www.tensorflow.org/
scikit-learn (machine learning in Python)	1.2.2	https://scikit-learn.org/
Prophet (forecasting at scale)	1.1.2	https://facebook.github.io/prophet/
pandas (Python data analysis library)	1.4.4	https://pandas.pydata.org/
Matplotlib (Visualization with Python)	3.7.1	https://matplotlib.org/
Flask (Python web microframework)	2.2.2	https://flask.palletsprojects.com/
Node.js (JavaScript runtime)	18.14.2	https://nodejs.org/
Express.js (Node.js web application framework)	4.16.1	https://expressjs.com/
React (JavaScript library for user interfaces)	18.2.0	https://react.dev/
Redux (JS library for state management)	4.2.1	https://redux.js.org/
ApexCharts.js (JavaScript charting library)	3.41.1	https://apexcharts.com/
Socket.IO (real-time bidirectional communication library)	4.7.1	https://socket.io/
MongoDB (document-oriented database)	6.0.3	https://www.mongodb.com/

Table 7. Software components and exact versions used in the experiments.

Software environment

To ensure reproducibility and modularity, the BeeViz system was developed using a multi-layered software stack supporting data preprocessing, model training, anomaly detection, and dashboard deployment.

Machine learning components were developed in *Python*, while the web services and dashboard relied on *JavaScript* technologies.

For AI model development, *TensorFlow* and *Scikit-learn* were used to implement and train the forecasting and anomaly detection models, with *Pandas* and *Matplotlib* supporting data preprocessing, time-series manipulation, and visualization. The trained models were deployed through *Flask* microservices, exposing REST APIs that enable real-time inference and anomaly monitoring.

On the application side, *Node.js* and *Express.js* handled backend APIs and communication with connected hives and the database. The web dashboard, implemented in *React.js* with *Redux* and *ApexCharts*, provides interactive visualization of sensor data, forecasts, and alerts.

Socket.IO was used to enable real-time bidirectional data streaming between the backend and the dashboard.

All data were stored in a *MongoDB* NoSQL database using a JSON-based schema, allowing efficient management of sensor records, predictions, and anomaly logs. The overall architecture ensures seamless integration between the AI services, database, and visualization components, enabling reliable real-time monitoring and decision support for connected beehives. The exact software components and versions used in the experiments are reported in Table 7.

All components are open-source and publicly available. The system was trained and tested on a workstation equipped with an Intel Core i7-11800H CPU, 64 GB RAM, and an NVIDIA GeForce RTX 3080 GPU (8 GB VRAM).

Discussion and future research directions

Several avenues for future research could further enhance BeeViz's capabilities. First, expanding the system to visualize, predict, and detect anomalies in additional metrics, such as CO₂ emissions, which indicate the metabolic state of the colony, or flight activity and external temperature, which influence internal hive conditions, would provide a more comprehensive assessment of hive health.

Second, given that our prediction and anomaly detection models are designed for general applicability, a collaborative learning approach could be explored. Instead of relying solely on the HOBOS dataset, the system could continuously improve its machine learning models by incorporating real-time data from connected hives.

This federated approach would allow for progressive refinement and adaptation of the models based on diverse environmental conditions and hive behaviors. Finally, a promising direction would be to analyze correlations between multiple hive parameters (e.g., humidity, temperature, and weight) to optimize predictive accuracy and anomaly detection. Understanding these interdependencies could lead to more robust and reliable insights for beekeepers³⁰.

Beyond the scope of this study, several recent works offer promising directions to enhance the robustness and intelligence of monitoring systems. For instance, DeepAutoD³¹ introduces a deep learning framework for unpacking protected Android applications, which could inspire secure data handling mechanisms in embedded hive sensors. Tan et al.³² leverage graph neural networks to infer BGP (Border Gateway Protocol) community attributes, an approach that could be adapted to model interactions across connected hives. Additionally, targeted adversarial example generation³³ and semantic-aware malware detection³⁴ suggest new ways to improve the resilience of predictive models against environmental noise. Research on community detection in graphs³⁵ and deep learning model security³⁶ further highlights the importance of designing adaptive, collaborative systems capable of evolving with real-world data. These contributions represent valuable foundations for future iterations of the BeeViz platform. By addressing these research directions, BeeViz could further strengthen data-driven hive management, contributing to more sustainable and informed beekeeping practices.

While the current system uses hive-specific models to ensure reliability, future work will explore collaborative learning and transfer learning strategies to improve generalization across diverse environments.

Conclusion

This paper presents BeeViz, a monitoring system designed to support precision beekeeping through continuous data collection, visualization, and intelligent analysis. The system ensures seamless communication with connected hives, enabling real-time acquisition and storage of key hive parameters such as internal temperature, humidity, and weight. Beyond simple data monitoring, BeeViz integrates predictive modeling and anomaly detection techniques to improve hive management. A comprehensive review of existing monitoring approaches has been conducted, highlighting how collected data can be leveraged for decision support in apiculture.

BeeViz generates web-based visualizations, allowing beekeepers to track hive dynamics over time and detect early warning signs of potential problems. In addition, the system incorporates advanced time-series analysis for anomaly detection and short-term forecasting, empowering beekeepers with actionable insights to anticipate and mitigate risks. By integrating intelligent analytics, BeeViz contributes to more efficient and proactive hive management.

Data availability

The datasets used in this study originate from the HOBOS (HoneyBee Online Studies) project, which monitored honeybee hives equipped with various environmental and biometric sensors. The data, covering parameters such as temperature, humidity, and hive weight, are publicly available through Kaggle at the following URL: <https://www.kaggle.com/datasets/se18m502/bee-hive-metrics>. These datasets were collected between 2016 and 2019 from instrumented beehives in Germany. The use of these data complies with the terms of the Kaggle dataset license, and no additional permissions were required for academic use.

Code availability

The code developed for the BeeViz monitoring system, including modules for time series forecasting and anomaly detection, is available from the corresponding author upon reasonable request.

Received: 24 June 2025; Accepted: 27 January 2026

Published online: 03 February 2026

References

1. FAO. Apimondia, CAAS, IZSLT: Good beekeeping practices for sustainable apiculture. In *Technical Report, United Nations, Food and Agriculture Organization (FAO)*. <https://openknowledge.fao.org/server/api/core/bitstreams/285dd834-945f-4964-8b7d-d2a4c5aab293/content> (2021). Accessed 21 May 2025.
2. European Parliament: Pollinator decline: causes, consequences and policy responses. https://www.europarl.europa.eu/pdfs/news/expert/2019/12/story/20191129STO67758/20191129STO67758_fr.pdf. Accessed Sep 2025 (2021).
3. ANSES: Les abeilles, des pollinisateurs essentiels dont la santé est menacée. <https://www.anses.fr/fr/content/les-abeilles-des-pollinisateurs-essentiels-dont-la-sante-est-menacee>. Accessed Sep 2025 (2024).
4. ADA. InterApi, ITSAP: Estimation de la production de miel 2024. In *Technical Report, Federation of the Regional Beekeeping Development Associations (ADA France)*. https://www.adafrance.org/wp-content/uploads/2024/11/Estimation_Prod_Miel_2024.pdf. Accessed 21 May 2025 (2024).
5. Kuan, Y. N., Goh, K. M. & Lim, L. L. Systematic review on machine learning and computer vision in precision agriculture: Applications, trends, and emerging techniques. *Eng. Appl. Artif. Intell.* **148**, 110401. <https://doi.org/10.1016/j.engappai.2025.110401> (2025).
6. Kriouile, Y., Ancourt, C., Wegrzyn-Wolska, K. & Bougueroua, L. Nested object detection using mask r-cnn: Application to bee and varroa detection. *Neural Comput. Appl.* **36**(35), 22587–22609. <https://doi.org/10.1007/s00521-024-10393-x> (2024).
7. Verbeke, W. et al. European beekeepers' interest in digital monitoring technology adoption for improved beehive management. *Comput. Electron. Agric.* **227**, 109556. <https://doi.org/10.1016/j.compag.2024.109556> (2024).
8. ITSAP. I.d.a. Enquête : les apiculteurs et le numérique. In *Technical Report*. <https://itsap.asso.fr/articles/enquete-les-apiculteurs-et-le-numerique>. Accessed 21 May 2025 (2020).
9. Kady, C. et al. Iot-driven workflows for risk management and control of beehives. *Diversity* <https://doi.org/10.3390/d13070296> (2021).
10. Zacepins, A., Stalidzans, E. & Meitalovs, J. Application of information technologies in precision apiculture. In *Proceedings of the 13th International Conference on Precision Agriculture (ICPA 2012)* (2012)

11. Tashakkori, R., Hernandez, N.P., Ghadiri, A., Ratzloff, A.P. & Crawford, M.B. A honeybee hive monitoring system: From surveillance cameras to raspberry PI. In *SoutheastCon 2017*. 1–7 (IEEE, 2017).
12. Bjerger, K. et al. A computer vision system to monitor the infestation level of varroa destructor in a honeybee colony. *Comput. Electron. Agr.* **164**, 104898. <https://doi.org/10.1016/j.compag.2019.104898> (2019).
13. Magnier, B., Gabbay, E., Bougamale, F., Moradi, B., Pfister, F. & Slangen, P. Multiple honey bees tracking and trajectory modeling. In *Multimodal Sensing: Technologies and Applications* (Stella, E. ed.). Vol. 11059. 110590 (SPIE, 2019). <https://doi.org/10.1117/12.526120> (International Society for Optics and Photonics).
14. Jeong, K. et al. IoT and AI systems for enhancing bee colony strength in precision beekeeping: A survey and future research directions. *IEEE Internet Things* **12**(1), 362–389. <https://doi.org/10.1109/JIOT.2024.3461775> (2025).
15. Cecchi, S., Spinsante, S., Terenzi, A. & Orcioni, S. A smart sensor-based measurement system for advanced bee hive monitoring. *Sensors* **20**(9), 2726. <https://doi.org/10.3390/s20092726> (2020).
16. Abou-Shaara, H., Owayss, A., Ibrahim, Y. & Basuny, N. A review of impacts of temperature and relative humidity on various activities of honey bees. *Insect. Soc.* **64**, 455–463. <https://doi.org/10.1007/s00040-017-0573-8> (2017).
17. Singh, I. & Singh, S. Honey moisture reduction and its quality. *J. Food Sci. Technol.* **55**(10), 3861–3871. <https://doi.org/10.1007/s13197-018-3341-5> (2018).
18. Polatidou, K., Nouska, C., Tananaki, C., Biliaderis, C. G. & Lazaridou, A. Physicochemical and rheological characteristics of monofloral honeys-kinetics of creaming-crystallization. *Foods* <https://doi.org/10.3390/foods14101835> (2025).
19. Ochoa, I.Z., Gutierrez, S. & Rodríguez, F. Internet of things: Low cost monitoring beehive system using wireless sensor network. In *2019 IEEE International Conference on Engineering Veracruz (ICEV)*. Vol. 1. 1–7 (IEEE, 2019).
20. Robustillo, M. C., Pérez, C. J. & Parra, M. I. Predicting internal conditions of beehives using precision beekeeping. *Biosyst. Eng.* **221**, 19–29. <https://doi.org/10.1016/j.biosystemseng.2022.06.006> (2022).
21. Erdal, B. & Tolga, T. Time series forecasting of honey production in Turkey. *Avrupa Bilim Teknol. Dergisi* **35**, 417–423. <https://doi.org/10.31590/ejosat.1066665> (2022).
22. Anwar, O., Keating, A., Cardell-Oliver, R., Datta, A. & Putrino, G. We-bee: Weight estimator for beehives using deep learning. In *36th AAAI Conference on Artificial Intelligence: AAAI-22* (2022)
23. Meireles Filho, M. & Sousa, A. et al. Application of artificial intelligence through neural networks to predict honey production based on weather patterns. In *ENCOINFO-Congresso de Computação e Tecnologias da Informação*. 15–21 (ENCOINFO, 2021).
24. Zacepins, A., Kviestis, A., Stalidzans, E., Liepniece, M. & Meitalovs, J. Remote detection of the swarming of honey bee colonies by single-point temperature monitoring. *Biosyst. Eng.* **148**, 76–80. <https://doi.org/10.1016/j.biosystemseng.2016.05.012> (2016).
25. Rubin-Delanchy, P., Lawson, D.J. & Heard, N.A. Anomaly detection for cyber security applications. In *Dynamic Networks and Cyber-security*. 137–156. <https://doi.org/10.1142/q0022> (World Scientific, 2016).
26. Chen, S.-H., Wang, J.-C., Lin, H.-J., Lee, M.-H., Liu, A.-C., Wu, Y.-L., Hsu, P.-S., Yang, E.-C. & Jiang, J.-A. A machine learning-based multiclass classification model for bee colony anomaly identification using an IOT-based audio monitoring system with an edge computing framework. *Expert Syst. Appl.* **255**, 124898. <https://doi.org/10.1016/j.eswa.2024.124898> (2024).
27. Davidson, P., Steininger, M., Lautenschlager, F., Krause, A. & Hotho, A. Anomaly detection in beehives: An algorithm comparison. In *International Conference on Sensor Networks*. 1–20 (Springer, 2020).
28. Senger, D., Johannsen, C., Melemenidis, A., Goncharskiy, A. & Kluss, T. Unsupervised anomaly detection on multisensory data from honey bee colonies. In *2020 IEEE International Conference on Data Mining (ICDM)*. 1238–1243 (IEEE, 2020).
29. Metidji, S.A., Huet, J.-C. & Bougueroua, L. Data preparation for prediction and anomaly detection in precision beekeeping. *Proc. Comput. Sci.* **246**, 4008–4017. <https://doi.org/10.1016/j.procs.2024.09.175> (2024) (28th international conference on knowledge based and intelligent information and engineering systems, KES, 2024).
30. Robustillo, M.C., Naranjo, L., Parra, M.I. & Pérez, C.J. Addressing multidimensional highly correlated data for forecasting in precision beekeeping. *Comput. Electron. Agric.* **226**, 109390. <https://doi.org/10.1016/j.compag.2024.109390> (2024).
31. Zhao, Y., Liu, Y. & Zhang, Y. et al. Deepautod: Research on distributed machine learning oriented scalable mobile communication security unpacking system. In *IEEE Transactions on Network and Service Management*. Vol. 18. 1234–1247. <https://doi.org/10.1109/tnse.2021.3100750> (2023).
32. Tan, Y., Huang, W., You, Y., Su, S. & Lu, H. Recognizing bgp communities based on graph neural network. *IEEE Netw.* <https://doi.org/10.1109/mnet.2024.3414113> (2024).
33. Hu, W., Gu, Z., Zhang, C., Wang, L. & Tang, K. Adversarial examples generation system based on gradient shielding of restricted region. In *Artificial Intelligence and Security (ICAIS). Communications in Computer and Information Science*. Vol. 1254. 81–91. https://doi.org/10.1007/978-981-15-8101-4_9 (2020).
34. Zhang, Y., Yang, S., Xu, L., Li, X. & Zhao, D. A malware detection framework based on semantic information of behavioral features. *Appl. Sci.* **13**(22), 12528. <https://doi.org/10.3390/app132212528> (2023).
35. Li, J., Lai, S. & Shuai, Z. et al. A comprehensive review of community detection in graphs. arXiv preprint [arXiv:2309.11798](https://arxiv.org/abs/2309.11798). <https://doi.org/10.48550/arXiv.2309.11798> (2023).
36. Wang, T., Bi, Z. & Zhang, Y. et al. Deep learning model security: Threats and defenses. arXiv preprint [arXiv:2412.08969](https://arxiv.org/abs/2412.08969). <https://doi.org/10.48550/arXiv.2412.08969> (2024).

Author contributions

J.C.H. and L.B. contributed equally to the conceptualization, validation, original draft writing, review and editing, visualization, supervision, and project administration. S.A.M. was responsible for formal analysis, investigation, data curation, and contributed to the review and editing of the manuscript. All authors have read and approved the final version of the manuscript.

Declarations

Competing Interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.-C.H.

Reprints and permissions information is available at www.nature.com/reprints.

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

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, 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 you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. 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-nc-nd/4.0/>.

© The Author(s) 2026